КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. И. АРАБАЕВА

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.РАЗЗАКОВА

На правах рукописи

УДК 004.8:631(575.2) (043)

САБИТОВ БАРАТБЕК РАХМАНОВИЧ

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ЗАДАЧАХ ЦИФРОВОГО СЕЛЬСКОГО ХОЗЯЙСТВА

Специальность: 05.13.16 – Применение вычислительной техники, математического моделирования и математических методов в научных исследованиях

Диссертация на соискание ученой степени
доктора физико-математических наук

Бишкек – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В диссертационной работе исследуется использование современных методов искусственного интеллекта применительно к задачам сельского хозяйства. Кыргызская республика является аграрной страной, и решение задач поддержки сельского хозяйства с развитием достижений науки и современных технологии является актуальной задачей на сегодня. Чрезвычайно важной задачей является и поддержка продовольственной безопасности страны. В работе подробно рассмотрены процессы построения различных моделей прогнозирования с применением машинного обучения, глубокого обучения и компьютерного зрения в сельском хозяйстве.

Актуальность темы диссертации. Важную роль во всем мире играет продовольственная безопасность, которая является важным рычагом в успехе экономике страны в целом. Продовольственная безопасность для любой страны, это продукты питания, и они рассматривается как основная потребность человека, которая может быть удовлетворена с помощью сельского хозяйства. Сельское хозяйство не только удовлетворяет основные потребности человека, но также считается источником занятости во всем мире. Сельское хозяйство считается основой экономики и источником занятости в аграрных и развивающихся странах, например, таких как Кыргызская Республика.

При этом основной составляющей сельского хозяйства является урожайность сельскохозяйственных культур. Урожайность, которая встречается во многих сельскохозяйственных проектах относиться к сложным категориям для моделирования и прогнозирования. А также может определяться как совокупность множества климатических факторов и естественных условий окружающей среды, где выращивается данная культура и требует всестороннего исследования. Таким образом, точное прогнозирование урожайности сельскохозяйственных культур требует фундаментального понимания функциональной зависимости урожайности с множеством факторов, которые могут влиять напрямую или косвенно, например, климатические факторы, как температура и влажность воздуха и периодичность, и качество осадков, а также эрозия, кислотность и состав почвы посевных площадей и т п. Для построения модели, также необходимо установить и другие взаимосвязи урожайности, например, с явлениями изменения климата и неустойчивости погодных условий. К дополнительным факторам можно отнести и характеристики почвы посевных площадей, принимая во внимание качественный состав почвы, обработанной различными удобрениями, а также требуют учета результаты семеноводческих исследований генотипа выращиваемой культуры в определенном регионе.

Важность этих факторов, подробно рассматривается во многих научных исследованиях. Очень важно для выявления взаимосвязи иметь достоверные наборы данных так и мощные алгоритмы моделирования и прогнозирования данного процесса. Машинное обучение является важным инструментом поддержки принятия решений для прогнозирования урожайности, включая поддержку решений в вопросе, какие культуры выращивать и что делать в течение вегетационного периода для выращивания сельскохозяйственных культур.

В связи с этим весьма актуальным является разработка методов и инструментов исследования, основанных на научных достижениях в задачах моделирования и прогнозирования урожайности сельскохозяйственных растений. Хотя изучение множества факторов, влияющих на здоровый рост выращиваемой культуры, который в свою очередь является залогом урожайности культуры не простейшая, но важная задача. Категорию урожайность, с точки зрения математического моделирования можно представить в виде сложной нелинейной функции, зависящей от природных явлений, различных характеристик почвы и многих других признаков, и решение задачи прогнозирования урожайности представляет собой трудную задачу.

Своевременная и точная интерпретация болезней выращиваемых культур можно классифицировать как подзадачу основной задачи прогнозирования урожайности культуры. Так как многие болезни растений, появляются при неправильном использовании пестицидов и могут привести к тому, что долгоживущие патогены вырабатывают устойчивость и серьезно снижают способность растения противостоять им. Крайне важно, чтобы производство было более здоровым за счет решения проблемы развития длительной устойчивости к патогенам и смягчения неблагоприятных последствий изменения климата для сохранности финансовых и других ресурсов.

Адекватное и своевременное выявление заболеваний культуры, включая ранние мероприятия препятствующие развитию болезни культуры, всегда были наиболее важными в изменяющихся условиях роста культуры. Патологии растений можно обнаружить несколькими способами. Некоторые заболевания растений не имеют явных синдромов, в таких ситуациях требуется тщательное обследование. Тем не менее, при большинстве болезней, на выращиваемых культурах появляются какие-то видимые признаки заболевания, так что квалифицированный профессиональный осмотр является основным методом обнаружения болезней растений. Например, опытные агрономы обладают наилучшими навыками наблюдения и распознавания болезней, чтобы идентифицировать характерные симптомы болезни, и получить точную

диагностику заболевания. Изменения симптомов у больных растений, могут привести к неточному диагнозу заболеваемости, так как агрономы новички и сельхозпроизводители, не имеющих сельскохозяйственного образования и многолетнего опыта, не могут точно распознать болезнь культуры, по сравнению с профессиональном. Решением этой проблемы может стать автоматизированная система, разработанная для определения состояния растения по внешнему виду и в качестве верификатора диагноза заболевания культуры по данным визуального осмотра и распознавания образа заболевания. Таким образом новички в садоводстве и опытные специалисты могут получить большую пользу от автоматизированных систем в области компьютерного зрения, которые открывают возможности для расширения и укрепления практики точной защиты растений и расширения рынка приложений для точного компьютерного зрения в сельском хозяйстве.

Глубокое обучение — это новая тенденция в машинном обучении, которая дает передовые результаты во многих областях исследований, включая компьютерное зрение. Глубокое обучение выигрывает от возможности использовать необработанные данные напрямую, без использования ручного труда. Использование глубокого обучения по двум основным причинам в последнее время дало хорошие результаты, как в академической, так и в промышленной сфере. Во-первых, генерируются большие объемы данных. Таким образом, эти данные могут быть использованы для разработки глубокой модели. Во-вторых, вычислительная мощность графического процессора, позволяет обучать глубокие модели и использовать их для параллелизма вычислений.

Цели и задачи исследования. В диссертационной работе изучены современные алгоритмы машинного обучения, которые были применены к задачам прогнозирования урожайности для различных сельскохозяйственных культур. Для поддержки исследований по прогнозированию и распознаванию болезней растений использованы новые технологии и методы глубокого обучения. В диссертационной работе приведено систематическое изучение современных литератур (SLR) для извлечения и синтеза алгоритмов и функций, которые используются в задачах по прогнозированию урожайности и распознаванию болезней растений. После тщательного изучения исследований и анализа использованных методов и их особенностей, внесено предложения для дальнейших исследований в этой области. Согласно анализу, наиболее часто используемыми функциями для обучения модели, являются температура, количество осадков и тип почвы, а наиболее применяемым алгоритмом в построении этих моделей являются нейронные сети. Согласно этому

дополнительному анализу, сверточные нейронные сети (CNN) являются наиболее широко используемым алгоритмом глубокого обучения в этих исследованиях, а другими широко используемыми алгоритмами глубокого обучения являются долговременная память (LSTM) и глубокие нейронные сети (DNN). Согласно этому дополнительному анализу, сверточные нейронные сети (CNN) являются наиболее широко используемым алгоритмом глубокого обучения в этих исследованиях, а другими широко используемыми алгоритмами глубокого обучения являются долговременная память (LSTM) и глубокие нейронные сети (DNN).

Такие технологии, как машинное обучение, глубокое обучение, облачные вычисления, граничные вычисления используются для получения данных, а также их предварительный анализ и ее подготовки к построению моделей. Приложения компьютерного зрения, машинного обучения, Интернета вещей помогут поднять производство, повысить качество и, в конечном итоге, повысить прибыльность фермеров и связанных с ними областей. Построение точной модели в области сельского хозяйства очень важно для повышения общей урожайности культур.

Представлено множество приложений машинного обучения, в разных областях. Алгоритмы глубокого обучения делают машинное обучение более мощным и точным. Используя автоматизированное машинное обучение, можно сократить потребность в специалистах по машинному обучению, автоматизировать конвейер машинного обучения с большей точностью.

Новизна диссертационной работы. В диссертационной работе с использованием методов машинного обучения исследуется обширный круг задач в сельском хозяйстве. Важной и ключевой категорией в секторе сельского хозяйства, является категория урожайность.

С применением технологий глубокого обучения и нейронных сетей исследуется сложный раздел компьютерного зрения болезни сельскохозяйственных растений.

Изучен, также применения самых последних методов системы машинного или компьютерного зрения, используемых для классификации и обнаружения болезней растений сельского хозяйства.

Построены самые различные модели по проблемам использования машинного обучения (МО), глубокого обучения (ГО) и их развертывания, которое можно имитировать как искусственный интеллекта (ИИ) для задач сельского хозяйства.

Разработаны искусственный интеллект основанное на Фреймворках Python.для фермеров и сельхозпроизводителей для оптимального управления

урожайностью и определения, и идентификации болезней растений с помощью веб приложений.

Были рассмотрены различные современные методы машинного обучения и глубокого обучения для создания моделей урожайности и болезни различных сельскохозяйственных растений, которые можно обобщить как трансферное обучение/

Технология глубокого обучения в данной работе используются совместно с методами компьютерного зрения. Использование CNN в сельском хозяйстве огромен, и он также получает замечательные результаты. Благодаря использованию глубины, другой структуры и аппаратной поддержки обучаемость и точность CNN значительно улучшаются.

В диссертационной работе построены приложения для прогнозирования различных болезней сельскохозяйственных растений. Тем не менее, есть проблемы, такие как сбор набора данных, время, необходимое для обучения и тестирования, поддержка оборудования, развертывание больших моделей на небольших устройствах, таких как веб сайты или телефоны Android.

Проблемы и рекомендации. Из этого обзора можно понять важность машинного обучения в области сельского хозяйства. На каждом этапе ведения сельского хозяйства, начиная с предуборочной компании и заканчивая послеуборочной, исследователи применяли алгоритмы машинного обучения для решения сложных задач.

Сегодняшняя потребность заключается в разработке точных и настраиваемых моделей машинного обучения, которые могут работать быстро, автоматически анализировать большие и сложные данные и помогать оптимизировать сельскохозяйственные процессы, такие как классификация, рекомендации или прогнозы. До появления современных технологий прогнозирования урожайности, основанных на научных подходах, фермеры использовали свой опыт выращивания культур, что приводило к различным результатам.

В отдельные годы, выращивания однотипных культур (например, только картофеля) для многих регионах одновременно в результате сбора урожаев сильно падала их стоимость на рынке и повторение этого процесса из года в год, приводило к процессу оттока фермеров из сельского хозяйства. Этот термин, введенный автором, требует отдельного исследования с применением машинного обучения. Интересный процесс требует изучения с точки зрения связи урожайности с учетом миграции населения, особенно молодежи. Есть множество факторов, которые оказывают влияние на урожайность сельскохозяйственных культур, таких как площадь засева растений, правильная

предобработка почвы, эффективное использование оросительных систем и их совершенствование, необходимо учитывать изменения погоды и особенности осадков, температура и влажность воздуха, поддержка и совершенствование компаний для вывода новых сортов сельскохозяйственных культур с учетом региональных особенностей. Правильное ведение агротехнического земледелия в использовании удобрений, ведение непрерывного наблюдения и своевременное выявление болезней сельскохозяйственных растений позволит повысить урожайность выращиваемых культур.

Преимущества машинного обучения и особенно глубокого в области сельского хозяйства огромны. Тем не менее, преимущества приходят с его проблемами. Некоторые из таких проблем при реализации алгоритмов машинного и построения сложных нейронных сетей с ее алгоритмами обучения в области сельского хозяйства перечислены ниже:

Структура диссертации. Диссертационная работа состоит из введения, где отражено обзор научного направления и анализ методов современных достижений в области машинного, глубокого обучения и компьютерного зрения, для задач сельского хозяйства, которые исследованы в диссертационной работе.

Диссертация состоит четырех глав, в которых отражено основное содержание, методология и методы исследования, а также полученные результаты по моделированию и прогнозированию урожайности и болезни сельскохозяйственных культур, заключения и списка литератур и приложений. Диссертация изложена на 252 страницах компьютерного текста, содержит 6 таблицы, 133 рис, 75 листингов, 25 диаграмм, а также содержит 15 приложений написанные на языке программирования Python, в которых приведены таблицы с данными, результаты по математическому моделирования и численные расчеты.

В первой главе диссертации исследовано современное состояние и анализ публикаций по применению машинного, глубокого обучения и компьютерного зрения, опубликованные в последние годы для задач сельского хозяйстве. Конкретно для задач распознавания изображений сделан обзор научных публикаций и технологий распознавания болезней широкого круга растений. Выделены основные направления для моделирования и прогнозирования задач сельского хозяйства.

Предложены основные алгоритмы машинного обучения и методы глубокого обучения, основанные на нейронных технологиях. Во второй главе диссертации отражено полный анализ и исследования, а также методы и методологии машинного обучения, ориентированной для задач цифрового сельского хозяйства и математическое описание методов машинного обучения.

Рассмотрено анализ построения моделей и методология применения алгоритмов машинного обучения для широкого круга прикладных задач сельского хозяйства в том числе.

Третья глава диссертации посвящена построению моделей для задач цифрового сельского хозяйства алгоритмами и методами машинного обучения. Представлен обзор соответствующих результатов, полученных автором на основе мощных алгоритмов машинного обучения для прогнозирования урожайности и распознавания болезней растений по различным сельскохозяйственным культурам.

Приведены численные результаты, выполненные с помощью методов оптимизации - градиентного спуска. Несмотря на множество публикуемых научных работ по этой области, численные подходы в задачах машинного обучения еще остается проблемой для многих сельскохозяйственных задач. Так как найти точный прогноз урожайности не удается во многих случаях, поэтому численные подходы для этой области необходимо расширить.

Современное состояние исследований в области моделирования и прогнозирования задач урожайности, обосновывает его как наукоемкую категорию и представляет с собой нелинейное моделирования, которая зачастую не удается решить обычными численными методами. Для его исследования необходимо применить новейшие методы и технологии искусственного интеллекта. В последнее время с применением современных технологий машинного обучения и глубокого обучения в различных областях, появились возможности исследовать задачи прогнозирования урожайности с большими данными о природных явлениях. Обзор научных исследований показывает, что сейчас более внимательно изучают применение машинного и глубокого обучения в этой области, для более точного моделирования, включая использования данных дистанционного зондирования и спутниковой архитектуры. Рассмотренные выше технологии машинного обучения и результаты, полученные автором по глубокому обучению, рассмотрены в третьей главе.

Серьезному направлению исследования задач сельского хозяйства, одного из точных и многообещающих методов искусственного интеллекта является глубокое обучение, и методы распознавания объектов с помощью компьютерного зрения с различными архитектурами и технологиями нейронных сетей, которое занимает лидирующее положение, рассмотрено в четвертой главе диссертационной работы. Изучению множества сложных нелинейных взаимосвязей между признаками и данными, влияющих на точное моделирование и прогнозирования урожайности и болезни растений по

изображениям растений, в настоящее время стало возможным благодаря появлению мощных алгоритмов машинного обучения и с проектированием нейронных сетей глубокого обучения различной архитектуры с предварительно обученными сетями. При этом второе направление, должно использовать различные архитектуры нейронных сетей, включая уже известные как трансферное обучение моделей, основой которых являются уже обученные модели на BigData.

Апробация результатов диссертации. В диссертационной работе исследованы методы машинного, глубокого обучения, компьютерного зрения и построения моделей для многочисленных задач сельского хозяйства. Основные результаты диссертации апробированы под непосредственным руководством автора во многих проектах МОиН за 2017-2023 годы. Задачи и проблемы в ходе выполнения проектов включены в Государственные программы по искусственному интеллекту, как приоритетные научные направления Кыргызской республики и Министерства образования и науки (МОиН), финансируемых МОиН КР. Работа выполнена на кафедре Прикладной информатики Киргизского государственного университета им. И. Арабаева.

Проекты

1.«Разработка и создание новых информационных технологий и интеллектуальной экспертной системы для сферы АПК КР (инвестиционные процессы, инфраструктура и логистика АПК)», 2017 год.

2.«Моделирование и прогнозирование в сфере АПК КР с применением интеллектуальных систем, Python технологий и нейронных сетей», 2018-2021гг.

3.«Технологии внедрения искусственного интеллекта в систему общеобразовательного образования», 2021-2022 гг.

4. «Искусственный интеллект в сельском хозяйстве» (2023-2025 гг.)

Научные результаты, полученные в исследовательской диссертационной работе, докладывались во многих международных и Вузовских конференциях КР в частности:

- Международная научная конференция “Технологии и перспективы современного инженерного образования, науки и производства”, посвященная 45-летию ФПИ – КТУ им. И. Раззакова – Бишкек, 1999 г.

- Международная научно-практическая конференция «Применения цифровых технологий в образовании: проблемы и перспективы». Вестник Кыргызского национального университета им. Ж.Баласагына, Труды. Бишкек. 2019 г.

- Международная научно-практическая конференция. «Научно-технологическое развитие АПК для целей устойчивого развития»

«Моделирование и прогнозирование задач сельского хозяйства на основе машинного обучения». Труды. Бишкек. 2022 г.

- Межвузовская научно-практическая конференция «Цифровые технологии в отраслях производства и социальной сфере», 27 окт. 2022, Астана, Дакка

- Международная научно-практической конференция. «Роль науки и инновационных технологий в устойчивом развитии горных территорий и экосистем». 27-28 октября 2022 г. Бишкек, Кыргызская Республика.

Практическая значимость полученных результатов. Все основные научно-исследовательские работы соискателя имеют прикладной характер, и основные научные результаты имеют значительную внедренческую ценность, в Государственную программу по продовольственной безопасности страны, сельское хозяйство в целом и искусственному интеллекту. Многие модели построенные, в диссертации основываются на реальных данных и имеют ценное практическое значение в прогнозировании задач сельского хозяйства.

Личный вклад соискателя. Все основные положения, концепции, теории, цели и задачи, научно-практические результаты получены соискателем.

Полнота отражения результатов диссертации в публикациях. Основные результаты диссертации отражены в публикациях автора [118]-[144],[181]-[182] в том числе в Web of Science, Scopus [181],[182] и посвящены задачам прогнозирования урожайности и распознавания болезней растений, основанные на данных Иссык-Кульской области за последние годы. Часть работ посвящены к задачам распознавания изображений. Получены результаты по выявлению болезней томатов Чуйской области, кукурузы в Иссык-Кульском регионе и других сельскохозяйственных растений на основе глубокого обучения. Результаты опубликованы в зарубежном индексированном журнале Web of Science [147] по распознаванию болезней груши на некоторых садовых участках изучаемого региона. Многие модели построены на основе трансферного обучения, т.е. на основе уже обученных моделей на данных большого размера.

В диссертации 134 рисунков, 75 листингов, 6 таблиц и наименований литератур 182, из них публикаций автора по тематике диссертации 29.

ГЛАВА 1
ОБЗОР НАУЧНЫХ ПУБЛИКАЦИЙ ПО ПРОГНОЗИРОВАНИЮ ЗАДАЧ СЕЛЬСКОГО ХОЗЯЙСТВА МЕТОДАМИ МАШИННОГО И ГЛУБОКОГО ОБУЧЕНИЯ

В данной главе диссертационной работы проведены обзор и анализ современных научных публикаций за последние годы по моделированию и прогнозированию задач сельского хозяйства с использованием методов машинного и глубокого обучения. Основное внимание уделено публикациям по прогнозированию урожайности и болезни сельскохозяйственных растений методами искусственного интеллекта.

Результаты расчетов также показали, что построенная модель на основе алгоритмов машинного обучения множественной регрессии (LR), регрессии Лассо (Lasso R), стохастический градиентный спуск (SGD), дерево решений (RT), которые дают ощутимый результат прогнозирования с расчетными показателями с MAPE=11%. Показано также, что мощные регрессионные алгоритмы машинного обучения как K – ближайших соседей (KNN), случайный лес (RF), метод опорных векторов (SVR) и градиентный бустинг (GBR) дают ощутимый результаты в прогнозирования по сравнению с другими методами машинного обучения, например, с MAPE=10 %.

Анализ результатов подробно приведены в 3 и 4 главах диссертации. Анализ результатов показали, что факторы окружающей среды (погодные условия) в большей степени влияют на урожайность, чем генотип почвы. Есть и другие факторы, которые оказывают влияние на урожайность сельскохозяйственных культур, таких как площадь засева растений с правильной предобработкой, эффективное использование оросительных систем и ее совершенствования, учитывать изменения погоды и особенности осадков и температуры. Со всеми вышеперечисленными условиями в данной работе исследовалось урожайность культур с применением методов и алгоритмов машинного обучения.

В этом направлении для урожайности пшеницы в [1] рассмотрели, применение K – ближайших соседей и дерево решений. В [2] с применением алгоритмов наивный Байесовский процесс и K – ближайших соседей с использованием данных компаний по выводу новых сортов сельскохозяйственных культур рассмотрена задача прогнозирования. В работе [3] на основе алгоритма случайный лес и линейной регрессии рассмотрена задача прогнозирования с учетом региональных и глобальных особенностей, при

выборе выращивания определенных видов растений. Для изучения урожайности риса в [4], с помощью метода опорных векторов получены результаты прогноза с учетом топографических признаков [5] местности с субтропическим климатом на основе алгоритмов дерево решений, логистической регрессии и K – ближайших соседей. В работе [6] с использованием методов наивный Байесовский процесс, случайный лес, нейронные сети, дерево решений и машины опорных векторов изучена задача классификации для задач урожайности. В [7], [8] исследовались прогнозирование урожайности с использованием анализа изображений фруктов с помощью нейронных сетей. В частности, изучалась задача распознавания с использованием сегментации изображений для обнаружения плодов и оценки урожайности в яблоневых садах. Правильное ведение агротехнического земледелия с использованием удобрений, вести непрерывное наблюдение и своевременное выявление потребностей сельскохозяйственных растений, играет ключевую роль в получении желаемого урожая изучалось в [9].

Исследования последних лет показывают, что результаты комплекса алгоритмов – ансамбля, составляющие, которых могут быть слабыми алгоритмами применяется ко многим прикладным задачам в том числе к задачам прогнозирования урожайности [10], [11], [12],[13].

Большую и особую роль в задачах сельского хозяйства сыграли нейронные сети глубокого обучения с технологиями компьютерного зрения для распознавания болезней и вредителей растений. Болезни и вредители растений являются важными факторами, определяющими урожайность и качество растений. Идентификацию болезней и вредителей растений можно проводить с помощью цифровой обработки изображений. В последние годы глубокое обучение совершило прорыв в области цифровой обработки изображений, намного превосходящий традиционные методы. В обзоре дается определение проблемы обнаружения болезней и вредителей растений, проводится сравнение с традиционными методами обнаружения болезней и вредителей растений.

В диссертации проведены анализ и перспективы будущих тенденций обнаружения болезней и вредителей растений на основе глубокого обучения. в этом исследовании обсуждаются возможные проблемы в практическом применении обнаружения болезней растений и вредителей на основе глубокого обучения. Кроме того, предлагаются возможные решения и исследовательские идеи для проблем, а также дается несколько предложений. Наконец, это исследование дает анализ и перспективы будущих тенденций обнаружения болезней и вредителей растений на основе глубокого обучения. В этом исследовании обсуждаются возможные проблемы в практическом применении

обнаружения болезней растений и вредителей на основе глубокого обучения. Кроме того, предлагаются возможные решения и исследовательские идеи для проблем, а также дается несколько предложений.

Обзор публикаций по глубокому обучению. Обнаружение болезней и вредителей растений является одной из важных подзадач исследований в области машинного зрения. Это технология, которая использует методы машинного зрения для получения изображений, чтобы определить, есть ли болезни и вредители на собранных изображениях растений [14]. В настоящее время технологии для обнаружения болезней растений и вредителей на основе машинного зрения первоначально применялось в сельском хозяйстве и в некоторой степени заменило традиционную идентификацию невооруженным глазом.

Для традиционного метода обнаружения болезней растений и вредителей на основе машинного зрения часто используются обычные алгоритмы обработки изображений или ручное проектирование признаков плюс классификаторы [15]. Этот тип метода обычно использует различные свойства болезней и вредителей растений для разработки схемы изображения и выбирает соответствующий источник света и угол съемки, что помогает получать изображения с равномерным освещением. Хотя тщательно построенные схемы визуализации могут значительно снизить сложность разработки классического алгоритма, они также увеличат стоимость приложения. В то же время в естественных условиях часто нереально ожидать, что классические алгоритмы призваны полностью исключить влияние смены сцены на результаты распознавания [16]. В реальной сложной природной среде обнаружение болезней растений и вредителей сталкивается со многими проблемами, такими как небольшая разница между площадью поражения и фоном, низкая контрастность, большие различия в масштабе области поражения и различных типов, много шума. на изображении поражения. Также много помех при сборе изображений болезней и вредителей растений в условиях естественного освещения. В это время традиционные классические методы часто оказываются бессильными, и добиться лучших результатов обнаружения сложно.

В последние годы, успешно применяются модели глубокого обучения, представленные сверточной нейронной сетью (CNN), которые нашли применения во многих областях компьютерного зрения таких как, обнаружение трафика [17], распознавание медицинских изображений [18], распознавание текста сценария [19], распознавание выражений лица [20], распознавание зрачка глаза [21] .

Определение болезней растений и обнаружение вредителей

По сравнению с определенными задачами классификации, обнаружения и сегментации в компьютерном зрении [22], требования обнаружения болезней растений и вредителей очень общие. По сути, его требования можно разделить на три различных уровня: что, где и как [23]. На рис.1.1. представлены методы исследования и обнаружения болезней расетний на основе глубокого обучения.

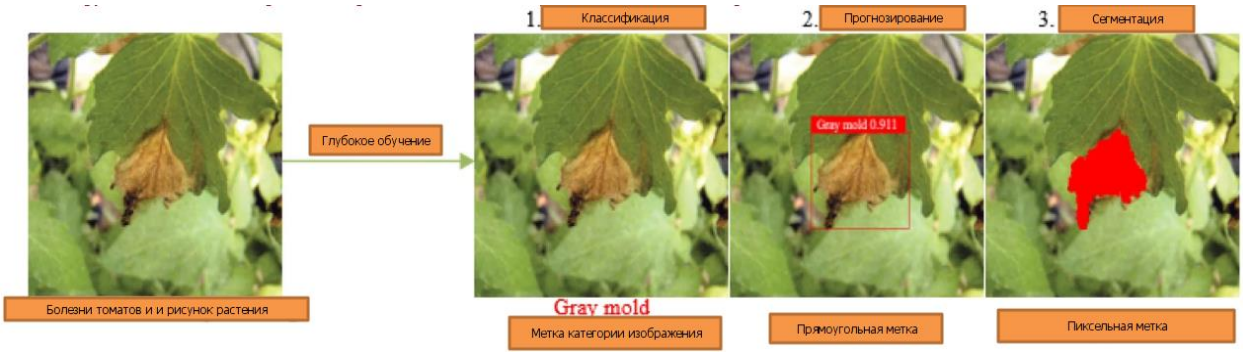


Рис.1.1. Обнаружения болезней и вредителей растений (болезнь Gray mold - серая плесень)

Сравнение с традиционными методами обнаружения болезней и вредителей растений. Чтобы лучше проиллюстрировать характеристики методов обнаружения болезней и вредителей растений, основанных на глубоком обучении, согласно существующим источникам [24 , 25 , 26 , 27 , 28], дано сравнение с традиционными методами обнаружения болезней и вредителей растений.

Теория глубокого обучения. Концепция глубокого обучения – DeepLearning (DL) возникла из статьи, опубликованной в журнале Science Hinton et al. [29] в 2006 году. Основная идея глубокого обучения заключается в использовании нейронной сети для анализа данных и изучения признаков. Признаки данных извлекаются несколькими скрытыми слоями, каждый скрытый слой можно рассматривать как персептрон, персептрон используется для извлечения низкоуровневых данных функции, а затем комбинировать низкоуровневые функции для получения абстрактных высокоуровневых функций, что может значительно облегчить проблему локального минимума. Глубокое обучение преодолевает тот недостаток, что традиционные алгоритмы полагаются на искусственно созданные функции, и привлекает все больше и больше внимания исследователей. В настоящее время он успешно применяется в компьютерном зрении, распознавании образов, распознавании речи, обработке естественного языка и системах рекомендаций [30].

Традиционные методы классификации изображений и распознавания признаков ручного проектирования могут извлекать только основные признаки, и сложно извлечь глубокую и сложную информацию о свойствах изображения [31]. И метод глубокого обучения может решить это узкое место. Он может напрямую проводить неконтролируемое обучение на исходном изображении для получения информации о многоуровневых функциях изображения, таких как функции низкого уровня, промежуточные функции и семантические функции высокого уровня.

В настоящее время методы глубокого обучения разработали множество хорошо известных моделей глубоких нейронных сетей, включая глубокую сеть доверия (DBN), глубокую машину Больцмана (DBM), автоэнкодер с шумоподавлением стека (SDAE) и глубокую сверточную нейронную сеть (CNN).[32].

В последние годы самой популярной средой глубокого обучения является глубокая сверточная нейронная сеть. В области распознавания изображений использование этих моделей глубоких нейронных сетей для реализации автоматического извлечения признаков из многомерного пространства признаков дает значительные преимущества по сравнению с традиционными методами извлечения признаков вручную.

Сверточная нейронная сеть. Сверточные нейронные сети, сокращенно CNN, имеют сложную сетевую структуру и могут выполнять операции свертки. Как показано на рис. 1.2, модель сверточной нейронной сети состоит из входного слоя, слоя свертки, слоя объединения, слоя полного соединения и выходного слоя. В одной модели слой свертки и слой объединения чередуются несколько раз, и когда нейроны слоя свертки соединены с нейронами слоя объединения, полного соединения не требуется. CNN — популярная модель в области глубокого обучения. Причина кроется в огромной емкости модели и сложной информации, вызванной основными структурными характеристиками CNN, что позволяет CNN играть преимущество в распознавании изображений. В то же время успехи CNN в задачах компьютерного зрения способствовали росту популярности глубокого обучения.

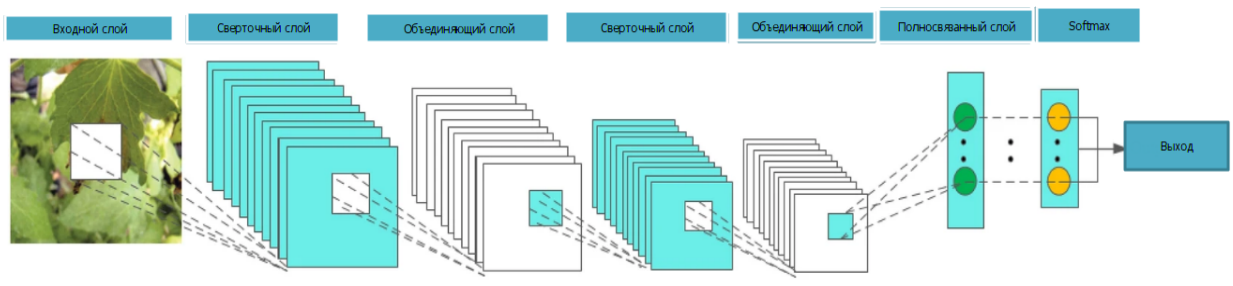


Рис. 1.2. Базовая структура CNN

В слое свертки сначала определяется ядро свертки. Ядро свертки можно рассматривать как локальное рецептивное поле, а локальное рецептивное поле является самым большим преимуществом нейронной сети свертки. При обработке информации о данных ядро свертки скользит по карте объектов, чтобы извлечь часть информации об объектах. После извлечения признаков из слоя свертки нейроны вводятся в объединяющий слой для повторного извлечения признаков. В настоящее время широко используемые методы объединения включают вычисление средних, максимальных и случайных значений всех значений в локальном рецептивном поле [33 , 34]. После поступления данных в несколько слоев свертки и слоев пула они попадают в слой полного соединения, а нейроны слоя полного соединения полностью связаны с нейронами верхнего слоя. Наконец, данные в слое полного соединения могут быть классифицированы методом softmax, а затем значения передаются на выходной слой для вывода результатов.

Инструменты с открытым исходным кодом для глубокого обучения.

Обычно используемые сторонние инструменты с открытым исходным кодом для глубокого обучения — это Tensorflow [35], Torch/PyTorch [36], Caffe [37], Theano [38]. Все четыре широко используемых сторонних инструмента глубокого обучения с открытым исходным кодом поддерживают кроссплатформенную работу, а платформы, которые можно запускать, включают Linux, Windows, iOS, Android. Torch/PyTorch и Tensorflow обладают хорошей масштабируемостью и поддерживают большое количество сторонних библиотек и структур глубокой сети, а также имеют самую высокую скорость обучения при обучении больших сетей CNN на графическом процессоре.

Методы обнаружения болезней и вредителей растений на основе глубокого обучения. В этом разделе дается краткий обзор методов обнаружения болезней и вредителей растений, основанных на глубоком обучении. Поскольку достигнутая цель полностью соответствует задаче компьютерного зрения, методы обнаружения болезней и вредителей растений, основанные на глубоком обучении, можно рассматривать как применение актуальных классических сетей в области сельского хозяйства.

Классификационная сеть. В реальной природной среде большие различия в форме, размере, текстуре, цвете, фоне, расположении и освещении изображений болезней растений и вредителей затрудняют распознавание. Из-за сильных возможностей CNN по извлечению признаков принятие сети классификации на основе CNN стало наиболее часто используемым шаблоном

для классификации болезней растений и вредителей. Как правило, часть извлечения признаков классификационной сети CNN состоит из каскадного слоя свертки + слоя объединения, за которым следует слой полного соединения (или средний слой объединения) + структура softmax для классификации. Существующие сети классификации болезней и вредителей растений в основном используют мутные сетевые структуры в компьютерном зрении, включая AlexNet [39], GoogleLeNet [40], VGGNet [41], ResNet [42], Inception V4 [43], DenseNets [44], MobileNet [45] и SqueezeNet [46]. Есть также некоторые исследования, которые разработали сетевые структуры, основанные на практических задачах [47 , 48 , 49 , 50]. Вводя тестовое изображение в классификационную сеть, сеть анализирует входное изображение и возвращает метку, которая классифицирует изображение. В соответствии с различием задач, решаемых с помощью метода сети классификации, его можно разделить на три подкатегории: использование сети в качестве экстрактора признаков, использование сети непосредственно для классификации и использование сети для определения местоположения поражений.

Использование сети в качестве средства извлечения признаков. В ранних исследованиях методов классификации болезней и вредителей растений, основанных на глубоком обучении, многие исследователи воспользовались мощными возможностями извлечения признаков CNN, и эти методы были объединены с традиционными классификаторами [51]. Сначала изображения вводятся в предварительно обученную сеть CNN для получения характеристик изображения, а затем полученные характеристики вводятся в обычный классификатор машинного обучения (например, SVM) для классификации. Ялчин Х. и др. [52] предложили архитектуру сверточной нейронной сети для извлечения признаков изображений при выполнении экспериментов с использованием классификаторов SVM с различными ядрами и дескрипторами признаков, такими как LBP и GIST, экспериментальные результаты подтвердили эффективность подхода. Фуэнтес А. и др. [53] выдвинули идею метаархитектуры на основе CNN с различными экстракторами признаков, а входные изображения включали здоровые и зараженные растения, которые были идентифицированы как соответствующие классы после прохождения метаархитектуры. Хасан М.Дж. и др. [54] идентифицировали и классифицировали девять различных типов болезней риса, используя признаки, извлеченные из модели DCNN и введенные в SVM, и точность достигла 97,5%.

Использование сети для классификации напрямую. Непосредственное использование сети классификации для классификации поражений является самым ранним распространенным средством CNN, применяемым для

обнаружения болезней растений и обнаружения вредителей. В соответствии с характеристиками существующей исследовательской работы ее можно подразделить на классификацию исходных изображений, классификацию после определения области интереса (ROI) и классификацию по нескольким категориям.

1. Оригинальная классификация изображений. То есть напрямую помещать собранное полное изображение болезней и вредителей растений в сеть для обучения. Тенможи К. и др. [55] предложили эффективную глубокую модель CNN, а трансферное обучение используется для точной настройки модели перед обучением. Виды насекомых были классифицированы по трем общедоступным наборам данных о насекомых с точностью 96,75%, 97,47% и 95,97% соответственно. Фанг Т. и др. [56] использовали ResNet50 для обнаружения болезней растений и вредителей. Функция потери фокуса использовалась вместо стандартной функции кросс-энтропийных потерь, а метод оптимизации Адама использовался для определения степени болезни листа, и точность достигла 95,61%.

2. Классификация после определения ROI. Для всего полученного изображения мы должны сосредоточиться на том, есть ли поражение в фиксированной области, поэтому мы часто заранее получаем область интереса (ROI), а затем вводим ROI в сеть, чтобы судить о категории болезней и вредителей. Нагасубраманян К. и др. [57] использовали новую трехмерную нейронную сеть глубокой свертки (DCNN) и метод визуализации карты значимости для идентификации здоровых и инфицированных образцов стеблевой гнили сои, и точность классификации достигла 95,73%.

3. Мультикатегориальная классификация. Когда количество классов болезней и вредителей растений, подлежащих классификации, превышает 2 класса, обычная сеть классификации болезней и вредителей растений аналогична исходному методу классификации изображений, то есть выходными узлами сети являются количество болезней растений и класс вредителей +1 (в т.ч. обычный класс). Однако методы классификации с несколькими категориями часто используют базовую сеть для классификации поражений и нормальных образцов, а затем совместно используют части извлечения признаков в той же сети для изменения или расширения ветвей классификации категорий поражений. Этот подход эквивалентен подготовке весового параметра перед обучением для последующей многоцелевой сети классификации болезней и вредителей растений, которая получается путем бинарного обучения между нормальными образцами и образцами болезней и вредителей растений. Пикон А. и др. [58] предложили архитектуру CNN для выявления 17 заболеваний в 5

культурах, которая легко интегрирует метаданные контекста, позволяя обучать одну модель для нескольких культур. Модель может достичь следующих целей: (a) получить более богатые и надежные общие визуальные характеристики, чем соответствующая отдельная культура; (b) не поражается различными болезнями, при которых разные культуры имеют сходные симптомы; (c) плавно интегрирует контекст для выполнения классификации условно-патогенных заболеваний сельскохозяйственных культур. Эксперименты показывают, что предложенная модель снимает проблему дисбаланса данных, а средняя сбалансированная точность составляет 0,98, что превосходит другие методы и устраняет 71% ошибок классификатора.

Проблема небольшого размера набора данных. В настоящее время методы глубокого обучения широко используются в различных задачах компьютерного зрения, обнаружение болезней растений и вредителей обычно рассматривается как специфическое применение в области сельского хозяйства. Имеется слишком мало образцов болезней сельскохозяйственных растений и вредителей. По сравнению с открытыми стандартными библиотеками наборы данных, собранные самостоятельно, имеют небольшой размер и трудоемки в маркировке данных. По сравнению с более чем 14 миллионами выборочных данных в наборах данных ImageNet наиболее серьезной проблемой, стоящей перед обнаружением болезней и вредителей растений, является проблема небольших выборок. На практике некоторые болезни растений имеют низкую заболеваемость и высокую стоимость получения изображений болезней, в результате чего собирается всего несколько или дюжина обучающих данных, что ограничивает применение методов глубокого обучения в области идентификации болезней растений и вредителей. В самом деле, для задачи малых выборок

Увеличение данных, синтез и генерация данных в глубоком обучении. Увеличение данных является ключевым компонентом обучения моделей глубокого обучения. Оптимизированная стратегия увеличения данных может эффективно улучшить эффект обнаружения болезней растений и вредителей. Наиболее распространенный метод расширения изображения болезней растений и вредителей заключается в получении большего количества образцов с использованием операций обработки изображений, таких как зеркальное отображение, вращение, смещение, деформация, фильтрация, регулировка контраста и т. д. для исходных образцов болезней растений и вредителей. Кроме того, генеративно-состязательные сети (GAN) [59] и вариационный автоматический кодировщик (VAE) [60] могут генерировать более разнообразные выборки для обогащения ограниченных наборов данных.

Трансферное обучение и тонкая настройка классической сетевой модели. Трансферное обучение (TL) переносит знания, полученные из общих больших наборов данных, в специализированные области с относительно небольшими объемами данных. Когда трансферное обучение разрабатывает модель для вновь собранных немаркированных образцов, оно может начаться с модели обучения с использованием аналогичного известного набора данных. После точной настройки параметров или изменения компонентов его можно применять для обнаружения локализованных болезней растений и обнаружения вредителей, что может снизить стоимость обучения модели и позволить сверточной нейронной сети адаптироваться к небольшим выборкам данных. Оппенгейм Д. и др. [61] собрали изображения зараженного картофеля разных размеров, оттенков и форм при естественном освещении и классифицировали путем точной настройки сети VGG. Результаты показали, что трансферное обучение и обучение новых сетей были эффективными. Too E. и др. [62] оценивал различные классические сети путем тонкой настройки и контраста. Экспериментальные результаты показали, что точность Dense-Nets улучшается с увеличением количества итераций. Чен Дж. и др. [63] использовали трансферное обучение и тонкую настройку для идентификации изображений болезней риса в сложных фоновых условиях и достигли средней точности 92,00%, что доказывает, что эффективность трансферного обучения лучше, чем обучение с нуля.

Разработав разумную сетевую структуру, можно значительно сократить требования к выборке. Чжан С. и др. [64] построили модель трехканальной сверточной нейронной сети для распознавания болезней листьев растений путем объединения трех цветовых компонентов. Каждый компонент канала TCCNN состоит из трех цветных RGB-изображений болезней листьев. Лю Б. и др. [65] представил усовершенствованный метод CNN для выявления болезней листьев винограда. В модели использовалась свертка с разделением по глубине вместо стандартной свертки, чтобы уменьшить переоснащение и уменьшить количество параметров. Для различных размеров поражений виноградных листьев к модели была применена исходная структура, чтобы улучшить возможность извлечения многомасштабных признаков. По сравнению со стандартными структурами ResNet и GoogLeNet эта модель имеет более высокую скорость сходимости и более высокую точность во время обучения. Точность распознавания этого алгоритма составила 97,22%.

Точное раннее выявление болезней растений необходимо для максимизации урожая [66]. При фактической ранней идентификации болезней и вредителей растений, из-за небольшого размера самого объекта поражения,

множественные процессы выборки в глубокой сети, извлечения признаков, как правило, приводят к игнорированию мелких объектов. Более того, из-за проблемы фонового шума на собранных изображениях крупномасштабный сложный фон может привести к большему количеству ложных срабатываний, особенно на изображениях с низким разрешением. Ввиду нехватки существующих алгоритмов анализируется направление улучшения алгоритма обнаружения мелких объектов, и предлагается несколько стратегий, таких как механизм внимания, для повышения производительности обнаружения мелких целей.

Использование механизма внимания позволяет более рационально распределять ресурсы. Суть механизма внимания заключается в том, чтобы быстро находить интересующую область и игнорировать второстепенную информацию. Изучая характеристики изображений болезней и вредителей растений, можно разделить признаки с помощью метода взвешенной суммы со взвешенным коэффициентом, а фоновый шум на изображении можно подавить. В частности, модуль механизма внимания может получить заметное изображение и отделить объект от фона, а функцию Softmax можно использовать для управления изображением элемента и объединения его с исходным изображением элемента для получения новых функций слияния для целей снижения шума. В будущих исследованиях по раннему распознаванию болезней и вредителей растений механизмы внимания могут использоваться для эффективного отбора информации и выделения дополнительных ресурсов интересующей области для достижения более точного обнаружения. Картик Р. и др. [67] применили механизм внимания к остаточной сети, и были проведены эксперименты с использованием набора данных PlantVillage, которые достигли общей точности 98%.

Раннее распознавание болезней и вредителей растений. При применении идентификации болезней растений и вредителей симптомы проявления не очевидны, поэтому ранняя диагностика очень сложна, будь то визуальное наблюдение или компьютерная интерпретация. Однако большее научное значение и востребованность имеет ранняя диагностика, которая в большей степени способствует предупреждению и борьбе с болезнями и вредителями растений, предотвращению их распространения и развития. Наилучшее качество изображения можно получить при достаточном солнечном свете, а съемка в пасмурную погоду усложнит предобработку изображения и снизит эффект распознавания. Кроме того, на ранней стадии появления болезней и вредителей растений трудно анализировать даже изображения высокого разрешения. Необходимо объединить метеорологические

данные и данные о защите растений, такие как температура и влажность, чтобы реализовать распознавание и прогнозирование болезней и вредителей. Судя по существующей исследовательской литературе, имеется несколько сообщений о ранней диагностике болезней растений и вредителей.

Междисциплинарные исследования. Только путем более тесной интеграции эмпирических данных с такими теориями, как агрономическая защита растений, мы можем создать модель полевой диагностики, которая больше соответствует правилам выращивания сельскохозяйственных культур и еще больше повысит эффективность и точность выявления болезней растений и вредителей. В дальнейшем необходимо перейти от анализа изображений на поверхностном уровне к выявлению механизма возникновения болезней и вредителей, а от простой экспериментальной среды перейти к практическим прикладным исследованиям, комплексно учитывающим закономерности роста сельскохозяйственных культур, факторы внешней среды и т.д.

Таким образом, с развитием технологии искусственного интеллекта фокус исследований в области обнаружения болезней растений и вредителей на основе машинного зрения сместился с классических методов обработки изображений и машинного обучения на методы глубокого обучения, которые решают сложные проблемы, которые не могут быть решены с помощью компьютерного зрения. Автором исследованы задачи прогнозирования урожайности и болезни растений различных сельскохозяйственных культур с применением алгоритмов машинного обучения и глубокого обучения, которые подробно описаны в работах [71]-[89]. Отдельные сельскохозяйственные задачи посвящены прогнозированию задач с применением глубокого обучения с технологиями компьютерного зрения и опубликованы в работах [90]- [104].

ГЛАВА 2
МЕТОДЫ И МЕТОДОЛОГИИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ЗАДАЧ ЦИФРОВОГО СЕЛЬСКОГО ХОЗЯЙСТВА

Глобальная задача машинного обучения — создать искусственный интеллект, который по своим аналитическим способностям будет равен или даже превосходить человеческий разум. Это очень сложная задача, которую тем не менее наука вполне может решить в ближайшие годы. Задачи машинного обучения можно разделить на четыре большие группы: классическое обучение, ансамблевые методы, обучение с подкреплением, нейросети и глубокое обучение.

Во второй главе диссертации исследуется классическое обучение с учителем. Рассматриваются наиболее распространенные в научных исследованиях методы классификации, регрессии, кластеризации и уменьшение размерности. Классификация используется для решения тех задач, где на основании признаков объектов требуется распределить их по заданным категориям. Например, в сельском хозяйстве распознать здоровый лист растения от нездорового. На производстве могут отделять детали с браком от хороших с помощью компьютерного зрения. Регрессия в теории вероятностей и математической статистике – это зависимость среднего значения какой-либо величины от некоторой другой величины или от нескольких величин. В задачах регрессии с помощью алгоритмов машинного обучения можно анализировать огромные массивы данных и делать прогнозы на их основе. Например, можно загрузить в компьютер данные об урожайности за последние 10 -20 лет и прогнозировать урожайность культур в текущем или последующие годы. Третья группа задач – кластеризация. Кластеризация – это распределение объектов по категориям, когда неизвестно, сколько категорий получится в итоге. Распределение происходит по заданному критерию. Например, кластеризацию можно использовать в задачах распознавания типов болезней растений по различным сельскохозяйственным культурам и на их основе прогнозировать какой группой болезнью болеют определенный класс растений. Например, компания может использовать кластеризацию для определения типов клиентов по паттернам их покупок и делать на основании этого персонализированные предложения товаров. Следующая сложная задача -уменьшение размерности или проклятие размерности. Уменьшение размерности помогает сократить количество признаков в данных без потери информации. Это упрощает их обработку и ускоряет алгоритмы машинного обучения, так как количество

данных, с которыми им предстоит работать, уменьшается. При распознавании изображений снижение размерности позволяет не анализировать каждый пиксель, а использовать только важные признаки. Например, чтобы распознать лист растения достаточно обнаружит желтые или бурые пятна или распознать больной лист растения среди здоровых. На рис.2.1. показаны структура машинного обучения в целом.

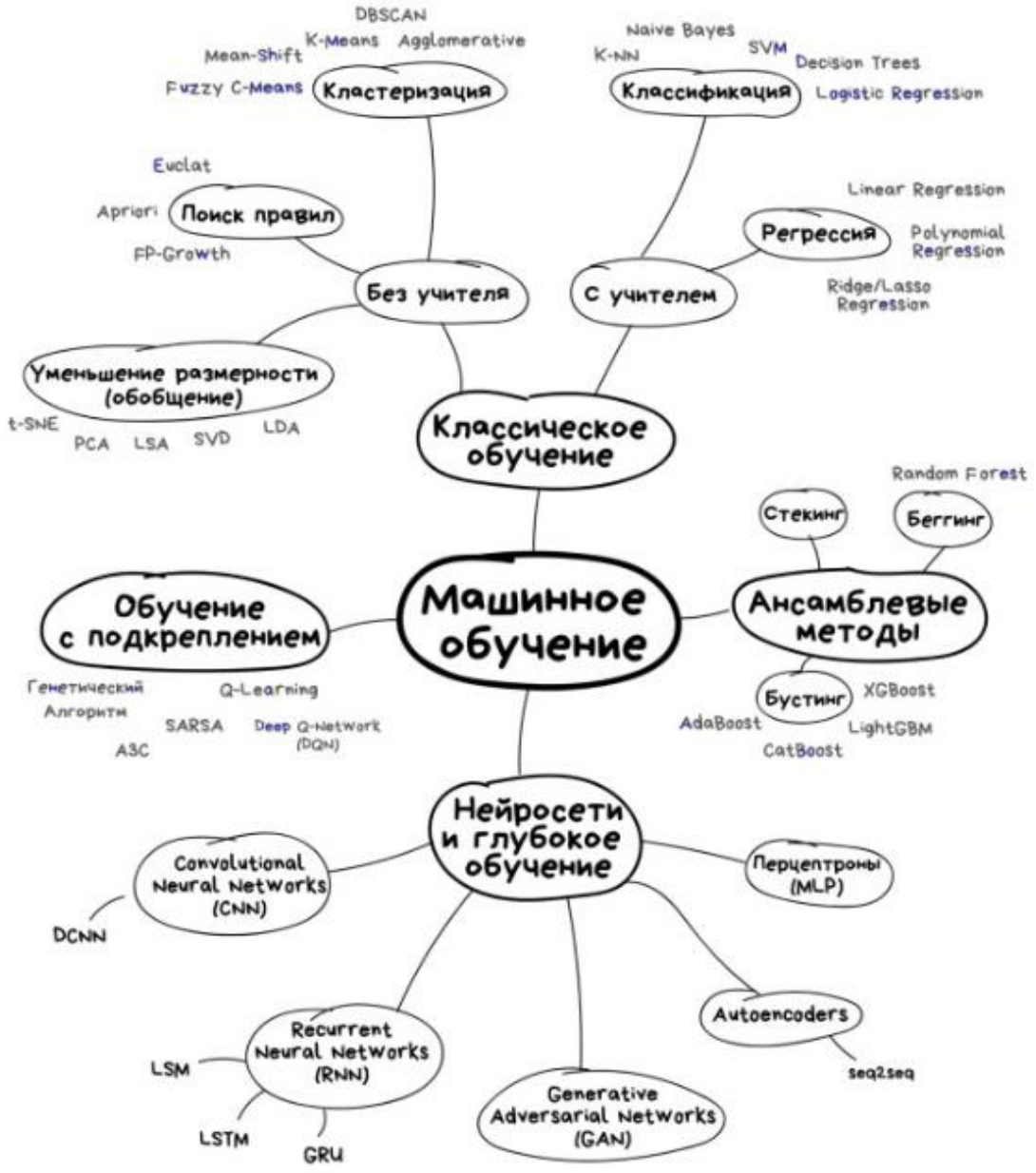


Рис.2.1. Структура машинного обучения

В данной главе диссертации подробно рассмотрены основные алгоритмы машинного обучения и их применение к задачам сельского хозяйства. Рассмотрены математическое описание и процесс реализации алгоритмов машинного обучения для построения различных моделей сельского хозяйства.

Задачам построения нелинейных моделей и прогнозирования сложных задач сельского хозяйства методами машинного обучения посвящена третья глава диссертации. Важным разделом машинного обучения является построения линейных и нелинейных моделей. Например, для построения линейной модели используются одномерная и множественная регрессия, различные варианты логистической регрессии, а также полиномиальная регрессия.

В данной главе рассмотрено описание алгоритмов этих методов, а в третьей главе диссертации подробно рассмотрен каждый из алгоритмов с реализацией для прикладных задач. Регрессионные задачи машинного обучения относятся к методам обучения с учителем. Они используются для решения задач регрессии. Регрессия - процесс поиска модели, которая предсказывает непрерывное значение на основе входных переменных. Например, он прогнозирует непрерывные значения, такие как температура, цена, продажи, зарплата и возраст. Границы использования данного алгоритма очень широкие. Линейная регрессия в основном используется для поиска линейной связи между целью и одним или несколькими предикторами. Другими словами, он предсказывает целевую переменную, подбирая линейную связь между зависимой (целевая переменная) и независимыми переменными (предикторами). Кроме того, он используется для прогнозирования и выяснения причинно-следственных связей между переменными.

2.1. Общая постановка задачи машинного обучения

Введем понятия, обучающей выборки и обучающего пространства. Пусть X^l= $\{(x_i, y_i)$, i=1,..,l$\}$ обучающая выборка, F=$|| f_j(x_i) ||_{lxn}$ признаковое пространство. Тогда предположим, пусть у нас имеется следующая модель, рис.2.2.

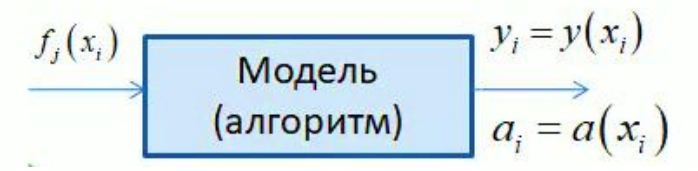


Рис. 2.2. Структура модели

Данная модель на обучающей выборке $f_j(x_i)$ для объекта x_i по правилу на выходе должна формировать целевое значение y_i. Вместо этого она выдает свой алгоритм a_i= $a(x_i)$, обученный на том же признаковом пространстве $f_j(x_i)$. Наша

задача построить алгоритм a_i максимально приближенный к y_i. Модель a(x) при этом определяется по формуле

$$\text{a (x) =g(x,}\theta\text{),} \qquad (2.1.)$$

где параметры $\theta = [\theta_1, \theta_2, \dots, \theta_n]^T$ должны быть выбраны из специальных условий. Схематически это выглядит таким образом.

$f_j(x_i)$→ Модель-Алгоритм → Должна выдать y_i =y (x_i), а выдает $a_i = a(x_i)$.

Ярким примером линейной модели служит линейная регрессия, когда входные x_i данные и выходные данные y_iмодели подбирается по условию

$$y_i = kx_i + b + \varepsilon_i, \qquad (2.2)$$

то есть модель выбираем как функцию

$$g(x, k, b) = kx + b,$$

в (2.2) ε_i означает Гауссово распределение, которое, в общем случае определяется по формуле

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad (2.3)$$

Поставим задачу, как выбрать критерий качества модели. Он должен быть выбран таким образом, чтобы минимизировать следующий функционал качества

$$\text{Q } (a, X^l) = \frac{1}{l}\sum_{i=1}^{l} L(a, x_i)\ , \qquad (2.4)$$

на множестве функций $a(x)= g(x, \theta)$, X xY. Примерами функционала L(a,x) могут быть следующие функции потерь:

L (a, x)=$|a(x) - y(x)|$ - абсолютная ошибка;
$L(a, x) = (a(x) - y(x))^2$ - квадратичная ошибка.

Приведенные выше рассуждения можно обобщить следующим образом.

1. Общая задача машинного обучения ставится как поиск модели $a(x)$, которая наилучшим образом описывает природу зависимости входных данных $\{x_i\}$ и целевых входных значений $\{y_i\}$.

2. Задачу поиска наилучшей модели часто сводят к задаче параметрической оптимизации функции вида $a(x) = g(x, \theta)$.

3. Для нахождения подходящих параметров θ вводится функция потерь $L(a, x)$ и определяется средний эмпирический риск $Q(a, X^l)$. Минимизируя показатель качества (2.4), получаем набор параметров θ по обучающей выборке

4. На основе найденной зависимости $a(x) = g(x, a)$ в дальнейшем вычисляются выходные значения $a_k = a(x'_k)$ при предъявлении нового входного вектора x'_k той же природы, что и при обучении.

Вот эти четыре этапа представляют собой общие принципы обучения всех алгоритмов машинного обучения. Теперь давайте введем понятие переобучение модели. Для заданной обучающей выборки $X^l = \{(x_i, y_i),\ i=1,..,l\}$ построим функцию потерь $L(a, x)$ и рассмотрим задачу оптимизации эмпирического риска $Q(a, X^l)$ определяемого функцией

$$Q(a, X^l) = \frac{1}{l} \sum_{i=1}^{l} L(a, x_l)\ . \tag{2.5}$$

Тогда задача оптимизации $Q(a, X^l) \to \min$ по $a(x)$ сводится к задаче нахождения такой функции $a(x)$, которая минимизирует эмпирический риск. Математически это можно записать в виде

$$\mu(X^l) = argmin\ Q(a, X^l), \text{ по } a \in A, \tag{2.6}$$

т.е. мы находим такой модель $a \in A$ из всех возможных моделей A, при которых функционал качества $Q(a, X^l)$ принимает минимальное значение. Модель $\mu(X^l)$ в данном случае называется оптимальным моделью. Это и есть общая постановка задачи обучения алгоритмов машинного обучения. Если же наша модель параметрическая и $a(x) = g(x, \theta)$, зависит от неизвестных параметров θ, тогда наша задача записывается следующим образом

$$\mu(X^l) = argmin\ Q(a, X^l), \text{ по } \theta. \tag{2.7}$$

Тогда после нахождения всех параметров θ, модель $a = \mu(X^l)$ используется для практических целей. В следующем параграфе мы рассмотрим общие линейные модели

$$a(x) = \sum_{i=1}^{n} \theta_i\, f_i(x),\ \theta = [\theta_1, \dots, \theta_n]^T\ . \tag{2.8}$$

Модель (2.8) является линейной, потому что при переходе, например в трехмерное пространство, модель $a(x)$ определяет гиперплоскость следующего вида, причем ориентация этой гиперплоскости определяется параметром θ, рис. 2.3.

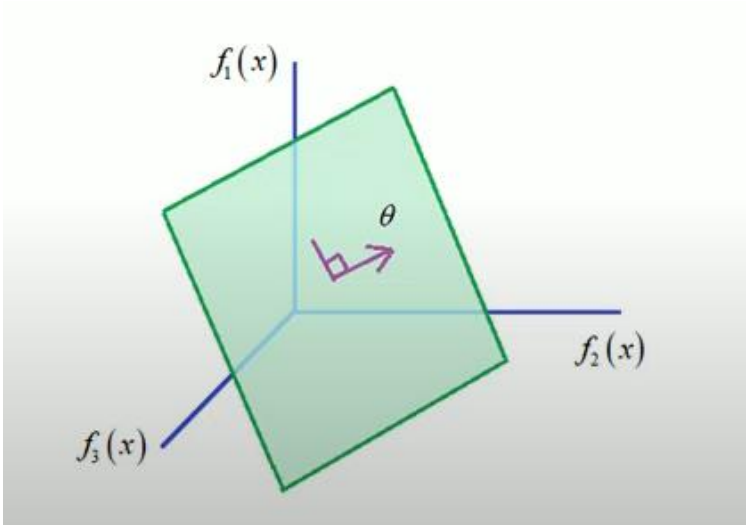


Рис. 2.3. Описание линейной модели гиперплоскостью в трехмерном пространстве

2.2. Алгоритмы машинного обучения. Обучение моделей регрессии

Рассмотрим алгоритм обучения линейных моделей. Независимые переменные (предикторы), входящие в линейные модели, должны быть линейно связаны с зависимыми, целевыми переменными. Между предикторами и целевой переменной должна быть линейная связь. Линейная зависимость – это такая зависимость, при которой увеличение или уменьшение одной переменной вызывает соответствующее увеличение или уменьшение и другой переменной. Независимость переменных можно проверить с помощью нескольких методов визуализации, таких как диаграмма рассеяния, парная диаграмма и тепловая карта.

Данные для обучения модели должны иметь нормальное распределение: нормальное распределение – это распределение вероятностей, которое симметрично относительно среднего, показывая, что данные, близкие к среднему, встречаются чаще, чем данные, далекие от среднего. В форме графика нормальное распределение будет отображаться в виде кривой колокола.

Логистическая регрессия. Это статистический метод, который используется для моделирования переменной бинарного отклика на основе переменных-предикторов. Хотя изначально этот метод был разработан для задач с двумя классами или бинарными ответами, этот метод можно обобщить и для

задач с несколькими классами. Однако в нашем примере данные представляют собой бинарный ответ или проблему двух классов, поэтому мы не будем рассматривать случай с несколькими классами. Задачи многоклассовой классификации будет рассматриваться отдельно. Логистическая регрессия очень похожа на линейную регрессию как концепцию, и ее можно рассматривать как проблему «оценки максимального правдоподобия», когда мы пытаемся найти статистические параметры, которые максимизируют вероятность выборки наблюдаемых данных из интересующего статистического распределения. Это также очень связано с общим подходом функции затрат / потерь, который мы видим в алгоритмах машинного обучения с учителем. В случае бинарных переменных отклика простая модель линейной регрессии, такая как $y_i = \beta_0 + \beta_1 x_i$, был бы плохим выбором, потому что он может легко генерировать значения за пределами 0 граница. Что нам нужно, так это модель, которая ограничивает нижнюю границу прогноза нулем, а верхнюю границу — 1. Первое, что нужно сделать для выполнения этого требования, — это по-иному сформулировать задачу. Если y_i может быть только 0 или 1, мы можем сформулировать y_i как реализация случайной величины, которая может принимать значения единица и ноль с вероятностями p_i и 1- p_i, соответственно. Эта случайная величина следует распределению Бернулли, и вместо предсказания двоичной переменной мы можем сформулировать проблему как $p_i \sim \beta_0 + \beta_1 x_i$. Однако наша первоначальная проблема остается в силе: простая линейная регрессия по-прежнему будет приводить к значениям, превышающим границы 0 и 1. Модель, удовлетворяющая граничному требованию, представляет собой логистическое уравнение, показанное ниже:

$$p_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}. \quad (2.9)$$

Это уравнение можно линеаризовать следующим преобразованием

$$\text{Logit}(p_i) = \ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_i. \quad (2.10)$$

Левая часть называется логитом, что означает «логистическая единица». Он также известен как логарифм шансов. В этом случае наша модель будет давать значения в логарифмическом масштабе, и с помощью приведенного выше логистического уравнения мы можем преобразовать значения в диапазон [0,1]. Теперь остается вопрос: «Каковы наилучшие оценки параметров для нашего обучающего набора». В рамках концепции максимального правдоподобия, наилучшие оценки параметров — это те, которые

максимизируют вероятность того, что статистическая модель действительно даст наблюдаемые данные. Вы можете думать об этой подгонке как о распределении вероятностей для наблюдаемого набора данных. Параметры распределения вероятностей должны максимизировать вероятность того, что наблюдаемые данные получены из рассматриваемого распределения. Если бы мы использовали распределение Гаусса, мы бы изменили параметры среднего значения и дисперсии до тех пор, пока наблюдаемые данные не станут более правдоподобными для извлечения из этого конкретного распределения Гаусса.

В логистической регрессии переменная отклика моделируется биномиальным распределением или его особым случаем распределения Бернулли. Значение каждой переменной ответа, y_i , равен 0 или 1, и нам нужно выяснить параметр p_i значения, которые могли бы генерировать такое распределение нулей и единиц. Если мы сможем найти лучшее значения для каждого образца данных , мы бы максимизировали функцию логарифмического правдоподобия модели по наблюдаемым данным. Функция максимального логарифмического правдоподобия для нашего случая бинарной переменной отклика показана в виде уравнения

$$\text{Ln(L)} = \sum_{i=1}^{N}[\ln(1 - p_i) + y_i \ln(\frac{p_i}{1-p_i})]. \quad (2.11)$$

Чтобы максимизировать это уравнение, мы должны найти оптимум значения p_i, которые зависят от параметров β_0 и β_1, а также зависящие от значений переменных-предикторов x_i. Мы можем изменить уравнение, заменив p_i с логистическим уравнением. Кроме того, многие функции оптимизации скорее минимизируют, чем максимизируют. Поэтому мы будем использовать отрицательную логарифмическую вероятность, которую также называют функцией «логарифмических потерь» или логистических потерь. Приведенная ниже функция представляет собой функцию потери. Мы заменили p_i с логистическим уравнением и упростили выражение

$$L_{log}(X, y, \beta_0, \beta_1) = \text{-ln(L)} = -\sum_{i=1}^{N}[-\ln\left(1 + e^{(\beta_0+\beta_1 x_i)}\right) + y_i(\beta_0+\beta_1 x_i)]. \quad (2.12)$$

В случае, когда имеются переобученность, задачу (2.12) можно переобучить следующими тремя задачами

$$L_{log}+\lambda \sum_{j=1}^{N} \beta_j^2 \Rightarrow \min,$$
$$L_{log}+\lambda \sum_{j=1}^{N} |\beta|_j \Rightarrow \min,$$

$$L_{log}+\lambda \sum_{j=1}^{N} \alpha\beta_j^2 + (1-\alpha)\,|\beta|_j \Rightarrow \min, \qquad (2.13)$$

где L_{log} определяется по формуле (2.12)

Теперь давайте посмотрим, как это работает на практике. Ниже представлен график бинарной задачи о влиянии пестицидов на рост растений до и после подкормки, рис. 2.4.

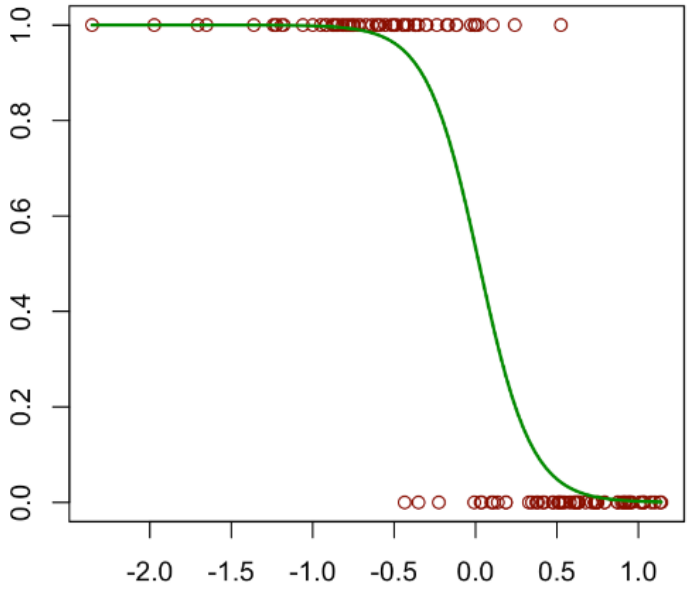


Рис.2.4. Бинарная задаче о росте растений с применением пестицидов с логистической регрессией.

Для определения урожайности рассматривается бинарная и многоклассовая задачи с применением логистической регрессии. Для переобученных моделей применяется метод регуляризации. Регуляризация применяется по умолчанию, что характерно для машинного обучения. Еще одним преимуществом регуляризации является то, что она улучшает численную стабильность т.е. сходимость градиентного спуска. Никакая регуляризация не равносильна установке параметра C на очень высокое значение.

Логистическая регрессия, несмотря на свое название, представляет собой скорее линейную модель классификации, чем регрессию. Логистическая регрессия также известна в литературе как логит-регрессия, классификация с максимальной энтропией или логарифмический линейный классификатор. В модели логистической модели вероятности, описывающие возможные исходы одного испытания, моделируются с помощью логистической функции.

Приведем задачи минимизации функционала ошибок в регуляризованной логистической регрессии в задачах Риджа, Лассо и эластической сети. Сначала в

качестве задачи оптимизации изучим бинарный класс L_2 логистической регрессии, которая минимизирует следующую функцию стоимости:

$$G(X,y) = \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \Rightarrow min, \text{ по } \omega, c \qquad (2.14)$$

Аналогичным образом, L_1 регуляризованная логистическая регрессия решает следующую задачу оптимизации:

$$Q(X,y) = ||\omega||_1 + C\sum_{i=1}^{N} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \Rightarrow min, \text{ по } \omega \text{ и } c \qquad (2.15)$$

Регуляризация Эластик Net представляет собой комбинацию L_1, а также L_2 , и минимизирует следующую функцию стоимости:

$$M(X,y) = \frac{1-\rho}{2}\omega^T\omega + \rho\left|\left|\omega\right|\right|_1 +$$
$$+ C\sum_{i=1}^{N} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \Rightarrow min, \text{ по } \omega, c \qquad (2.16)$$

где ρ контролирует силу L_1 регуляризация по сравнению с L_2 регуляризация. Обратите внимание, что в этих обозначениях предполагается, что цель y_i принимает значения в множестве $[-1,1]$. Легко заметить, что Elastic-Net эквивалентен L_1, когда ρ=1 и эквивалентно L_2, когда ρ=0.

Обобщенная линейная регрессия. Обобщенные линейные модели (GLM) расширяют линейные модели двумя способами. Во-первых, прогнозируемые значения $\hat{y}$ связаны с линейной комбинацией входных переменных X через функцию обратной ссылки h по формуле:

$$\hat{y}(\omega, X) = h(X\omega), \qquad (2.17)$$

Во-вторых, квадрат функции потерь заменяется единичным отклонением d распределения в экспоненциальном семействе (или, точнее, модели репродуктивной экспоненциальной дисперсии (EDM)).

Веса ω при этом определяются из задачи минимизации, которая имеет вид:

$$L(y,\hat{y}) = \frac{\alpha}{2}||\omega||_2^2 + \frac{1}{2N}\sum_{i=1}^{N} d(y_i, \hat{y}_i) \Rightarrow min, \text{ по } \omega \qquad (2.18)$$

где α является штрафом за регуляризацию L_2. При указании весов выборки среднее значение становится средневзвешенным.

Дерево решений. С помощью алгоритма дерево решений можно прогнозировать сложные нелинейные процессы, например в задачах сельского хозяйства. В связи с этим он применяется к прикладным задачам, где требуются нестандартные решения изучаемой проблемы. Сначала изучим дерево решений для одномерного случая. Рассмотрим простейший пример дерево решений для подбора синусоидальной кривой с добавлением зашумленного наблюдения. В результате он изучает локальные линейные регрессии, аппроксимирующие синусоиду. Легко заметить, что, если максимальная глубина дерева (контролируемая max_depth параметром) установлена с большим числом, тогда деревья решений прогнозирует более мелкие детали обучающих данных и обучаются на шуме, т.е. они переобучаются, рис. 2.5.

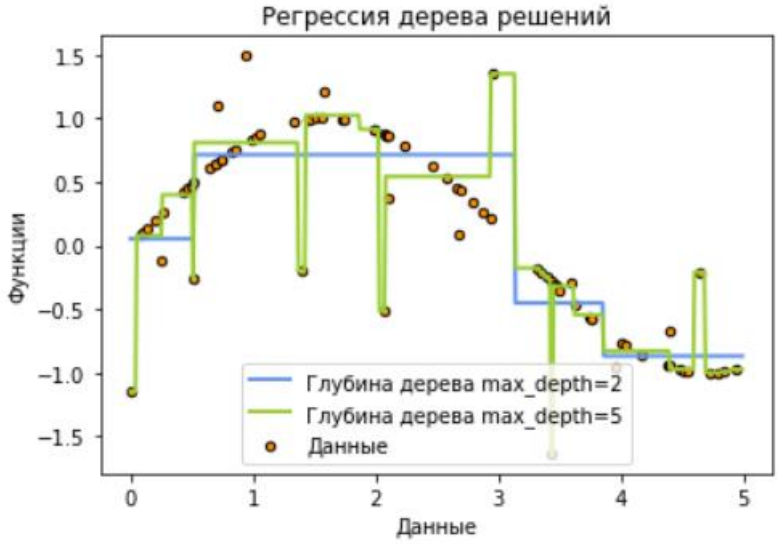


Рис.2.5. Алгоритм дерево решений с различными глубинами.

Деревья решений также можно применять к задачам регрессии, используя параметры данного класса. Как и в настройке классификации, метод fit будет принимать в качестве аргументов массивы X и y, только в этом случае ожидается, что y будет иметь значения с плавающей запятой вместо целых значений.

2.3. Алгоритмы бэггинга и случайные лес

Метод решающих деревьев могут восстанавливать очень сложные закономерности, но при этом неустойчивы к малым изменениям в данных. Из-за этого данный метод не очень подходить для решения задач моделирования, однако при объединении в композицию с другими алгоритмами они показывают очень хорошие результаты. Одним из подходов к построению композиций

является бэггинг, который независимо строит несколько моделей и усредняет их ответы. На данном разделе диссертации мы изучим математический инструмент, который поможет нам в анализе бэггинга декомпозицию ошибки на компоненты смещения и разброса (bias-variance decomposition) а затем перейдем к самим методам. Также существует другой подход к построению композиций, называемый бустингом, который строит модели последовательно, и каждая следующая модель исправляет ошибки предыдущей. Данный мощный метод рассмотрим в следующем разделе.

Рассмотрим простой пример построения композиции алгоритмов. Пусть дана конечная выборка X = (x_i,y_i) с вещественными ответами. Будем решать задачу линейной регрессии. Сгенерируем подвыборку с помощью бутстрепа. Равномерно возьмем из выборки ℓ объектов с возвращением. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X_1. Повторив процедуру N раз, сгенерируем N подвыборок X_1, . . . , X_N. Обучим по каждой из них линейную модель регрессии, получив базовые алгоритмы b_1(x), . . . , b_N(x). Предположим, что существует истинная функция ответа для всех объектов y(x), а также задано распределение на объектах p(x). В этом случае мы можем записать ошибку каждой функции регрессии

$$\varepsilon_j(x) = b_j(x) - y(x),\ j=1,..,N$$

и записать математическое ожидание среднеквадратичной ошибки

$$E_x(b_j(x) - y(x))^2 = E_x(\varepsilon_j(x))^2.$$

Средняя ошибка построенных функций регрессии имеет вид

$$E_1 = \frac{1}{N}\sum_{j=1}^{N} E_x\, \varepsilon_j{}^2(x).$$

Предположим, что ошибки не смещены и не коррелированы:

$$E_x\varepsilon_j(x) = 0;\ E_x\varepsilon_i(x)\varepsilon_j(x) = 0,\ i \neq j. \qquad (2.19)$$

Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций:

$$A(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x), \qquad (2.20)$$

Найдем ее среднеквадратичную ошибку:

$$E_N = E_x\left(\frac{1}{N}\sum_{j=1}^{N} b_j(x) - y(x)\right)^2 = E_x\left(\frac{1}{N}\sum_{j=1}^{N} \varepsilon_j(x)\right)^2 =$$
$$= \frac{1}{N^2} E_x\left(\sum_{j=1}^{n} \varepsilon_j^2 + \sum_{i \neq j} \varepsilon_i(x)\, \varepsilon_j(x)\right)$$

Учитывая $\sum_{i \neq j} \varepsilon_i(x)\, \varepsilon_j(x)) = 0$, получаем

$$E_N = \frac{1}{N} E_1. \qquad (2.21)$$

Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки в N раз. Следует отметить, что рассмотренный нами пример, не очень применим на практике, поскольку мы сделали предположение о некоррелированности ошибок (2.19), что редко выполняется. Если это предположение неверно, то уменьшение ошибки оказывается не таким значительным. Позже мы рассмотрим более сложные методы объединения алгоритмов в композицию, которые позволяют добиться высокого качества в реальных задачах.

Разложение ошибки дисперсии. Bias-Variance. Допустим, у нас есть некоторая выборка, на которой линейные методы работают лучше решающих деревьев с точки зрения ошибки на контроле. Чем можно объяснить превосходство определенного метода обучения? Оказывается, ошибка любой модели складывается из трех факторов: сложности самой выборки, сходства модели с истинной зависимостью ответов от объектов в выборке, и богатства 3 семейства, из которого выбирается конкретная модель. Между этими факторами существует некоторый баланс, и уменьшение одного из них приводит к увеличению другого. Такое разложение ошибки носит название разложения на смещение и разброс, и его формальным выводом мы сейчас займемся. Рассмотрим задачу регрессии, пусть задана выборка X = (x_i, y_i) i=1,…,n с вещественными значениями $y_i \in$ R. В пространстве X × Y существует распределение p(x, y), из которого можно сгенерировать выборку X и значения на ней. Рассмотрим квадратичную функцию потерь

$$L(y, a) = (y - a)^2, \qquad (2.22)$$

и соответствующий ей среднеквадратичный риск

$$R(a)= E_{x,y}[(y - a(x))^2]=\int\int p(x,y)\,(y - a(x))^2 dxdy$$

Данный функционал усредняет ошибку модели в каждой точке пространства x и для каждого возможного ответа y, причём вклад пары (x, y), по сути, пропорционален вероятности, получит её в выборке p(x, y). Разумеется, на практике мы не можем вычислить данный функционал, поскольку распределение p(x, y) неизвестно. Тем не менее, в теории он позволяет измерить качество модели на всех возможных объектах, а не только на обучающей выборке.

Минимум среднеквадратичного риска. Покажем, что минимум среднеквадратичного риска достигается на функции, возвращающей условное математическое ожидание ответа при фиксированном объекте:

$$a_*(x) = E[y|x] = \int yp(y|x)\,dy= \operatorname*{argmin}_{\alpha} R(\alpha). \qquad (2.23)$$

Преобразуем функцию потерь (2.22):

$$L(y, a(x))=(y - a)^2= (y - E(y|x) + E(y|x) - a(x))^2=$$
$$=(y - E(y|x))^2+2(\, y - E(y|x))\,(\, E(y|x) - a(x)) + (E(y|x) - a(x))^2.$$

Разберемся сначала с последним слагаемым. Перейдём от математического ожидания $E_{x,y}$[f(x, y)] к цепочке математических ожиданий

$$E_{x,y}[\text{f(x, y)}|x]=\iint f(x,y)\,p(y|x)dyp(x)dx$$

и заметим, что величина (E(y | x)−a(x)) не зависит от y, и поэтому ее можно вынести за математическое ожидание по y:

$$E_x E_y[(y - E(y|x))(E(y|x) - a(x)|x)] =E_x[\big(E(y|x) - a(x)\big)\times$$
$$\times E_y[(\text{y-E(y|x))|x}]=E_x((E(y|x) - a(x)\ (\text{E(y|x)- E(y|x)}))=0.$$

Получаем, что функционал среднеквадратичного риска имеет вид

$$R(\alpha)= E_{x,y}(y - E(y|x))^2+E_{x,y}(E(y|x) - a(x))^2. \qquad (2.24)$$

От алгоритма a(x), (2.20) зависит только второе слагаемое (2.23), и оно достигает своего минимума, если a(x) = E(y | x). Таким образом, оптимальная модель регрессии для квадратичной функции потерь имеет вид (2.25)

$$a_*(\mathrm{x}) = E(y|x) = \int yp(y|x)\mathrm{dy}. \quad (2.25)$$

Иными словами, мы должны провести «взвешенное голосование» по всем возможным ответам, причем вес ответа равен его апостериорной вероятности.

Ошибка метода обучения. Для того, чтобы построить идеальную функцию регрессии, необходимо знать распределение на объектах и ответах p(x, y), что, как правило, невозможно. На практике вместо этого выбирается некоторый метод обучения $\mu : (X \times Y)^\ell \to A$, который при произвольной обучающей выборке ставит в соответствие некоторый алгоритм из семейства A. В качестве меры качества метода обучения можно взять усредненный по всем выборкам среднеквадратичный риск алгоритма, выбранного методом μ по выборке:

$$L(\mu)=E_x\left[E_{x,y}\left[\left(y-\mu(X)(x)\right)^2\right]\right]=\iint\left(y-\right.$$
$$\left.-\mu(X)(x)\right)^2 p(x,y)\ \prod_{i=1}^{l} p(x_i,y_i)dxdydx_1dy_1\ldots dx_ldy_l,$$
$$x\in(X\times Y)^l,\ y\in X\times Y, \quad (2.26)$$

здесь математическое ожидание $E_x[\cdot]$ берется по всем возможным выборкам $\{(x_1,y_1), \ldots, (x_l,y_l)\}$ из распределения $\prod_{i=1}^{n} p(x_i,y_i)$. Выше мы показали, что среднеквадратичный риск на фиксированной выборке X можно расписать как

$$E_{x,y}\left[\left(y-\mu(X)\right)^2\right]=E_{x,y}[(y-E[y|x])^2]+E_{x,y}[(E[y|x]-\mu(X))^2].$$

Тогда $L(\mu)$ можно переписать в виде

$$L(\mu)=E_{x,y}[(y-E[y|x])^2]+E_x\left[(E_X[\mu(X)]-E[y|x])^2\right]+E_x[\,E_X[(\mu(X)-$$
$$E_X[\mu(X)])^2]\,, \quad (2.27)$$

Бэггинг. Пусть имеется некоторый метод обучения μ(X). Построим на его основе метод $\tilde{\mu}$ (X), который генерирует случайную под выборку $\tilde{X}$с помощью бутстрапа и подает ее на вход метода μ: $\tilde{\mu}$ (X) = μ($\tilde{X}$). Напомним, что бутстрап представляет собой сэмплирование ℓ объектов из выборки с возвращением, в результате чего некоторые объекты выбираются несколько раз, а некоторые ни разу. Помещение нескольких копий одного объекта в бутстрапированную выборку соответствует выставлению веса при данном объекте соответствующее ему слагаемое несколько раз войдет в функционал, и поэтому штраф за ошибку на нем будет больше. В бэггинге (bagging, bootstrap aggregation) предлагается

обучить некоторое число алгоритмов b_n(x) с помощью метода $\tilde{\mu}$, и построить итоговую композицию как среднее данных базовых алгоритмов:

$$a_N(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(\mathrm{x}) = \frac{1}{N}\sum_{n=1}^{N} \tilde{\mu}\,(\mathrm{X})(\mathrm{x})\,. \qquad (2.28)$$

Заметим, что в методе обучения для бэггинга появляется ещё один источник случайности взятие подвыборки. Чтобы функционал качества L(μ) был детерминированным, мы будем далее считать, что матожидание $E_X[\cdot]$ берётся не только по всем обучающим выборкам X, но ещё и по всем возможным подвыборкам $\tilde{X}$, получаемым с помощью бутстрапа. Это вполне логичное обобщение, поскольку данное матожидание вводится в функционал именно для учёта случайностей, связанных с процедурой обучения модели.

Найдём смещение из разложения (2.27) для бэггинга:

$$E_{x,y}\left[E_X\left[\frac{1}{N}\sum_{n=1}^{N}(\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x}) - \mathrm{E}[\mathrm{y}|\mathrm{x}]\right)^2\right]\right] = E_{x,y}\left[\left[\frac{1}{N}\sum_{n=1}^{N} E_x[\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x})] - \mathrm{E}[\mathrm{y}|\mathrm{x}]\right)^2\right]\right] = E_{x,y}[E_X[\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x})] - \mathrm{E}[\mathrm{y}|\mathrm{x}])^2], \qquad (2.29)$$

Мы получили, что смещение композиции, полученной с помощью бэггинга, совпадает со смещением одного базового алгоритма. Таким образом, бэггинг не ухудшает смещенность модели.

Теперь перейдём к разбросу. Запишем выражение для дисперсии композиции, обученной с помощью бэггинга:

$$E_{x,y}\left[E_X\left[\left(\frac{1}{N}\sum_{n=1}^{N}\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x}) - E_X[\frac{1}{N}\sum_{n=1}^{N}\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x})]\right)^2\right]\right], \qquad (2.30)$$

Рассмотрим выражение, стоящее под матожиданием:

$$\left(\frac{1}{N}\sum_{n=1}^{N}\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x}) - E_X[\frac{1}{N}\sum_{n=1}^{N}(\,\tilde{\mu}\,(\mathrm{X})(\mathrm{x}))\right)^2 =$$
$$= \frac{1}{N^2}\sum_{n=1}^{N}\,[\tilde{\mu}\,(\mathrm{X})(\mathrm{x}) - E_X\tilde{\mu}\,(\mathrm{X})(\mathrm{x})]]\Big)^2 = \frac{1}{N^2}\sum_{n=1}^{N}\,[\tilde{\mu}\,(\mathrm{X})(\mathrm{x}) - E_X\tilde{\mu}\,(\mathrm{X})(\mathrm{x})]]\Big)^2 +$$
$$+\frac{1}{N^2}\sum_{n_1\neq n_2}(\tilde{\mu}\,(x)(x) - E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])(\tilde{\mu}\,(\mathrm{X})(x) - E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})]), \qquad (2.31)$$

Возьмем теперь математическое ожидание от этого выражения, учитывая, что все базовые алгоритмы одинаково распределены относительно X:

$$E_{x,y}\left[E_X\left[\frac{1}{N^2}\sum_{n=1}^{N}(\tilde{\mu}\,(\mathrm{X})(\mathrm{x})-E_X[(\tilde{\mu}\,(\mathrm{X})(\mathrm{x})]\right)^2+\frac{1}{N^2}\sum_{n_1\neq n_2}(\tilde{\mu}\,(x)(x)-E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])(\tilde{\mu}\,(\mathrm{X})(x)-E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])\right]\right]=$$

$$=\frac{1}{N}E_{x,y}\left[E_X\left[(\tilde{\mu}\,(\mathrm{X})(\mathrm{x})-E_X[(\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])^2\right]+\frac{N(N-1)}{N^2}E_{x,y}[E_X[\tilde{\mu}\,(\mathrm{X})(x)-E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])(\tilde{\mu}\,(\mathrm{X})(x)-E_X[\tilde{\mu}\,(\mathrm{X})(\mathrm{x})])\right]\right], \quad (2.32)$$

Первое слагаемое это дисперсия одного базового алгоритма, деленная на длину композиции N. Второе ковариация между двумя базовыми алгоритмами. Мы видим, что если базовые алгоритмы не коррелированы, то дисперсия композиции в N раз меньше дисперсии отдельных алгоритмов. Если же корреляция имеет место, то уменьшение дисперсии может быть гораздо менее существенным.

Алгоритм случайный лес. Как мы выяснили, бэггинг позволяет объединить несмещенные, но чувствительные к обучающей выборке алгоритмы в несмещенную композицию с низкой дисперсией. Хорошим семейством базовых алгоритмов здесь являются решающие деревья, они достаточно сложны и могут достигать нулевой ошибки на любой выборке. Наряду с этим имеют низкое смещение, но в то же время легко переобучаются. Метод случайных лесов основан на бэггинге над решающими деревьями. Выше мы отметили, что бэггинг сильнее уменьшает дисперсию базовых алгоритмов, если они слабо коррелированы. В случайных лесах корреляция между деревьями понижается путем рандомизации по двум направлениям: по объектам и по признакам. Во-первых, каждое дерево обучается по бутстрапированной подвыборке. Во-вторых, в каждой вершине разбиение ищется по подмножеству признаков. Вспомним, что при построении дерева последовательно происходит разделение вершин до тех пор, пока не будет достигнуто идеальное качество на обучении. Каждая вершина разбивает выборку по одному из признаков относительно некоторого порога. В случайных лесах признак, по которому производится разбиение, выбирается не из всех возможных признаков, а лишь из их случайного подмножества размера m. Рекомендуется в задачах классификации брать m = ⌊√d⌋, а в задачах регрессии $m = [d/3]$, где d число признаков. Также рекомендуется в задачах классификации строить каждое дерево до тех пор, пока в каждом листе не окажется по одному объекту, а в задачах регрессии пока в каждом листе не окажется по пять объектов.

Случайные леса один из самых сильных методов построения композиций. На практике он может работать немного хуже градиентного бустинга, но при этом он гораздо более прост в реализации.

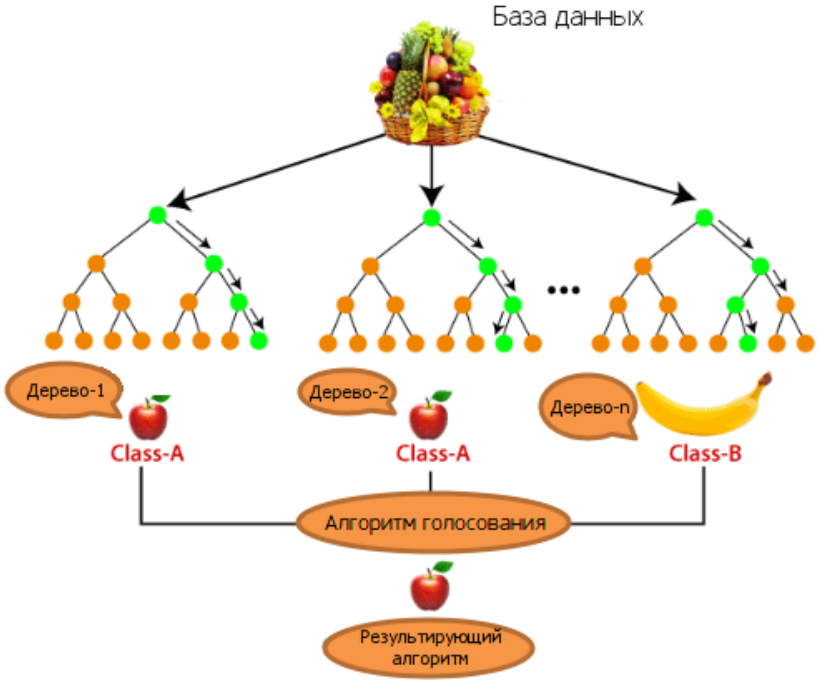


Рис.2.6. Реализация алгоритма случайный лес.

Градиентный бустинг. Мы изучили наиболее часто применяемые алгоритмы в прикладных задачах, бэггинг и случайные леса, основная идея которых лежит в построении композиции алгоритмов, которые независимо обучают каждый базовый алгоритм по некоторому подмножеству обучающих данных. Основная идея следующего более мощного алгоритма, которая строится по принципу, каждая следующая модель, в данном алгоритме исправляет ошибки предыдущих алгоритмов. Данная технология предложено Фридманом. Ниже мы рассмотрим алгоритм градиентного бустинга. Методы алгоритма градиентного бустинга работают для любых дифференцируемых функционалов потерь. Данный алгоритм является одним из наиболее мощных и универсальных на сегодняшний день алгоритмом машинного обучения, с помощью которого моделируется наибольшее число прикладных задач любой сложности и с большой точностью. С помощь этого метода в третьей главе рассмотрены задачи сельского хозяйства.

2.4. Алгоритмы градиентного бустинга в задачах регрессии

Рассмотрим задачу минимизации квадратичного функционала:

$$\text{L} = \frac{1}{2}\sum_{i=1}^{l}(a(x_i) - y_i)^2 \to \min_a L, \quad (2.33)$$

Будем искать итоговый алгоритм в виде суммы базовых моделей b_n(x):

$$a_N(\text{x}) = \sum_{n=1}^{N} b_n(\text{x}), \quad (2.34)$$

где в (2.34) базовые алгоритмы $b_n(x)$ принадлежат некоторому семейству A. Построим первый базовый алгоритм:

$$b_1(\text{x}) = \text{argmin}\ (\frac{1}{2}\sum_{i=1}^{l}(b(x_i) - y_i)^2),\ \text{b} \in A\ , \quad (2.35)$$

Решение такой задачи не представляет трудностей для многих семейств алгоритмов. Теперь мы с учетом (2.35), можем посчитать остатки на каждом объекте расстояния от ответа нашего алгоритма до истинного ответа:

$$s_i^{(1)} = y_i\text{-}b_1(x_i), \quad (2.36)$$

Если прибавить эти остатки к ответам построенного алгоритма, то он не будет допускать ошибок на обучающей выборке. Значит, будет разумно построить второй алгоритм так, чтобы его ответы были как можно ближе к остаткам:

$$b_2(\text{x}) := argmin\frac{1}{2}\sum_{i=1}^{l}(b(x_i) - s_i^{(1)})^2,\ \text{b} \in A, \quad (2.37)$$

Каждый следующий алгоритм тоже будем настраивать на остатки предыдущих алгоритмов:

$$s_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i),\ \text{i=1,…,l;} \quad (2.38)$$

$$b_N(\text{x}) := argmin\frac{1}{2}\sum_{i=1}^{l}(b(x_i) - s_i^{(N)})^2,\ \text{b} \in A, \quad (2.39)$$

Описанный метод (2.38) - (2.39) прост в реализации, хорошо работает и может быть найден во многих библиотеках, например, в scikit-learn. Заметим, что остатки могут быть найдены как антиградиент функции потерь по ответу модели, посчитанный в точке ответа уже построенной композиции:

$$s_i^{(N)} = y_i - a_{N-1}(x_i) = -\frac{\partial}{\partial z}\frac{1}{2}(z - y_i)^2 \,,\ z = a_{N-1}(x_i), \qquad (2.40)$$

Получается, что выбирается такой базовый алгоритм (2.95), который как можно сильнее уменьшить ошибку композиции это свойство вытекает из его близости к антиградиенту функционала на обучающей выборке. Попробуем разобраться с этим свойством подробнее, а также попытаемся обобщить его на другие функции потерь.

Градиентный бустинг в задаче регрессии. Пусть дана некоторая дифференцируемая функция потерь L(y, z). Будем строить взвешенную сумму базовых алгоритмов:

$$a_N(x) = \sum_{n=0}^{N} \gamma_n \, b_n(x). \qquad (2.41)$$

Заметим, что в композиции (2.41) имеется начальный алгоритм b_0(x). Как правило, коэффициент γ_0при нем берут равно единице, а сам алгоритм выбирают очень простым, например:

- нулевым b_0(x) = 0;
- возвращающим самый популярный класс (в задачах классификации): b_0(x) =$\underset{y \in Y}{\operatorname{argmax}} \sum_{i=1}^{l} [y_i = y]$
- возвращающим средний ответ (в задачах регрессии): b_0(x)= $\frac{1}{l}\sum_{i=1}^{l} y_i$

Допустим, мы построили композицию $a_{N-1}(x)$ из N −1 алгоритма, и хотим выбрать следующий базовый алгоритм $b_N(x)$ так, чтобы как можно сильнее уменьшить ошибку:

$$\min_{b_N, \gamma_N} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i)), \qquad (2.42)$$

Ответим в первую очередь на следующий вопрос: если бы в качестве алгоритма $b_N(x)$ мы могли выбрать совершенно любую функцию, то какие значения ей следовало бы принимать на объектах обучающей выборки? Иными словами, нам нужно понять, какие числа $s_1, \ldots, s_l$ из (2.40) надо выбрать для решения следующей задачи:

$$\min_{s_1, s_2, \ldots, s_l} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + s_i). \qquad (2.43)$$

Понятно, что можно требовать $s_i = y_i - a_{N-1}(x_i)$, но такой подход никак не учитывает особенностей функции потерь L(y, z) и требует лишь точного

совпадения предсказаний и истинных ответов. Более разумно потребовать, чтобы сдвиг s_i был противоположен производной функции потерь в точке $z = a_{N-1}(x_i)$,

$$s_i = -\frac{\partial L}{\partial z}, \text{ при } z = a_{N-1}(x_i).$$

В этом случае мы сдвинемся в сторону скорейшего убывания функции потерь. Заметим, что вектор сдвигов $s = (s_1,..., s_l)$ совпадает с антиградиентом:

$$-\frac{\partial L}{\partial z}|_{z=a_{N-1}(x_i),i=1,...,l} = -\nabla_z \sum_{i=1}^{l} L(y_i, z_i)|_{z_i = a_{N-1}(x_i)}$$

При таком выборе сдвигов s_i мы, по сути, сделаем один шаг градиентного спуска, двигаясь в сторону наискорейшего убывания ошибки на обучающей выборке. Отметим, что речь идет о градиентном спуске в ℓ-мерном пространстве предсказаний алгоритма на объектах обучающей выборки. Поскольку вектор сдвига будет свой на каждой итерации, правильнее обозначать его как $s_i^{(N)}$, но для простоты будем иногда опускать верхний индекс. Итак, мы поняли, какие значения новый алгоритм должен принимать на объектах обучающей выборки. По данным значениям в конечном числе точек необходимо построить функцию, заданную на всем пространстве объектов. Это классическая задача обучения с учителем, которую мы уже хорошо умеем решать. Один из самых простых функционалов среднеквадратичная ошибка. Воспользуемся им для поиска базового алгоритма, приближающего градиент функции потерь на обучающей выборке:

$$b_N(\mathrm{x}) = \arg\min_{b \in A} \sum_{i=1}^{l} (b(x_i) - s_i)^2, \qquad (2.44)$$

Отметим, что здесь мы оптимизируем квадратичную функцию потерь независимо от функционала исходной задачи вся информация о функции потерь L находится в антиградиенте s_i, а на данном шаге лишь решается задача аппроксимации функции по ℓ точкам. Разумеется, можно использовать и другие функционалы, но среднеквадратичной ошибки, как правило, оказывается достаточно. Ещё одна причина для использования среднеквадратичной ошибки состоит в том, что от алгоритма требуется как можно точнее приблизить направление наискорейшего убывания функционала (то есть направление $((s_i)_i)$; совпадение направлений вполне логично оценивать через косинус угла между ними, который напрямую связан со среднеквадратичной ошибкой. После того, как новый базовый алгоритм найден, можно подобрать коэффициент при нем по аналогии с наискорейшим градиентным спуском:

$$\gamma_N = \underset{\gamma \in R}{\operatorname{argmin}} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i)) , \qquad (2.45)$$

Описанный подход с аппроксимацией антиградиента базовыми алгоритмами и называется градиентным бустингом. Данный метод представляет собой поиск лучшей функции, восстанавливающей истинную зависимость ответов от объектов, в пространстве всех возможных функций. Ищем мы данную функцию с помощью «псевдоградиентного» спуска каждый шаг делается вдоль направления, задаваемого некоторым базовым алгоритмом. При этом сам базовый алгоритм выбирается так, чтобы как можно лучше приближать антиградиент ошибки на обучающей выборке.

Стохастический градиентный бустинг. Еще одним способом улучшения качества градиентного бустинга является внесение рандомизации в процесс обучения базовых алгоритмов. А именно, алгоритм b_Nобучается не по всей выборке X, а лишь по ее случайному подмножеству $X^k \subset$ X. В этом случае понижается уровень шума в обучении, а также повышается эффективность вычислений. Существует рекомендация необходимо брать подвыборки размер, которого вдвое меньше исходной выборки.

Функции потерь. Регрессия. При вещественном целевом векторе, как правило, используют квадратичную функцию потерь, формулы для которой уже были приведены в разделе 1. Другой вариант модуль отклонения L(y, z) = |y−z|, для которого антиградиент вычисляется по формуле

$$s_i^{(N)} = -sign(a_{N-1}(x_i) - y_i), \qquad (2.46)$$

Классификация. В задаче классификации с двумя классами разумным выбором является логистическая функция потерь, с которой уже сталкивались при изучении линейных методов:

$$L(y, z) = \log (1 + \exp (-yz)), \qquad (2.47)$$

Задача поиска базового алгоритма (2.96) с ней принимает вид

$$b_N = \arg\min_{b \in A} \sum_{i=1}^{l} (b(x_i) - \frac{y_i}{1+\exp (y_i a_{N-1}(x_i))})^2 , \qquad (2.48)$$

Логистическая функция (2.47) потерь имеет интересную особенность, связанную со взвешиванием объектов. Заметим, что ошибка на N-й итерации может быть записана как

$$Q(a_N) = \sum_{i=1}^{l} \log\left(1 + \exp\left(-y_i a_{N-1}(x_i)\right)\right) = \sum_{i=1}^{l} \log\left(1 + \right.$$
$$\left. + \exp\left(-y_i a_{N-1}(x_i)\right) \exp\left(-y_i \gamma_N b_N(x_i)\right)\right), \quad (2.49)$$

Если отступ $y_i a_{N-1}(x_i)$ на i-м объекте большой положительный, то данный объект не будет вносить практически никакого вклада в ошибку, и может быть исключен из всех вычислений на текущей итерации без потерь. Таким образом, величина

$$\omega_i^{(N)} = \exp\left(-y_i a_{N-1}(x_i)\right), \quad (2.50)$$

может служить мерой важности объекта x_i на N-й итерации градиентного бустинга.

Взвешивание объектов. Одним из первых широко распространённых методов построения композиций является AdaBoost, в котором оптимизируется экспоненциальная функция потерь

$$L(y, z) = e^{-yz}. \quad (2.51)$$

Благодаря её свойствам удаётся свести задачу поиска базового алгоритма к минимизации доли неверных ответов с весами при объектах. Эти веса возникают и в градиентном бустинге при использовании экспоненциальной функции потерь:

$$L(a, X) = \sum_{i=1}^{l} \exp\left(-y_i \sum_{n=1}^{N} \gamma_n b_n(x_i)\right). \quad (2.52)$$

Найдем компоненты ее антиградиента после (N – 1)-й итерации:

$$s_i = -\frac{\partial}{\partial z} L(y_i, z)|z = a_{N-1}(x_i) = y_i \exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i)\right) = y_i \omega_i, \quad (2.53)$$

где в (2.53) введено обозначение:

$$\omega_i = \exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i)\right). \quad (2.54)$$

Заметим, что антиградиент (2.53) представляет собой ответ на объекте, умноженный на его вес. Если все веса будут равны единице, то следующий

базовый классификатор будет просто настраиваться на исходный целевой вектор y_i, i=1,…,l штраф за выдачу ответа, противоположного правильному, будет равен 4 (поскольку при настройке базового алгоритма используется квадратичная функция потерь). Если же какой-либо объект будет иметь большой отступ, то его вес окажется близким к нулю, и штраф за выдачу любого ответа будет равен 1. Отметим, что многие функционалы ошибки классификации выражаются через отступы объектов:

$$L(a_{N-1}, X^l) = \sum_{i=1}^{l} L(a_{N-1}(x_i), y_i) = \sum_{i=1}^{l} \tilde{L}(y_i a_{N-1}(x_i)), \qquad (2.55)$$

В этом случае антиградиент $\tilde{L}(y_i a_{N-1}(x_i))$ принимает вид

$$s_i = y_i\left(-\frac{\partial \tilde{L}(y_i a_{N-1}(x_i))}{\partial a_{N-1}(x_i)}\right) = y_i \omega_i\ , \qquad (2.56)$$

то есть тоже взвешивает ответы с помощью ошибки на них.

Влияние шума на обучение. Выше (2.53) мы находили формулу для антиградиента при использовании экспоненциальной функции потерь:

$$s_i = y_i \exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n\, b_n(x_i)\right) = y_i \omega_i,$$

где $\omega_i = \exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n\, b_n(x_i)\right)$.

Заметим, что если отступ на объекте большой и отрицательный (что обычно наблюдается на шумовых объектах), то вес становится очень большим, причем он никак не ограничен сверху. В результате базовый классификатор будет настраиваться исключительно на шумовые объекты, что может привести к неустойчивости его ответов и переобучению. Рассмотрим теперь логистическую функцию потерь, которая также может использоваться в задачах классификации:

$$L(a, X^l) = \sum_{i=1}^{l} \log\left(1 + \exp\left(-y_i a(x_i)\right)\right), \qquad (2.57)$$

Найдем ее антиградиент (2.58) после (N – 1)-го шага:

$$s_i = y_i \frac{1}{1+\exp\left(y_i a_{N-1}(x_i)\right)} = y_i \omega_i^{(N)}, \qquad (2.58)$$

Теперь веса ограничены сверху единицей. Если отступ на объекте большой отрицательный (то есть это выброс), то вес при нем будет близок к единице; если же отступ на объекте близок к нулю (то есть это объект, на котором классификация неуверенная, и нужно ее усиливать), то вес при нем будет

примерно равен ½. Таким образом, вес при шумовом объекте будет всего в два раза больше, чем вес при нормальных объектах, что не должно сильно повлиять на процесс обучения.

XGBoost градиентный бустинг. Мы уже разобрались с двумя типами методов построения композиций бустингом и бэггингом, и познакомились с градиентным бустингом и случайным лесом, которые являются наиболее яркими представителями этих классов. На практике реализация градиентного бустинга оказывается очень непростой задачей, в которой успех зависит от множества тонких моментов. Мы рассмотрим конкретную реализацию градиентного бустинга пакет XGBoost. Ниже будут рассмотрены задачи регрессии с применением данной технологии. В третьей и четвертой главах диссертации данный пакет использовано для построения моделей для широкого круга задач сельского хозяйства.

Выше мы при использовании алгоритма градиентного бустинга, на каждой итерации градиентного бустинга вычисляли вектор сдвигов s, который показывает, как нужно скорректировать ответы композиции на обучающей выборке, чтобы как можно сильнее уменьшить ошибку с помощью следующего соотношения.

$$s = (-\frac{\partial L}{\partial z}|_{z=a_{N-1}(x_i),i=1,\dots,l}) = -\nabla_z \sum_{i=1}^{l} L(y_i, z_i)\,|_{z_i = a_{N-1}(x_i)}, \qquad (2.59)$$

После этого новый базовый алгоритм обучается путем минимизации среднеквадратичного отклонения от вектора сдвигов s (2.59):

$$b_N(x)=\arg\min_{b\in A} \sum_{i=1}^{l} (b(x_i) - s_i)^2, \qquad (2.60)$$

Альтернативный подход. Использование среднеквадратичной функции потерь лучше тем, что она наиболее проста для оптимизации. Обоснуем ее. Найдем новый алгоритм b(x), который решает следующую задачу.

$$\min_b \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + b(x_i)), \qquad (2.61)$$

Разложим функцию L в каждом слагаемом в ряд Тейлора до второго члена с центром композиции $a_{N-1}(x_i)$:

$$\sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + b(x_i)) \approx \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) - s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)), \qquad (2.62)$$

где в (2.62) через h_i обозначены вторые производные по сдвигам:

$$h_i = \frac{\partial^2 L}{\partial z^2}\Big|_{z=a_{N-1}(x_i)}$$

Первое слагаемое (2.62) не зависит от нового базового алгоритма, и поэтому его можно не учитывать. Получаем функционал

$$\min_b \sum_{i=1}^{l}(-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)), \qquad (2.63)$$

Покажем, что он очень похож на среднеквадратичный из формулы (2.62). Преобразуем его:

$\sum_{i=1}^{l}(b(x_i) - s_i)^2 = \sum_{i=1}^{l}(b^2(x_i) - 2b(x_i)s_i - s_i^2)$ = {Последняя слагаемое не зависит от b}$= \sum_{i=1}^{l}(b^2(x_i) - 2b(x_i)s_i) = 2\sum_{i=1}^{l}(\frac{1}{2}b^2(x_i) - b(x_i)s_i) \rightarrow \min_b 2\sum_{i=1}^{l}(\frac{1}{2}b^2(x_i) - b(x_i)s_i)$.

Видно, что последняя формула совпадает с (2.62) с точностью до константы, если положить $h_i = 1$. Таким образом, в обычном градиентном бустинге мы используем аппроксимацию второго порядка при обучении очередного базового алгоритма, и при этом отбрасываем информацию о вторых производных (то есть считаем, что функция имеет одинаковую кривизну по всем направлениям).

Выводы к главе 2. В данной главе было обосновано основные алгоритмы машинного обучения для построения различных моделей для прикладных задач. Основной нашей целью было математический обзор наиболее часто применяемых алгоритмов в задачах сельского хозяйства. В последние годы очень хорошие результаты дают ансамблевые алгоритмы на основе многомерной линейной регрессии и градиентного спуска для задач, где требуются обработка данных с требованиями установления линейной связи между данными. Для задач сельского хозяйства очень важно исследовать процесс построения линейных моделей. Ведущее место для рассмотрения нелинейных взаимосвязей между данными, для задач сельского хозяйства, где требуются учитывать в моделях очень сложные взаимосвязи между категориями и явлениями природы, а также связанные человеческими разумом интеллектуальные способности моделей предсказывать, прогнозировать будущее. В данном направлении в работе исследованы задачи сельского хозяйства с применением технологий и алгоритмов машинного обучения как дерево решений, градиентного бустинга, стохастического градиентного бустинга и ведущие алгоритмы сегодняшнего дня

варианты алгоритма XGBoost . Градиентный бустинг в XGBoost имеет ряд важных особенностей. Базовый алгоритм приближает направление, посчитанное с учетом вторых производных функции потерь. Функционал регуляризуется добавляются штрафы за количество листьев и за норму коэффициентов.

Базовый алгоритм приближает градиент, посчитанное с учетом вторых производных функции потерь. Функционал регуляризуется добавляются штрафы за количество листьев и за норму коэффициентов, например в методе случайный лес. При построении дерева используется критерий информативности, зависящий от оптимального вектора сдвига. Критерий останова при обучении дерева также зависит от оптимального сдвига. В третьей главе построены и проведены подробные анализы с помощью описанных в данной главе алгоритмы для различных задач сельского хозяйства. Методов построения моделей с применением нейронных сетей для крупных задач сельского хозяйства рассмотрены в четвертой главе настоящей диссертационной работы.

ГЛАВА 3
МОДЕЛИРОВАНИЕ И ПРОГНОЗИРОВАНИЕ ЗАДАЧ СЕЛЬСКОГО ХОЗЯЙСТВА НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

В данной главе диссертации исследуется результаты, полученные автором по применению машинного обучения к задачам сельского хозяйства. При исследовании моделирования и прогнозирования задач сельского хозяйства в данной работе использовались алгоритмы машинного обучения множественная регрессия (LR), регрессия Лассо (Lasso R), стохастический градиентный спуск (SGD), дерево решений (RT), которые дают хорошие результаты для многих сельскохозяйственных задач. При этом весьма важными и результативными при моделировании и прогнозировании данного класса задач, где требуются исследовать множества основных влияющих факторов, оказываются продвинутые алгоритмы машинного обучения K – ближайших соседей (KNN), случайный лес (RF), метод опорных векторов (SVR), варианты градиентного бустинга (GBR) и основанная на технологиях бустинга передовой алгоритм XGBoost. Естественно, здесь необходимо отметить все еще мало использованную технологию нейронных технологий для прогнозирования задач сельского хозяйства, в том числе к сложным категориям как урожайность, болезни растений, борьба с сорняковыми растениями, исследования задач сельского хозяйства в условиях экологических рисков заморозки, проблемы деградации почвы, засухи.

Многообещающей для многих задач сельского хозяйства, является применение технологий компьютерного зрения, которая тесно связаны с распознаванием болезни и роста растений в период вегетации. Исследованию задач такого класса полностью посвящена четвертая глава диссертации.

Алгоритмы и методы, которые использовалось для обучения моделей машинного обучения и многие коды реализованы с использованием программирования Python, пакетов Sklearn и среды для реализации блокнот Jupyter, а также сервиса GoogleColaboratory. Для создания базы данных, прогнозирования урожайности, которая зависит от многих факторов использованы, зависимости от климата, погоды, почвы, правильной обработки посевных площадей и подбора генотипа. Использование достоверных, близких к природным явлениям данных в настоящее время, также играет ключевую роль в получении точных прогнозов и желаемых результатов для урожайности.

Основной целью данной главы является использование методов машинного обучения к задачам сельского хозяйства для задач прогнозирования урожайности, с использованием различных данных, полученных от регионов Иссык-Кульской области.

Данные, используемые в работе, были извлечены из источников пяти районов Иссык-Кульской области Кыргызской Республики и представляют собой вносимые и обработанные поля, удобрения (азот, фосфор и калий), температуру, влажность, осадки и кислотность почвы для пяти наименований районов, а также урожайность сельскохозяйственных культур для каждого из районов

Для удобства работы с библиотеками Python при сборе данных учитывалось особенности данных каждого из регионов и была представлена в виде обобщенного .csv файла. Одним из важных факторов при формировании модели урожайности, является, проблема оттока клиентов. Очевидно, что без участия фермеров получить соответствующий урожай не получиться. Для регионов области исследован процесс оттока фермеров и получены результаты исследования, по ведении земледелия в виде следующих гистограмм, рис.3.1.

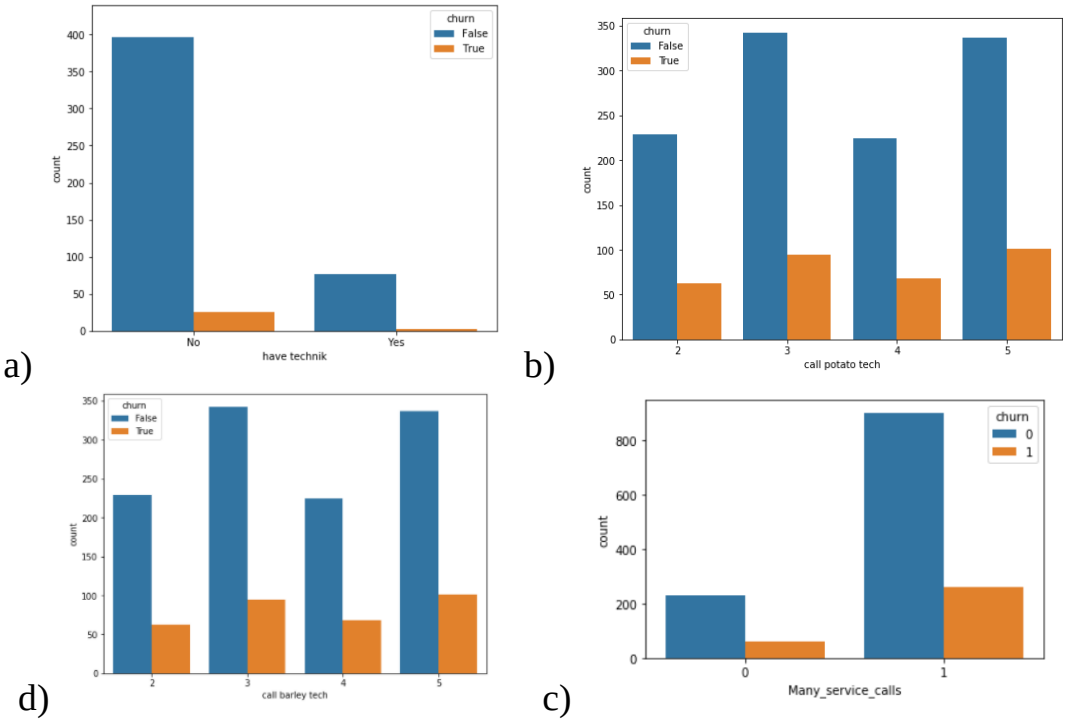


Рис.3.1. Гистограммы оттока фермеров (0- нет оттока, 1-отток) для a) отток при использовании сельхозтехники b, c) отток при аренде сельхозтехники для выращивания ячменя и картофеля d) использование множества других сервисов

Как видно из рисунка отток фермеров в конкретном случаях, зависит от различных факторов. В нашем случае исследован данный процесс для фермеров, которые имеют и не имеют собственную сельскохозяйственную технику, как ни странно, это присуще для фермеров для аграрных стран, где фермеры не в силах купить, содержать собственную сельхозтехнику и вынуждены арендовать их,

иногда, за не окупаемую расходы. Процесс оттока фермеров отчетливо видно из графика гистограмм. Видим, что, когда арендуются сельхозтехника, доля оттока намного выше. Возможно, большие и плохо контролируемые траты при использовании аренды конфликтогенны и приводят к недовольству производителей сельхозпродукций и, соответственно, к их оттоку.

Следующим важным фактором являются сорняки. С момента посева семян до сбора урожая посевы необходимо защищать от сорняков, насекомых и болезней, а также от засухи и наводнений, жары и холода. Все более нестабильная погода и более экстремальные явления, такие как наводнения и засухи, меняют вегетационные периоды, ограничивая доступ к воде, которые способствуют росту различных сорняков, вредителей и грибков и могут намного снизить урожайность.

Необходимы решения защищающие сельскохозяйственные культуры от сорняков, насекомых и болезней. Они должны включать технологии, которые поддерживают сильный и здоровый рост растений. В диссертационной работе изучено задача о влиянии сорняковых растений на рост растений и их классификации в наборе данных сорняковых растений состоящий из однолетней, многолетней и паразитных сорняковые растений, который можно встречать во всех полях наших полей изучаемого региона. Каждый из видов сорняков имеет свой определённый вред урожайности к соответствующим растениям.

Например, в работе определено что, однолетние сорняки присуще корнеплодным растениям, рис.3.2.

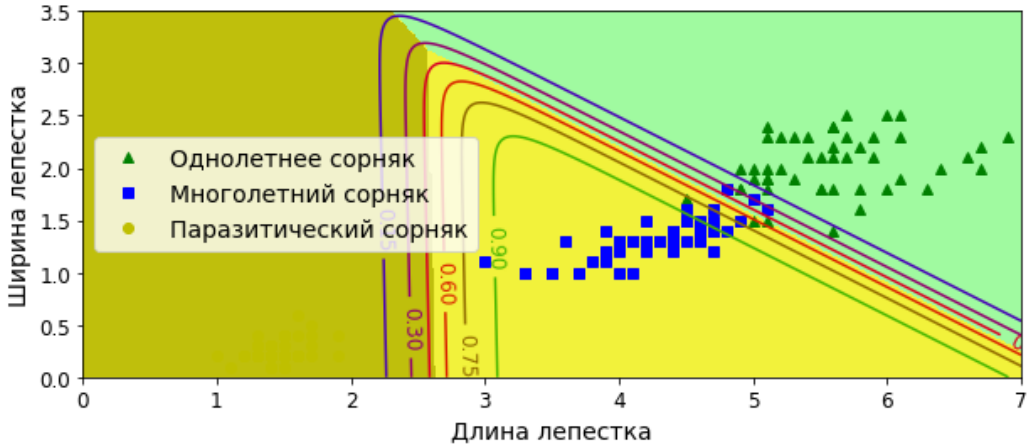


Рис.3.2. Результаты классификации сорняковых растений на три класса-однолетние, многолетние и паразитные

При сборе данных учитывалось особенности данных каждого из регионов и были представлены в виде .csv файлов для удобства работы с библиотеками Python. Все данные урожайности, подчиняется нормальному закону

распределения. Корреляционная матрица, рис.3.3. изучаемой базы данных распределена следующим образом

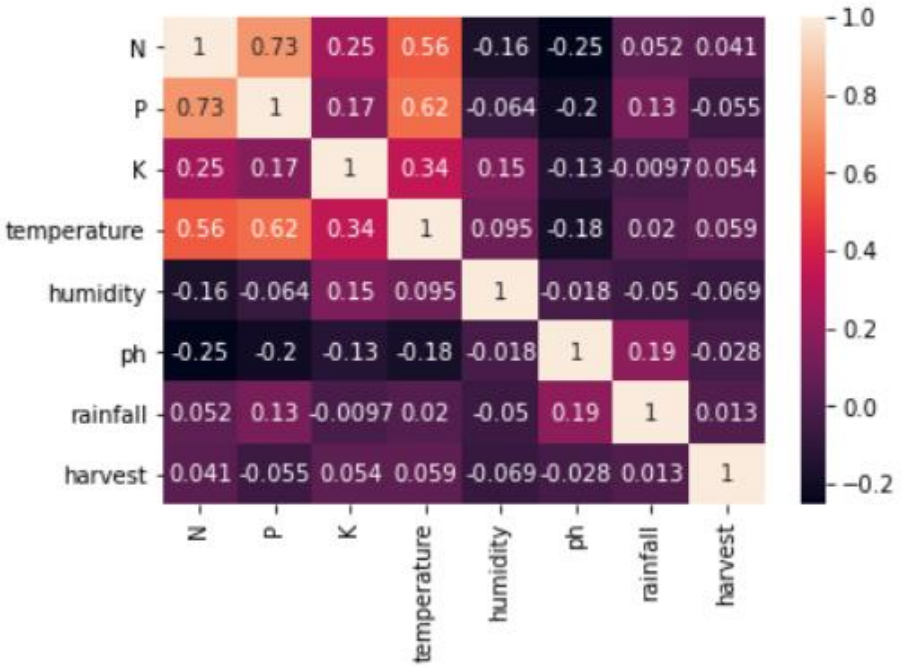


Рис.3.3. Результаты построения корреляционной матрицы прогнозируемого фактора с другими переменными

Из корреляционной матрицы мы видим, что наши независимые данные факторов допускают небольшую мультиколлинеарность или ее отсутствие. Данное распределение, позволяет исследовать задачи прогнозировании зависимой переменной урожайности. В обобщении множественная зависимая переменная harvest –урожайность, в нашем случае не коррелирует с другими независимыми переменными. Все наши изучаемые данные факторов подчиняются нормальному закону распределения, рис.3.4.

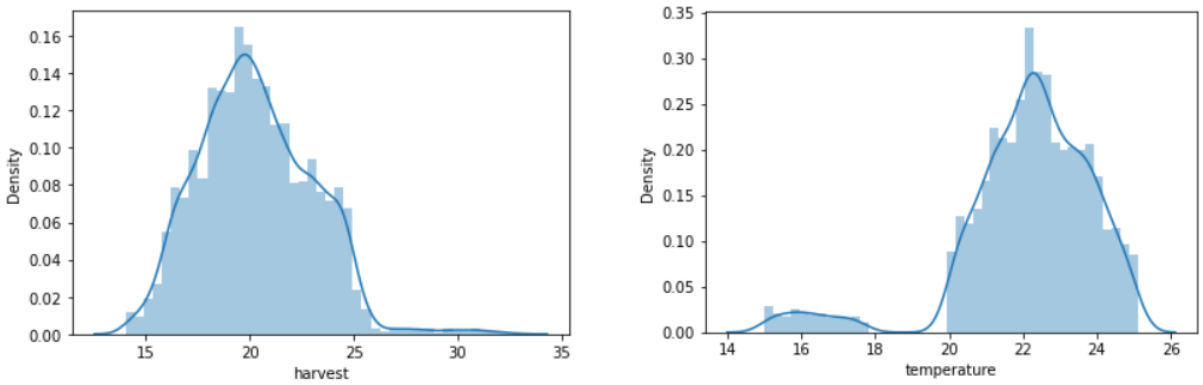


Рис.3.4. Распределение данных, например, прогнозируемого фактора урожайности и температуры по нормальному закону

3.1. Обучение линейных моделей задач сельского хозяйства на основе машинного обучения

Во второй главе мы подробно изучили алгоритм процесса построения линейных моделей. Рассмотрим следующую задачу из сельского хозяйства. Построим модель линейной регрессии для урожайности сельскохозяйственных культур, в которой мы стремимся прогнозировать p наблюдений переменной отклика урожайности Y=harvest с линейной комбинацией p переменных предикторов X и нормально распределенной ошибкой с дисперсией σ^2, которое можно представить формулой

$$\text{Y=X}\omega + \varepsilon, \text{ где } \varepsilon \sim \text{N } (0,\sigma^2) \quad (3.1)$$

где в (3.1) N (μ, σ^2) нормальное распределение с математическим ожиданием $\mu = 0$ и дисперсией σ^2 , которая определяется следующей формулой

$$\text{N } (\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (3.2)$$

Здесь необходимо, чтобы зависимая переменная урожайность Y=harvest не должна коррелировать с другими независимыми переменными X. Зависимая переменная Y урожайности в качестве первоначального представления можно представить в виде модели множественной регрессии, которая имеет следующий вид

$$\text{Y=}\omega_0+\omega_1 X_1+\omega_2 X_2+\ldots+\omega_p X_p+\varepsilon, \quad (3.3)$$

в которой Y – зависимая переменная урожайность, X_p – независимые переменные составляющие множественной регрессии, ω_p- неизвестные весовые коэффициенты, а ε-ошибка модели, определяемая формулой $\varepsilon \sim$ N $(0,\sigma^2)$. Алгоритм определения неизвестных коэффициентов ω_i входящие в данную модель, аналитически можно определить из условия минимума следующего квадратичного функционала:

$$\text{L}=\frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2=\frac{1}{2n}\sum_{i=1}^{n}(y_i - \widehat{\omega}_0 - \widehat{\omega}_1 X_{i1} - \widehat{\omega}_2 X_{i2} - \cdots - \widehat{\omega}_p X_{ip})^2, \quad (3.4)$$

где $\hat{y}_i$ – прогнозируемые значения для точного значения y_i. Выражение (3.4), можно переписать в векторной форме:

$$L(X,\vec{y},\vec{\omega}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \vec{\omega}^T\vec{x}_i)^2 = \frac{1}{2n}||\vec{y} - X\vec{\omega}||_2^2 =$$
$$= \frac{1}{2n}(\vec{y} - X\vec{\omega})^T(\vec{y} - X\vec{\omega}), \quad (3.5)$$

Тогда из условия минимума функционала (3.5) следует

$$\frac{\partial L}{\partial\vec{\omega}} = \frac{\partial}{\partial\vec{\omega}}\frac{1}{2n}(\vec{y}^T\vec{y} - 2\vec{y}^T X\vec{\omega} - \vec{\omega}^T X^T X\vec{\omega}) = \frac{1}{2n}(-2X^T\vec{y} + 2X^T X\vec{\omega}) = -X^T\vec{y} +$$
$$+ X^T X\vec{\omega} = 0,$$

Определяем, весовые коэффициенты:

$$X^T X\vec{\omega} = -X^T\vec{y} \rightarrow \vec{\omega} = (X^T X)^{-1}X^T\vec{y}, \quad (3.6)$$

Таким образом, мы построили алгоритм построения линейной модели. Определили известные оценки параметров МНК как вектор $\vec{\omega}$, которая определяется по формуле (2.14). Однако алгоритм (3.6) является неустойчивым к малым изменениям исходных данных, т.е. задача в данном случае некорректно поставлено.

Модель созданный данным алгоритмом будет переобученным, построенная модель в этом случае начинает интерполировать данные, вместо экстраполяции и она теряет обобщающую способность. Обобщение модели на тестовых данных по данному алгоритму не получиться. Когда функции коррелируют, а столбцы матрицы X имеют приблизительно линейную зависимость, матрица становится близкой к сингулярной. В результате оценка модели методом наименьших квадратов становится очень чувствительной к случайным ошибкам в наблюдаемой цели, что приводит к большой дисперсии.

Для многих задач такая ситуация мультиколлениарности может возникнуть, например, когда данные собираются случайно и без учета эксперимента. Рассмотрим теперь, один конкретный пример метода наименьших квадратов. Рассмотрим реализацию алгоритма (2.14), МНК для конкретной задачи сельского хозяйства. Ниже представлена задача построения линейной модели влияния пестицидов на характеристику плодородия почвы. Реализация данного примера проведено на Python (Приложение 3.1).

```
Shape of X is (35,)
Head of X is [0.14465792 0.18239477 0.27044742 0.48428956 0.50944746 0.57234221
 0.66039485 0.74215802 0.81134225 1.04405281]
```

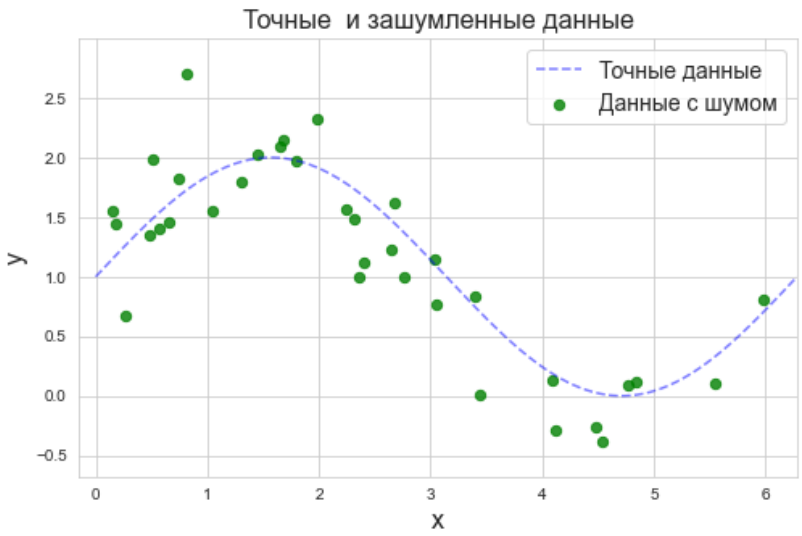


Рис.3.5. Графическое представление данных

Приведем множество данных обучения. Листинг 3.1. Данные обучения

```
Ввод [16]: data['x_train']

Out[16]: array([0.14465792, 0.18239477, 0.27044742, 0.48428956, 0.50944746,
                0.57234221, 0.66039485, 0.74215802, 0.81134225, 1.04405281,
                1.30821075, 1.45286867, 1.65413187, 1.67928977, 1.79250031,
                1.98118456, 2.2453425 , 2.32081619, 2.35226357, 2.40257937,
                2.64157941, 2.67302678, 2.76107943, 3.03152684, 3.05039527,
                3.40260586, 3.43405323, 4.08815861, 4.11331651, 4.48439552,
                4.54100079, 4.77371136, 4.83660611, 5.54731676, 5.98757999])

Ввод [17]: data['y_train']

Out[17]: array([ 1.55002898,  1.44731394,  0.66667142,  1.35402697,  1.98840254,
                 1.41003155,  1.4515063 ,  1.81897548,  2.69989517,  1.5580805 ,
                 1.79259586,  2.03029097,  2.08731367,  2.15125198,  1.97580626,
                 2.32010894,  1.56692727,  1.48232926,  0.9988114 ,  1.11533069,
                 1.22120994,  1.61965078,  0.99161669,  1.14579185,  0.77201729,
                 0.84180525,  0.00386517,  0.12872393, -0.28845764, -0.26534265,
                -0.37745804,  0.08778455,  0.12110027,  0.10260282,  0.80806673])
```

Результат применения алгоритма (3.6) к данной задаче приводит к следующему результату (Приложение 2.2)

```
Верхние ряды данных обучения  X
[[1.         0.20126319]
 [1.         0.43397376]
 [1.         0.54089483]
 [1.         0.67926328]
 [1.         0.78618435]]
Коэффициенты регрессии
[ 2.12714404 -0.35715516]
```

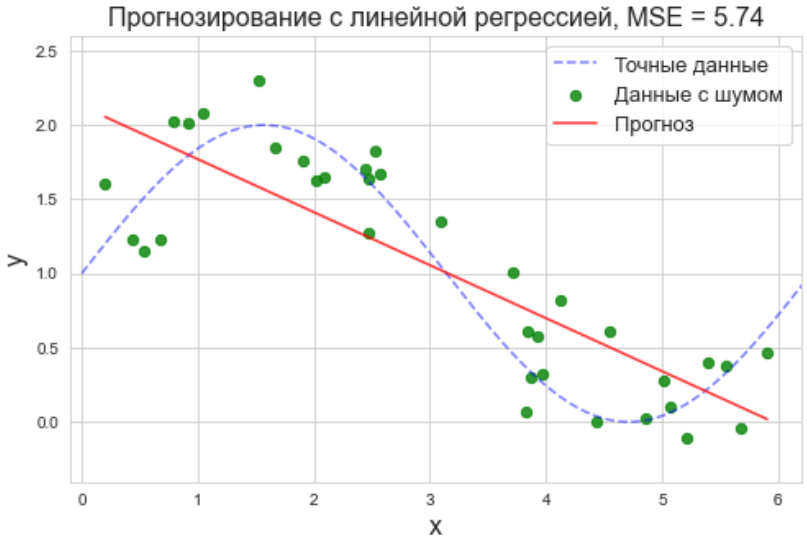


Рис.3.6. Применение алгоритма (3.6) к практической задаче и ее графическое представление

Вообще говоря, расчеты, проведенные по алгоритму (3.6) в одномерном и по аналогичной формуле в многомерном случае приводят к переобученным моделям и не обладают обобщающими свойствами, теряя при этом смысл для тестовых данных. Модель созданный данным алгоритмом будет сколь угодно интерполировать данные вместо экстраполяции при тестировании данных. Для решения данной проблемы вводится некоторая регуляризующая функция $R(\vec{\omega})$. Сформулируем теперь задачу определения весовых коэффициентов $\vec{\omega}$ с преобразованным функционалом ошибок в виде (3.7)

$$\Omega\,(X,\vec{y},\vec{\omega}\,)= L(X,\vec{y},\vec{\omega}\,)+\lambda\, R(\vec{\omega}) \;\Rightarrow min \qquad (3.7)$$

где λ -называется коэффициентом регуляризации, E-единичная матрица. При реализации данной задачи обычно в качестве $R\,(\vec{\omega}\,)$ используют (5) для задачи Риджа:

$$R\,(\vec{\omega}\,)=\frac{1}{2}\,||\vec{\omega}\,||_2^2=\frac{1}{2}\vec{\omega}^{T}\vec{\omega}\,, \qquad (3.8)$$

и (3.12) для задачи Лассо:

$$R(\vec{\omega}) = \lambda||\vec{\omega}||, \quad (3.9)$$

В данном случае алгоритм нахождения коэффициентов $\vec{\omega}$ определяется по формуле (3.10)

$$\vec{\omega} = (X^T X + \lambda E)^{-1} X^T \vec{y}, \quad (3.10)$$

Ниже приведены полученные результаты, которые подробно анализируются ниже, с применением регуляризующих алгоритмов задачи (3.1), (3.4) и (3.7) с учетом (3.5). Результаты, полученные при создании моделей для одномерного и множественной регрессии прогнозирования урожайности применительно к исследуемой базе данных по регионам показаны на рис. 3.7 и рис.3.8

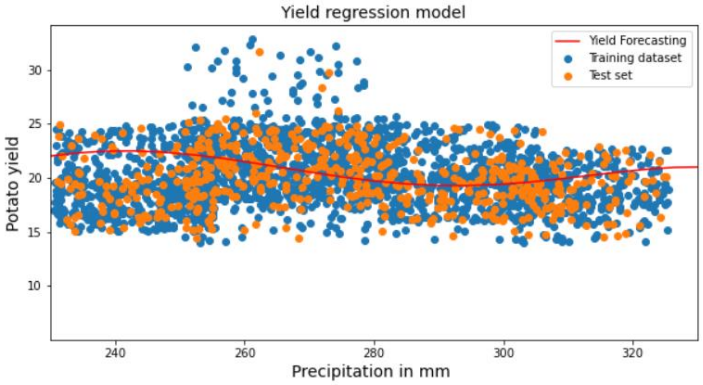


Рис.3.7. Прогноз урожайности регрессии полученная с применением регуляризирующего алгоритма

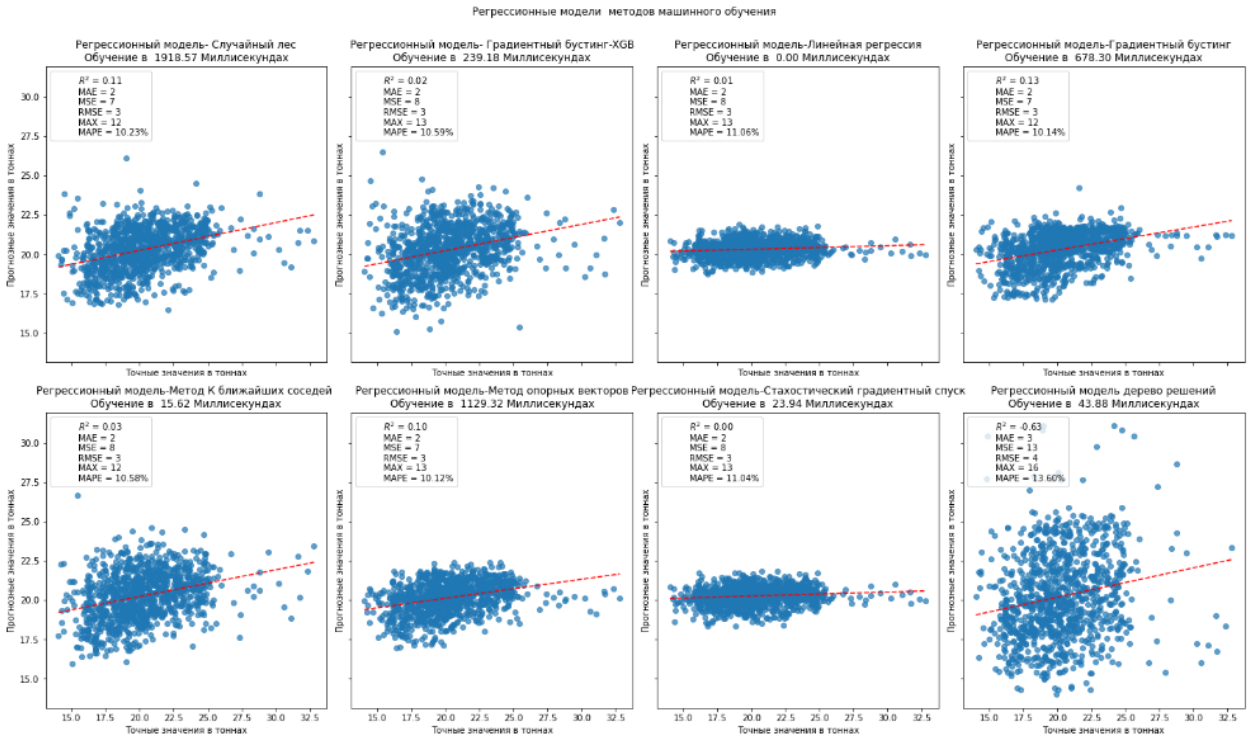


Рис.3.8. Обобщающие результаты алгоритмов машинного обучения в задачах регрессии и скопления данных вокруг линии регрессии по многим факторам

Вот результат регрессионного уравнения для конкретного случая, когда урожайность, зависит от азотного удобрения, температуры и осадков.

Regression Equation: harvest =
18.196024931744926+0.007244397628626693*N+
+0.05654913521333674*temperature+0.0002257625611471747 5*rainfall , (3.11)

Использование алгоритма случайный лес приведен на рис.3.9.

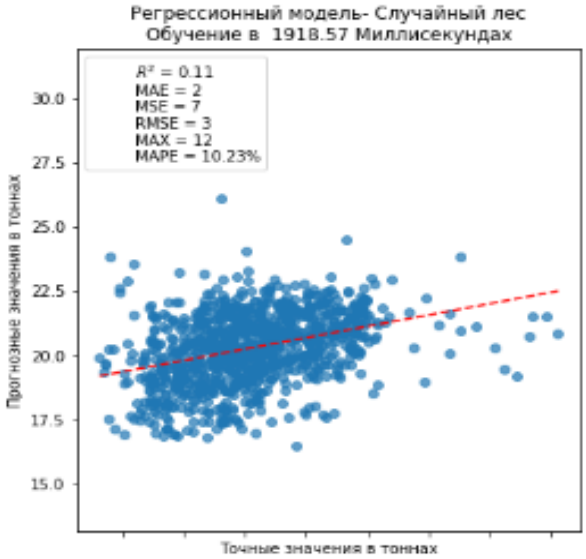


Рис.3.9. Скопления данных вокруг линия регрессии при применении случайного леса

Относительно результатов, полученных с использованием алгоритма Лассо для прогнозирования задач урожайности, для переобученных моделей ниже описан алгоритм для данного метода. Функционал ошибок, который используется для нахождения коэффициентов регрессии может быть использован как оператор LASSO. В данном методе, который является и методом регуляризации, коэффициенты при коррелированных признаках, исключаются, т.е. данным методом их коэффициент устанавливается равным нулю . К функционалу ошибок добавляется, так называемы штрафной член ($|\omega_i|$) к модели линейной регрессии в Тихонова-Лассо, которая может уменьшать коэффициенты до нуля (регуляризация L1). Функция потерь Тихонова-Лассо в данном случае выглядит следующим образом.

$$L = \sum(y_i - \hat{y}_i)^2 + \alpha\sum|\omega_i|, \quad (3.12)$$

где в (3.12), знак суммирование распространяется по $i = 1, .., n, \alpha$- параметр отвечающее за скорость сходимости, который необходимо определить перед

выполнением обучающей задачи. Эффективность данного алгоритма и результаты прогноза для нашего случая, приведены на рис.3.9 и Таблица 3..

Таблица 3.1. Производительность основных алгоритмов машинного обучения

№	Estimates/ML Algorithm	R2score	MAE	RMSE
1	Lasso Regression -all parametrs	0.0351341	0.7237154	0.8865015
2	Lasso Regression-selected parametrs	0.9980028	0.0393173	0.0403324
3	SVR –all parametrs	0.1411921	0.6776377	0.6776377
4	SVR -selected parametrs	0.9958232	0.05250301	0.0583266
5	Random Forest Regression – selected parametrs	0.9999996	0.0003116	0.0005596
6	Random Forest Regression –all parametrs	0.0750289	0.6847263	0.8679807
7	Gradient Descent Algorithm–all parametrs	0.0047446	0.7329265	0.9003539
8	Gradient Descent Algorithm – selected parametrs	0.3576148	0.5885566	0.7233421

3.2. Моделирование и прогнозирование урожайности на основе множественной регрессии

Данный раздел диссертации полностью посвящен практической реализации алгоритмов машинного обучения к задачам сельского хозяйства, математические основы которых и особенности их применения к прикладным задачам рассмотрены во второй главе. Для задач прогнозирования обучение линейных моделей играет особую, фундаментальную роль. С этой целью будем использовать множественную линейную регрессию для прогнозирования урожайности. Прогнозирование урожайности — это рациональный и научный способ предсказания будущих событий в сельском хозяйстве — уровень производственных эффектов. Его основная цель – снижение риска при принятии решений. Данный процесс, влияет на урожай с точки зрения количества и качества. Первоначальная цель заключается в создании линейной модели для прогнозирования урожайности различных сельскохозяйственных культур. Линейная модель в данном случае создается на основе множественной линейной регрессионного анализа (MLR), которая является базовым для изучения сложнейших нелинейных моделей. Обычно нелинейные модели создаются с использованием искусственных нейронных сетей (ИНС), которые рассматриваются в Главе 4. Построенную модель мы будем проверять с

использованием следующих метрик- ошибки прогноза. Это глобальная относительная ошибка аппроксимации (RAE), среднеквадратическая ошибка (RMSE), средняя абсолютная ошибка (MAE) и среднюю абсолютную процентную ошибку (MAPE). Рассмотрим метрики ошибок и как оценивать свои прогнозы. При рассмотрении производительности любой модели прогнозирования необходимо оценивать получаемые ею прогнозные значения. Это делается путем расчета подходящих метрик ошибок. Метрика ошибок — это способ количественной оценки эффективности модели, который позволяет прогнозисту количественно сравнивать различные модели. Они дают нам возможность более объективно оценить, насколько хорошо модель выполняет свои задачи. В этом сообщении блога мы рассмотрим некоторые часто используемые показатели для прогнозирования временных рядов, способы их интерпретации, а также ограничения каждого показателя. Как работают эти показатели. Вот основные формулы.

В этом разделе диссертации мы начнем с исследования линейной регрессионной модели (linear regression model) урожайности, одной из простейших и доступных моделей. Мы обсудим два очень разных способа ее обучения.

Получим алгоритм обучения линейных моделей урожайности с применением прямого уравнения в аналитическом виде, непосредственно вычисляющего параметры модели, которые лучше всего подгоняют модель к обучающему набору (т.е. параметры модели, сводящие к минимуму значение функции издержек на обучающем наборе).

Численный подход с использование подхода итеративной оптимизации, называемого градиентным спуском (Gradient Descent - GD), который постепенно корректирует параметры модели урожайности, чтобы довести до минимума значение функции издержек на обучающем наборе, в итоге сходясь к тому же самому набору параметров, что и при первом способе.

С применением этих алгоритмов создадим модель прогнозирования урожайности, являющееся объемной категорией для задач сельского хозяйства. Мы рассмотрим, также несколько вариантов численной реализации данной задачи, градиентного спуска, которая периодически будет применяться при изучении нейронных сетей, четвертая глава настоящей диссертации, пакетный градиентный спуск (Batch GD - BGD), мини-пакетный градиентный спуск (mini-batch GD) и стохастический градиентный спуск (stochastic GD). Далее мы изучим применение полиномиальной регрессии (polynomial regression),более сложную модель, которая может подгоняться к нелинейным наборам данных . Поскольку такая модель имеет больше параметров, чем линейная регрессионная модель, она более предрасположена к переобучению обучающими данными, поэтому мы посмотрим, как обнаруживать подобную ситуацию, используя кривые обучения

(learning curve), и опишем несколько приемов регуляризации, которые способны снизить риск переобучения обучающим набором. В заключение мы рассмотрим еще две модели, широко применяемые для задач классификации: логистическую регрессию (logistic regression) и многопеременную логистическую регрессию (softmax regression). Построим конкретные модели с применением данной технологии к задачам прогнозирования сельского хозяйства.

Постановка задачи урожайности. Предполагает, что выходная переменная h(x)-урожайность или цель могут быть предсказаны с помощью следующей модели:

$$H(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x, \quad (3.13)$$

где $x_0 = 1$, n — количество признаков, θ_i— неизвестные параметры. Когда n = 1, гипотеза сводится к уравнению следующей простейшей задачи урожайности:

$$\text{harvest} = \theta_0 + \theta_1 * \text{rainfall}$$

В данном случае rainfall – это осадки. Для n = 2, h(x) сводится к виду:

$$\text{harvest} = \theta_0 + \theta_1 \text{rainfall} + \theta_2 \text{humidity},$$

где humidity –означает влажность среды. Задача алгоритма обучения состоит в том, чтобы определить значения параметров из обучающей выборки, т.е. оценить θ из заданных значений данных. В дальнейшем мы построим линейную модель урожайности, которая будет, зависит от многих факторов т.е. наиболее полную модель для прогнозирования урожайности. Точность алгоритма обучения обычно измеряется функцией стоимости (ошибки), которая естественным образом выбирается следующей функционала:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2, \quad (3.14)$$

где $y^{(i)}$ - наблюдаемые целевые значения. Параметр θ определяется из условий наилучшей оценки минимума функционала $J(\theta)$. т.е. из условия $J(\theta) \rightarrow \min$, по θ (3.15)

Аналитическая минимизация. Нормальное уравнение урожайности.

В силу линейности предполагаемой модели урожайности из условий минимизации функционала стоимости J(θ) можно получить аналитическое точное решение. Решение может быть получено с использованием терминологии линейной алгебры. Но нам нужно брать производные по матрицам. Учитывая функцию, которую нам нужно найти

$$\nabla_\theta \text{ J}= \begin{pmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{pmatrix} \in R^{n+1}$$

Итерация градиентного спуска запишем в виде соотношения:

$$\theta^{n+1} = \theta^n - \alpha\nabla_\theta \text{ J}(\theta^n), \qquad \text{n=0,1,…} \qquad (3.16)$$

где левая и правая части представляют собой n + 1-мерные векторы.

Определение. Если f:R^{mxn} →R или f(A), A∈ R^{mxn}. Тогда

$$\nabla_A f(A) = \begin{pmatrix} \frac{\partial f}{\partial A_{11}} \cdots \frac{\partial f}{\partial A_{1m}} \\ \vdots \\ \frac{\partial f}{\partial A_{m1}} \cdots \frac{\partial f}{\partial A_{mn}} \end{pmatrix}$$

Известно, что $z^T z = \sum_i z_i^2$, тогда

$$\frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) = \frac{1}{2}\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \text{J}(\theta)$$

Таким образом, мы получили алгоритм (3.16) вычисления производной J(θ) и определили параметр θ по формуле.

$$\theta = (X^T X)^{-1} X^T y, \qquad (3.17)$$

Таким образом, мы получили точное решение задачи урожайности (3.17) в виде нормального уравнения. Это и есть алгоритм обучения нашей линейной модели. Однако при использовании модели (3.13) для больших значений признаков т.е. при увеличении параметров модели, матрица $X^T X$ становится, сингулярной т.е. плохо обусловленной. Обратная матрица в данном случае не существует. В таких случаях используют метод регуляризации.

Теперь на основе вышеизложенных алгоритмов мы построим модели для прогнозирования, конкретных задач сельского хозяйства. С помощью множественной линейной регрессии построим модель прогнозирования урожайности. Для этой цели будем использовать множественную линейную регрессию для прогнозирования урожайности. Прогнозирование урожайности — это рациональный и научный способ предсказания будущих событий в сельском хозяйстве — уровень производственных эффектов. Его основная цель – снижение риска при принятии решений. Это процесс, влияющий на урожай с точки зрения количества и качества. Цель заключается в создании линейной модели для прогнозирования урожайности различных сельскохозяйственных

культур. Линейная модель в данном случае создается на основе множественной линейной регрессионного анализа (MLR). Обычно нелинейные модели создаются с использованием искусственных нейронных сетей (ИНС). При рассмотрении производительности любой модели прогнозирования необходимо оценивать получаемые ею прогнозные значения. Это делается путем расчета подходящих метрик ошибок. Метрика ошибок — это способ количественной оценки эффективности модели, который позволяет аналитику по прогнозирования количественно сравнивать различные модели. Они дают нам возможность более объективно оценить, насколько хорошо модель выполняет свои задачи. Построенную модель мы будем проверять с использованием следующих метрик ошибки прогноза. Это глобальная относительная ошибка аппроксимации (RAE), среднеквадратическая ошибка (RMSE), средняя абсолютная ошибка (MAE) и среднюю абсолютную процентную ошибку (MAPE). Теперь рассмотрим конкретный пример. Загружаем библиотеку программного обеспечения, для построения линейной регрессии. Ниже приведена структура базы данных урожайности целевая переменная (harvest), как зависимый признак и независимые признаки базы данных N, P, K –азотное, фосфорное и калийное удобрения соответственно. Функции temperature, humidity и rainfall определяют температуру, влажность воздуха и осадки, которые входят в базу данных как погодные характеристики, ph – означает кислотность почвы, где выращивается данная культура. Листинг 3.2. Необходимый пакет программ для загрузки данных:

```
#Импортируем библиотеки
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Загружаем базу данных
dataset = pd.read_csv("data_harvest_potato_orig.csv")
```

```
dataset.head()
```

Прежде чем построит модель линейной регрессии, мы должны предварительно обработать базу данных. Листинг 3.3. база данных.

	N	P	K	temperature	humidity	ph	rainfall	harvest
0	87.061966	59.225440	56.501270	15.438161	61.188624	7.111946	309.758114	19.227750
1	87.673701	60.270827	67.905067	16.943861	56.056799	7.124829	317.056731	19.003378
2	88.548707	58.155658	62.924899	17.008231	56.348194	7.602639	320.557945	18.777324
3	89.332462	58.205528	67.702136	15.228555	59.279004	8.177541	310.698694	18.914045
4	90.966950	59.743297	56.636310	15.661283	55.958781	7.581785	316.473301	18.010575

```
dataset.tail()
```

	N	P	K	temperature	humidity	ph	rainfall	harvest
5024	103.695292	73.980198	71.044300	21.371892	44.604063	7.340779	305.132219	17.438029
5025	115.404442	71.899455	73.481243	21.531391	38.830613	8.058029	294.097090	17.051708
5026	118.219000	74.197155	72.236984	21.188428	42.803255	7.327458	301.477242	18.206521
5027	106.505420	76.944341	77.660698	22.212777	41.600163	7.456819	293.704992	17.130518
5028	115.866384	71.962656	73.560316	20.363222	40.196063	6.970762	290.715123	18.715612

Ниже предельно ясно представлены в Листинг 3.3 некоторые результаты о свойствах базы данных:

Листинг 3.4. Анализ размерности пропущенных значений в базе данных

```
dataset.shape
```

```
(5029, 8)
```

```
Анализ пропущенных значений в таблице данных
```

```
dataset.isna().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
harvest        0
dtype: int64
```

```
Вывод. В нашем случае нет пропущенных значений в табл
```

Проверим на повторяющихся строк

```
dataset.duplicated().any()
```

```
False
```

Повторяющихся значений нет

Теперь изучим выбросы данных по признакам. Вот соответствующие результаты.

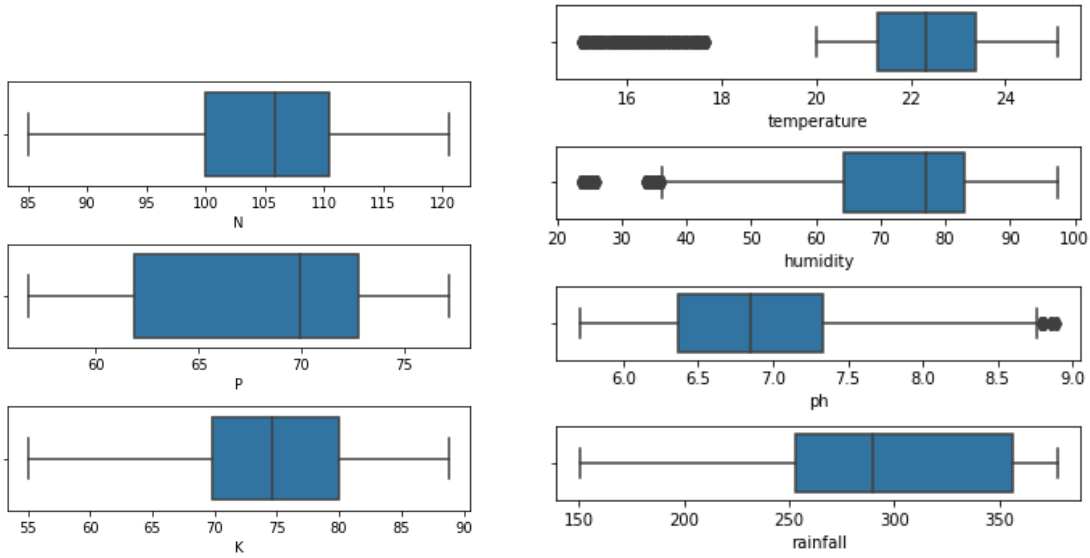


Рис.3.10. Выбросы в базе данных по функциям.

Как видно из рис.3.11, нет экстремальных значений в базе данных. Теперь изучим плотность функций распределения целевой и составляющих переменных.

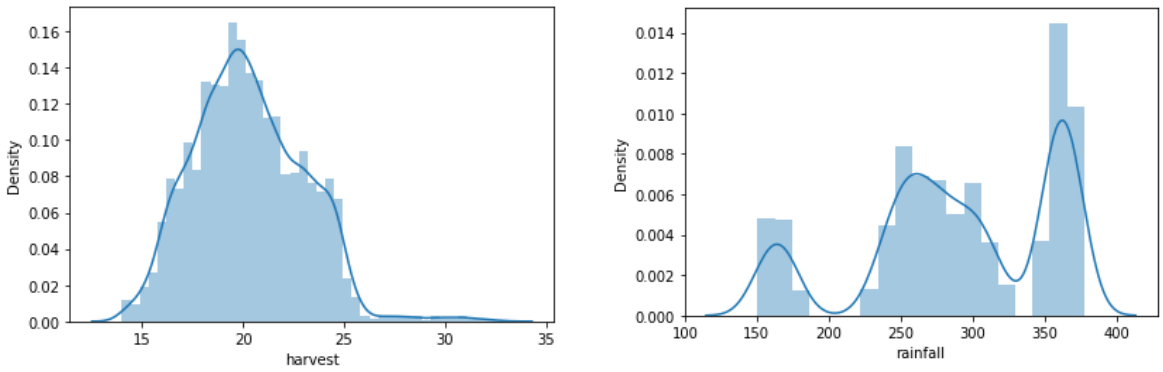


Рис.3.11. Пример плотности функций распределения урожайности и осадки.

Целевая переменная и осадки распределены по нормальному закону. Ниже представлены результаты, как урожайность связаны с другими переменными.

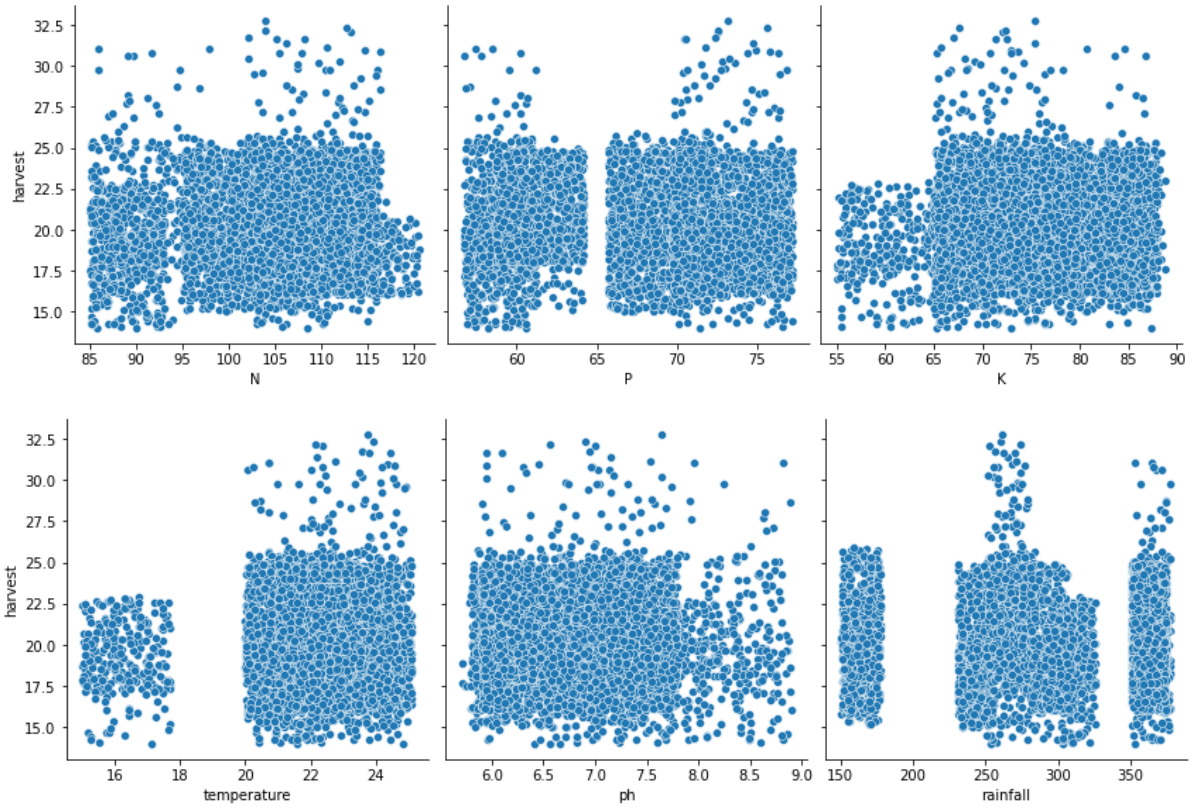


Рис.3.12. Результаты связи целевой переменной и другими функциями.

В целом признаки относительно целевой переменной некоррелированы. Только малая часть процентов, 30 коррелируют с целевой переменной. Для просмотра связи между целевой и независимыми переменными выведем корреляционную матрицу:

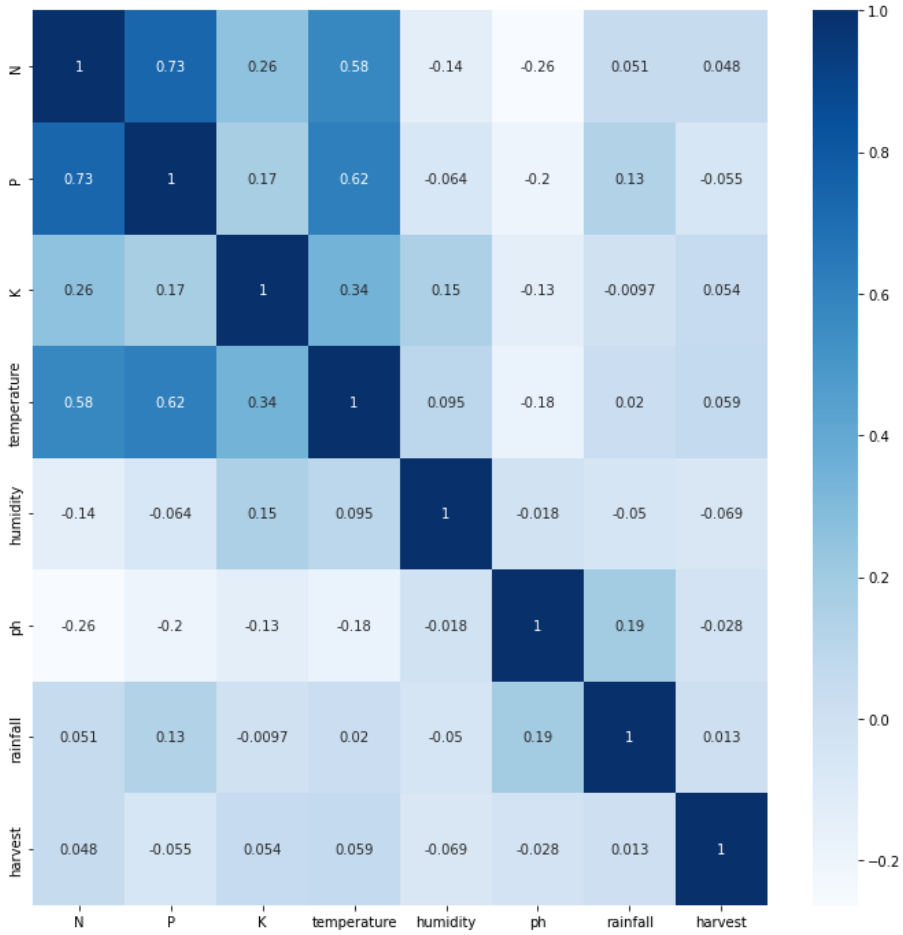


Рис.3.13. Корреляционная матрица связи целевой переменной и другими функциями.

Регрессионная модель урожайности. Теперь построим уравнение одномерной модели урожайности. В данном случае мы имеем только одну переменную, например rainfall и одну переменную harvest. Это подход к прогнозированию количественной связи с использованием одной функции. Он устанавливает связь между двумя переменными с помощью прямой линии. Линейная связь здесь пытается провести линию, которая ближе всего подходит к данным, путем нахождения наклона и точки пересечения, которые определяют линию и минимизируют ошибки регрессии. Формула в одномерном случае выглядит таким образом:

$$\text{harvest}=\beta_0+\beta_1 rainfall+\varepsilon, \quad (3.18)$$

здесь, harvest - зависимая переменная (Целевая переменная); β_0 - пересечение линии регрессии; β_1 - наклон кривой регрессии, которая сообщает, увеличивается или уменьшается линия; rainfall - независимая переменная / переменная-предиктор; ε – ошибка.

В нашем случае мы можем написать, например зависимость:

$$harvest=\beta_0+\beta_1 rainfall+\varepsilon.$$

Множественная регрессия урожайности. Создадим теперь множественное уравнение регрессии относительно переменных базы данных data_harvest_potato_orig.csv. В нашем случае это будет, независимые переменные: 'N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall' и целевая зависимая переменная 'harvest'. В общем случае уравнение выглядит следующим образом:

$$\begin{aligned} harvest = \beta_0 + \beta_1 N + \beta_2 P + \beta_3 K + \beta_4 * temperature + \\ + \beta_5 * humidity + \beta_6 * ph + \beta_7 * rainfall, \end{aligned} \quad (3.19)$$

В нашем случае уравнение (3.19) с неизвестными коэффициентами $\beta_0, \beta_1, \ldots, \beta_7$ называется уравнением регрессии урожайности, относительно нашей базы данных. Отметим также, что уравнение (3.19) может содержать множество других неизвестных параметров в зависимости от структуры исходной базы данных. Для программной реализации и построения уравнения регрессии, необходимо определить переменные x и y , как входные и выходные данные для обучения модели, например по программному пакету mlr= LinearRegression() из системы sklearn. После нормировки входных данных и ее преобразования в массив numpy с разделением данных на обучающие x_train и y_train наши данные преобразуются в двумерные и одномерные массивы соответственно в виде Листинга 3.5:

Листинг 3.5. Данные после нормирования

```
x_train
```

```
array([[0.63733899, 0.29289957, 0.54845238, ..., 0.65814405, 0.18463285,
        0.42303133],
       [0.08989596, 0.06227135, 0.0270424 , ..., 0.50244207, 0.46238809,
        0.70706431],
       [0.02782894, 0.13184142, 0.25651565, ..., 0.77180203, 0.41930445,
        0.73439934],
       ...,
       [0.20201106, 0.0819131 , 0.36203616, ..., 0.75813708, 0.33603267,
        0.67167168],
       [0.05657502, 0.19239375, 0.29594306, ..., 0.43815314, 0.15240818,
        0.6927117 ],
       [0.71360918, 0.75364412, 0.54165328, ..., 0.78044623, 0.39682825,
        0.92968058]])
```

```
y_train
```

```
array([[0.30129696],
       [0.20873341],
       [0.24501794],
       ...,
       [0.09821491],
       [0.41164126],
       [0.37839467]])
```

После реализации мы можем выписать искомое уравнение множественной регрессии урожайности в виде линейного уравнения. Уравнение регрессии урожайности:

$$harvest = 0.31675191+0.07841115*N-0.12146211*P+0.01389853*K+0.10198519*temperature-0.06513732*humidity-0.01610677*ph+0.01503798*rainfall, \quad (3.20)$$

Уравнение (3.20) представляет с собой, линейную модель урожайности harvest и устанавливает связь между независимыми переменных и зависимой переменной. Теперь рассмотрим одномерный случай. Для примера определим линейную связь между использования азотных удобрений и урожайностью. Определим, также следующие метрики: MAE, который определяется как среднее значение абсолютной разницы между прогнозируемыми значениями и истинными значениями.

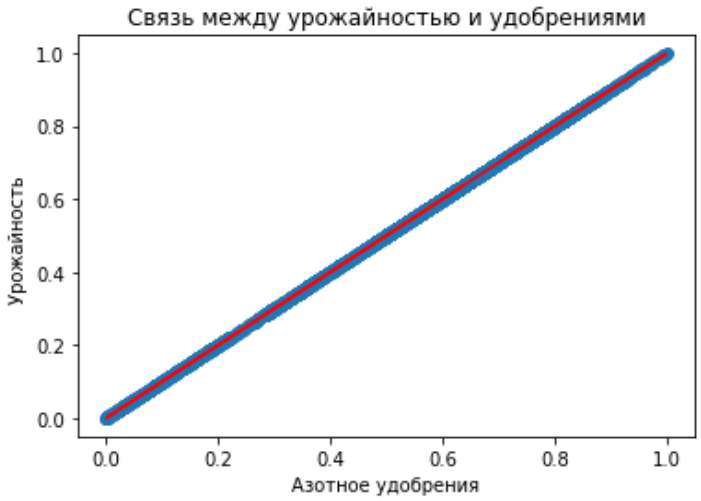


Рис.3.14. Корреляционная связь целевой переменной урожайности и применения удобрений.

MSE определяется как среднее значение квадратов ошибки. А также мы определим показатель RMSE, который является расширением MSE и определяется как квадратный корень из среднеквадратичной ошибки. Вот соответствующие результаты:

Листинг 3. 6. Вычисление метрик.

```
print('MAE-Mean Absolute Error:', meanAbErr)
print('MSE-Mean Square Error:', meanSqErr)
print('RMSE-Root Mean Square Error:', rootMeanSqErr)
```

```
MAE-Mean Absolute Error: 8.201684076388092e-17
MSE-Mean Square Error: 1.0919489619699138e-32
RMSE-Root Mean Square Error: 1.0449636175340813e-16
```

С помощью уравнения (3.20), можно устанавливать различные связи между нашими функциями. Ниже приведены результаты прогноза в многомерном случае.

Листинг 3. 7. Сравнение данных прогнозирования

	Actual value	Predicted value
3948	23.301191	20.358309
2453	20.681503	20.310101
302	21.148933	20.124360
3633	19.389529	20.426233
1490	19.390689	20.106113
...	...	...
1463	18.783205	20.166900
4992	19.927974	20.222249
1821	19.353832	20.253464
1551	23.877601	20.120557
205	17.391741	19.791394

1509 rows × 2 columns

Полиномиальная регрессия как расширение линейных моделей базисными функциями. Одним из распространенных методов машинного обучения является использование линейных моделей, обученных на нелинейных функциях данных. Этот подход поддерживает в целом высокую производительность линейных методов, позволяя им соответствовать гораздо более широкому диапазону данных.

Например, простую линейную регрессию можно расширить, построив полиномиальные признаки из коэффициентов. В случае стандартной линейной регрессии у вас может быть модель, которая выглядит следующим образом для двумерных данных:

$$\hat{y}(\omega, x)=\omega_0+\omega_1 x_1+\omega_2 x_2, \quad (3.21)$$

Если мы хотим подогнать к данным параболоид вместо плоскости, мы можем объединить признаки в полиномы второго порядка, чтобы модель выглядела так:

$$\hat{y}(\omega, x) = \omega_0+\omega_1 x_1+\omega_2 x_2 + \omega_3\omega_2 x_2 + \omega_4 x_1^2 + \omega_5 x_5^2 . \quad (3.22)$$

Линейную модель (3.22), с помощью обозначения

$$z = [\, x_1 , x_2 , x_1 x_2 , x_1^2 , x_2^2 \,], \quad (3.23)$$

формулу (3.22), мы можем переписать в виде формулы линейной модели

$$\hat{y}(\omega, z) = \omega_0+\omega_1 z_1+\omega_2 z_2 + \omega_3 z_3 + \omega_4 z_4 + \omega_5 z_5 . \quad (3.24)$$

Таким образом, мы убедились, что после небольших преобразований мы нелинейную модель преобразовали в линейную модель. Рассмотренный нами пример показывает, что эти два преобразователя хорошо подходят для моделирования нелинейных эффектов с помощью линейной модели. При этом мы используем два конвейера для добавления нелинейных функций. Методы ядра расширяют эту идею и могут создавать пространства признаков очень высокой (даже бесконечной) размерности. Определим функции, которую собираемся аппроксимировать, и подготовим ее график. Далее, чтобы сделать его интересным, делаем только небольшое подмножество точек для обучения. Вот график, которая показывает, насколько хорошо они интерполируются между собой.

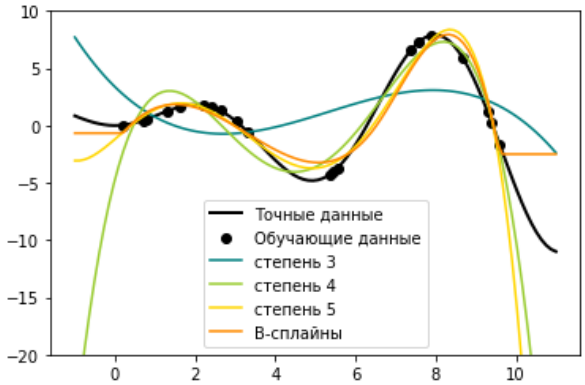


Рис.3.15. Моделирования нелинейных эффектов с помощью линейной модели

Ниже представлен результат аппроксимации полиномами высокой степени для задачи о пестицидах, для которой в простейшем случае мы построили регрессионную картину с применением алгоритма нашего алгоритма. Результат применения полиномов различной степени приведено ниже.

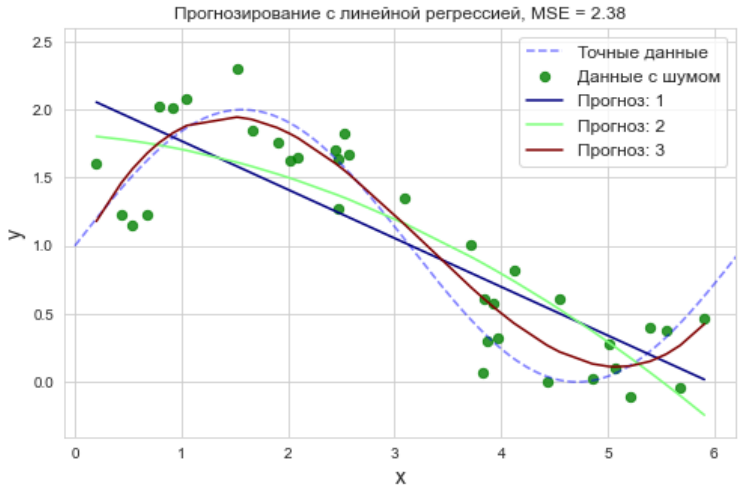


Рис.3.16. Прогнозирование с помощью полиномов низкой степени

Здесь представлено аппроксимация точной модели с помощью полиномов высшей степени.

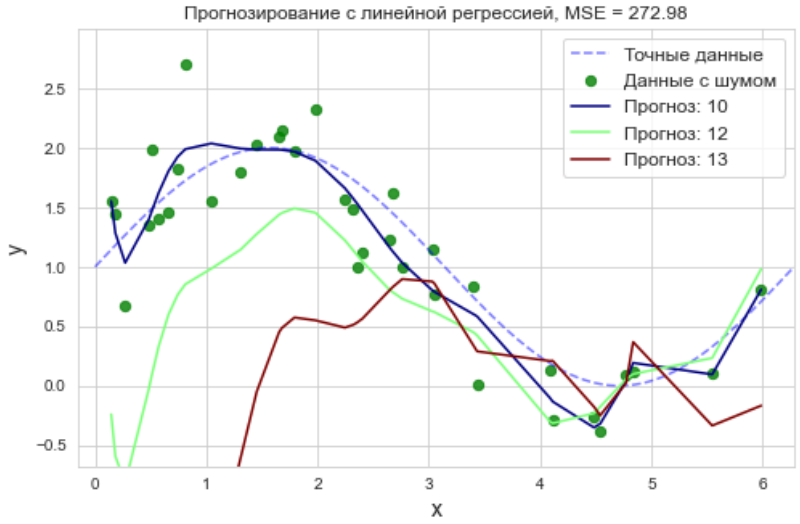


Рис.3.17. Прогнозирование с помощью полиномов высокой степени

Но в то же время слишком большие степени могут показать, нежелательное колебательное поведение и особенно опасны для экстраполяции за пределы диапазона подобранных данных. В этом случае модель переобучается, и она не обладает способностью к обобщению. Вот коэффициенты аппроксимации в случае аппроксимации полиномом степени 13 видно на рис.3.18.

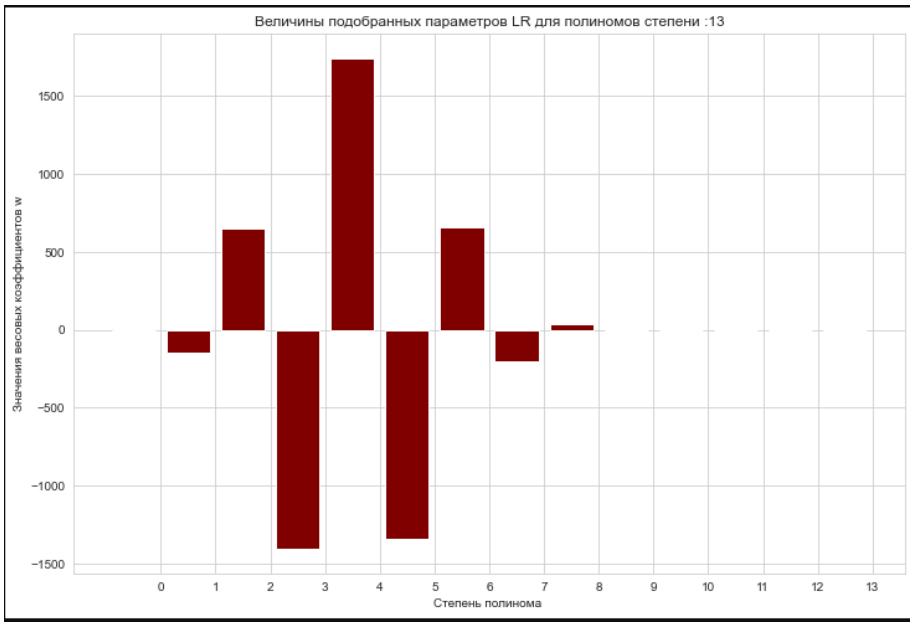


Рис.3.18. Коэффициенты регуляризации

Таким образом, мы видим, что с одной стороны полиномы высшей степени отлично прогнозируют или аппроксимирует обучающую выборку, но на тестовых выборках аппроксимация совершенно не отражает истинную модель. Модели, построенные с помощью полиномами высоких степеней, не будет обладать обобщающими способностями.

3.3. Переобученные модели технологии регуляризации в машинном обучении

Как было показано выше, основная проблема машинного обучения, это склонность моделей, построенных алгоритмами машинного обучения к переобучению. Приступим к решению этой важной проблемы. Переобучение модели происходит, когда модель «слишком хорошо учится» на данных обучения. Это может показаться преимуществом, но это не так. Когда модель перетренирована на обучающих данных, она хуже всего работает на тестовых данных или любых новых предоставленных данных. Технически модель изучает детали, а также шум данных в обучающей выборке. Это помешает производительности любых новых данных, предоставленных модели, поскольку изученные детали и шум не могут быть применены к новым данным. Это тот случай, когда мы говорим, что производительность модели некорректна.

Рис.3.19. График переобученной модели.

Из рисунка видно, что весовые коэффициенты ближе к точке 10 сильно меняется. В нашем случае прогнозное значение –зеленые точки расходится с обучающими значениями. Такой процесс мы называли эффектом переобучения модели. Существует несколько способов избежать переобучения модели, таких как перекрестная проверка, в K-кратном порядке, повторная выборка, уменьшение количества признаков и т. д. Один из способов – применить к модели регуляризацию. Предложенная ниже регрессия Тихонова-Риджа решает некоторые проблемы , которые возникают в методах наименьших квадратов, налагая штраф на размер коэффициентов. Коэффициенты хребта минимизируют оштрафованную остаточную сумму квадратов. Для решения данной проблемы вводится некоторая регуляризующая функция. Регуляризация – это метод, который снижает весовые коэффициенты. В модели с переобучением эти коэффициенты обычно завышены. Таким образом, регуляризация добавляет штрафы к параметрам и позволяет избежать больших коэффициентов в функции стоимости. Коэффициенты добавляются к функционалу минимизации или функции стоимости линейного уравнения. Таким образом, если коэффициент увеличивается, функция стоимости будет увеличиваться. И модель линейной регрессии попытается оптимизировать коэффициент, чтобы минимизировать функцию стоимости.

Рассмотрим случаи метода регуляризации Лассо и Тихонова-Риджа. Метод регуляризации L_1, которую определим ниже, также известен как LASSO регуляризация или оператор наименьшего абсолютного сокращения и выбора. При этом штрафной член, добавляемый к функции стоимости, представляет собой сумму абсолютных значений коэффициентов. Поскольку

используется абсолютное значение коэффициентов, он может уменьшить коэффициент до 0, и такие функции могут быть полностью отброшены в рядах LASSO. Таким образом, мы можем сказать, что LASSO помогает как в регуляризации, так и в выборе функций. Регуляризация Лассо, это линейная модель, которая оценивает разреженные коэффициенты. Данная технология предпочтительнее для моделей с меньшим количеством ненулевых коэффициентов. Он эффективно уменьшает количество функций, от которых зависит данное решение. По этой причине Лассо регуляризация и его варианты имеют большое применение на практике. При определенных условиях он может восстановить точный набор ненулевых коэффициентов. Рассмотрим пример удержания влаги в почве в зависимости от количества дней. Функция имеет вид

$$y(x) = \frac{1}{1+10x^2},\ x \in [0; 10]. \qquad (3.25)$$

Приведем результаты расчета. Расчет коэффициентов полиномиальной модели представлен на Листинге 3.8.

Листинг 3.8. Расчет коэффициентов модели (3.25)

```
import numpy as np
def predict_poly(x, koeff):
    res = 0
    xx = [x ** (len(koeff) - n - 1) for n in range(len(koeff))]
    for i, k in enumerate(koeff):
        res += k * xx[i]
    return res
x = np.arange(0, 10.1, 0.1)
y = 1 / (1 + 10 * np.square(x))
x_train, y_train = x[::2], y[::2]
N = len(x)
z_train = np.polyfit(x_train, y_train, 10)
print(z_train)
```

```
z_train = np.polyfit(x_train, y_train, 10)
print(z_train)
```

```
[-1.26252985e-07  5.76781414e-06 -1.05030156e-04  9.09286286e-04
 -2.58708457e-03 -1.92393634e-02  2.13797583e-01 -8.96937855e-01
  1.95728683e+00 -2.19007649e+00  1.02203201e+00]
```

Здесь мы видим , что появляется диспропорция в коэффициентах, сначала они имеют маленькие значения , но с ростом степени полинома они быстро растут.Например в случае полинома степени 50, ниже даны результаты расчета.

Листинг 3.9. Результаты обучения полиномиальной модели высокой степени

```
z_train = np.polyfit(x_train, y_train, 50)
print(z_train)
```

```
[ 2.31162714e-41 -6.73073223e-40  2.39431313e-39  5.33539207e-38
  1.53942493e-37 -3.22162173e-36 -4.63625274e-35 -2.29124668e-34
  1.62113038e-33  4.12516261e-32  3.85265931e-31  1.09940723e-30
 -2.27735372e-29 -4.23352124e-28 -3.53837161e-27 -7.03689552e-27
  2.62188463e-25  4.33001587e-24  3.32323145e-23  1.20181660e-23
 -3.23377808e-21 -4.48024395e-20 -2.58472450e-19  1.29732997e-18
  4.37309522e-17  4.03895493e-16  2.43073908e-16 -4.02027272e-14
 -4.67381428e-13 -5.65702478e-13  4.48497370e-11  4.59942315e-10
 -1.26709294e-09 -5.89196341e-08 -1.78791227e-07  5.87598283e-06
  3.10471097e-05 -6.82431530e-04 -1.22718028e-03  8.77854615e-02
 -8.76922199e-01  4.91937135e+00 -1.82882893e+01  4.75029867e+01
 -8.74511393e+01  1.13106455e+02 -9.95100541e+01  5.53557696e+01
 -1.59345493e+01  1.82303713e-01  1.00005583e+00]
```

Для решения этой проблемы рассмотрим следующий вариант решения. Рассмотрим задачу.

$$\hat{L}_i(\omega) = L_i(\omega) + \frac{\lambda}{2}||\omega||^2 = L_i(\omega) + \frac{\lambda}{2}\sum_{j=1}^{n}\omega_j^2 \to \min_{\omega}\hat{L}_i(\omega), \qquad (3.26)$$

Важно заметить, что в данную сумму ω_0 не входить. Например, в процессе аппроксимации множество точек линейной функцией график с учетом этого коэффициента ω_0 имеет определенное значение. В случае ее входа в (3.26) нам давала бы неправильную аппроксимацию. Вот правильная аппроксимация.

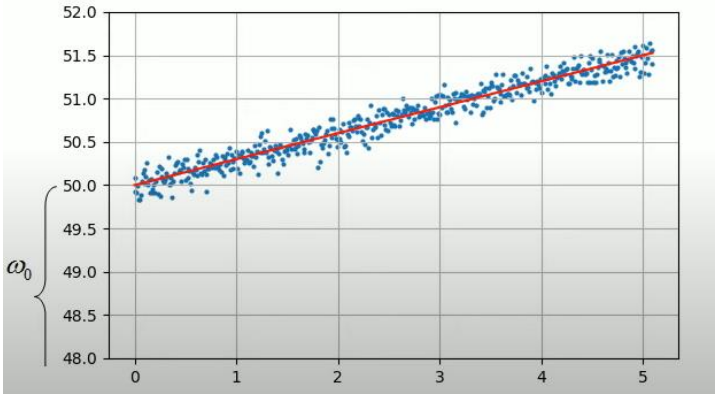


Рис.3.20. График аппроксимации модели линейной функцией с участием BIAS

Вычислим градиент данного функционала. Получим цепочку равенств.

$$\nabla\hat{L}_i(\omega) = \nabla L_i(\omega) + \lambda\omega\ , \qquad (3.27)$$

Учитывая, (3.27) имеем

$$\omega^{(t+1)} = \omega^{(t)} - \eta\nabla\hat{L}_i\left(\omega^{(t)}\right), \quad (3.28)$$

$$\omega^{(t+1)} = \omega^{(t)} - \eta(\nabla L_i(\omega^{(t)}) + \lambda\omega^{(t)})$$

$$\omega^{(t+1)} = \omega^{(t)}(1 - \eta\lambda) - \eta(\nabla L_i(\omega^{(t)}), \quad (3.29)$$

Здесь мы видим, что с каждой итерацией $\omega^{(t)}$ уменьшается на величину $(1 - \eta\lambda)$ и она будет стремиться к нулю, не давая весовым коэффициентам стремиться к + и – бесконечности. Таким образом, выбирая коэффициент затухания λ, мы можем регулировать наши весовые коэффициенты. Данный метод получил название метод регуляризации или гребневой регрессией. В случае, когда функционал $L_i(\omega)$ имеет вид:

$$L_i(\omega) = (a(x_i, \omega) - y_i)^2, \text{ где } a(x, \omega) = \langle\omega, x\rangle = \omega^T x. \quad (3.30)$$

задача регуляризации решается следующим образом. Строится функционалы

$$\hat{L}_i(\omega) = (\omega^T x_i - y_i)^2 + \frac{\lambda}{2}\omega^T\omega$$

$$Q(a, X^l) = \frac{1}{2}(X\omega - Y)^T(X\omega - Y) + \frac{\lambda}{2}\omega^T\omega, \text{ где}$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_l \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{l1} & x_{l2} & \dots & x_{ln} \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_l \end{pmatrix}$$

Из условия минимума функционала Q получим точку минимума ω_*:

$$\omega_* = (X^T X + \lambda I)^{-1} X^T Y, \quad (3.31)$$

Таким образом, добавление диагонального элемента гарантирует существования обратной матрицы, и весовые коэффициенты будут адекватными. Примером реализации могут, служит следующие графики. График между точным, до применения регуляризации, и график модели приближенной модели выглядят следующим образом.

```
[ 5.00000010e+02  3.00018437e+00 -9.99994391e+00  1.00152604e+00
  1.10210577e-03 -8.46005926e-05 -1.25511065e-04 -1.09688656e-05
 -1.40751996e-06 -9.98781808e-08  1.01012370e-09 -1.34466516e-10]
```

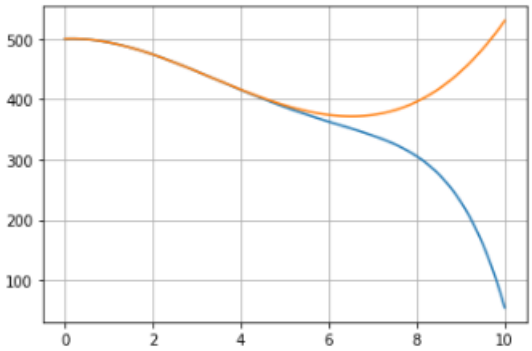


Рис.3.21. Обучение модели до применения метода регуляризации.

Как видим, присутствует переобучение модели. При использовании метода регуляризации имеем следующий результат.

```
[ 4.97627884e+02 -6.45936650e-01 -1.05785321e+00 -1.26016919e+00
 -9.36213848e-01 -4.63575138e-02  3.90347522e-01 -1.60627053e-01
  3.03083443e-02 -3.06331974e-03  1.61140721e-04 -3.47596060e-06]
```

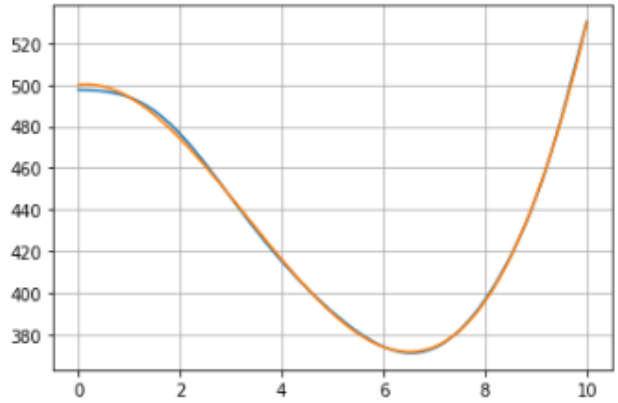


Рис.3.22. Обучение модели с применения метода регуляризации

В данном случае переобучение отсутствует. При этом размер признакового пространства не изменилось. В обоих случаях полином имеет степень n=12, только разница в применении метода регуляризации. Добавлением малой величины $\frac{\lambda}{2}\omega^T\omega$ мы добились эффекта регуляризации. Теперь рассмотрим некоторые конкретные примеры задач сельского хозяйства. Функционал качества перепишем в виде L_2 ргегуляризатора.

$$\hat{L}_i(\omega) = L_i(\omega) + \frac{\lambda}{2}||\omega||^2 = L_i(\omega) + \frac{\lambda}{2}\sum_{j=1}^{n}\omega_j^2, \qquad (3.32)$$

Мы видели, что с применением и без применения технологии регуляризации мы получаем требуемое решение задачи минимизации (3.32). Теперь давайте рассмотрим L_1 регуляризацию. Выпишем сам функционал качества. Для задачи минимизации функционала:

$$\hat{L}_i(\omega) = L_i(\omega) + \lambda||\omega|| = L_i(\omega) + \lambda \sum_{i=1}^{n} |\omega_i| \to min, \quad (3.33)$$

которая, не имеет аналитического решения, необходимо чтобы существовало первая производная от функции $\hat{L}_i(\omega)$. Однако, функция |x|, не имеет производную в точке 0, будем использовать численные методы. Для применения градиентного метода выпишем градиент

$$\nabla\hat{L}_i(\omega) = \nabla L_i(\omega) + \lambda sign(\omega). \quad (3.34)$$

Знаковая функция определяется как

$$sign(x) = \begin{cases} +1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$$

Будем использовать стохастический градиентный спуск

$$\omega^{(t+1)} = \omega^{(t)} - \eta\nabla L_i(\omega^{(t)}) - \lambda\, sign(x)\,. \quad (3.35)$$

Рассмотрим пример, изучим задачу классификации больных растений. Будем считать растение здоровая, если целевая функция принимает +1, и -1 в противном случае. Рассмотрим множество данных

$$x_i = [\omega_i, h_i]^T \text{ , y={-1; +1}, i=1,m} \quad (3.36)$$

Для реализации метода регуляризации L_1, расширим обучающее множество данных, перепишем ее в виде:

$$x_i = [\omega_i, h_i, 10\omega_i, 10h_i, 5(\omega_i + h_i)]^T, a(x) = sign(\langle\omega, x\rangle), \quad (3.37)$$

В (3.37) специально выберем 3-ю и 4-ю набор данных как, линейно зависимые от 1-ой и 2-ой данных. Построим функционал

$$Q(a, X^l) = \sum_{i=1}^{l} [\, a(x_i) \neq y_i] \leq \sum_{i=1}^{l} L_i(\omega, x_i), \quad (3.38)$$

$$\frac{d}{d\omega}\mathrm{L}\,(\omega, x_i, y_i) = \frac{2}{(1+e^{x_i y_i \omega^T})^2}\, e^{x_i y_i \omega^T} y_i {x_i}^T, \quad (3.39)$$

Для реализации данного метода для (3.37) - (3.39) выпишем градиентный метод

$$\omega^{(t+1)} = \omega^{(t)} - \eta \frac{d}{d\omega}\mathrm{L}\,(\omega, x_i, y_i) - \lambda\, sign(x)\,, \quad (3.40)$$

Реализация алгоритма сходимости градиентного метода.

```
[ 2.89867809e-05  4.68383794e-05  6.48998678e-02 -3.41616162e-02
  6.09125802e-04]
0.033978430565527566
```

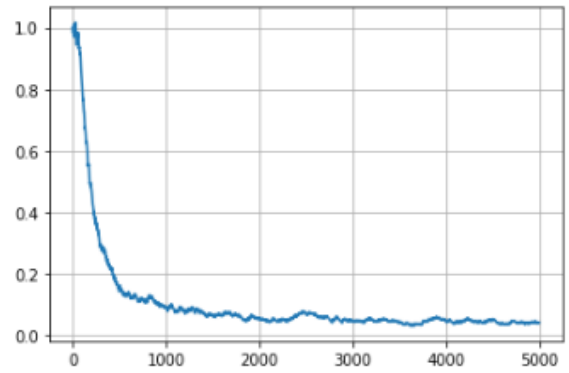


Рис.3.23. Сходимость метода регуляризации в L_1

Их результатов расчета можно заметить следующее. Всего коэффициентов 5, первые два почти равно 0, 3 и 4 коэффициенты означают, что они наиболее значимые. Последний коэффициент тоже не значимо при моделировании. Таким образом, метод регуляризации нам выдал результаты с нужными весовыми коэффициентами. В случае, когда не используется L_1, регуляризирующий оператор результат будет следующим:

```
[0.00121008 0.00080543 0.01210077 0.00805435 0.01007756]
0.9995188080164328
```

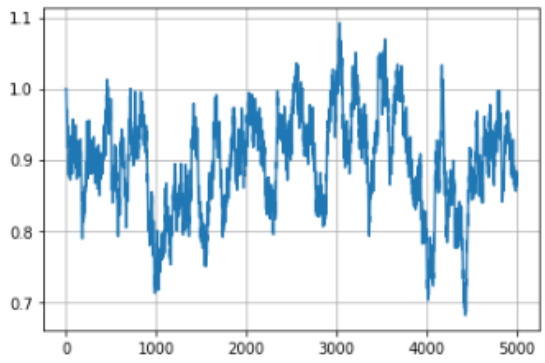


Рис.3.24. График сходимости, когда не используется метод регуляризации в L_1

В данном случае мы видим, что наш алгоритм не сходится. Регуляризация L_1 или LASSO. Ниже приведена векторная запись функционала стоимости со штрафной нормой в L_1:

$$\Omega(X,\vec{y},\vec{\omega})= L(X,\vec{y},\vec{\omega}) +\lambda R(\vec{\omega}) \Rightarrow min, \quad (3.41)$$

где, λ параметр регуляризации, функцию $L(X,\vec{y},\vec{\omega})$ определим как разность в развернутом виде:

$$L(X,\vec{y},\vec{\omega}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - (\omega_0 + \omega_1 X_1 + \omega_2 X_{2+\cdots+}\omega_N X_N))^2, \quad (3.42)$$

Функцию $R(\omega)$ в (2.43) возьмем в виде

$$R(\vec{\omega})=\sum_{i=0}^{N}|\omega_i|, \quad (3.43)$$

Задачу (3.41), с учетом (3.42) - (3.43) перепишем в виде

$$\Omega(X,\vec{y},\vec{\omega})= \frac{1}{N}\sum_{i=1}^{N}(Y_i - (\omega_0 + \omega_1 X_1 + \omega_2 X_{2+\cdots+}\omega_N X_{1N}))^2 + \lambda\sum_{i=0}^{N}|\omega_i| \Rightarrow min \quad (3.44)$$

где λ - называется коэффициентом регуляризации.

К сожалению, задача в этом виде не имеет общего аналитического решения, но мы можем воспользоваться методом градиентного спуска для поиска оптимального значения параметров модели выпишем градиент:

$$\frac{\partial\Omega(X,\vec{y},\vec{\omega})}{\partial\omega_j} = \frac{1}{N}\sum_{i=o}^{N}(\vec{x}_i^T\vec{\omega} - y_i)\vec{x_i}+\lambda \text{sign}(\vec{\omega}), \quad (3.45)$$

Формулу для итеративного обновления весов с применением градиентного спуска, можно записать в виде:

$$\vec{\omega}_{new} = \vec{\omega}-\alpha\frac{\partial}{\partial\vec{\omega}}\Omega(X,\vec{y},\vec{\omega}), \quad (3.46)$$

где α – это скорость обучения или размер градиентного шага. Таким образом, функционал Лассо решает задачу минимизации функционала со штрафной функцией по методу наименьших квадратов с нормой $\lambda||\omega||_1$.

Ниже представлен результат прогноза урожайности y в тоннах картофеля в зависимости от обработанной площади в x га с применением данной технологии.

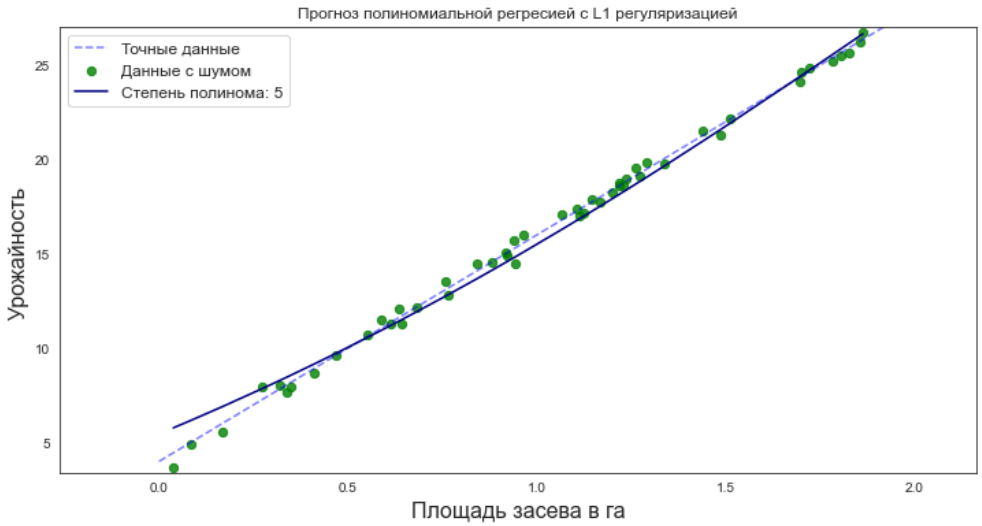


Рис.3.25. Результат применения технологии регуляризации L_1 для прогнозирования урожайности картофеля

Выбор параметров регуляризации в задачах регрессии Лассо играет существенную роль в построении линейной модели для практических задач. Рассмотрим следующие технологии с помощью перекрестной проверки или с использованием информационного критерия. Для этой цели изучим два важных критерия AIC или BIC. Определение AIC (и, следовательно, BIC) может отличаться в литературе. Рассмотрим математическое обоснование критерия выборов параметров регуляризации, вычисляемое с известным пакетом scikit-learn. Критерий AIC определяется как:

$$\text{AIC}= -2\log(\hat{L})+2*d$$

где $\hat{L}$ – максимальная вероятность модели и d - количество параметров, называемых степенями свободы. Определение BIC определяется заменой константы 2= log(N) в AIC:

$$\text{BIC} = -2\log(\hat{L})+\log(N)d, \text{ где N это количество образцов.}$$

Для линейной модели Гаусса максимальное логарифмическое правдоподобие определяется как:

$$\log(\hat{L})=-\frac{n}{2}\log(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{2*\sigma^2}, \quad (3.47)$$

где σ^2- оценка дисперсии шума y_i, а также соответственно $\widehat{y_i}$ - точная истинная и прогнозируемая цели, и n - количество образцов.

Подстановка максимального логарифмического правдоподобия в формулу AIC дает:

$$\text{AIC}=\text{n}\log(2\pi\sigma^2)+\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sigma^2}+2\text{d}, \qquad (3.48)$$

Первое слагаемое в (3.48) иногда отбрасывают, так как он является константой. Однако в строгом смысле оно эквивалентно только с точностью до некоторой постоянной и мультипликативного множителя.

Наконец, мы упоминали выше, что σ^2, является оценкой дисперсии шума. Если в Лассо-Ларс параметр noise_variance не указан по умолчанию, дисперсия шума оценивается с помощью несмещенного оценщика, определяемого как:

$$\sigma^2=\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{n-p}. \qquad (3.49)$$

где p - это количество признаков и $\hat{y}_i$ является прогнозируемой целью с использованием обычной регрессии наименьших квадратов. Обратите внимание, что эта формула справедлива только при N=n_samples > n_features

Сравнение с параметром регуляризации SVM. Эквивалентность между alpha и параметром регуляризации SVM C задается или, в зависимости от оценщика и точной целевой функции, оптимизированной моделью:

alpha = 1 / C или alpha = 1 / (n_samples * C).

Теперь рассмотрим практическое реализацию метода регуляризации. В качестве набора данных выберем, следующие данные по урожайности.

Листинг 3.10. Загрузка данных урожайности

```
df = pd.read_csv('data_harvest_potato.csv')
```

```
df.head()
```

	N	P	K	temperature	humidity	ph	rainfall	harvest
0	87.061966	59.225440	56.501270	15.438161	61.188624	7.111946	309.758114	19.227750
1	87.673701	60.270827	67.905067	16.943861	56.056799	7.124829	317.056731	19.003378
2	88.548707	58.155658	62.924899	17.008231	56.348194	7.602639	320.557945	18.777324
3	89.332462	58.205528	67.702136	15.228555	59.279004	8.177541	310.698694	18.914045
4	90.966950	59.743297	56.636310	15.661283	55.958781	7.581785	316.473301	18.010575

Кроме того, мы добавляем некоторые случайные числа к исходным данным, чтобы лучше проиллюстрировать выбор признаков, выполненный моделью Лассо. После этого набор данных выглядит таким образом

Листинг 3.11. Модель Лассо с шумовыми наборами данных

	N	temperature	rainfall	random_02	random_02	random_02	random_02	random_05	random_05	random_05	...	random_05	random_05	rando
0	87.061966	15.438161	309.758114	0.647689	0.647689	0.647689	0.647689	-0.234137	-0.234137	-0.234137	...	-0.234137	-0.234137	0.6
1	87.673701	16.943861	317.056731	-0.469474	-0.469474	-0.469474	0.542560	-0.465730	-0.465730	-0.465730	...	-0.465730	0.241962	-0.4
2	88.548707	17.008231	320.557945	-1.724918	-1.724918	-1.724918	-1.012831	0.314247	0.314247	0.314247	...	0.314247	-1.412304	-1.7
3	89.332462	15.228555	310.698694	1.465649	1.465649	1.465649	-1.424748	-1.424748	-1.424748	-1.424748	...	-1.424748	-1.150994	1.4
4	90.966950	15.661283	316.473301	-1.150994	-1.150994	-1.150994	-0.601707	-0.291694	-0.291694	-0.291694	...	-0.291694	-1.057711	-1.1

5 rows × 35 columns

Лассо предоставляет средство оценки Лассо, которое использует информационный критерий AIC или информационный критерий Байеса (BIC) для выбора оптимального значения параметра регуляризации альфа. Перед обучения модели мы стандартизируем данные с помощью файла StandardScaler. Кроме того, мы измерим время, необходимое для подбора и настройки гиперпараметра альфа, чтобы сравнить его со стратегией перекрестной проверки. Сначала мы обучим модель Лассо к критерию AIC. Мы сохраняем метрику AIC для каждого значения альфы, используемого во время обучения. Точно так же проведем тот же анализ с использованием критерия BIC. Мы можем проверить, какое значение alpha приводит к минимальным значениям AIC и BIC. Приведем результат реализации обеих критериев.

Листинг 3.12 . Виды критериев в методе Лассо.

alphas	AIC criterion	BIC criterion
0.188247	24380.468009	24380.468009
0.163081	24376.304000	24382.826976
0.156916	24375.264921	24388.310874
0.147805	24363.191369	24382.760299
0.111416	24313.337164	24339.429070
0.065361	24252.045404	**24284.660286**
0.062549	24251.097704	24290.235563
0.051763	24240.843414	24286.504249
0.044624	24235.602521	24287.786332
0.042079	24235.117291	24293.824079
0.033493	24229.306798	24294.536563
0.010457	**24218.286259**	24290.039000
0.008427	24219.763081	24298.038798
0.000000	24220.736872	24305.535566

Теперь, мы можем построить значения AIC и BIC для различных значений альфа. Вертикальные линии на графике соответствуют альфе, выбранной для каждого критерия. Выбранная альфа соответствует минимуму критерия AIC или BIC. Вот график реализации.

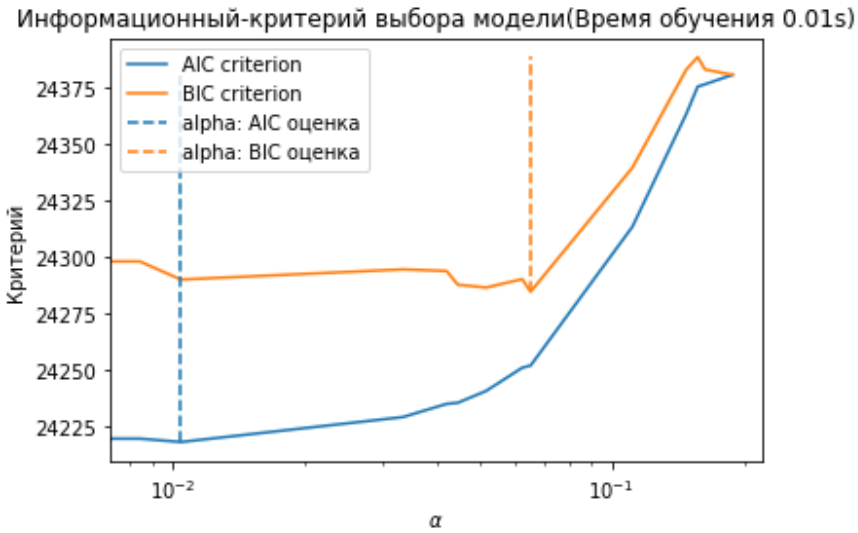


Рис.3.26. Выбор параметров регуляризации для модели урожайности картофеля по критериям AIC и BIC

Настройка гиперпараметров через координатный спуск.

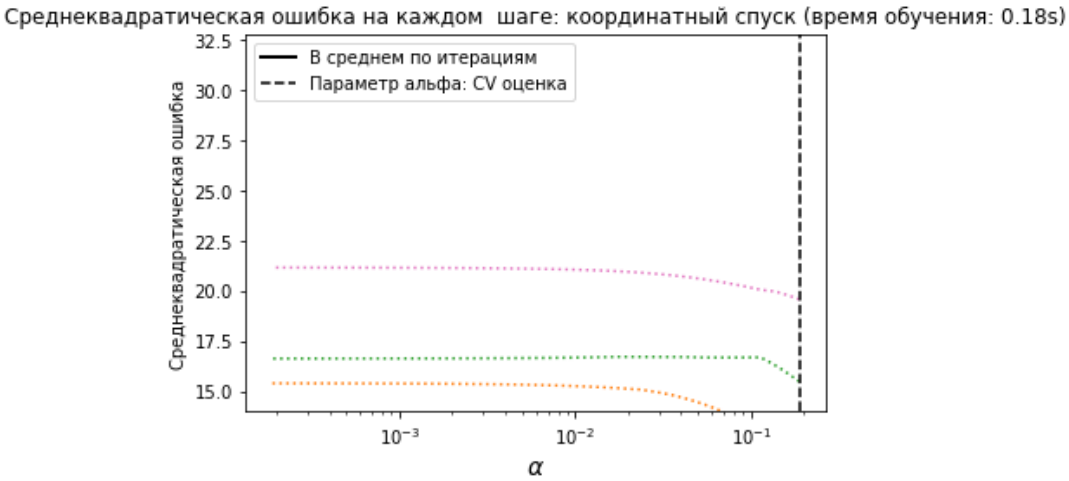


Рис.3.27. Выбор параметров регуляризации для модели урожайности картофеля по критериям AIC и BIC

Подбор модели по информационному критерию происходит очень быстро. Он основан на вычислении критерия для набора в выборке, предоставленного fit. Оба критерия оценивают ошибку обобщения модели на основе ошибки обучающей выборки и штрафуют эту чрезмерно оптимистичную ошибку.

Однако этот штраф зависит от правильной оценки степеней свободы и дисперсии шума. Оба получены для больших выборок (асимптотические результаты) и предполагают, что модель верна, т. е. что данные фактически генерируются этой моделью. Таким образом мы изучили два подхода к выбору лучшего гиперпараметра alpha в методе регуляризации Лассо.

Регуляризация L_2 или гребень. Опишем еще одну из важную технологию, регуляризации L_2, известной как Тихонова-Риджа регуляризации. При этом штрафной член, добавляемый к функции стоимости L(X,$\vec{y}, \vec{\omega}$), представляет собой сумму квадратов значений коэффициентов. В отличие от регуляризации LASSO, термин Тихонова-Риджа использует квадраты значений коэффициента и может уменьшить значение коэффициента почти до 0. Ниже приведено функционал стоимости после добавления штрафа в L_2. Сформулируем задачу.

$$\Omega(\mathrm{X},\vec{y},\vec{\omega})=\frac{1}{N}\sum_{i=1}^{N}\left(Y_i-(\omega_0+\omega_1X_1+\omega_2X_{2+\cdots+}\omega_NX_{1N})\right)^2+ \\ +\lambda\sum_{i=0}^{N}|\omega_i|^2\Rightarrow min, \qquad (3.50)$$

здесь λ – как и выше коэффициент регуляризации. Для удобства (21) можно переписать в виде

$$\Omega(\mathrm{X},\vec{y},\vec{\omega})=||X\omega-y||^2+\lambda||\omega||_2^2\Rightarrow min, \qquad (3.51)$$

При реализации норму для задачи Риджа данной задачи представим в векторной форме. Имеем

$$||\vec{\omega}||_2^2=\frac{1}{2}\vec{\omega}^T\vec{\omega}\,, \qquad (3.52)$$

Целевую функцию (3.51) с условием (3.52) перепишем в векторной форме (3.53):

$$\mathrm{L}(\mathrm{X},\vec{y},\vec{w})=\frac{1}{2}(\vec{y}-X\vec{\omega})^T(\vec{y}-X\vec{\omega})+\frac{\lambda}{2}\vec{\omega}^T\vec{\omega}, \qquad (3.53)$$

Приравняем к нулю производную функционала L(X,$\vec{y}, \vec{\omega}$) по весовым коэффициентам ω:

$$\frac{\partial L}{\partial\vec{\omega}}=\frac{\partial}{\partial\vec{\omega}}\left(\frac{1}{2}(\vec{y}-X\vec{\omega})^T(\vec{y}-X\vec{\omega})+\frac{\lambda}{2}\vec{\omega}^T\vec{\omega}\right)=0, \qquad (3.54)$$

Из (3.54) найдем решение искомой регуляризованной задачи для определения параметров модели $\vec{\omega}$ в виде (3.55):

$$\vec{\omega}=(X^TX+\lambda E)^{-1}X^T\vec{y}\,, \qquad (3.55)$$

Такой метод регрессии называется алгоритмом Тихонова-Риджа. Параметр сложности $\lambda > 0$ контролирует величину стабилизирующего слагаемого: чем больше значение λ, тем больше величина стабилизирующего слагаемого и, следовательно, коэффициенты становятся более устойчивыми к коллениарности.

При увеличении параметра регуляризации, матрица $X^T X + \lambda E$ становится "менее сингулярной", т.е. регулярной, а задача становится более определенной. Для такой матрицы число обусловленности будет равно: $(e^{max}+\lambda)/ (e^{min}+\lambda)$, e^x где — это собственные числа матрицы. Таким образом, увеличивая параметр регуляризации мы, уменьшаем число обусловленности. Прогнозирование влияния количества вносимых пестицидов с применением полиномиальной регрессии на почвенные характеристики с применением регуляризации Тихонова-Риджа, выглядит следующим образом

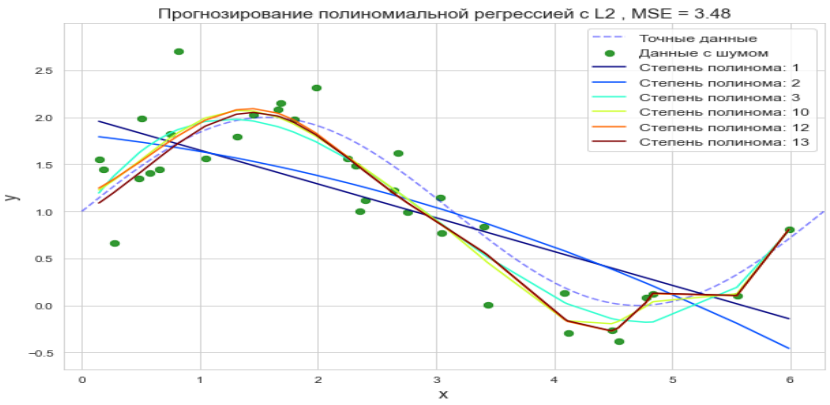


Рис.3.28. Результат применения технологии регуляризации для прогнозирования влияние пестицидов на урожайность картофеля

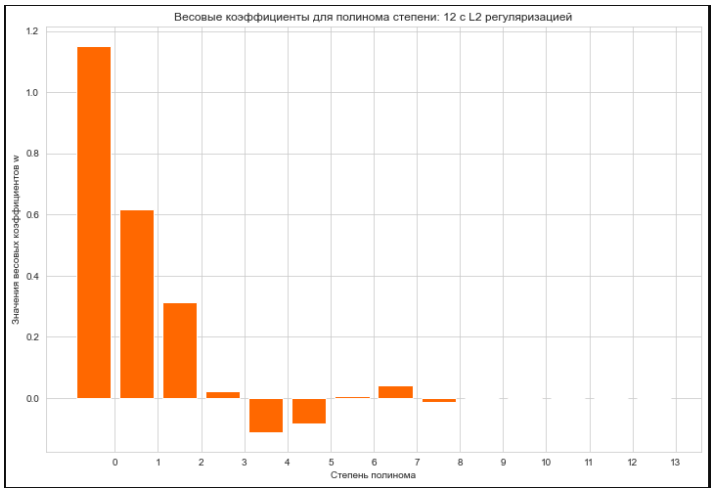


Рис.3.29. Значения весовых коэффициентов полиномов высокой степени после применения L_2 регуляризации

Ниже приведен метод регуляризации в L_1. (Приложение 2.11)

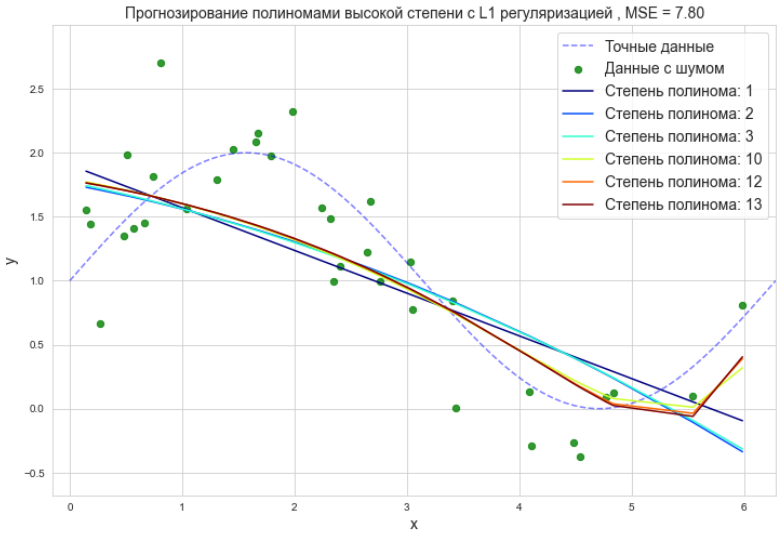


Рис.3.30. Значения весовых коэффициентов полиномов высокой степени после применения L_1 регуляризации

Вот коэффициенты регуляризации.

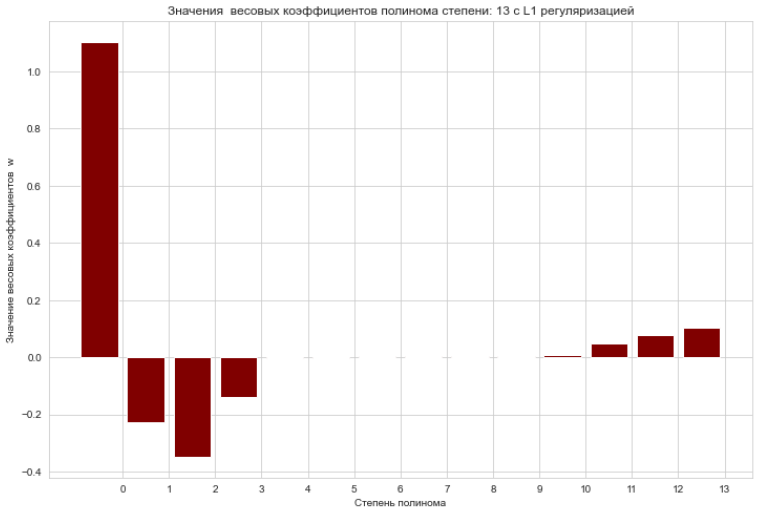


Рис.3.31. Диаграмма значения весовых коэффициентов полиномов высокой степени после применения L_1 регуляризации.

Из рисунка видно, что многие коэффициенты регуляризации ряда имеет нулевые коэффициенты, что приводит к разреженной матрице.

3.4. Численные методы прогнозирования урожайности на основе градиентного спуска

Исследуется сельскохозяйственный проект по применению численных методов к задачам прогнозирования урожайности, которая является наукоемкой и самой важной категорией по обеспечению продовольственной программы любой страны или региона. Рассматривается урожайность сельскохозяйственных продукций по различным регионам Кыргызской республики. Реализовывать данную задачу будем численно. Среди численных методов градиентный спуск представляет собой самый общий алгоритм оптимизации, способный находить оптимальные решения широкого диапазона прикладных задач. Основная идея градиентного спуска заключается в том, чтобы итеративно подстраивать параметры для сведения к минимуму функции издержек, например урожайности.

С этой целью изучим различные варианты градиентного спуска для численной реализации задач сельского хозяйства. Рассмотрим сначала точные методы в виде нормального уравнения и реализуем алгоритм нахождения точного решения, которое мы вывели в второй главе. Весовые коэффициенты оправляются по известной нам формуле

$$\vec{\omega} = (X^T X)^{-1} X^T \vec{y}$$

Для реализации загружаем базу данных и соответствующее программное обеспечение. Вот база данных.

Листинг 3. 13. Просмотр базы данных

```
# Загружаем сначала обучающее множество
data = pd.read_csv('train_data_yield.csv')
data
```

	id_fermer	yield	sqear	pestid_spring	pestid_summer	watering	pruning	view	abnormal_days	sunny_days	type3_grade
0	7739100155	750000	1.75	2850	11860	1.0	0	0	3	9	2850
1	6099400053	145000	1.00	1010	5490	1.0	0	0	3	6	1010
2	7468900270	140000	1.00	1090	10114	1.0	0	0	4	7	1090
3	396100025	339999	2.00	1740	6369	1.0	0	0	5	6	870
4	1160000255	311000	1.00	1120	8631	1.0	0	0	3	7	1120
...	...	...	...	...	...	...	...	...	...	...	...
18228	5272200045	378000	1.50	1000	6914	1.0	0	0	3	7	1000
18229	9578500790	399950	2.50	3087	5002	2.0	0	0	3	8	3087
18230	7202350480	575000	2.50	2120	4780	2.0	0	0	3	7	2120
18231	1723049033	245000	0.75	380	15000	1.0	0	0	3	5	380
18232	6147650280	315000	2.50	3130	5999	2.0	0	0	3	7	3130

18233 rows × 20 columns

База данных содержит объемную информации об урожайности с учетом множества признаков. В базе данных рассматриваются информации

урожайности, площади засева, весеннее и летняя обработка растений с пестицидами, количество полива, подвержен ли листья к болезням, количество аномально жарких и солнечных дней в месяц, механическая обработка почвы. Прогнозируется урожайность наиболее стратегических сельхозпродукций картофеля, ячменя, кукурузы и кормы для животных сенаж. Обучаем линейную модель по нашим данным X_train и y_train:

Листинг 3. 14. Обучение линейной регрессией

```
%%time
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_val)
```

```
CPU times: total: 15.6 ms
Wall time: 22 ms
```

и в результате мы получаем прогнозные значения y_pred_lin полученное по нормальному уравнению, которая дает следующий результат точности:

Листинг 3. 15. Точность обучение модели линейной регрессией

```
lin_r2 = r2_score(y_val, y_pred_lin)
lin_r2
```

```
0.7225927050639922
```

Визуализация результатов расчета:

Рис.3.32. Регрессионная связь целевой переменной урожайности с одной из переменных признаков

Отметим, что прогнозные значения записаны в виде массива Numpy. Далее мы будем использовать численные методы. Рассмотрим варианты градиентного спуска, алгоритмы, которых у нас подробно рассмотрены в Главе 2. Переменная $\vec{\theta}$ итеративно минимизирует функцию потерь J($\vec{\theta}$), начиная с начального случайного значения (скажем, $\vec{\theta} = \vec{0}$) и обновление значений параметров с использованием определенного шага итерации. Обновленную точку можно получить, выбрав направление наибольшего спуска, потому что цель состоит в том, чтобы достичь минимума как можно быстрее. Недостаток состоит в том, что для некоторой сложной функции J(θ) с несколькими минимумами она может сходиться к локальной минимум J(θ), т. Е. начало итерации из другой начальной точки может привести к другому минимальному значению. На каждой итерации значение θ обновляется по следующему правилу

$$\theta^{n+1}=\theta^{n} - \frac{\partial}{\partial\theta_i}\mathrm{J}(\theta^{n}) , \tag{3.56}$$

где α — размер шага или скорость обучения. Если α слишком мало, алгоритм минимизации для сходимости требуется много времени, если α слишком велико, алгоритм может расходиться. Для реализации подберем параметры $\alpha = 0{,}005$, а эпоху равную 500. Ниже приведены расчеты, проведенные по пакетному градиентному спуску BatchGDLinearRegression.

Листинг 3. 16. Результаты прогноза градиентным спуском

```
y_pred_bgd
```

```
array([[910411.61115297],
       [539437.35427848],
       [434889.19957279],
       ...,
       [671241.77477595],
       [500289.17106246],
       [268540.28190973]])
```

Визуализация результатов расчета

Рис.3.33. Применения пакетного градиентного спуска к задаче регрессии

Ниже приведены результаты реализация стохастического градиентного спуска- StochasticGDLinearRegression, со скоростью обучения eta=0.005 и эпохой 20.Вот результат точности и визуализации и процесс обучения.

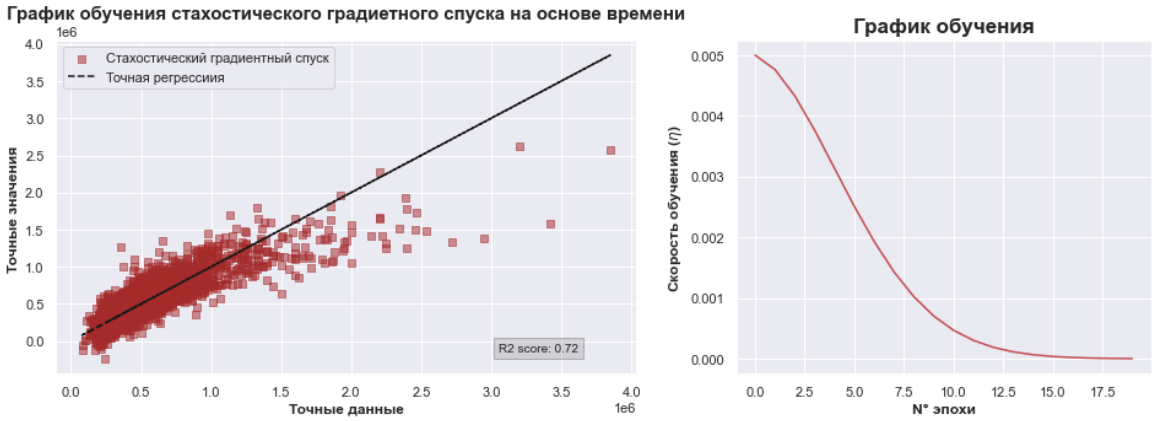


Рис.3.34. Применения стохастического градиентного спуска к задаче регрессии. Обучение метода

Теперь реализуем данную задачу с применением одной из часто применяемых численных подходов в прикладных задачах мини пакетный градиентный спуск. Вот результаты.

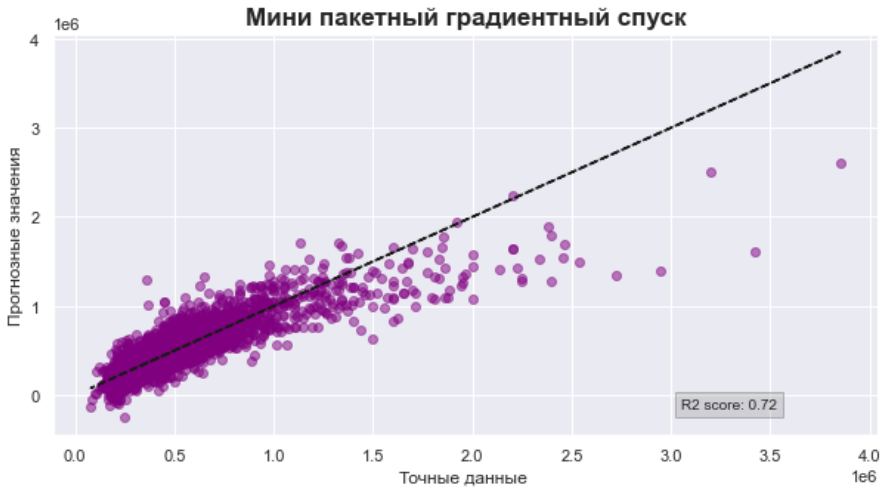


Рис.3.35. Мини пакетный градиентного спуска к задаче регрессии

Все варианты рассмотренные выше зависят от подбора параметров и размера эпохи. Ниже приведены сходимости этих методов, для нашей задачи прогнозирования.

Рис.3.36. Сходимости градиентного спуска

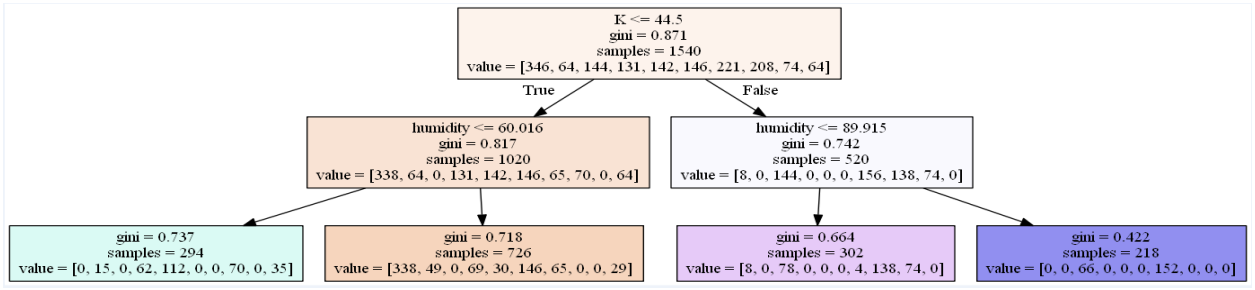
3.5. Ансамблевые методы прогнозирования задач сельского хозяйства с применением машинного обучения

Во многих случаях методы ансамбля т.е. одновременное использование нескольких алгоритмов машинного обучения дает очень хорошие результаты. Одним из важных составляющих является дерево алгоритм решений. Рассмотрим пример в работе построения модели с применением данного алгоритма модели на базе данного алгоритма. Рассмотрим многоклассовую задачу классификации. В данном случае для расширения изучения задачи

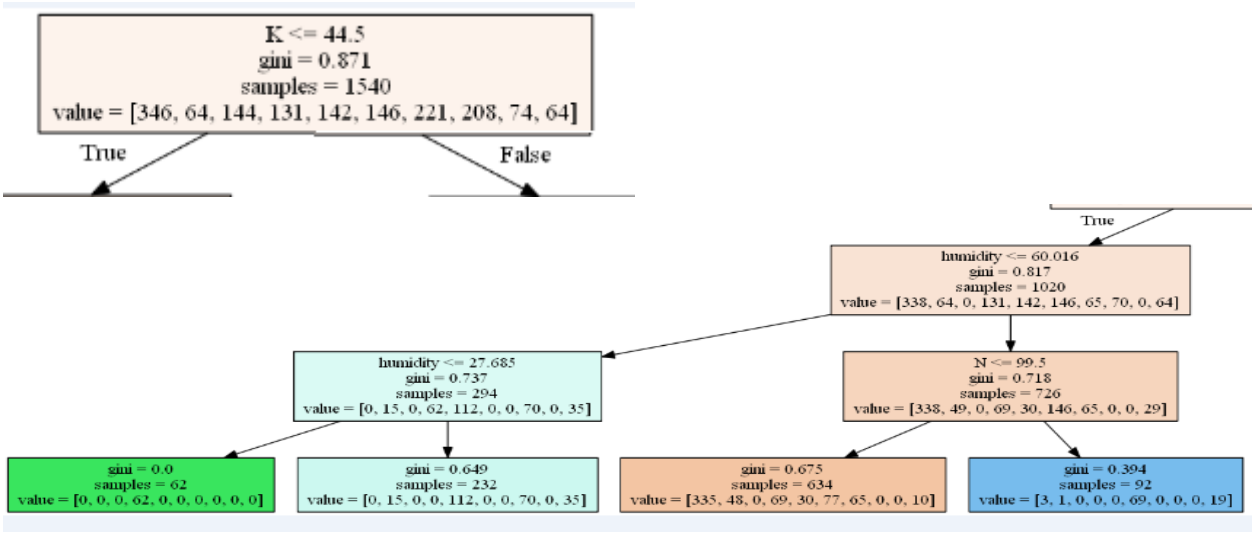
урожайности для нескольких растений используется предыдущая преобразованная база данных с учетом введенных ниже меток

label = {'potato' : 1, 'maize' : 2, 'barley':3, 'black currant':4, 'apricot':5, 'alfalfa':6, 'apple':7, 'pear':8, 'cherry':9, 'corn':10}

Применение алгоритма дерево решений, построенной в виде модели с древовидной структурой является сложной по природе. Результат, модель является сильно нелинейной сложной функцией для различных глубин дерева, в случаях подбора параметров max_depth=2 и max_depth=3 дает следующий результат соответственно рис.3.37.



a)



b)

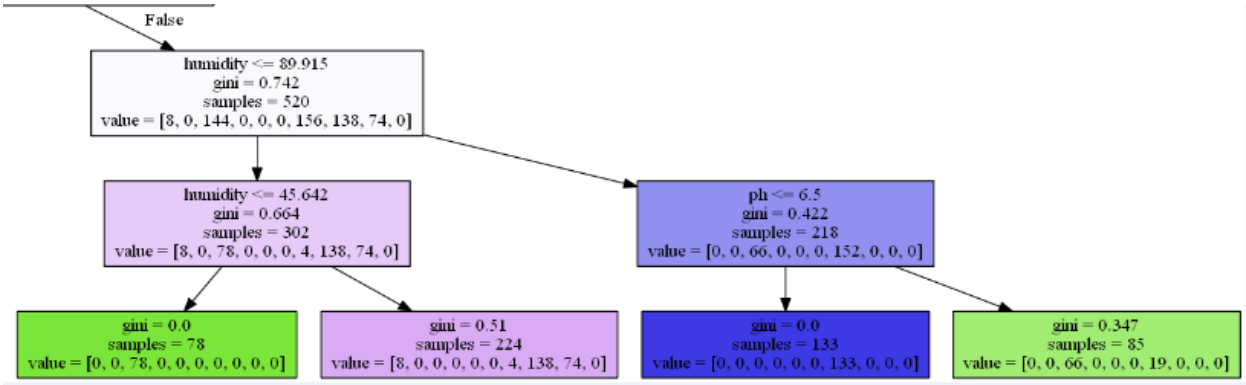


Рис.3.37. Графическое представление модели с помощью дерево решений при a)max_depth=2 и b) max_depth=3 соответственно

Данную технологию можно распространить на сложную задачу регрессию. Ниже представлена задача регрессии с применением дерево решений. В качестве примера приведем задачу прогнозирования удержания влаги почвой в зависимости от количества дней.

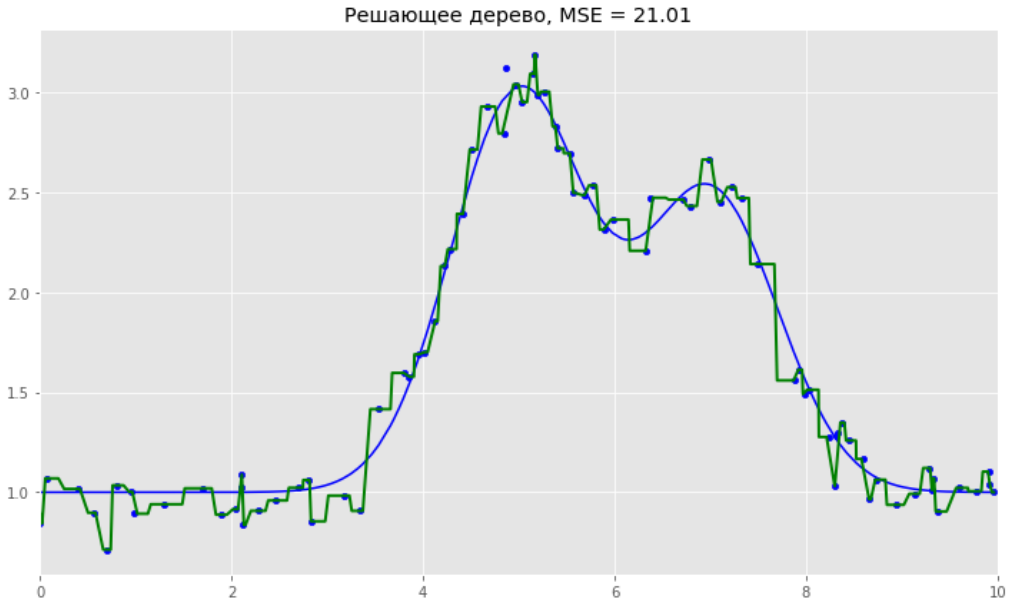


Рис.3.38. Графическое представление модели с помощью дерево решений

Мы выше уже вывели уравнение и расчетные данные свободного члена и коэффициентов при исследовании множественной линейной регрессии. Для исследуемой базы данных уравнение множественной линейной регрессии урожайности от независимых признаков удобрений, температуры, влажности, кислотности почвы и осадки с обновленными данными на основании полученных коэффициентов имеет вид:

$$\text{Harvest} = 0.31813690723082944 + 0.08049244*N - 0.11737906*P + 0.01466049*K + 0.10570846*\text{temperature} - 0.06511367*\text{humidity} - 0.01802083*\text{ph} + 0.01505539*\text{rainfall}, \quad (3.57)$$

Уравнение (3.57), представляет собой, простейшее регрессионное уравнение – модель урожайности. Рассчитаем теперь оценки модели прогноза множественной регрессии в различных метриках.

Mean Absolute Error=0.116981, Mean Square Error=0.020748,
Root Mean Square Error=0.144041, (3.58)

Теперь приведем результаты применение более продвинутых алгоритмов машинного обучения. Ниже приведены результаты прогнозирования, в виде Таблица 3.2, которые получены с помощью следующих основных четырех алгоритмов с регулировкой параметров.

Таблица 3.2. - Производительность основных алгоритмов машинного обучения

№	Estimates/ML Algorithm	R2score	MAE	RMSE
1	Lasso Regression -all parametrs	0.0351341	0.7237154	0.8865015
2	Lasso Regression-selected parametrs	0.9980028	0.0393173	0.0403324
3	SVR –all parametrs	0.1411921	0.6776377	0.6776377
4	SVR -selected parametrs	0.9958232	0.05250301	0.0583266
5	Random Forest Regression –selected parametrs	0.9999996	0.0003116	0.0005596
6	Random Forest Regression –all parametrs	0.0750289	0.6847263	0.8679807
7	Gradient Descent Algorithm–all parametrs	0.0047446	0.7329265	0.9003539
8	Gradient Descent Algorithm – selected parametrs	0.3576148	0.5885566	0.7233421

Из данных Таблица 3.2, мы видим, что все MAE близки к нулю. В идеале мы должны иметь MAE равно, нулю. Некоторые из результатов прогнозирования для нашей базы данных, приведенные в Таблица 3.2 и на рисунках рис.3.37 и рис.3.38. Результаты и сравнительный анализ производительности моделей для урожайности картофеля, показали что, точность градиентным спуском ниже чем другие алгоритмы. В расчетах приведенное ниже для переобученных моделей везде использовано различные технологии машинного обучения. Ниже приведены результаты применения алгоритмов машинного обучения, как составляющие некоторого ансамбля, для исследования регрессионных моделей, а также их оценки точности по прогнозу урожайности. Полученные результаты регрессии урожайности с применением алгоритмов машинного обучения в виде

Таблица 3.3 и в виде визуализации расчетов урожайности показаны на рисунках (Рис.3.39, Рис.3.40). Из

Таблица 3.3 видно, что алгоритмы градиентного бустинга с MAPE =10.14% и случайный лес с MAPE =10.19% дают хорошие результаты. Лидером прогноза оказалось метод опорных векторов с 10.12%.

Таблица 3.3. - Результаты оценок моделей, полученных алгоритмами машинного обучения

№	Estimates/ML Algorithm	R^2	MAE	MSE	RMSE	MAX	MAPE in %
1	Linear Regression	0,01	2	8	3	13	11,06
2	Decision Tree Regression	-0,61	3	12	4	16	13,49
3	Stochostic Gradient Descent Regression	0	2	8	3	13	11,12
4	K – Nearest Neighbour 5	0,03	2	8	3	12	10,58
5	SVR	0,10	2	9	3	13	10,12
6	Gradient Boosting Regression	0,12	2	7	3	12	10,14
7	Random Forest Regression	0,11	2	7	12	12	10,19

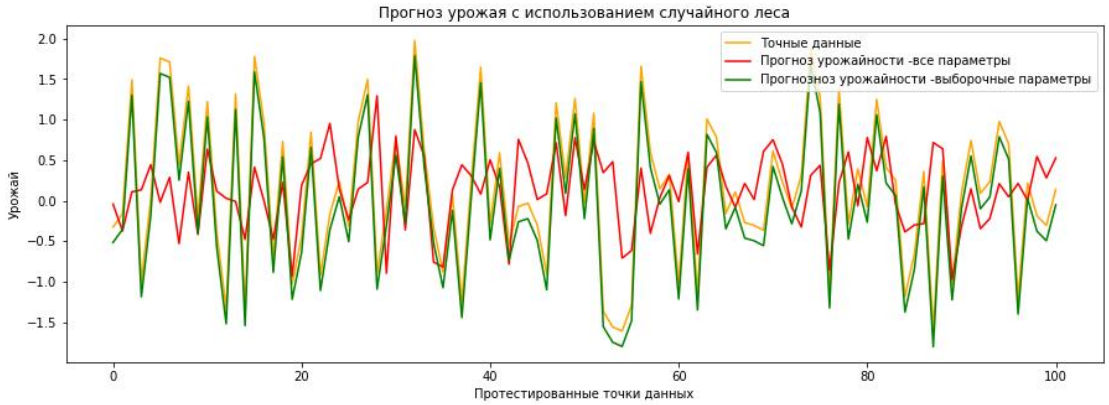


Рис.3.39. Анализ производительности алгоритма случайный лес для прогноза урожайности

Рис.3.40. Производительности алгоритма метода Лассо для прогноза урожайности

Существенную роль, во всех алгоритмах, для оценки точности модели играет выбор параметров алгоритмов. Рассмотрены случаи с выбором всех параметров и с частичным выбором параметров. Анализ полученных расчетов проведенная с применением алгоритма машинного обучения: метод опорных векторов, Лассо регрессия показали удовлетворительные результаты. Результаты прогнозирования урожайности показали, что наиболее точными прогноза являются алгоритмы Random Forest, Lasso Regression и SVR с выбранными коэффициентами. При использовании ансамблевых алгоритмов, или как ее называют ленивое прогнозирование Lazy Prediction для задач прогнозирования урожайности сельскохозяйственных культур с несколькими, составляющими ансамбля получены продвинутые результаты в виде рис.3.41. Здесь уместно отметить, что составляющие алгоритмы могут быть любыми. Ниже применяя данную библиотеку со специальным подбором параметров алгоритмов машинного обучения, мы вычислили производительность 10 самых мощных известных классификаций или моделей регрессии на нескольких матрицах производительности. Вычисления на ансамблевых алгоритмах для задач регрессии, которые приведены ниже рис.3.41. вычислены в Google Colaboratory.

```
 21%|██        | 9/42 [00:02<00:10,  3.12it/s]GammaRegressor model failed to execute
Some value(s) of y are out of the valid range for family GammaDistribution
 76%|███████   | 32/42 [00:19<00:06,  1.55it/s]PoissonRegressor model failed to execute
Some value(s) of y are out of the valid range for family PoissonDistribution
100%|██████████| 42/42 [00:26<00:00,  1.61it/s]
```

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
NuSVR	0.08	0.14	0.84	1.58
SVR	0.07	0.13	0.84	1.68
HistGradientBoostingRegressor	0.06	0.13	0.84	0.62
LGBMRegressor	0.06	0.13	0.84	0.20
RandomForestRegressor	0.03	0.10	0.86	3.61
MLPRegressor	0.02	0.09	0.86	5.50
BaggingRegressor	0.02	0.09	0.86	0.38
ExtraTreesRegressor	-0.01	0.06	0.87	1.35
HuberRegressor	-0.04	0.04	0.89	0.05
SGDRegressor	-0.04	0.04	0.89	0.03

Рис.3.41. Результаты применения ансамблевого метода к задачам прогнозирования урожайности

Вышеизложенные технологии построения моделей и прогнозирование на основе алгоритмов машинного обучения можно распространить для различных прикладных задач. Рассмотрим конкретные реальные задачи из сельского хозяйства. Изучим построение нелинейной модели урожайности сельскохозяйственных культур на базе алгоритма дерево решений.

Листинг 3.17. Структура базы данных по Иссык-Кульской области.

aiyl okmot	id fermer	use udobr	have technik	u yach	plosh yach	charge yach	call yach tech	u kart	plosh kart	charge kart	call kart man	call kart tech	u seno	charge seno	plosh seno	call man seno	call seno tech	ch
Ак-Шыйракский	1553	No	No	32.89	1.78	5.87	4	11.132111	2.349559	9.907052	5	4	325.040246	146.998972	2.960933	5	4	T
Ак-Шыйракский	1554	No	No	40.61	2.27	13.18	3	8.424757	2.904278	11.362902	5	3	190.529747	198.395243	2.108808	5	3	F
Ак-Шыйракский	1555	No	No	41.99	1.51	7.08	5	10.788050	2.504698	11.963904	3	5	321.546607	179.641636	2.284478	3	5	F
Ак-Шыйракский	1556	No	No	28.90	1.72	12.19	2	7.569101	1.875705	7.190054	4	2	281.450256	203.908994	2.964149	4	2	F
Ак-Шыйракский	1557	Yes	Yes	43.47	2.64	8.53	4	10.736328	2.537654	10.339240	5	4	203.986219	180.479539	2.586299	5	4	F

Информация о базе данных с числовыми характеристиками представлено в Листинге 2.9.

Листинг 3.18.Информация о базе данных с числовыми характеристиками

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1457 entries, 0 to 1456
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   u yach          1457 non-null   float64
 1   plosh yach      1457 non-null   float64
 2   charge yach     1457 non-null   float64
 3   call yach tech  1457 non-null   int64
 4   u kart          1457 non-null   float64
 5   plosh kart      1457 non-null   float64
 6   charge kart     1457 non-null   float64
 7   call kart man   1457 non-null   int64
 8   call kart tech  1457 non-null   int64
 9   u seno          1457 non-null   float64
 10  charge seno     1457 non-null   float64
 11  plosh seno      1457 non-null   float64
 12  call man seno   1457 non-null   int64
 13  call seno tech  1457 non-null   int64
 14  churn           1457 non-null   bool
dtypes: bool(1), float64(9), int64(5)
memory usage: 160.9 KB
```

Столбцы представляют u yach- урожай ячменя, plosh yach-площадь посева ячменя, charge yach-продажная цена ячменя, call yach tech-использование сельхозтехники. Аналогично для картофеля и кормовой травы. Переменная churn –означает будет ли урожайность выше среднего. Программная реализация основан на алгоритме дерево решений. При реализации сначала необходимо импортировать пакеты Python и дерево решений, также пакеты разделения данных на обучающие, валидации и тестирования. Полный код приложения приведена в Приложении 5. Результаты построения дерево решений в зависимости от глубины дерева выглядит следующим образом.

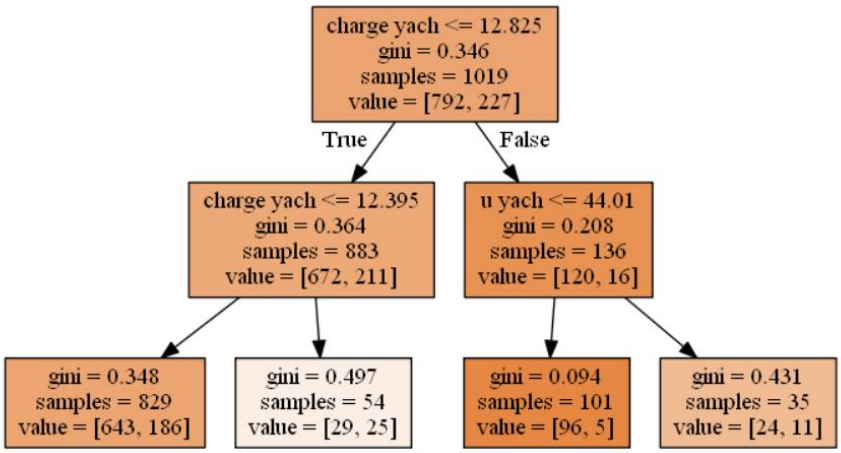


Рис.3.42. Дерево решений с глубиной max_depth=2

Дерево читается следующим образом. В начале было 1019 объектов, 792 - одного класса и 227 – другого. Энтропия начального состояния была 0.346. Затем было сделано разбиение объектов на 2 группы в зависимости от сравнения признака $x1$= charge_yach со значением 0.364 (найдите этот участок границы на

рисунке выше, до дерева). При этом энтропия в левой увеличилось, и в правой группе объектов уменьшилась до gini=0.208. Причем количество объектов в левом уменьшилось до 883 всего,672 одного класса 211 другого класса. И так далее, дерево строится до глубины 3,4,5. При такой визуализации чем больше объектов одного класса, тем цвет вершины ближе к темно-оранжевому и, наоборот, чем больше объектов второго класса, тем ближе цвет к темно-синему. В начале объектов одного класса поровну, поэтому корневая вершина дерева – обычно белого цвета. В принципе дерево решений можно построить до такой глубины, чтоб в каждом листе был ровно один объект. Но на практике это не делается из-за того, что такое дерево будет переобученным – оно слишком настроится на обучающую выборку и будет плохо работать на прогноз на новых данных. Где-то внизу дерева, на большой глубине будут появляться разбиения по менее важным признакам.

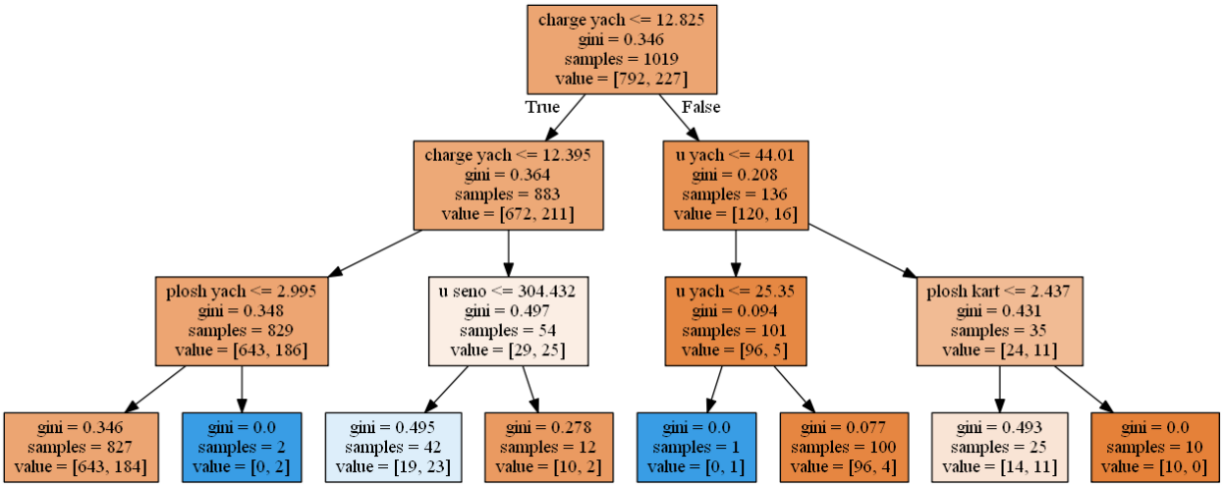


Рис.3.43. Дерево решений с глубиной max_depth=3

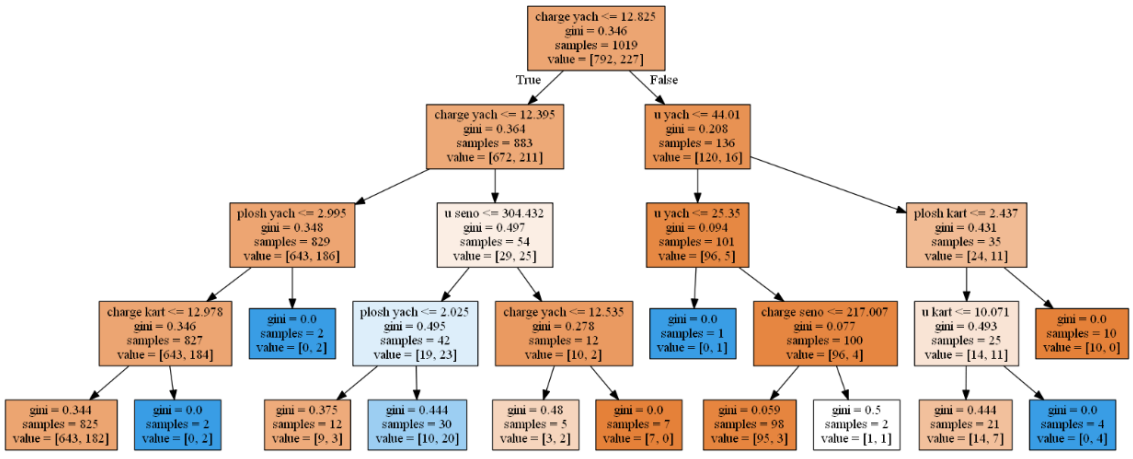


Рис.3.44.Дерево решений с глубиной max_depth=4

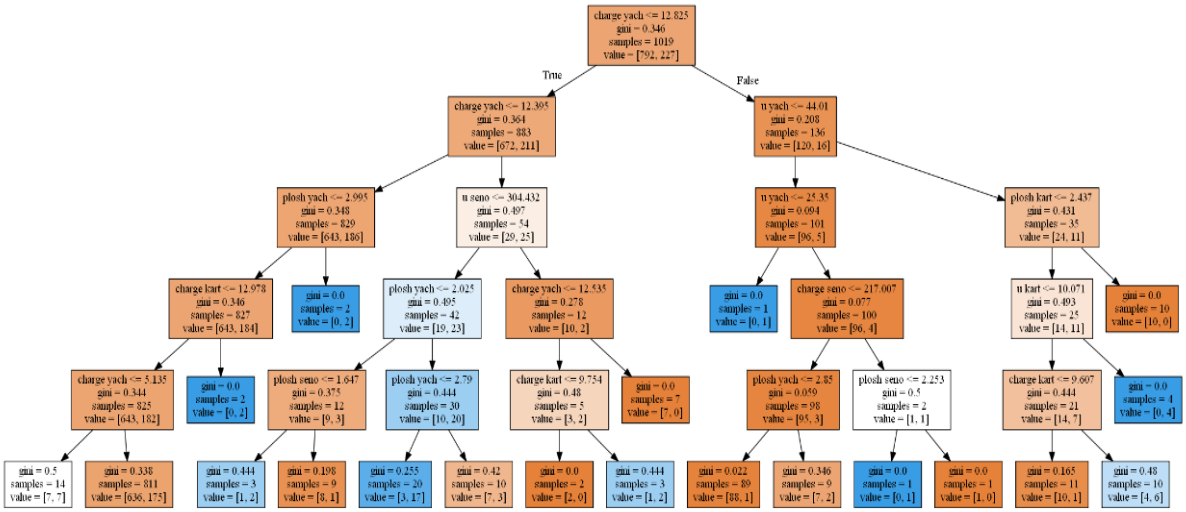


Рис.3.45. Дерево решений с глубиной max_depth=5

Метод K-ближайших соседей. Классификация ближайших соседей. Классификация на основе соседей — это тип обучения на основе экземпляров или не обобщающего обучения: он не пытается построить общую внутреннюю модель, а просто сохраняет экземпляры обучающих данных. Классификация вычисляется простым большинством голосов ближайших соседей каждой точки: точке запроса назначается класс данных, который имеет наибольшее количество представителей среди ближайших соседей точки.

В системе scikit-learn реализуется два разных классификатора ближайших соседей: Классификация метода реализует обучение на основе K-ближайших соседей каждой точки запроса, где k целочисленное значение, указанное пользователем. Алгоритм, также реализует обучение на основе количества соседей в фиксированном радиусе r каждой тренировочной точки, где r представляет собой значение с плавающей запятой, указанное пользователем.

Отметим, что задача классификация K-ближайших соседей, является наиболее часто используемым методом. Оптимальный выбор значения k сильно зависит от данных: как правило, более k подавляет влияние шума, но делает границы классификации менее четкими.

В случаях, когда выборка данных неравномерна, классификация соседей на основе радиуса может быть лучшим выбором. Пользователь указывает фиксированный радиус r, так что точки в более разреженных окрестностях используют меньше ближайших соседей для классификации. Для многомерных пространств параметров этот метод становится менее эффективным из-за так называемого «проклятия размерности».

Базовая классификация ближайших соседей использует единые веса: то есть значение, присвоенное точке запроса, вычисляется на основе простого большинства голосов ближайших соседей. В некоторых случаях лучше взвешивать соседей так, чтобы более близкие соседи вносили больший вклад в подгонку. Это можно сделать с помощью weights ключевого слова. Значение по умолчанию , присваивает одинаковые веса каждому соседу. присваивает веса, пропорциональные обратному расстоянию от точки запроса. В качестве альтернативы для вычисления весов может быть задана определяемая пользователем функция расстояния, weights = 'uniform' и weights = 'distance'

Пример использования классификации ближайших соседей для задачи

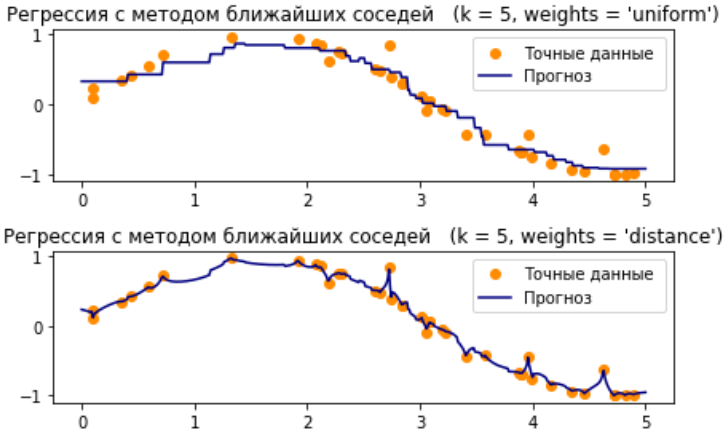


Рис.3.46. Результаты прогнозирования по методу ближайших соседей

Во многих случаях при численной реализации задачи прогнозировании урожайности с базой данных, которая содержит информацию, о составе почв по районам исследуемой области точность моделей намного улучшается. Добавим данные новые признаки в базу данных, где имеются почвенные данные разных культур, выращиваемые в сельском хозяйстве и их урожайность по годам. Ниже приведены результаты прогнозирования урожайности с выбором всех параметров базы данных и с частичной выборкой признаков базы данных. В каждом из этих случаев мы отдельно обучаем и прогнозируем с предварительно обработанными данными. Рассмотрим базу данных и загружаем его, используя DataFrame из библиотеки Pandas.

Листинг 3. 19. Загрузка базы данных на основе DataFrame

```
data = pd.DataFrame(pd.read_csv("Dataset_for_Prediction.csv"))
data.head()
```

	Year	Area	Production	Yield	Predominantly reddish medium texture	Red desert soils	Sandy loam	Soil are lithosolsat foot hills alluvials in plains	brown soils	clay loam	...
0	1997	56600	30400.0	0.537102	0	0	0	0	1	0	...
1	1997	105900	34600.0	0.326723	0	0	0	0	1	0	...
2	1997	24700	28900.0	1.170040	0	0	0	0	1	0	...
3	1997	36700	25400.0	0.692098	0	0	0	0	1	0	...
4	1997	79300	144500.0	1.822194	0	0	0	0	1	0	...

5 rows × 71 columns

Сначала о прогнозировании с выбором всех параметров. Предварительно нормируя данные и обучаем модель. Для точности модели мы вычисляем отклонения прогноза от точных данных по метрикам r2_score, mean_squared_error, mean_absolute_error.На тестовых данных точность градиентного спуска дает следующий результат.

pred_all= (array([[-0.03754773], [0.16217191], [0.18255649], [-0.10688531],
[0.11688714], [0.09693025], [0.05486058], [-0.09005193],
y_test= array([[-1.62125744e-01],
[2.88291569e-01], [2.13741156e-01], [-1.43945712e-01], [1.53665950e-01], [-3.52347898e-02], [-9.30681142e-02], [-1.08106470e-01], [1.51950269e-01], [2.02310720e-01]]

Листинг 3. 20. Оценка точности прогноза

```
auc_test = 100 - np.mean(np.abs(pred_all - y_test)*100)
auc_test
```

```
92.09301584177175
```

Оценка точности моделей по вышеизложенным метрикам, с учетом всех 71 параметров получены следующие оценки:

Листинг 3. 21. Оценка точности прогноза с учетом всех параметров

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

print("R2score:", r2_score(y_test,pred_all))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred_all)))
print('MAE:', mean_absolute_error(y_test, pred_all))
```

```
R2score: 0.7373477673958941
RMSE: 0.09606976094443122
MAE: 0.07906984158228256
```

Точно также оценка по выборочным коэффициентам, где учтено 42 параметров имеет следующий вид

Листинг 3. 22. Оценка точности прогноза с выборочными коэффициентами

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

print("R2score:", r2_score(y_test,pred_sel))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred_sel)))
print('MAE:', mean_absolute_error(y_test, pred_sel))
```

```
R2score: 0.6721446803351905
RMSE: 0.10733399124642685
MAE: 0.08833348923247232
```

Теперь приведем сравнительную диаграмму результатов прогнозирования с применением градиентного спуска.

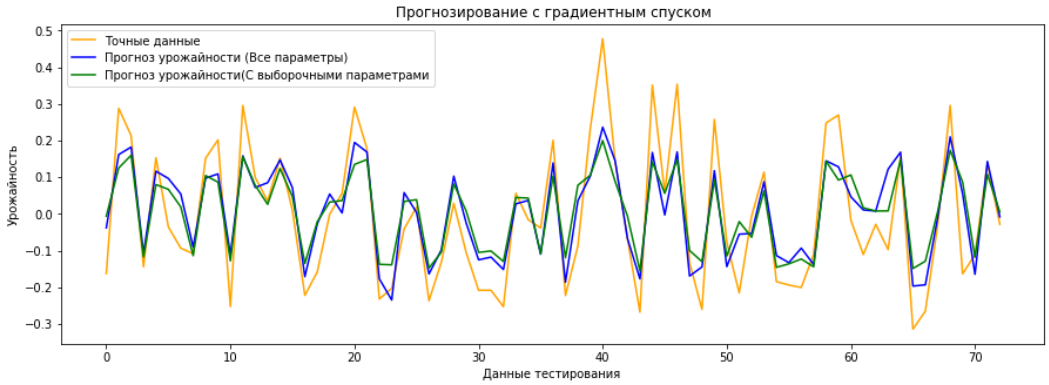


Рис.3.47. Результаты прогнозирования по градиентному спуску.

3.6. Обучение нелинейных регрессионных моделей задач урожайности с применением ансамблевых алгоритмов машинного обучения

Во второй главе мы описали алгоритмы машинного обучения случайный лес, метод опорных векторов и метод основанной на регуляризации переобученных моделей алгоритм Лассо. С помощью этих алгоритмов мы обучим и построим модели для прогнозирования задачи урожайности с базой данных с предыдущего раздела. Рассмотрим задачу регрессии. Программный пакет для алгоритма случайный лес, с учетом всех параметров из

sklearn.ensemble возьмем в виде RandomForestRegressor. Обучим модель с использованием данного регрессора на обучающих данных. Выбранное из базы данных. Сначала оценим точность модели в различных метриках. Как и в предыдущем случае, оценим MAE, которая определяется как среднее значение абсолютной разницы между прогнозируемыми значениями и истинными значениями, в нашем случае MAE: 0.02171092742447838. Следующий показатель MSE определяется как среднее значение квадратов ошибки. Он определяет качество модели прогнозирования и включает дисперсию -разброс прогнозируемых значений друг от друга. MSE почти всегда положительный, а значения ближе к нулю лучше. Чем ближе к нулю MSE, тем лучше. Наши результаты случайного леса RMSE:0.03210438555877871. Теперь рассмотрим R2score. В процентном отношении данный показатель 97.06%, что показывает высокий результат прогноза. Когда R2score*100%= 100% означает идеальную корреляцию. Вот листинг результатов.

Листинг 3. 23. Прогнозирование урожайности с помощью случайный лес и оценка ее точности

```
from sklearn.metrics import r2_score
r = r2_score(y_test,pred_selected)
print('Алгоритм случайный лес')
print("R2score: ",r)
print('RMSE:',np.sqrt(mean_squared_error(y_test, pred_selected)))
print('MAE:',mean_absolute_error(y_test, pred_selected))
```

```
C:\Users\Admi\AppData\Local\Temp\ipykernel_6916\1735190508.py:4:
array was expected. Please change the shape of y to (n_samples,),
  model.fit(x1_train,y_train)
```

```
Алгоритм случайный лес
R2score:  0.9668300027823048
RMSE: 0.034140421539900156
MAE: 0.022710501840947014
```

Ниже на рис.3.48 приведена сравнительная диаграмма производительности модели, построенная с помощью алгоритма случайный лес.

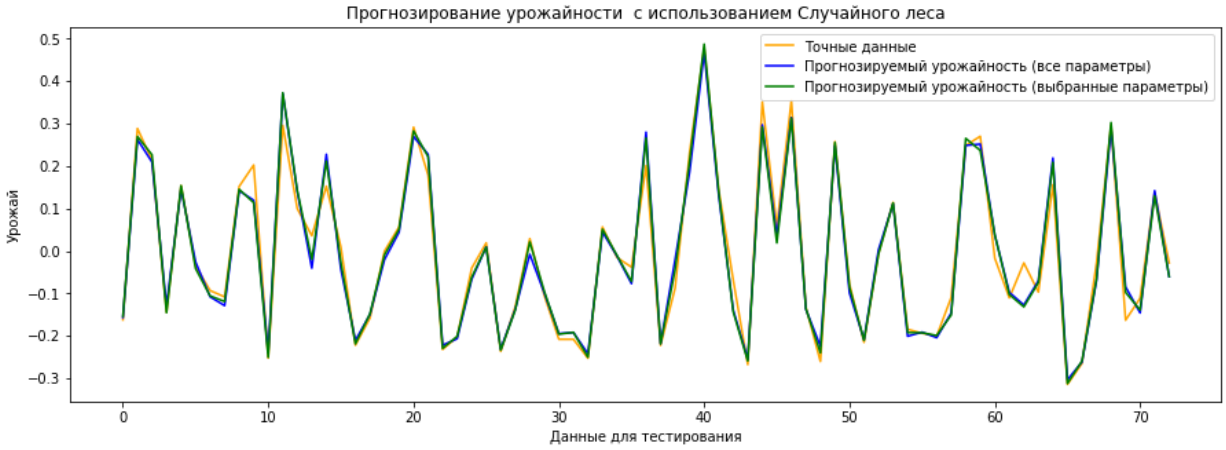


Рис.3.48. Результаты прогнозирования по методу случайный лес.

В методе опорных векторов получаем следующие результаты.

Листинг 3. 24. Прогнозирование урожайности методом опорных векторов

```
Метод опорных векторов с использованием всех параметров
R2 score: 0.8982433161230685
RMSE: 0.0597967574251987
MAE: 0.04785225303334516
```

```
Метод опорных векторов с использованием выбранных  параметров
R2 score: 0.9033044480088024
RMSE: 0.05829071870687769
MAE: 0.047442737811823434
```

Производительность данной модели показана ниже.

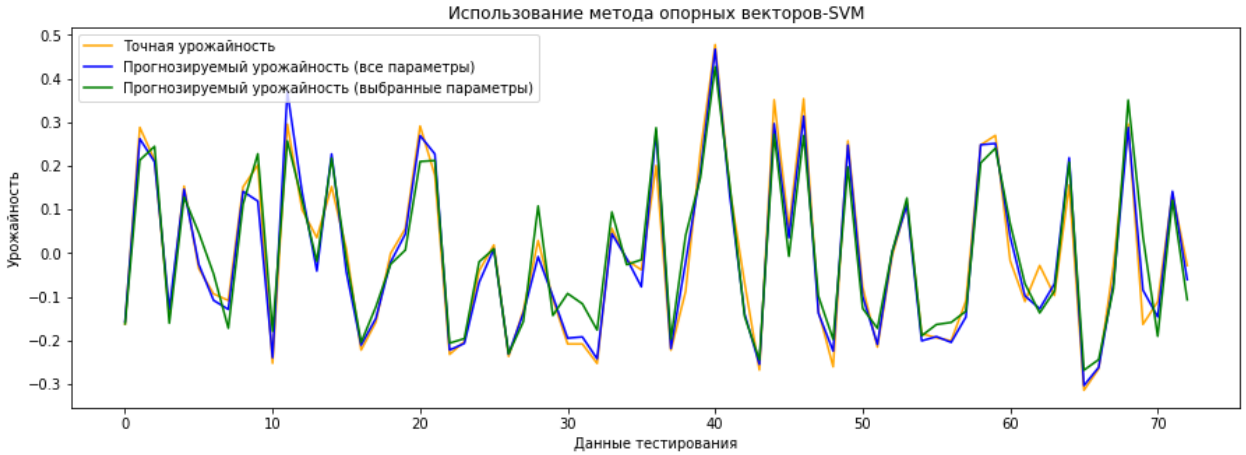


Рис.3.49. Результаты прогнозирования по методу опорных векторов.

Приведем результаты применения алгоритм метода регуляризации Лассо для прогнозирования урожайности.

Листинг 3. 25. Прогнозирование урожайности с помощью регуляризации Лассо и оценка ее точности

```
print('R2 score:',r2_score(y_test, pred_all))
print('RMSE:',np.sqrt(mean_squared_error(y_test,pred_all)))
print('MAE:',mean_absolute_error(y_test, pred_all))
```

```
Метод регуляризации с использованием всех параметров
R2 score: 0.8146824212217434
RMSE: 0.08069645755381687
MAE: 0.0588093331954562
```

```
print('R2 score:',r2_score(y_test, pred_sel))
print('RMSE:',np.sqrt(mean_squared_error(y_test,pred_sel)))
print('MAE:',mean_absolute_error(y_test, pred_sel))
```

```
Метод регуляризации с использованием выбранных  параметров
R2 score: 0.7929212370102475
RMSE: 0.08530292881366584
MAE: 0.06279053471454211
```

Вот результаты Рис.3.50. производительности данной модели.

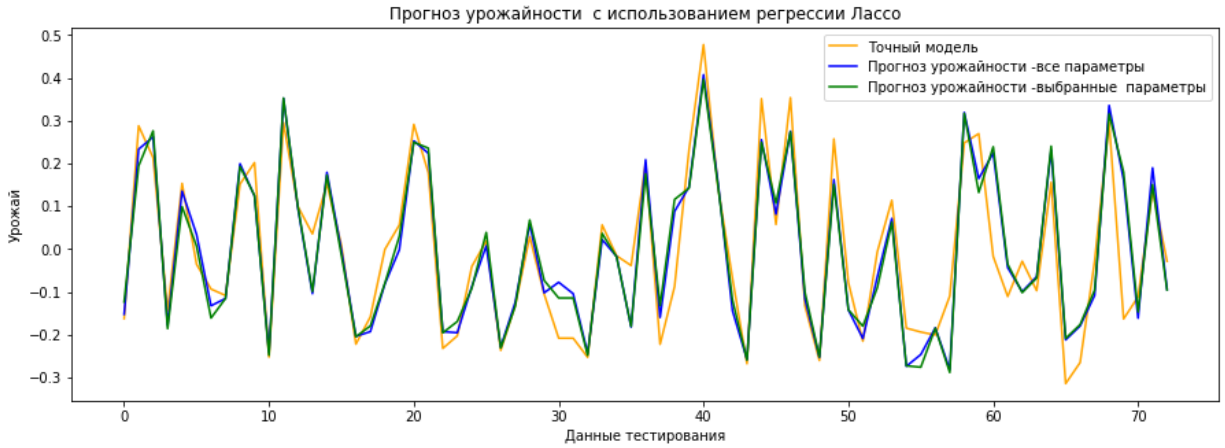


Рис.3.50. Прогнозирования урожайности по методу регуляризации Лассо.

Теперь мы расширим применимость методов машинного обучения. Будем прогнозировать регрессионную задачу. Построим регрессионные модели, которые наиболее часто используемые в задачах сельского хозяйства. Рассмотрим следующую базу данных урожайности. В качестве признаков в данном случае использованы данные о пестицидах и погодные условия. Данные категории являются наиболее влиятельными для задач прогнозирования урожайности многих сельскохозяйственных продукций. В данной базе данных также учтено кислотность почвы отдельных районов изучаемых районов. Сначала, как всегда, мы загружаем пакет программ для реализации алгоритмов машинного обучения. Далее для удобства работы с базой данных урожайности мы исходный файл в excel преобразуем в .csv файл. Вот структура базы данных.

Листинг 3. 26. Загрузка базы данных

```
harvest_data = pd.read_csv('data_harvest_potato.csv')
```

```
harvest_data.head()
```

	N	P	K	temperature	humidity	ph	rainfall	harvest
0	87.061966	59.225440	56.501270	15.438161	61.188624	7.111946	309.758114	19.227750
1	87.673701	60.270827	67.905067	16.943861	56.056799	7.124829	317.056731	19.003378
2	88.548707	58.155658	62.924899	17.008231	56.348194	7.602639	320.557945	18.777324
3	89.332462	58.205528	67.702136	15.228555	59.279004	8.177541	310.698694	18.914045
4	90.966950	59.743297	56.636310	15.661283	55.958781	7.581785	316.473301	18.010575

Количество фермеров с соответствующими информациями об урожайности следующая:

```
harvest_data.shape
```

```
(5029, 8)
```

Так как данные у нас разные мы их должны преобразовать т.е. мы должны их масштабировать. Для создания модели разделяем данные на зависимую переменную, в нашем случае y = data['harvest'] и независимые функции, которые поместим в датафрейме X = data.drop('harvest', axis=1) .Вот структура переменных:

```
y
```

```
0       19.227750
1       19.003378
2       18.777324
3       18.914045
4       18.010575
          ...
5024    17.438029
5025    17.051708
5026    18.206521
5027    17.130518
5028    18.715612
Name: harvest, Length: 5029, dtype: float64
```

В данном случае прогнозируемое переменная на масштабируется.

Листинг 3. 27. Структура базы данных

	N	P	K	temperature	humidity	ph	rainfall
0	0.047736	0.122804	0.043374	0.042807	0.510028	0.439772	0.701309
1	0.062071	0.173827	0.380745	0.191989	0.440386	0.443818	0.733472
2	0.082576	0.070589	0.233411	0.198367	0.444340	0.593904	0.748902
3	0.100942	0.073023	0.374742	0.022039	0.484113	0.774489	0.705454
4	0.139245	0.148079	0.047369	0.064913	0.439055	0.587354	0.730901
...	...	...	...	...	...	...	...
5024	0.906198	0.842958	0.473616	0.630713	0.284963	0.511651	0.680923
5025	0.711911	0.741401	0.545711	0.646516	0.206614	0.736948	0.632293
5026	0.777867	0.853547	0.508901	0.612536	0.260525	0.507466	0.664816
5027	0.972050	0.987633	0.669357	0.714027	0.244198	0.548100	0.630565
5028	0.722736	0.744485	0.548051	0.530775	0.225144	0.395424	0.617389

5029 rows × 7 columns

Разделяем данные на обучающие и проверочные в процентном соотношении 80%:20%. Для этой цели используем пакет train_test_split из системы sklearn.model_selection. Для оценки точности модели, т.е. разности y_true, y_pred пишем следующую функцию:

Листинг 3. 28. Функция определения точности прогнозирования

```
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Создадим регрессионные модели. В дальнейшем мы будем использовать обозначения для алгоритмов дерево решений и метод К-ближайших соседей dtr и knn соответственно:

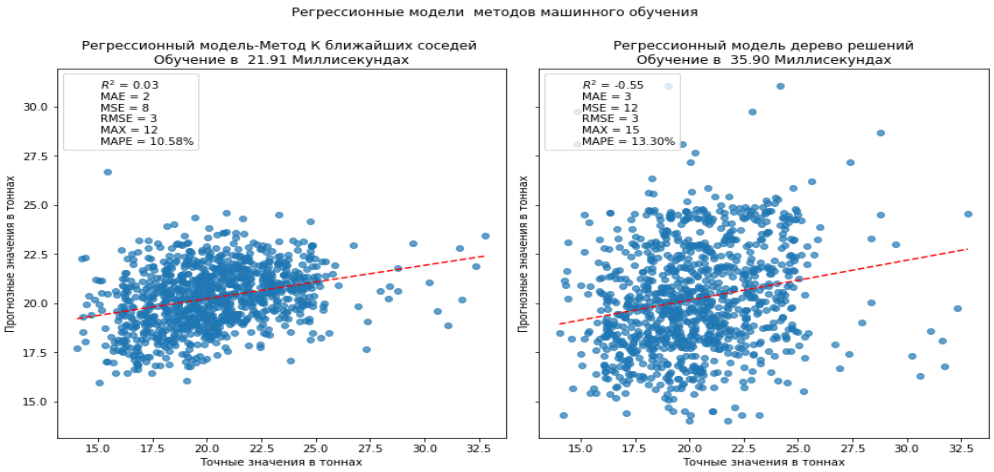
dtr = DecisionTreeRegressor()

knn = KNeighborsRegressor(n_neighbors=5)

Используемые алгоритмы:

estimators = [('Регрессионный модель-Метод К ближайших соседей ', knn),
('Регрессионный модель дерево решений', dtr)]

Точное решение определим из нормального уравнения. На Рис.3.51 указан красной линией.

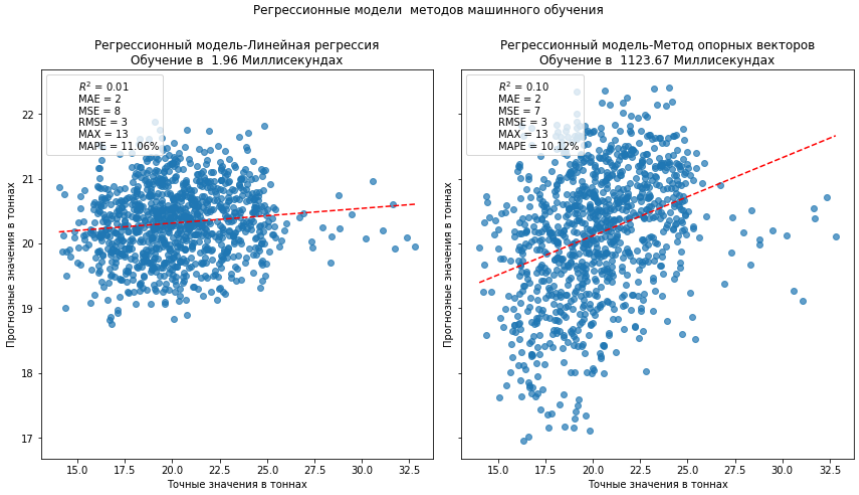


a) b)

Алгоритм	R2 Score	MAE	MSE	RMSE	MAPE
К ближайших соседей	0.03		3	2	10.58%
Дерево решений	0.55		3	3	13.30%

Рис.3.51. a) Регрессионные модели с применением машинного обучения к-ближайших соседей и дерево решений; b) Оценка точности моделей

lin = LinearRegression() svr=SVR()
Используемые регрессии:
estimators = [('Регрессионный модель-Линейная регрессия', lin), ('Регрессионный модель-Метод опорных векторов', svr)]

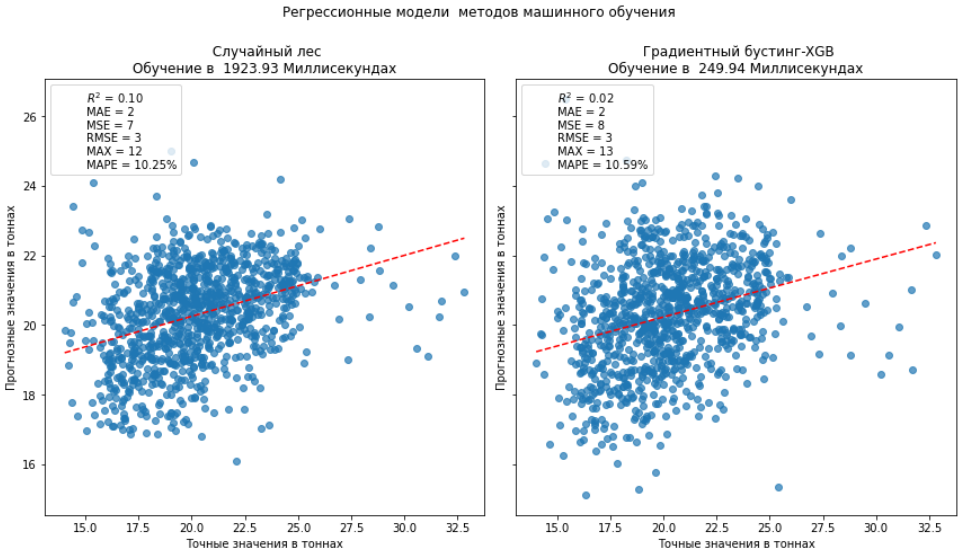


a) b)

Алгоритм	R2 Score	MAE	MSE	RMSE	MAPE
Линейная регрессия	0.01	2	8	3	11.06%
Метод опорных векторов	0.1	2	7	3	10.12%

Рис.3.52. a) Регрессионные модели с применением машинного обучения линейной регрессии и метода опорных вектров; b) Оценка точности моделей

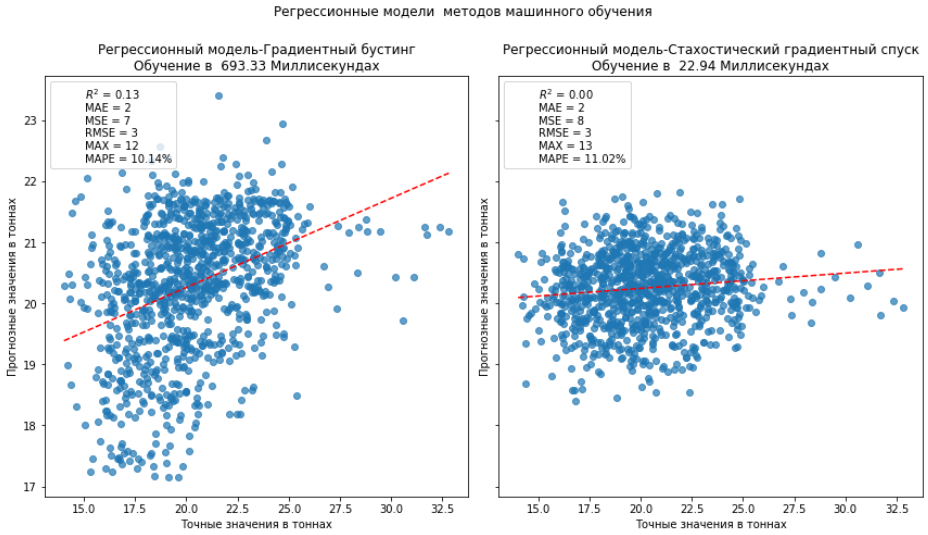
Алгоритм случайный лес и градиентный бустинг
rfr = RandomForestRegressor() xgb=XGBRegressor()
Используемые регрессии:
estimators = [(' Случайный лес', rfr),(' Градиентный бустинг-XGB', xgb)]



a) b)

Алгоритм	R2 Score	MAE	MSE	RMSE	MAPE
Случайный лес	0.01	2	8	3	11.06%
Градиентный бустинг XGB	0.1	2	7	3	10.12%

Рис.3.53. a) Регрессионные модели с применением машинного обучения случайный лес и градиентного бустинга -XGB; b) Оценка точности моделей



a) b)

Алгоритм	R2 Score	MAE	MSE	RMSE	MAPE
Градиентный бустинг	0.01	2	8	3	11.06%
Стохастический градиетный спуск	0.1	2	7	3	10.12%

Рис.3.54. a) Регрессионные модели с применением машинного обучения градиентного бустинга и стохастического градиентного спуска; b) Оценка точности моделей

3.7. Прогнозирование задач сельского хозяйства с учетом рисков

Во многих случаях урожайность зависит от различных рисков. Это продолжительность аномально жарких дней в период вегетации. Град – одна из разновидностей осадков, выпадающих из облаков, которая связано с грозой. Это самые частые случаи для наших земледельческих районов. Град наносить иногда большой ущерб урожаю и входить в рисковые зоны многих земледельческих районов нашей страны. Процесс этот может многократно повторяться в период

вегетационного периода растения с различной степенью тяжести. Риски могут быть связаны и с другими различными факторами, это может быть неправильное использования пестицидов, насекомые вредители растений, приносящие большие потери урожайности. В связи с этим рассмотрим следующую актуальную задачу повреждения урожайности. Это наиболее часто встречающаяся задача в сельском хозяйстве. Создадим модель с использованием различных алгоритмов продвинутых методов машинного обучения. Рассмотрим моделирование классификации и проверим их точность.

Основная наша задача, это определить исход урожая с учетом сезона, т.е. будет ли урожай здоровым (живым), поврежденным пестицидами или поврежденным другими причинами, которые мы указали выше.

Вот описание базы данных: будем использовать два набора данных для обучения и тестирования.

ID: UniqueID

Estimated_Insects_Count: Расчетное количество насекомых на квадратный метр

Crop_Type: Категория культуры (0,1)

Soil_Type: Категория почвы (0,1)

Pesticide_Use_Category: Тип использования пестицидов (1- никогда, 2-использовалось ранее, 3-в настоящее время Использование)

Number_Doses_Week: количество доз в неделю

Number_Weeks_Used: количество использованных недель

Number_Weeks_Quit: количество недель прекращения использования

Season: категория сезона (1,2,3)

Crop_Damage: категория повреждения урожая (0=живой, 1=повреждение по другим причинам, 2=Ущерб от пестицидов). Описания процесса моделирования.

Сначала импортируем библиотеки и набор данных урожайности.

Необходимо импортировать библиотек, таких как NumPy, pandas, matplotlib и seaborn.

Набора данных, как в других задачах заданы в формате CSV и конвертируем его в pandas DataFrame и проверяем несколько верхних строк для анализа данных.

Листинг 3. 29. Структура базы данных

	ID	Estimated_Insects_Count	Crop_Type	Soil_Type	Pesticide_Use_Category	Number_Doses_Week	Number_Weeks_Used	Number_Weeks_Quit
0	F00000001	188	1	0	1	0	0.0	0
1	F00000003	209	1	0	1	0	0.0	0
2	F00000004	257	1	0	1	0	0.0	0
3	F00000005	257	1	1	1	0	0.0	0
4	F00000006	342	1	0	1	0	0.0	0

(продолжение таблицы)

Season	Crop_Damage	source
1	0	train
2	1	train
2	1	train
2	1	train
2	1	train

Предварительная обработка набора данных состоит из следующих этапов.

1. Проверяем в наличие нулевых значений.

2. Проверка типов данных: Должны проверить типы данных всех столбцов, для обнаружения несоответствия в типах данных

3. Проверка уникальных значений. Теперь нам нужно понять уникальные значения, если они присутствуют в столбцах, что поможет уменьшить размерность при будущей обработке.

4. В случае присутствия нулевых значения, обычно их заменяют средними значениями столбцов, где она обнаружена.

Теперь мы проведем исследовательский анализ данных и получим общее представлении о данных. Для этой цели рассмотрим следующие различные свойства базы данных количественные показатели базы, уникальность признаков, названия признаков и общая длина записей в базе данных.

ID 88858

Estimated_Insects_Count 71

Crop_Type 2

[1 0]

Soil_Type 2

[0 1]

Pesticide_Use_Category 3

[1 3 2]

Number_Doses_Week 20

Number_Weeks_Used 65

Number_Weeks_Quit 51

Season 3

[1 2 3]
Crop_Damage 3
[0 1 2]
source 1
['train']

Листинг 3. 30. Уникальность признаков базы данных

```
df.nunique()
```

```
ID                         88858
Estimated_Insects_Count       71
Crop_Type                      2
Soil_Type                      2
Pesticide_Use_Category         3
Number_Doses_Week             20
Number_Weeks_Used             64
Number_Weeks_Quit             51
Season                         3
Crop_Damage                    3
source                         1
dtype: int64
```

```
df.columns
```

```
Index(['ID', 'Estimated_Insects_Count', 'Crop_Type', 'Soil_Type',
       'Pesticide_Use_Category', 'Number_Doses_Week', 'Number_Weeks_Used',
       'Number_Weeks_Quit', 'Season', 'Crop_Damage', 'source'],
      dtype='object')
```

```
df.shape
```

```
(88858, 11)
```

Вот информация о типах данных.

Листинг 3. 31. Информация о базе данных

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88858 entries, 0 to 88857
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       88858 non-null  object
 1   Estimated_Insects_Count  88858 non-null  int64
 2   Crop_Type                88858 non-null  int64
 3   Soil_Type                88858 non-null  int64
 4   Pesticide_Use_Category   88858 non-null  int64
 5   Number_Doses_Week        88858 non-null  int64
 6   Number_Weeks_Used        88858 non-null  float64
 7   Number_Weeks_Quit        88858 non-null  int64
 8   Season                   88858 non-null  int64
 9   Crop_Damage              88858 non-null  int64
 10  source                   88858 non-null  object
dtypes: float64(1), int64(8), object(2)
memory usage: 7.5+ MB
```

Часто очень полезно знание о связи независимых функций с прогнозируемой функцией в виде корреляции. Выведем корреляционную матрицу.

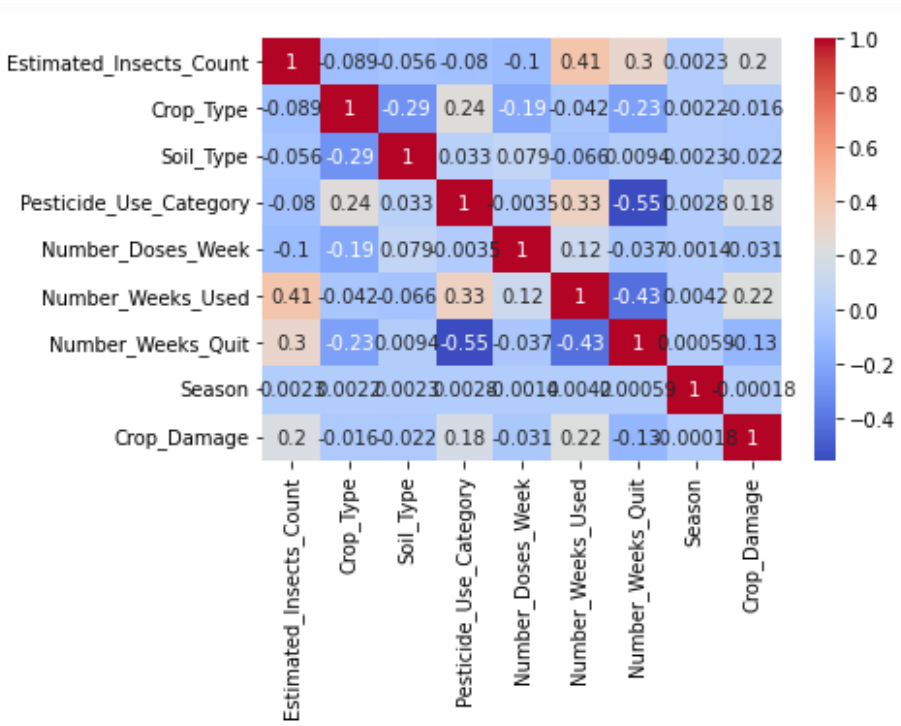


Рис.3.55. Корреляционная связь между признаками базы данных.

Выводы, сделанные из тепловой карты, следующие:

1.Estimated_Insects_count, Pesticide_use_category и Number_weeks_used положительно коррелируют с повреждением урожая.

2. Число_использованных_недель положительно коррелирует с Расчетным_числом_насекомых и категорией_использования_пестицидов.

3. Number_weeks_Quit сильно отрицательно коррелирует с Pesticide_use_category и Number_weeks_used.

Теперь анализ данных. Поврежденный урожай, различного типа.

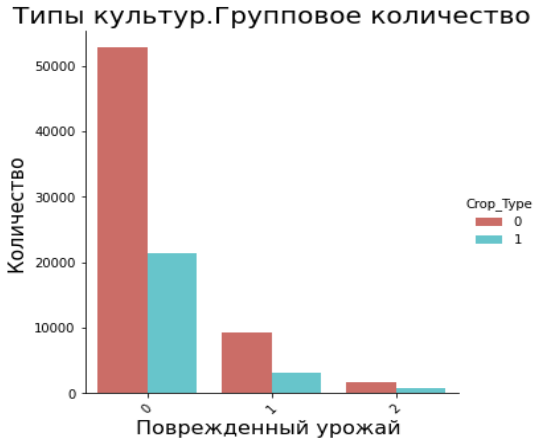


Рис.3.56. Гистограмма поврежденных культур из базы данных.

Выводы следующие. Повреждение урожая из-за пестицидов меньше по сравнению с ущербом из-за других причин. Культура типа 0 имеет более высокие шансы на выживание по сравнению с культурой типа 1.

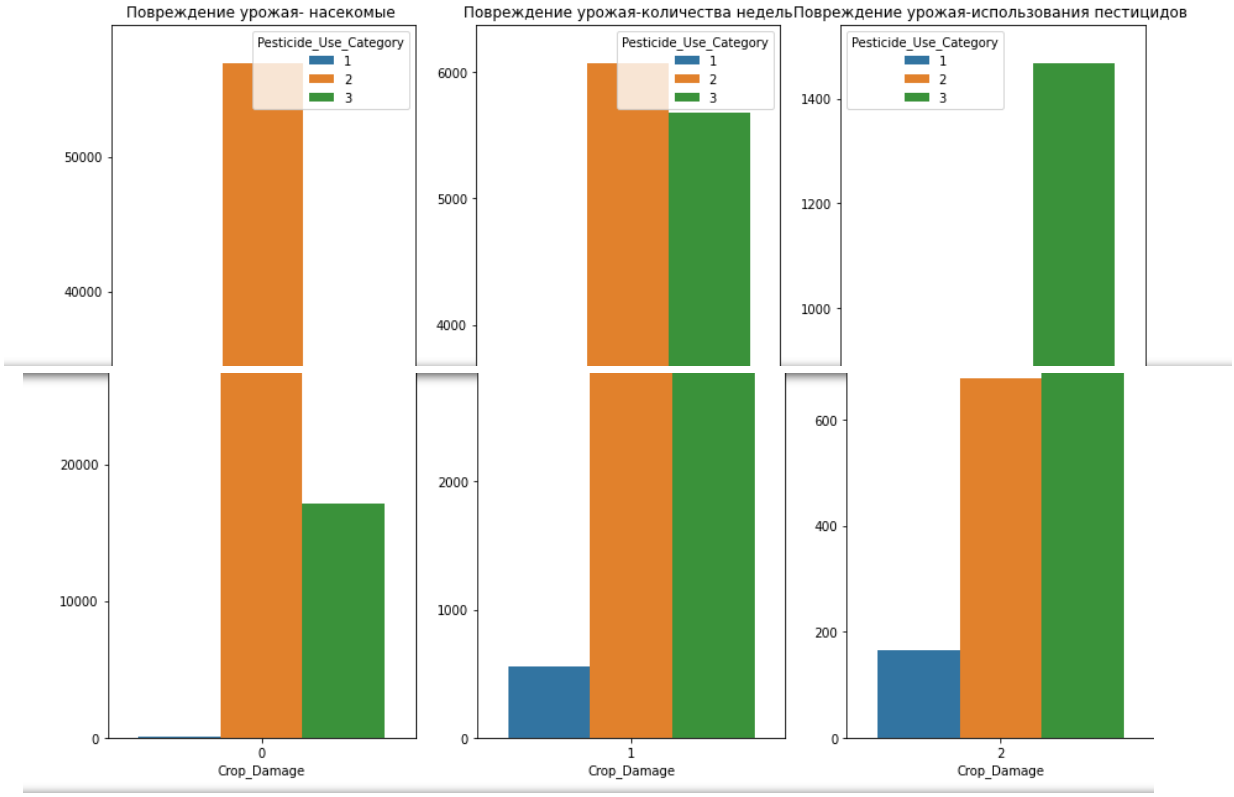


Рис.3.57. Гистограмма поврежденных культур от насекомых, увеличения числа недель использования пестицидов, от типа пестицидов

Пестицид типа 2 намного безопаснее в использовании по сравнению с пестицидом типа 3. Пестицид типа 3 наносит наибольший ущерб сельскохозяйственным культурам, связанный с пестицидами.

Еще один график в одномерном анализе для получения дополнительной информации.

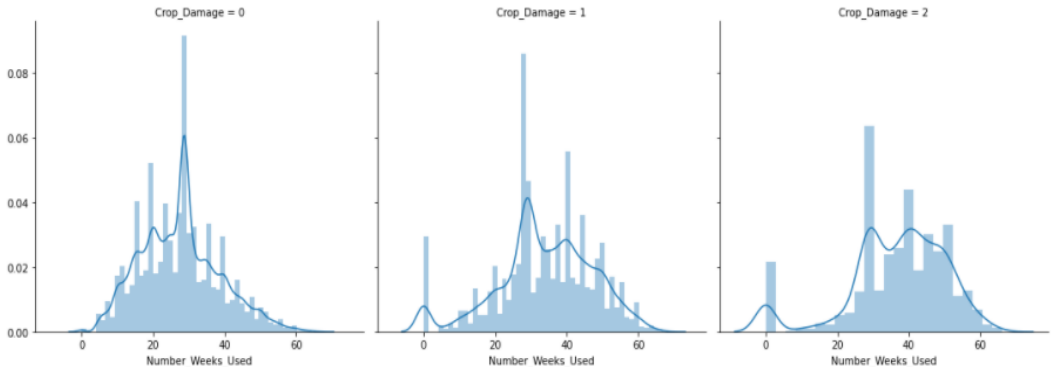


Рис.3.58. Гистограмма о поврежденных культурах

Выводы:

1. Из графика 1 можно сделать вывод, что до 20-25 недель вред от пестицидов незначителен.

2. Из графика 3 видно, что через 20 недель ущерб от применения пестицидов значительно возрастает.

Проведем теперь двумерный анализ данных:

Построенная гистограмма связи между поврежденным урожаем Crop_Damage и количеством вредителей -Estimated_Insects_Count

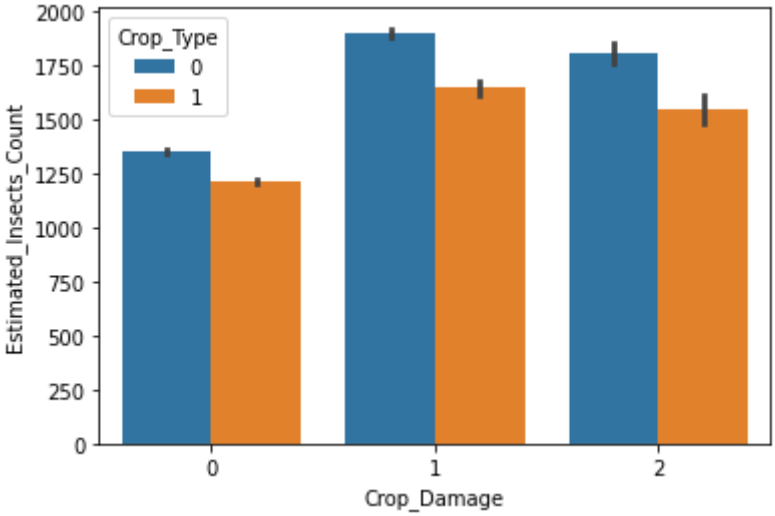


Рис.3.59. Уязвимость первого типа культур на повреждение

Из приведенного выше графика ясно видно, что большинство атак насекомых совершается на культуре типа 0.

Гистограмма между Crop_Type и Number_Weeks_Used.

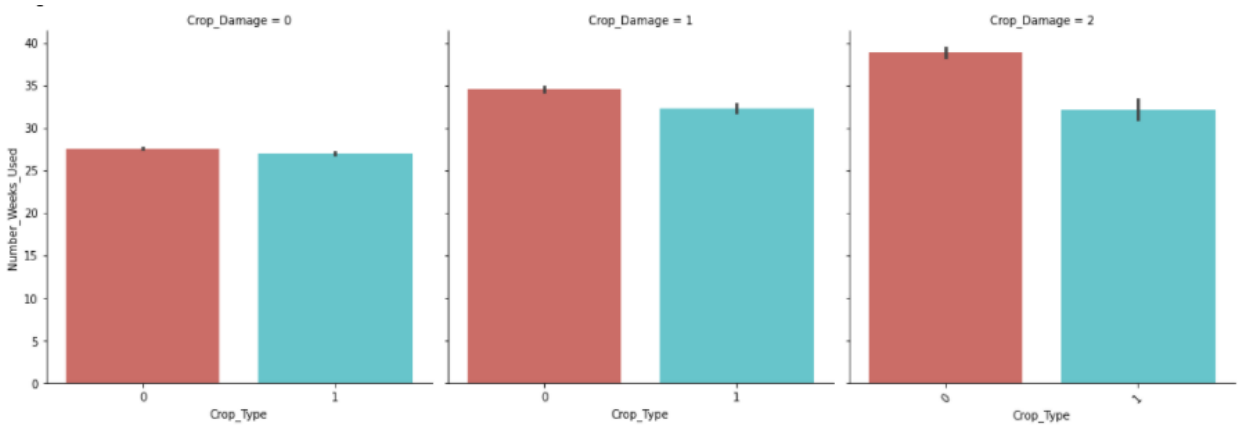


Рис.3.60. Гистограмма уязвимости типа культур на повреждение

Выводы:

Культура типа 0 более уязвима к пестицидам и другим повреждениям по сравнению с культурой типа 1.

Средняя продолжительность повреждения, связанного с пестицидами, ниже для культур типа 1.

Критические данные. Проведем анализ выбросов. Мы проверим наличие выбросов с помощью Boxplot .

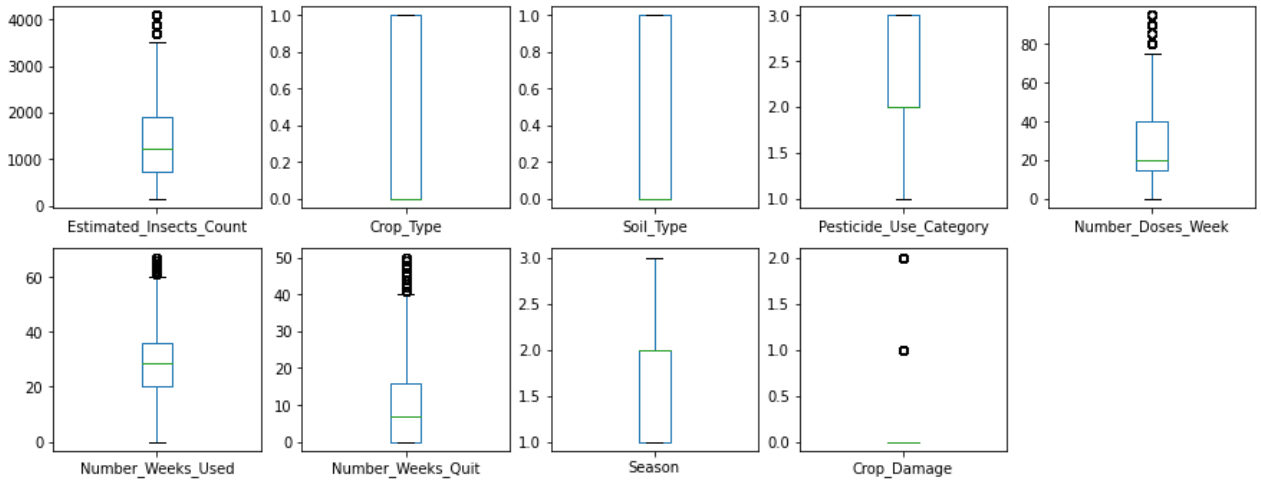


Рис.3.61. Данные выбросов до удаления.

Очевидно, что в столбцах Insect_Count, Doses_week и number_weeks_quit присутствуют некоторые выбросы. Теперь для удаления этих выбросов каждого столбца, заменим их средними значениями. Картина после удаления выбросов, имеет вид.

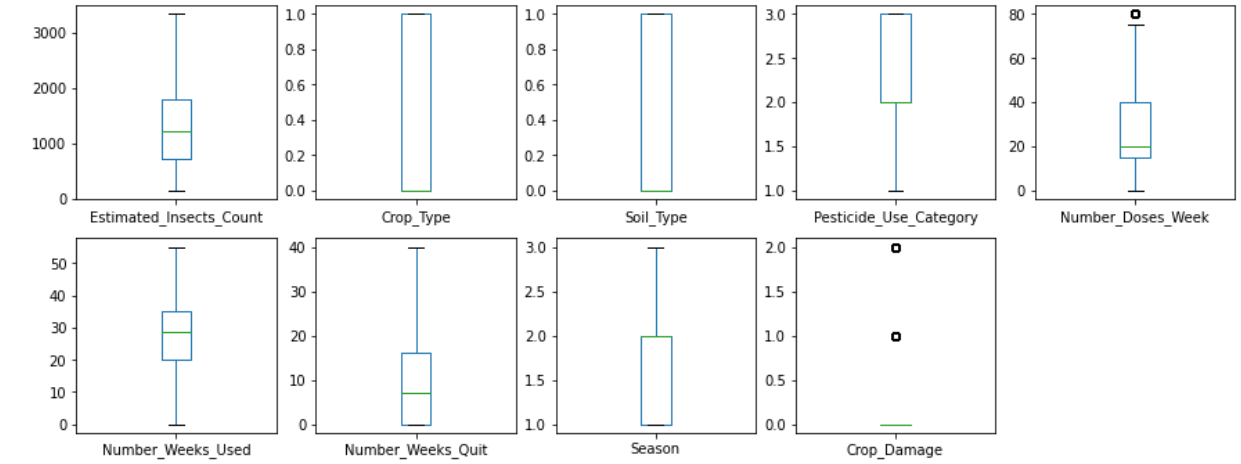


Рис.3.62. Процесс удаления выбросов. Замена выбросов средним значением.

Анализ перекоса:

Проверяем асимметрию наших данных с помощью histplot и наблюдаем, что все данные нормально распределены. Из рисунка ниже видно , что все данные нормально распределены.

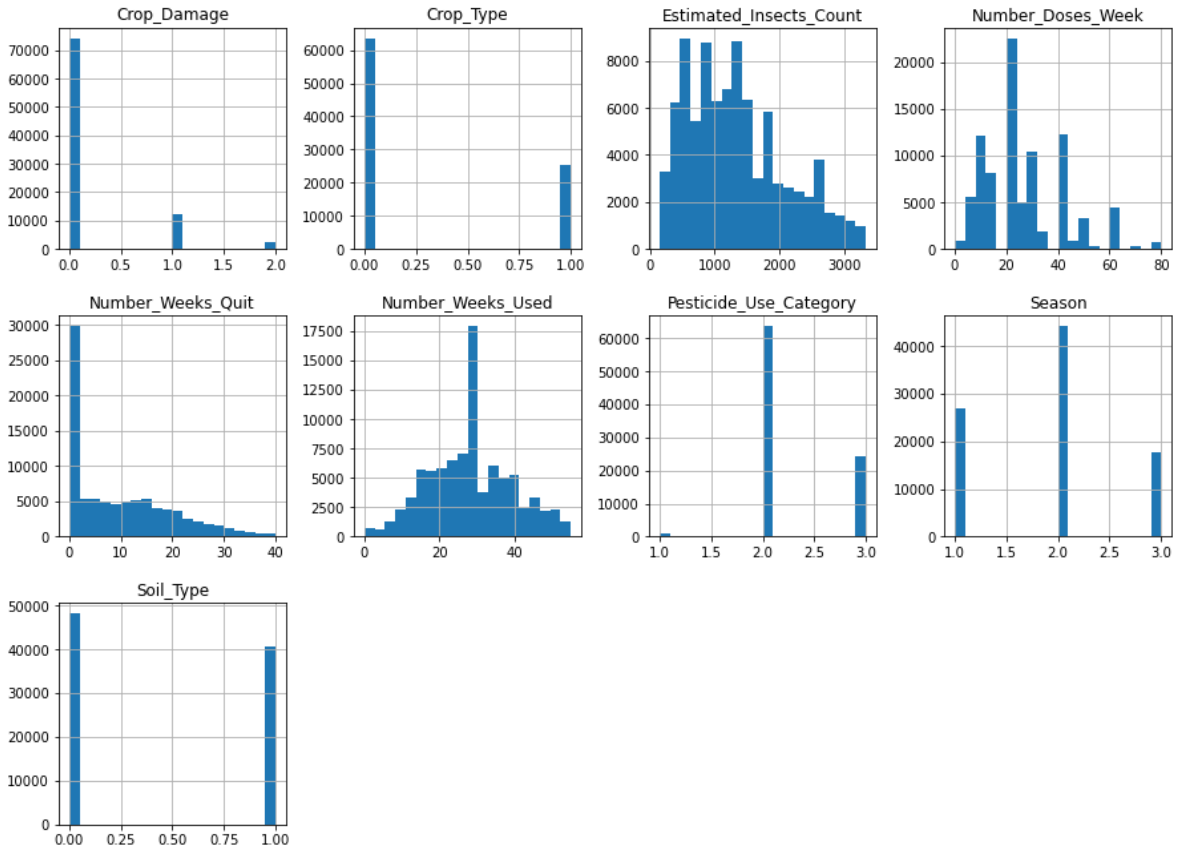


Рис.3.63. Асимметрии в данных-перекосы. Данные распределены нормально

Теперь наш набор данных готов для использования в модели машинного обучения для классификационного анализа.

Построение модели машинного обучения. Загружаем весь необходимый объемный пакет для построения модели. Далее для построения модели первым шагом удаляем из датафрейма целевую переменную, а затем масштабируем набор данных с помощью стандартного масштабатора, чтобы обеспечить нормальное распределение данных. После предварительной обработки теперь разделяем данные на подмножества обучения/тестирования. После процесса обучения с помощи нагруженных алгоритмов машинного обучения мы имеем следующие результаты оценки моделей. Оценка моделей:

Листинг 3.32. Результаты оценки точности моделей машинного обучения по различным метрикам

```
Results for model :  Random Forest
max accuracy score is 0.827308689128359
Mean accuracy score is :  0.8227283851917901
Std deviation score is :  0.0017259364106887492
Cross validation scores are :   [0.82568085 0.82286743 0.82191087 0.82161949 0.82297001]
**********************************************************************************************************
Results for model :  KNN
max accuracy score is 0.8304119492565817
Mean accuracy score is :  0.8270498878034861
Std deviation score is :  0.0017037082599439077
Cross validation scores are :   [0.82939455 0.82787531 0.82494936 0.82780935 0.82522087]
**********************************************************************************************************
Results for model :  Decision Tree Classifier
max accuracy score is 0.7536488882826354
Mean accuracy score is :  0.7475973026583367
Std deviation score is :  0.0026009823178113167
Cross validation scores are :   [0.75326356 0.74611749 0.74651137 0.7461032  0.74852287]
**********************************************************************************************************
Results for model :  Gaussian NB
max accuracy score is 0.8249215659528032
Mean accuracy score is :  0.8212091186402223
Std deviation score is :  0.0017104126392989223
Cross validation scores are :   [0.8239928  0.81859104 0.82117938 0.82105678 0.82122559]
**********************************************************************************************************
```

Из исходных значений точности модели мы видим, что KNN работает лучше, чем другие. Он имеет максимальную оценку точности и минимальные стандартные отклонения. Алгоритм дерево решений - DTC является худшим исполнителем с точностью 74%. А наивысший результат у алгоритма К-ближайших соседей 83.04%. Он имеет максимальную оценку точности и минимальные стандартные отклонения. Теперь я нахожу лучший параметр, то есть n_neighbors, используя GridSearchCV, взяв диапазон n_neighbors = (2,30), cv = 5 и оценку = 'точность' для нашей модели KNN, и нашел n_neighbor = 22.

Снова запускаем модель KNN с лучшими параметрами, т.е. n_neighbor = 22. Результат:

Листинг 3.33. Результаты оценки точности к-ближайших соседей при n=22

```
Results for model :  KNeighbors Classifier
max accuracy score is 0.8432001091256309
Mean accuracy score is :  0.842028849081089
Std deviation score is :  0.0012047855035650036
Cross validation scores are :   [0.84284267 0.84318028 0.84059194 0.84300264 0.8405267 ]
**********************************************************************************************************
```

Чтобы проверить производительность модели, мы теперь нанесем на график различные показатели производительности. Точность и отзыв, а также матрица путаницы оценки модели.

Листинг 3.34. Оценки точности и к-ближайших соседей при n=22

```
accuracy score is :  0.8402543326581139
classification report
              precision    recall  f1-score   support

           0       0.86      0.98      0.91     14848
           1       0.48      0.17      0.25      2461
           2       0.33      0.00      0.00       463

    accuracy                           0.84     17772
   macro avg       0.56      0.38      0.39     17772
weighted avg       0.79      0.84      0.80     17772

<matplotlib.axes._subplots.AxesSubplot at 0x1ee20a883a0>
```

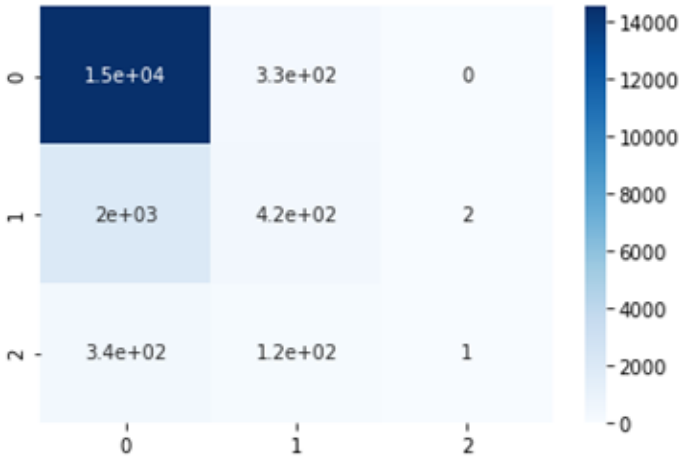


Рис.3.64. Проверка точности и полноты модели.

Из наблюдений мы обнаружили приличную точность (~ 0,84) , точность и полноту для модели. Это указывает на то, что модель хорошо подходит для предсказания. Для лучших результатов можно выполнить настройку гиперпараметров, которая поможет повысить точность модели. Но в целом KNN дает хорошую точность среди всех моделей, поэтому мы сохраняем ее как нашу окончательную модель. Проверим еще один классификатор - AdaBoosting. Вот результат. Листинг 3.36. Результаты полноты моделей.

```
0.8400855277965339
[[14830    18     0]
 [ 2361   100     0]
 [  429    34     0]]
              precision    recall  f1-score   support

           0       0.84      1.00      0.91     14848
           1       0.66      0.04      0.08      2461
           2       0.00      0.00      0.00       463

    accuracy                           0.84     17772
   macro avg       0.50      0.35      0.33     17772
weighted avg       0.79      0.84      0.77     17772
```

На листинге выше видно, что AdaBoostClassifier работает лучше, чем остальные алгоритмы, включая К-ближайших соседей. поэтому мы можем использовать его для наших окончательных прогнозов. Сохраним его.

ВЫВОДЫ К ГЛАВЕ 3:

В данной главе диссертационной работы, показано эффективность использования алгоритмов машинного обучения для моделирования и прогнозирования сельскохозяйственных задач.

Исследования по данным районов Иссык-Кульской области по урожайности картофеля и других культур, с применением машинного обучения, показали результативность по выявлению основных характеристик для построения моделей и прогнозов. Исследован важный раздел прогнозирования отток фермеров, как один из важных факторов урожайности. Сильная сторона прогнозирования оттока — это выявление потенциала выращивания урожая, человеческого фактора.

Еще одним немаловажным фактором исследованное в работе, это борьба с сорняками на весь период вегетации растения. Рассмотрено задача классификации сорняковых растений по виду растений на однолетние, многолетние и паразитные на основе длины, и ширины листьев растения. Показано, что для корнеплодных растений в основном влияющим фактором является однолетние сорняки. Получены результаты в виде уравнений множественной регрессии урожайности от независимых факторов и их оценки.

Приведены расчеты, построенные по алгоритмам машинного обучения, которые дали результаты отклонения точных и данных прогноза на основе регрессионных моделей.

Изучены обобщающие результаты применения алгоритмов машинного обучения для задач регрессии, на примере случайного леса в виде графического представления, как показателя урожайности картофеля по региону в целом. В данном случае интенсивность скопления данных вокруг линии регрессии, как мы видим наибольшая.

Наилучшая производительность получилось для оценки точности, средняя абсолютная ошибка MAPE =10.19% у алгоритма случайный лес для прогнозирования урожайности.

Другие алгоритмы как, метод опорных векторов, градиентный бустинг, метод ближайших соседей близки со значениями MAPE к результату случайного леса.

Показаны результаты применения ансамблевого метода к задачам прогнозирования урожайности, здесь выбраны и получены результаты для десяти наилучших моделей ансамбля.

Изучены, также урожайность наиболее распространенных культур, которые выращиваются в указанных районах и получены модель с помощью дерево решений с различными выборами параметров дерева и их глубин на примере случайный лес, получены результаты производительности моделей и даны оценки точности модели.

Визуализировано производительность алгоритма метода Лассо для прогноза урожайности. К задачам прогнозирования урожайности различных культур реализованы продвинутые алгоритмы машинного обучения, например метод опорных векторов, К –ближайших соседей, варианты градиентного бустинга и случайный лес.

Для сравнительного анализа оценки точности моделей сравнивались с результатами множественной регрессии. Использование машинного обучения — это выявления некоторых скрытых особенностей факторов, влияющих на урожайность для выбранного региона.

Для дальнейшего исследования региона — это сбор данных включать данные с более расширенными диапазонами и категориями, которая учитывается при построении модели. Сильное влияние на урожайность при этом, оказывают изменении климата, различные случайные природные явления как выпадения града, резкое увеличение дневных и ночных температур –заморозки, усиление солнечной активности и риски связанные с длительными аномально жаркими днями в летнее время, которая разрушает почву приводит к эрозии площадей засева, не исследован влияние на урожайность как частые селевые потоки и борьба маловодьем, которые нередки для данного региона, явления.

Все эти факторы приводят к потере урожайности в колоссальных размерах. Некоторые задачи, связанные с этими классами задач, рассматривается, с использованием технологии глубокого обучения в четвертой главе диссертации. Результаты полученные в данной главе диссертации опубликованы в международных индексируемых журналах Scopus.

ГЛАВА 4
СОЗДАНИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ЗАДАЧ СЕЛЬСКОГО ХОЗЯЙСТВА МЕТОДАМИ ГЛУБОКОГО ОБУЧЕНИЯ С ТЕХНОЛОГИЯМИ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Глубокое обучение-Deep Learning – это набор алгоритмов машинного обучения, которые моделируют высокоуровневые абстракции в данных, используя архитектуры, состоящие из множества нелинейных преобразований. В данной главе диссертационной работы рассматривается современные методы прогнозирования задач сельского хозяйства с использованием методов глубокого обучения. Было проведено исследование по глубокому обучению и его гибридные методы, такие как искусственная нейронная сеть, глубокая нейронная сеть, вопросы оптимизации нейронных сетей для различных задач сельского хозяйства. Исследовано применение мощного метода глубокого обучения применительно к задачам сельского хозяйства- сверточная нейронная сеть. Особенность данной технологии заключается, использование технологий компьютерного зрения для задач распознавания болезни растений различного класса.

Данный метод в диссертации использовано как метод управления растениеводством и является важным инструментом исследования борьбы с болезнями и вредителями растений, методы глубокого обучения позволяют оптимизировать процесс использования удобрений и как метод борьбы с сорняковыми растениями. С применением глубокого обучения построены различные модели для прогнозирования задач сельского хозяйства. На базе данных с учетом климатических условий изучалось задача распознавания болезней различных растений и культур. Ниже на Рис.4.1. приведены некоторые классы приложений применения искусственного интеллекта в сельском хозяйстве.

Рис.4.1. Приложения глубокого обучения в сельском хозяйстве

4.1. Обзор методов глубокого обучения для задач сельского хозяйства

Глубокое обучение является неотъемлемой частью искусственного интеллекта (ИИ), где она отображает структуру человеческого мозга и его процессы познания. С развитием больших данных и общих параллельных вычислительных блоков быстро развивались и приложения, основанные на глубоком обучении.

Глубокое обучение — это нейронная сеть с несколькими скрытыми слоями. Подобно мелким нейронным сетям, глубокие нейронные сети могут моделировать сложные нелинейные системы. Сети скрытых слоев со многими слоями обеспечивают более высокий уровень абстракции модели, тем самым улучшая возможности модели.

Самая большая разница между глубоким обучением и другим машинным обучением заключается в том, как находить функции наиболее влияющие на результат прогнозирования, выявление скрытых связей между данными с любым сложной связью. Процесс извлечения признаков является абстрактным процессом. Абстракция глубокого обучения имитирует способ передачи информации нейронами человека.

В диссертации для распознавания болезней растений широко использованы различные архитектуры сверточных нейронных сетей. Сверточные нейронные сети — это типичная глубокая нейронная сеть.

Структура глубокого обучения в основном называется нейронной сетью, который обрабатывается со скрытыми слоями для улучшения обучения.

Идентификация болезней растений.

В диссертации глубокое обучение играет важную роль как важный инструмент исследований в идентификация растений и выявление болезней растений, которая в условиях изменения климата является сложной задачей сельского хозяйства.

Заболевание растений можно определить с помощью метод глубокого обучения, называемый сверточной нейронной сетью (CNN), в котором изображения зараженные листья захватывают, а затем проверяют, чтобы обнаружить заболевание растения. С применением данной технологии и открытой системы базы данных в работе исследованы листья из 25 видов растений, находившихся на разных стадиях вегетации, обучены около 80 000 изображений, в результате модель получит четкое представление о конкретном растении и выдаст желаемое правильный результат.

Сверточная нейронная сеть (CNN) играет важную роль, когда она определяет точное положение пораженной части, что помогает поставить высокоточный диагноз. Заболевание растений также можно остановить до возникновения, если удавалось выявить факторы, вызывающие заболевания. Как правило, существуют факторы, влияющие на растения климат и внешние факторы, такие как насекомые. Для аграриев было бы полезнее, если заболевания были выявлены в момент воздействия внешних факторов, вызывающих повреждение. Грибок, бактериальные инфекции были вызваны климатическими изменениями, периодическими вредителями и насекомыми.

Глубокий метод обучения CNN помогает идентифицировать различные части больных растений и вредителей, лежащих на нем. Это помогает фермерам, чтобы обнаружить болезнь растений с большей скоростью и уменьшить ущерб урожаю.

Прогнозирование урожайности.

В настоящее время фермеры, начали обращать внимание на автоматизированные системы, которые позволяют предварительно обнаруживать различные болезни растений.

В случае сельского хозяйства автоматизированная система может быть реализовано на основе различных сайтов и мобильных приложений основанные на моделях ИИ.

Результаты ИИ могут быть достигнуты с помощью техники глубокого обучения, где он использует концепцию нейронных сетей, сеть, которая изображает метод функционирования человеческого мозга. В глубоком

обучении различные типы нейронных сетей и их комбинации используются, чтобы найти метод классификации и регрессии для урожайности одновременно.

Искусственная нейронная сеть (ИНС).

При прогнозировании урожайности в работе использованы искусственная нейронная сеть. Данные в задачах, например урожайности задаются в виде числовых или категориальных данных.

Типы или классы болезней, например в задачах прогнозирования болезней растений задаются в виде изображений с метками – названиями болезней. А свойства почвы и факторы окружающей среды являются числовыми данными.

Изображения листьев растений периодически снимаются и анализируются на рост и выявления болезней. Уровень роста измеряется для определения правильности роста по отношению к высоте, определяются болезни и различные стадии их развития для прогнозирования процента урожая. Урожайность можно прогнозировать с использованием факторов окружающей среды, таких как температура, осадки и орошение, где на основании которых можно определить устойчивость урожая. Нейронная сеть в данной работе обучалась на данных факторов окружающей среды и урожая и изображения таких культур, как картофель, кукуруза, пшеница и ячмень. Для получения прогноза для более высокой урожайности необходимо анализировать факторы окружающей среды с использованием нескольких методов, в том числе множественной линейной регрессией и других алгоритмов машинного обучения.

Климатические факторы окружающей среды, такие как температура, влажность, осадки и свойства почвы периодически меняются в зависимости от времени. При упомянутых факторах окружающей среды было бы труднее иметь конкретные характеристики почвы для каждой культуры, поэтому с алгоритмом машинного обучения K-ближайших соседей она стала более эффективной.

Сверточная нейронная сеть (CNN). Техника CNN в основном используется в глубоком обучении для обработки изображений листьев растений. В сверточной нейронной сети изображения анализируются попиксельно в матричном формате, затем формируется в фильтр, который был назван сверткой. В работе для широкого класса задач было использовано сверточная основа нейронных сетей.

4.2. Математическое обоснование методов глубокого обучения

В данном разделе рассмотрено математическое описание методов глубокого обучения. Рассмотрены глубокие сети прямого и обратного распространения, а также технологии регуляризации А.Н.Тихонова для переобученных моделей. Изучается вопросы оптимизации глубоких нейронных

сетей и подбор параметров обучения нейронной сети. Широко посвящен вопрос, практической реализации глубокого обучения с помощью сверточных нейронных сетей для реальных задач сектора сельского хозяйства. Рассматривается современное состояние глубокого обучения – как оно применяется для решения практических задач, в частности компьютерном зрении для задач распознавания. Современное глубокое обучение предлагает развитую инфраструктуру обучения с учителем. Благодаря добавлению дополнительных слоев и блоков в пределах одного слоя глубокая сеть может представлять все более и более сложные функции. Большинство задач, сводящихся к отображению входного вектора на выходной, с которыми человек справляется легко и непринужденно, может быть решено методами глубокого обучения при наличии достаточно больших моделей и наборов данных. Другие задачи, которые нельзя описать как ассоциирование одного вектора с другим, или настолько трудные, что человеку нужно время для их решения, пока не поддаются глубокому обучению. В этой части диссертации мы рассмотрим базовые технологии аппроксимации параметрических функций, лежащие в основе почти во всех практических приложений глубокого обучения. Рассмотрено модели глубокой сети прямого распространения или полносвязанные нейронные сети, используемой для представления таких функций. Для переобученных моделей, являющейся, рассмотрено передовые методы регуляризации и оптимизации. Для масштабирования моделей на большой объем входных данных, например на изображения высокого разрешения, или на длинные временные последовательности, необходима специализация. Для задач связанное с распознаванием болезней растений использованы в основном сверточные нейронные сети.

Полносвязанные глубокие нейронные сети. Глубокие сети прямого распространения, которые называют также нейронными сетями прямого распространения, или многослойными перцептронами (МСП),самые типичные примеры моделей глубокого обучения. Цель сети прямого распространения аппроксимировать некоторую функцию f *. Например, в случае классификатора y = f *(x) отображает вход x в категорию y. Сеть прямого распространения определяет отображение y = f(x; θ) и путем обучения находит значения параметров θ, дающие наилучшую аппроксимацию. Слова «прямое распространение» означают, что распространение информации начинается с x, проходит через промежуточные вычисления, необходимые для определения f, и заканчивается выходом y. Не существует обратных связей, по которым выходы модели подаются на ее вход. Обобщенные нейронные сети, включающие такие обратные связи, называются рекуррентными, ее мы рассмотрим ниже. Сети

прямого распространения исключительно важны для практического применения машинного обучения. Они лежат в основе многих важных коммерческих приложений. Например, сверточные сети, используемые для распознавания объектов на фотографиях, это частный случай сетей прямого распространения. Сети прямого распространения концептуальная веха на пути к рекуррентным сетям, стоящим за многими приложениями в области естественных языков. Нейронные сети прямого распространения называются сетями, потому что они, как правило, образованы композицией многих различных функций. С моделью ассоциирован ориентированный ациклический граф, описывающий композицию. Например, можно связать три функции f^1, f^2, и f^3, в цепочку f(x) = $f^3(f^2(f^1(x)))$. Такие цепные структуры чаще всего используются в нейронных сетях. В данном случае f^1 называется первым слоем сети, f^2, – вторым слоем и т. д. Общая длина цепочки определяет глубину модели. Название «глубокое обучение» непосредственно связано с этой терминологией. Последний слой сети прямого распространения называется выходным. В ходе обучения нейронной сети мы стремимся приблизить f(x) к f^*(x). Обучающие данные – это зашумленные приближенные примеры f^*(x), вычисленные в различных точках. Каждый пример x сопровождается меткой y ≈ f^*(x). Обучающие примеры напрямую указывают, что в выходном слое должно соответствовать каждой точке x, – это должно быть значение, близкое к y. Поведение остальных слоев напрямую обучающими данными не определяется. Алгоритм обучения должен решить, как использовать эти слои для порождения желаемого выхода, но обучающие данные ничего не говорят о том, что должен делать каждый слой. Алгоритму обучения предстоит самостоятельно решить, как с помощью этих слоев добиться наилучшей аппроксимации f^*. Поскольку обучающие данные не определяют выходов каждого из этих слоев, они называются скрытыми слоями.

Один из способов разобраться в сетях прямого распространения состоит в том, чтобы начать с линейных моделей и подумать, как преодолеть их ограничения. Линейные модели, в т. ч. логистическая регрессия и линейная регрессия, так привлекательны, потому что дают эффективную и надежную аппроксимацию – в замкнутой форме или посредством выпуклой оптимизации. Но у линейных моделей есть очевидный недостаток – емкость модели ограничена линейными функциями, поэтому модель неспособна понять произвольную связь между двумя величинами. Чтобы обобщить линейную модель на представление нелинейных функций от x, мы можем применить ее не к самому x, а к результату вычисления ϕ(x), где ϕ – нелинейное преобразование. Для получения нелинейного алгоритма обучения, основанного на неявном

применении преобразования ϕ, можно использовать метод опорных векторов. Можно считать, что ϕ дает набор признаков, описывающих x, или новое представление x. Тогда вопрос сводится к выбору отображения ϕ.

1. Один из вариантов – взять очень общее ϕ, например бесконечномерное, неявно используемое в ядерных методах, основанных на радиально-базисном ядре. Если размерность ϕ(x) достаточна велика, то емкости модели хватит для аппроксимации обучающего набора, но обобщение модели на тестовые данные в данном случае не получается.

2. Другой вариант – спроектировать ϕ вручную до применения методами глубокого обучения. Но для каждой задачи требуются десятилетия человеческого труда и специалисты в соответствующей предметной области, например распознавания речи или компьютерного зрения, а передачи знаний между разными областями почти нет.

Технологии регуляризации переобученных моделей в глубоком обучении.

Основной недостаток моделей машинного обучения — это переобучение. С помощью технологий регуляризации в данном разделе мы создадим алгоритм машинного обучения, который будет хорошо работать не только на обучающих данных, но и на данных тестирования. Данный результат достигается с помощью метода регуляризации.

В глубоком обучении существует много вариантов регуляризации. Разработка эффективных алгоритмов регуляризации является основной целью метода.Опишем регуляризацию, уделив особое внимание стратегиям регуляризации для глубоких моделей или моделей, которые используются в качестве их строительных блоков. Регуляризацию можно определить как «любую модификацию алгоритма обучения, предпринятую с целью уменьшить его ошибку обобщения, не уменьшая ошибки обучения».

Смысл регуляризация оценки – в увеличении смещения в обмен на уменьшение дисперсии. Эффективным считается регуляризатор, который находит выгодный компромисс, т. е. значительно уменьшает дисперсию, не слишком увеличивая смещение. Существует три случае , когда обучаемое семейство моделей либо (1) не включает истинный процесс, порождающий данные, – это соответствует недообучению и провоцирует смещение, либо (2) соответствует истинному порождающему процессу, либо (3) включает как истинный порождающий процесс, так и много других потенциальных процессов, – это режим переобучения, когда в ошибке оценивания превалирует дисперсия, а не смещение.

Цель регуляризации – перевести модель из третьего режима во второй. На практике мы почти никогда не имеем доступа к истинному процессу, поэтому не можем знать наверняка, включает оцениваемое семейство этот процесс или нет. Однако алгоритмы глубокого обучения по большей части применяются в предметных областях, где истинный процесс почти наверняка не входит в семейство моделей. Алгоритмы глубокого обучения обычно используются в чрезвычайно сложных областях – обработка изображений, звуковых последовательностей и текстов, где истинный процесс, по существу, сводится к моделированию сложных процессов.

В реальных приложениях глубокого обучения наилучшая эмпирическая модель, это управление сложностью модели, где основной целью является регуляризированная модель. Далее мы дадим обзор нескольких стратегий создания большой регуляризированной глубокой модели.

Регуляризация параметров по норме L^2. Один из самых простых и распространенных видов штрафа параметров – по норме L^2 , – который часто называют снижением весов. Цель такой стратегии регуляризации – выбирать веса, близкие к началу координат, за счет прибавления к целевой функции члена регуляризации

$$\Omega(\theta) = \frac{1}{2}||\omega||_2^2 . \quad (4.1)$$

Регуляризацию по норме L^2 называют также гребневой регрессией, или регуляризацией Тихонова. Получить качественное представление о поведении регуляризации методом снижения весов можно, изучив градиент регуляризированной целевой функции. Для простоты опустим параметр смещения, т. Е. будем считать, что θ совпадает ω. Полная целевая функция в такой модели с учетом (4.1) имеет вид:

$$J(\omega; X, y) = (\alpha/2)\omega^T \omega + J(\omega; X, y), \quad (4.2)$$

а градиент по параметрам ω:

$$\nabla_\omega J(\omega; X, y) = \alpha\, \omega + \nabla_\omega J(\omega; X, y). \quad (4.3)$$

Итерация обновления весов с целью уменьшения градиента имеет вид:

$$\omega = \omega - \varepsilon\big(\alpha\omega + \nabla_\omega J(\omega; X, y)\big). \quad (4.4)$$

То же самое можно переписать в виде:

$$\omega = (1 - \varepsilon\alpha)\omega - \varepsilon\nabla_{\omega} J\,(\omega;\, X, y). \tag{4.5}$$

Как видим, добавление члена сложения весов изменило правило обучения: теперь мы на каждом шаге умножаем вектор весов на постоянный коэффициент, меньший 1, перед тем как выполнить стандартное обновление градиента. Итак, мы описали, что происходит на одной итерации. Что получится в целом в процессе обучения.Еще упростим анализ, предположив квадратичную аппроксимацию целевой функции в окрестности того значения весов, при котором достигается минимальная стоимость обучения без регуляризации, ω^*= $\arg\min_{\omega}$ J(ω). Если целевая функция действительно квадратичная, как в случае модели линейной регрессии со среднеквадратической ошибкой, то такая аппроксимация идеальна. Аппроксимация J – описывается формулой:

$$\hat{J}(\theta) = J(\omega^*) + \frac{1}{2}(\omega - \omega^*)^T \mathrm{H}(\omega - \omega^*), \tag{4.6}$$

где H – матрица Гессе J относительно ω, вычисленная в точке ω^*. В этой квадратичной аппроксимации нет члена первого порядка, потому что ω^*, по определению, точка минимума, в которой градиент обращается в нуль. Из того, что ω^*– точка минимума J, следует также, что матрица H положительно полуопределенная. Минимум $\hat{J}$ достигается там, где градиент

$$\nabla_{\omega}\hat{J}(\omega) = \mathrm{H}\,(\omega - \omega^*) = 0. \tag{4.7}$$

Чтобы изучить эффект снижения весов, модифицируем уравнение (7), прибавив градиент снижения весов. Теперь мы можем найти из него минимум регуляризированного варианта $\hat{J}$ обозначим через $\tilde{\omega}$ положение точки минимума.

$$\begin{aligned} &\alpha\tilde{\omega} + \mathrm{H}\,(\tilde{\omega} - \omega^*) = 0 \\ &(\mathrm{H} + \alpha I)\tilde{\omega} = \mathrm{H}\,\omega^*, \\ &\tilde{\omega} = (\mathrm{H} + \alpha I)^{-1}\mathrm{H}\omega^*. \end{aligned} \tag{4.8}$$

Когда α стремится к 0, регуляризированное $\tilde{\omega}$ решение стремится к ω^*. Но что происходит, когда α возрастает? Поскольку матрица H вещественная и симметричная, мы можем разложить ее в произведение диагональной матрицы

Λ и ортогональной матрицы собственных векторов Q: H = QΛ Q^T. Подставляя это разложение в уравнение (4.8), получаем

$$\tilde{\omega} = (Q\Lambda Q^T + \alpha I)^{-1}\, Q\Lambda Q^T \omega^* = [(Q(\Lambda + \alpha I)Q^T]^{-1}\, Q\Lambda Q^T \omega^* = \\ = (Q(\Lambda + \alpha I)]^{-1}\, \Lambda Q^T \omega^*. \qquad (4.9)$$

Мы видим, что результатом снижения весов является масштабирование ω^* вдоль направлений собственных векторов H. Точнее, компонента ω^*, параллельная i -му собственному вектору H, умножается на коэффициент $\lambda_i/(\lambda_i + \alpha)$. Вдоль направлений, для которых собственные значения H относительно велики, например, когда $\lambda_i \gg \alpha$. эффект регуляризации сравнительно мал. Те же компоненты, для которых $\lambda_i \ll \alpha$, сжимаются почти до нуля. Относительно неизменными остаются только те направления, в которых параметры дают сильный вклад в уменьшение целевой функции. Если направление не дает вклада в уменьшение целевой функции, то собственное значение гессиана мало, т. е. движение в этом направлении не приводит к заметному возрастанию градиента. Компоненты вектора весов, соответствующие таким малозначимым направлениям, снижаются почти до нуля благодаря использованию регуляризации в ходе обучения. До сих пор мы обсуждали снижение весов в терминах воздействия на оптимизацию абстрактной квадратичной функции стоимости. Но как эти эффекты проявляются конкретно в машинном обучении? Это можно выяснить на примере изучения линейной регрессии – модели, в которой истинная функция стоимости квадратичная и потому поддается проведенному выше анализу. Повторяя те же рассуждения, мы получим для этого частного случая результат, сформулированный в терминах обучающих данных. Для линейной регрессии функция стоимости равна сумме квадратов ошибок:

$$(\mathrm{X}\omega - \mathrm{y})^{\mathrm{T}}(\mathrm{X}\omega - \mathrm{y}). \qquad (4.10)$$

После добавления L^2 - регуляризации целевая функция принимает вид:

$$(\mathrm{X}\omega - \mathrm{y})^{\mathrm{T}}(\mathrm{X}\omega - \mathrm{y}) \; + \frac{1}{2}\, \alpha\omega^{\mathrm{T}}\omega. \qquad (4.11)$$

В результате нормальные уравнения, из которых ищется решение:

$$\omega = \; (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{y}, \qquad (4.12)$$

принимают вид

$$\omega = (X^T X + \alpha I)^{-1} X^T y\ . \qquad (4.13)$$

Матрица $X^T X$ в уравнении (4.13) пропорциональна ковариационной матрице (1/m) $X^T X$. В уравнении (4.13) применение L^2-регуляризации заменяет эту матрицу на $(X^T X + \alpha I)^{-1}$. Новая матрица отличается от исходной матрицы только прибавлением α ко всем диагональным элементам. Диагональные элементы этой матрицы соответствуют дисперсии каждого входного признака. Таким образом, L^2 – регуляризация заставляет алгоритм обучения «воспринимать» вход X как имеющий более высокую дисперсию и, следовательно, уменьшать веса тех признаков, для которых ковариация с выходными метками мала, по сравнению с добавленной дисперсией.

L^1 – регуляризация. Регуляризация по норме L^2– самая распространенная форма снижения весов, но есть и другие способы штрафовать за величину параметров модели. Одним из них является L^1– -регуляризация.

Формально L^1– регуляризация параметров модели ω определяется по формуле

$$\Omega(\theta) = ||\omega||_1 = \sum_{i=1}^{n} |\omega_i|, \qquad (4.14)$$

т. Е. как сумма абсолютных величин отдельных параметров. Обсудим влияние L^1– регуляризации на простую модель линейной регрессии без параметра смещения – ту самую, что рассматривалась в ходе анализа L^2-регуляризации. Особенно нас интересуют различия между двумя видами регуляризации. Как и в предыдущем случае, сила регуляризации контролируется путем умножения штрафа на положительный гиперпараметр α. Таким образом, регуляризированная целевая функция $\tilde{J}(\omega, X, y)$ описывается формулой:

$$\tilde{J}(\omega, X, y) = \alpha\, ||\omega||_1 + J(\omega; X, y), \qquad (4.15)$$

а ее градиент (точнее, частная производная) равен

$$\nabla_\omega \tilde{J}(\omega, X, y) = \alpha \mathrm{sign}(\omega) + \nabla_\omega J(\omega, X, y), \qquad (4.16)$$

где $\mathrm{sign}(\omega)$ означает, что функция sign применяется к каждому элементу ω.

Из уравнения (4.16) сразу видно, что эффект L^1– -регуляризации совсем не такой, как L^2– -регуляризации. Теперь вклад регуляризации в градиент уже не масштабируется линейно с ростом каждого ω_i, а описывается постоянным слагаемым, знак которого совпадает с $\mathrm{sign}(\omega_i)$. Одним из следствий является

тот факт, что мы уже не получим изящных алгебраических выражений квадратичной аппроксимации J (ω; X, y) , как в случае L^2-регуляризации.

В нашей простой линейной модели имеется квадратичная функция стоимости, которую можно представить ее рядом Тейлора. Можно вместо этого считать, что это первые члены ряда Тейлора, аппроксимирующие функцию стоимости более сложной модели. Градиент в этой конфигурации равен

$$\nabla_{\omega} \hat{J}(\omega, \mathrm{X}, \mathrm{y}) = H(\omega - \omega^*), \tag{4.17}$$

где H –матрица Гессе J относительно ω, вычисленная в точке ω^*.

Поскольку штраф по норме L^1– не допускает простого алгебраического выражения в случае полного гессиана общего вида, то мы сделаем еще одно упрощающее предположение: будем считать матрицу Гессе диагональной

$$\mathrm{H} = \mathrm{diag}([\mathrm{H}_{1,1}, \mathrm{H}_{2,2}, \ldots, \mathrm{H}_{\mathrm{n,n}}])$$

где все $\mathrm{H}_{\mathrm{i,i}} > 0$. Это предположение справедливо, если данные для задачи линейной регрессии были подвергнуты предварительной обработке для устранения корреляции между входными признаками, например методом главных компонент. Нашу квадратичную аппроксимацию L_1 - регуляризированной целевой функции можно представить в виде суммы по параметрам:

$$\hat{J}(\omega, X, y) = \mathrm{j}\ (\omega^*, X, y) + \sum_{i=1}^{n} [\frac{1}{2} H_{i,i} (\omega - \omega^*)^2 + \alpha |\omega_i|], \tag{4.18}$$

У задачи минимизации этой приближенной функции стоимости имеется аналитическое решение (для каждого измерения *i*) вида:

$$\omega_i = \mathrm{sign}\ (\omega_i^*) \max\{|\omega_i^*| - \frac{\alpha}{H_{i,i,}}, 0\ \}, \tag{4.19}$$

Предположим, что $\omega_{\mathrm{i}}^* > 0$ для всех i. Тогда есть два случая:

1) $\omega_{\mathrm{i}}^* \leq \alpha / H_{i,i}$, Тогда оптимальное значение ω_{i} для регуляризированной целевой функции будет просто $\omega_{\mathrm{i}} = 0$. Причина в том, что вклад J (ω; X, y), в регуляризированную целевую функцию перевешивается – в направлении i – L^1 регуляризацией, которая сдвигает значение ω_i в нуль;

2) $\omega_{\mathrm{i}}^* > \alpha / H_{i,i}$.Тогда регуляризация не сдвигает оптимальное значение ω_i в нуль, а просто смещает его в этом направлении на расстояние $\alpha / H_{i,i}$.

Аналогичное рассуждение проходит, когда $\omega_i^*<0$, только L^1-штраф увеличивает ω_i на $\alpha/H_{i,i}$ или обращает в 0.

По сравнению с L^2-регуляризацией, L^1-регуляризация дает более разреженное решение. В этом контексте под разреженностью понимается тот факт, что у некоторых параметров оптимальное значение равны 0. Разреженность L^1 −регуляризации является качественным отличием от поведения L^2 −регуляризации. Уравнение (4.9) дает решение $\widetilde{\omega}$ для L^2- регуляризации. Применив к нему предположение о диагональности и положительной определенности гессиана H, которое мы приняли для анализа L^1 −регуляризации, найдем, что $\omega_i^* = \frac{H_{i,i}}{H_{i,i}+\alpha}\omega_i^*$. Если ω_i^* было не равно 0, то и $\widetilde{\omega}_i$ окажется не равным нулю. Это значит, что L^2 −регуляризация не приводит к разреженности параметров, тогда как в случае L^2- регуляризации это возможно, если α достаточно велико.

Свойство разреженности, присущее L^1- регуляризации, активно эксплуатировалось как механизм отбора признаков, идея которого состоит в том, чтобы упростить задачу машинного обучения за счет выбора некоторого подмножества располагаемых признаков. В частности, хорошо известная модель LASSO объединяет L^1- штраф с линейной моделью и среднеквадратической функцией стоимости. Благодаря L^1- штрафу некоторые веса обращаются в 0, и соответствующие им признаки отбрасываются.

Штраф по норме как оптимизация с ограничениями.

Рассмотрим функцию стоимости, регуляризированную путем добавления штрафа по норме параметров:

$$\tilde{J}(\theta; X, y)=J(\theta; X, y) +\alpha\Omega(\theta), \quad (4.19)$$

Для минимизации функции при наличии ограничений мы можем построить обобщенную функцию Лагранжа, состоящую из исходной целевой функции плюс набор штрафов. Каждый штраф имеет вид произведения коэффициента, называемого множителем Каруша–Куна–Таккера (ККТ), на функцию, показывающую, удовлетворяется ли ограничение. Если мы хотим, чтобы ограничение $\Omega(\theta)$, было меньше некоторой константы k, то можем определить такую обобщенную функцию Лагранжа:

$$\mathscr{L}(\theta, \alpha; X, y) = J(\theta; X, y)+ \alpha(\Omega(\theta) – k), \quad (4.20)$$

Решение задачи с ограничениями имеет вид:

$$\Theta^* = \mathscr{A}(\theta, \alpha). \quad (4.21)$$

Для решения этой задачи необходимо модифицировать θ, α. Есть также много других процедур – в одних используется метод градиентного спуска, в других аналитически ищутся точки, в которых градиент равен нулю, но в любом случае α должно увеличиваться, когда Ω(θ)>k, и уменьшаться, когда Ω(θ) < k.

Любое положительное α побуждает Ω(θ) к уменьшению. Оптимальное значение α* поощряет уменьшение Ω(θ), но не настолько сильно, чтобы Ω(θ) оказалось меньше k.

Чтобы составить представление о влиянии ограничения, зафиксируем α* и будем рассматривать задачу как функцию только от θ:

$$\theta^* = \mathscr{A}(\theta, \alpha^*) = J(\theta; X, y) + \alpha^*\Omega(\theta) \quad (4.22)$$

Это в точности то же самое, что регуляризированная задача обучения путем минимизации $\tilde{J}$. Таким образом, можно считать, что штраф по норме параметра налагает ограничение на веса. Если в качестве нормы Ω берется L^2, то веса должны лежать в единичном L^2 −шаре. Если же Ω – это норма L^1, то веса должны лежать в области с ограниченной нормой L^1. Обычно нам неизвестен размер области ограничений, соответствующей снижению весов с коэффициентом α*, потому что значение α* не говорит прямо о значении k. В принципе, можно решать уравнение относительно k, но связь между k и α* зависит от вида $J(\theta; X, y)$. Но хотя мы не знаем точный размер области ограничений, мы можем грубо управлять им, уменьшая или увеличивая α с целью увеличить или уменьшить область ограничений. Чем больше α, тем меньше область ограничений, и наоборот.

Функцию стоимости J можно смоделировать с помощью квадратичной аппроксимации в окрестности эмпирического оптимума весов ω^*:

$$\hat{J} = J(w^*) + \tfrac{1}{2}(\omega - \omega^*)^T H (\omega - \omega^*), \quad (4.23)$$

где H – гессиан J относительно ω, вычисленный в точке ω^*. Поскольку мы предположили, что ω^*– точка минимума J (w), то H является положительно полуопределенной.

Аппроксимируя разложением в ряд Тейлора, получаем выражение для градиента:

$$\nabla_\omega \hat{J}(\omega) = H(\omega - \omega^*). \quad (4.24)$$

Мы изучим траекторию вектора параметров в процессе обучения. Для простоты, пусть начальный вектор параметров совпадает с началом координат1, w(0) = 0. Мы составим приближенное представление о поведении градиентного спуска по J, проанализировав градиентный спуск по $\hat{J}$:

$$\omega^{(\tau)} = \omega^{(\tau-1)} - \varepsilon\nabla_{\omega}\,\hat{J}\left(\omega^{(\tau-1)}\right) - \varepsilon H(\omega^{(\tau-1)} - \omega^{*}), \tag{4.25}$$

$$\omega^{(\tau)} - \omega^{*} = (I - \varepsilon H)\,(\,\omega^{(\tau-1)} - \omega^{*})\,. \tag{4.26}$$

Перепишем это выражение в пространстве собственных векторов H, воспользовавшись спектральным разложением H: H = QΛQ^{T} где Λ – диагональная матрица, а Q –ортогональная матрица собственных векторов:

$$\omega^{(\tau)} - \omega^{*} = (I - \varepsilon \mathrm{Q}\Lambda Q^{T})\,(\,\omega^{(\tau-1)} - \omega^{*})\,, \tag{4.27}$$

$$Q^{T}(\omega^{(\tau)} - \omega^{*}) = (I - \varepsilon\Lambda)Q^{T}(\,\omega^{(\tau-1)} - \omega^{*}). \tag{4.28}$$

В предположении, что $\omega^{(0)}$= 0 и что ε достаточно мало, чтобы выполнялось условие

$$|1 - \varepsilon\,\lambda_i| < 1,$$

траектория параметров в процессе обучения после τ обновлений параметров описывается уравнением:

$$Q^{T}\omega^{(\tau)} = -\omega^{*} - (I - \varepsilon\Lambda)Q^{T}\omega^{*}, \tag{4.29}$$

$$Q^{T}\omega^{(\tau)} = [(\mathrm{I} - (\mathrm{I} - \varepsilon\Lambda)^{\tau}]Q^{T}\omega^{*}. \tag{4.30}$$

Выражение $Q^{T}\tilde{\omega}$ в уравнении (4.29) L^{2} регуляризацию можно переписать в виде:

$$Q^{T}\tilde{\omega} = (\Lambda + \alpha\mathrm{I})^{-1}\Lambda Q^{T}\omega^{*}, \tag{4.31}$$

$$Q^{T}\tilde{\omega} = [(\mathrm{I} - (\Lambda + \alpha\mathrm{I})^{-1}\alpha]Q^{T}\omega^{*}. \tag{4.32}$$

Сравнивая уравнения (4.31) и (4.32), мы заключаем, что если выбрать гиперпараметры ε, α и τ, так чтобы

$$(\mathrm{I} - \varepsilon\,\Lambda)^{\tau} = (\Lambda + \alpha\mathrm{I})^{-1}\alpha, \tag{4.33}$$

то L^2регуляризацию и раннюю остановку можно считать эквивалентными (по крайней мере, в предположении о квадратичной аппроксимации целевой функции). Мы можем пойти даже дальше: прологарифмировав и воспользовавшись разложением в ряд функции log(1 + x), приходим к выводу, что если все λ_i малы (то есть $\varepsilon\lambda_i \ll 1$ 1 и λ_i /α ≪ 1), то

$$\tau = \frac{1}{\varepsilon\alpha}, \tag{4.34}$$

$$\alpha = \frac{1}{\tau\varepsilon}, \tag{4.35}$$

Таким образом, в этих предположениях число итераций обучения τ играет роль величины, обратно пропорциональной параметру L^2-регуляризации, а число, обратное ετ, – роль коэффициента снижения весов.

Значения параметров, соответствующие направлениям сильной кривизны целевой функции, регуляризируются меньше, чем в направлениях меньшей кривизны.

В контексте ранней остановки это в действительности означает, что параметры, соответствующие направлениям сильной кривизны, обучаются раньше параметров, соответствующих направлениям меньшей кривизны.

Выкладки, приведенные в этом разделе, показывают, что траектория длины τ обрывается в точке, соответствующей минимуму L^2-регуляризированной целевой функции. Конечно, ранняя остановка – больше, чем простое ограничение на длину траектории; ранняя остановка обычно подразумевает наблюдение за ошибкой на контрольном наборе, чтобы оборвать траекторию в удачной точке пространства. Поэтому, по сравнению со снижением весов, у ранней остановки есть то преимущество, что она автоматически определяет правильную степень регуляризации, тогда как при использовании снижения весов требуется много экспериментов с разными значениями гиперпараметра.

4.3. Оптимизаторы в глубоком обучении

Алгоритмы оптимизации являются ключевой частью процесса обучения моделей глубокого обучения. Они отвечают за **настройку параметров модели** для минимизации функции потерь, которая измеряет, насколько хорошо

модель может делать прогнозы для данного набора данных. Доступны различные алгоритмы оптимизации, выбор которых может существенно повлиять на производительность модели.

Оптимизаторы в глубоком обучении — это алгоритмы, используемые для настройки параметров модели для минимизации функции потерь. Выбор оптимизатора может сильно влияет на производительность и скорость обучения модели. В данной диссертации рассмотрены некоторые наиболее часто используемых оптимизаторов в глубоком обучении, в том числе:

- Градиентный спуск
- Стохастический градиентный спуск (SGD)
- Метод импульсов
- Адаград
- Ададельта
- Адам
- RMSprop
- Адам и Адамакс

Каждый оптимизатор имеет уникальные характеристики и преимущества, а выбор оптимизатора зависит от задачи и архитектуры модели. Мы обсудим детали каждого оптимизатора и предоставим фрагменты кода, которые помогут вам понять, как они работают на практике.

Потребность в оптимизаторах в глубоком обучении. Выбор подходящего оптимизатора для модели глубокого обучения важен, поскольку это может существенно повлиять на ее производительность. Алгоритмы оптимизации имеют разные сильные и слабые стороны и лучше подходят для определенных задач и архитектур.

Например**,** стохастический градиентный спуск — это простой и эффективный оптимизатор, который широко используется, но ему может потребоваться помощь для решения задач со сложными невыпуклыми функциями потерь. С другой стороны, Адам — более сложный оптимизатор, который сочетает в себе идеи импульса и скорости адаптивного обучения и часто считается одним из наиболее эффективных оптимизаторов в глубоком обучении.

Критерии для выбора оптимизатора:

- Определение ключевой проблемы и архитектуру модели, для выбора оптимизатора
- Выбор различных оптимизаторов глубокого обучения, которая лучше подходить для изучаемой архитектуры нейронной сети
- Выбор основных параметров оптимизатора, такие как скорость обучения, чтобы увидеть, улучшит ли это производительность.

- Выбор оптимизатора является не единственным фактором, влияющий на производительность модели.
- Важными факторами являются и выбор архитектуры, качество данных и объем доступных данных для обучения модели

В диссертации для построения нейронных сетей рассмотрено наиболее часто используемых оптимизаторов, начиная с более простых и заканчивая более сложными.

Градиентный спуск. Градиентный спуск — это простой алгоритм оптимизации, который обновляет параметры модели, чтобы минимизировать функцию потерь.

Данный алгоритм оптимизации, основанный на выпуклой функции, который итеративно настраивает ее параметры, чтобы минимизировать данную функцию до ее локального минимума. Градиентный спуск итеративно уменьшает функцию потерь, перемещаясь в направлении, противоположном направлению наибольшего подъема. Это зависит от производных функции потерь при поиске минимумов. Использует данные всей обучающей выборки для расчета градиента функции стоимости к параметрам, что требует большого объема памяти и замедляет процесс.

Мы можем записать базовую форму алгоритма следующим образом:

$$\theta = \theta - \alpha \nabla_{\theta} \mathrm{L}(\theta), \tag{4.36}$$

где θ параметр модели, L (θ) функция потерь, и α это скорость обучения. В начальный момент времени выбор параметра α можно осуществить по следующим правилам

$$\alpha = 0{,}1;\ 0{,}01;\ 0{,}001;\ \ldots$$

$$\alpha = \alpha_n = \frac{1}{\min(n+1, mn)},\ n = 0,1,2,\ \ldots$$

$$\alpha_{min} = \frac{1}{mn}.$$

Преимущества градиентного спуска:

- Простота реализации.
- При оптимальной настройке скорости обучении может достигать хороших результатов

Минусы:

- Он может сходиться медленно, особенно для сложных моделей или больших наборов данных.
- Данный алгоритм чувствителен к выбору скорости обучения.

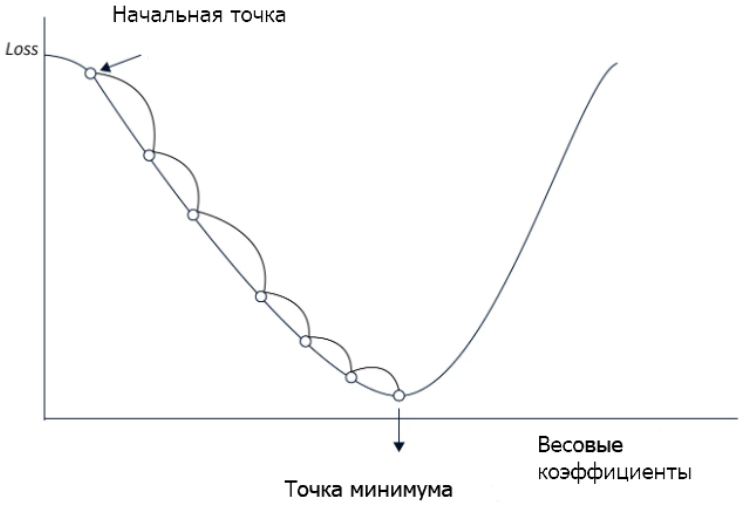


Рис.4.2. Сходимость метода градиентного спуска.

Стохастический градиентный спуск. Стохастический градиентный спуск (SGD) — это вариант градиентного спуска, который включает обновление параметров на основе небольшого, случайно выбранного подмножества данных (т. е. «мини-пакета»), а не полного набора данных. Мы можем записать базовую форму алгоритма следующим образом:

$$\theta = \theta - \alpha \nabla_{\theta} \mathrm{L}\,(\theta;\, x^{(i)}; y^{(i)})\,, \tag{4.37}$$

где $x^{(i)}, y^{(i)}$ представляет собой мини-пакет данных.

Плюсы:

- Это может быть быстрее, чем стандартный градиентный спуск, особенно для больших наборов данных.
- Может легче избежать локальных минимумов.

Минусы:

- Он может быть шумным, что приводит к снижению стабильности.
- Для получения хорошей производительности может потребоваться дополнительная настройка гиперпараметров.

  В следующем рисунке представлено сходимость стохастического градиентного спуска.

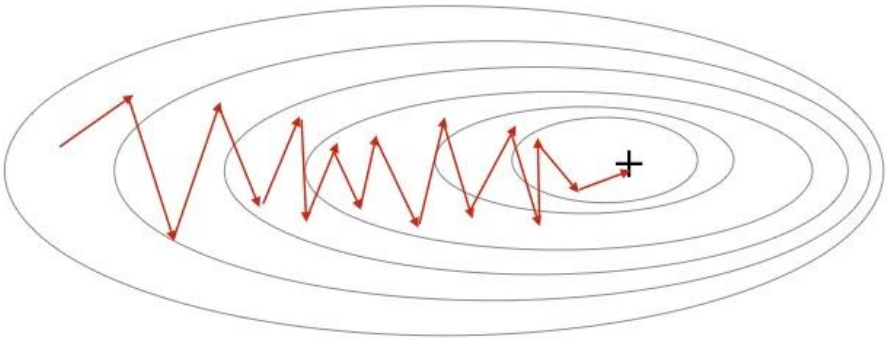


Рис.4.3. Сходимость стохастического градиентного спуска.

Стохастический градиентный спуск с импульсом. SGD с импульсом — это вариант SGD, который добавляет к правилу обновления термин «импульс», который помогает оптимизатору продолжать движение в том же направлении, даже если локальный градиент невелик. Термину импульса обычно присваивается значение от 0 до 1. Мы можем написать правило обновления следующим образом:

$$\upsilon = \beta\upsilon + (1-\beta)\nabla_\theta \mathrm{L}\,(\theta;\, x^{(i)}; y^{(i)})\,, \qquad (4.38)$$
$$\theta = \theta - \alpha \cdot \upsilon$$

где υ вектор импульса и β— гиперпараметр импульса.

Плюсы:

- Это может помочь оптимизатору более эффективно перемещаться по «плоским» областям функции потерь.
- Это может помочь уменьшить колебания и улучшить сходимость.

Минусы:

- Может промахнуться с хорошими решениями и остановиться на неоптимальных, если импульс слишком высок.
- Требует настройки гиперпараметра импульса.

Ниже представлен случаи сходимости стохастического градиентного спуска SGD без импульса - momentum и с импульсом- momentum.

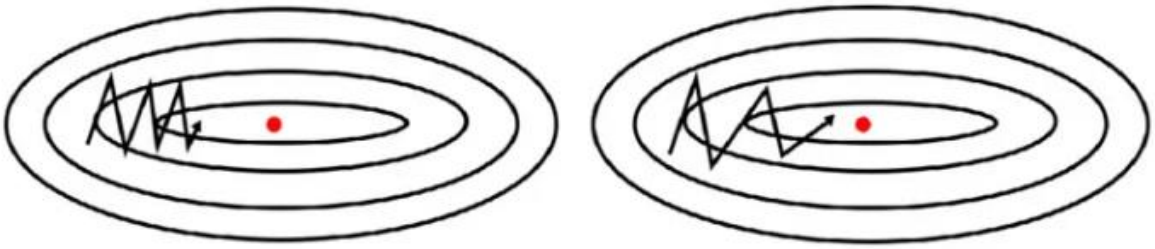


Рис.4.4. Сходимость стохастического градиентного спуска SGD без импульса и с импульсом

Мини-пакетный градиентный спуск. Мини-пакетный градиентный спуск аналогичен SGD, но вместо использования одной выборки для вычисления

градиента он использует небольшой **«мини-пакет» фиксированного размера** . Правило обновления такое же, как и для SGD, за исключением того, что градиент усредняется по мини-пакету. Это может уменьшить шум при обновлениях и улучшить сходимость.

Плюсы:

- Это может быть быстрее, чем стандартный градиентный спуск, особенно для больших наборов данных.
- Может легче избежать локальных минимумов.
- Может уменьшить шум при обновлениях, что приведет к более стабильной конвергенции.

Минусы:

- Может быть чувствителен к выбору размера мини-партии.
-

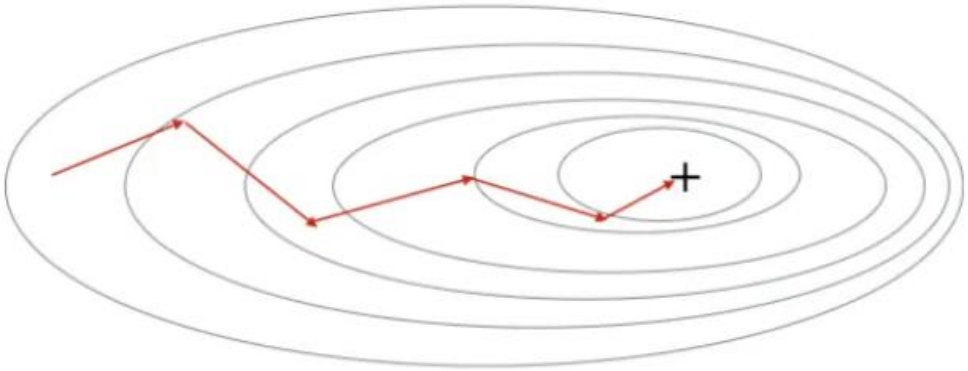


Рис.4.5. Сходимость мини пакетного градиентного спуска.

ADAGRAD. Adagrad — это алгоритм оптимизации, который **использует адаптивную скорость обучения для каждого параметра**. Скорость обучения обновляется на основе исторической информации о градиенте, так что параметры, которые получают много обновлений, имеют более низкую скорость обучения, а параметры, которые получают меньше обновлений, имеют более высокую скорость обучения. Правило обновления можно записать следующим образом:

$$g = \nabla_\theta \mathrm{L}\,(\theta;\, x^{(i)}; y^{(i)})\,, \tag{4.39}$$

$$G = G + g \odot g$$

$$\theta = \theta - \frac{\alpha}{\sqrt{G + \varepsilon}} \odot g$$

где g представляет собой матрицу, которая накапливает квадраты градиентов, а ε — небольшая константа, добавленная во избежание деления на ноль, в (4.2.3) и всюду в дальнейшем знак $\odot$ означает поэлементное умножение двух матриц.

Плюсы:

- Он может хорошо работать с разреженными данными.
- Автоматически регулирует скорость обучения на основе обновлений параметров.

Минусы:

- Может сходиться слишком медленно из-за некоторых проблем.
- Может вообще прекратить обучение, если скорость обучения станет слишком низкой.

RMSProp (Root Mean Square Propagation). RMSProp — это алгоритм оптимизации, аналогичный Adagrad, но он **использует экспоненциально убывающее среднее значение** квадратов градиентов, а не сумму. Это помогает уменьшить монотонное снижение скорости обучения Адаграда и улучшить сходимость. Мы можем написать правило обновления следующим образом:

$$g = \nabla_\theta \mathrm{L}\,(\theta;\ x^{(i)}; y^{(i)})\,, \qquad (4.40)$$
$$G = \ \beta G + (1-\beta)\cdot g \odot g$$
$$\theta = \ \theta \ - \frac{\alpha}{\sqrt{G+\varepsilon}} \odot g$$

где g представляет собой матрицу, накапливающую квадраты градиентов, ε — небольшая константа, добавляемая во избежание деления на ноль, и β— гиперпараметр скорости затухания.

Плюсы:

- Он может хорошо работать с разреженными данными.
- Автоматически регулирует скорость обучения на основе обновлений параметров.
- Может сходиться быстрее, чем Адаград.

Минусы:

- Из-за некоторых проблем он все еще может сходиться слишком медленно.
- Требует настройки гиперпараметра скорости затухания.

AdaDelta. AdaDelta — это алгоритм оптимизации, аналогичный RMSProp, но не требующий скорости обучения гиперпараметров . Вместо этого он использует **экспоненциально затухающее среднее значение** градиентов и квадраты градиентов для определения обновленного масштаба. Мы можем написать правило обновления следующим образом:

$$g = \nabla_\theta \mathrm{L}\,(\theta;\ x^{(i)}; y^{(i)})\,, \tag{4.41}$$
$$G = \beta G + (1-\beta)\cdot g \odot g$$
$$\Delta\theta = -\frac{\sqrt{S+\varepsilon}}{\sqrt{G+\varepsilon}} \odot g$$
$$\mathrm{S} = \beta \mathrm{S} + (1-\beta)\cdot \Delta\theta \odot \Delta\theta$$
$$\theta = \theta + \Delta\theta$$

где G и S являются матрицами, которые накапливают градиенты и квадраты обновлений соответственно, и ε — небольшая константа, добавленная во избежание деления на нуль.

Плюсы:

- Может хорошо работать с разреженными данными.
- Автоматически регулирует скорость обучения на основе обновлений параметров.

Минусы:

- Может сходиться слишком медленно из-за некоторых проблем.
- Может вообще прекратить обучение, если скорость обучения станет слишком низкой.

Оптимизатор АДАМ. Адам (сокращение от «адаптивная оценка момента») — это алгоритм оптимизации, сочетающий в себе идеи SGD с импульсом и RMSProp. Он **использует экспоненциально затухающее среднее значение градиентов** и квадраты градиентов для определения обновленного масштаба, аналогично RMSProp . Он также использует термин импульса, чтобы помочь оптимизатору более эффективно перемещаться по функции потерь. Правило обновления можно записать следующим образом:

$$g = \nabla_\theta \mathrm{L}\,(\theta;\ x^{(i)}; y^{(i)})\,, \tag{4.42}$$
$$m = \beta_1 m + (1-\beta_1)\cdot g$$
$$\upsilon = \beta_2 \upsilon + (1-\beta_2)\cdot g \odot g$$
$$\widehat{m} = \frac{m}{1-\beta_1^t},\ \widehat{\upsilon} = \frac{\upsilon}{1-\beta_2^t}\,, \tag{4.43}$$
$$\theta = \theta - \frac{\alpha}{\sqrt{\widehat{\upsilon}+\varepsilon}} \odot \widehat{m}$$

где m и υ – векторы импульса и скорости соответственно, β_1 и β_2– скорости затухания импульса и скорости.

Этот метод, который вычисляет скорость адаптивного обучения для каждого параметра. Он хранит как затухающее среднее значение прошлых градиентов, аналогичное импульсу, так и затухающее среднее значение прошлых квадратов градиентов, аналогично RMS-Prop и Adadelta. Таким образом, он сочетает в себе преимущества обоих методов.

Плюсы:

- Может сходиться быстрее, чем другие алгоритмы оптимизации.
- Может хорошо работать с зашумленными данными.

Минусы:

- Это может потребовать большей настройки гиперпараметров, чем другие алгоритмы.
- Может работать лучше при решении некоторых типов задач.

Как оптимизаторы работают в глубоком обучении? Оптимизаторы глубокого обучения корректируют параметры модели, чтобы минимизировать функцию потерь. Функция потерь измеряет, насколько хорошо модель может делать прогнозы для данного набора данных, а цель обучения модели — найти набор параметров модели, который дает минимально возможные потери.

Оптимизатор использует алгоритм оптимизации для поиска параметров, которые минимизируют функцию потерь. Алгоритм оптимизации использует градиенты функции потерь к параметрам модели, чтобы определить направление, в котором нам следует корректировать параметры.

Градиенты вычисляются **с** использованием обратного распространения ошибки, которое включает применение правила цепочки для вычисления градиентов функции потерь к каждому из параметров модели.

Затем алгоритм **оптимизации** корректирует параметры модели, чтобы минимизировать функцию потерь. Этот процесс повторяется до тех пор, пока функция потерь не достигнет минимума или оптимизатор не достигнет максимального количества разрешенных итераций.

На рисунке 2.25 представлено, реализация по сравнению сходимости само настраиваемых оптимизаторов SGD, SGD -momentum, Nag, Adagrad, Adadelta Adam и Rmsprop.

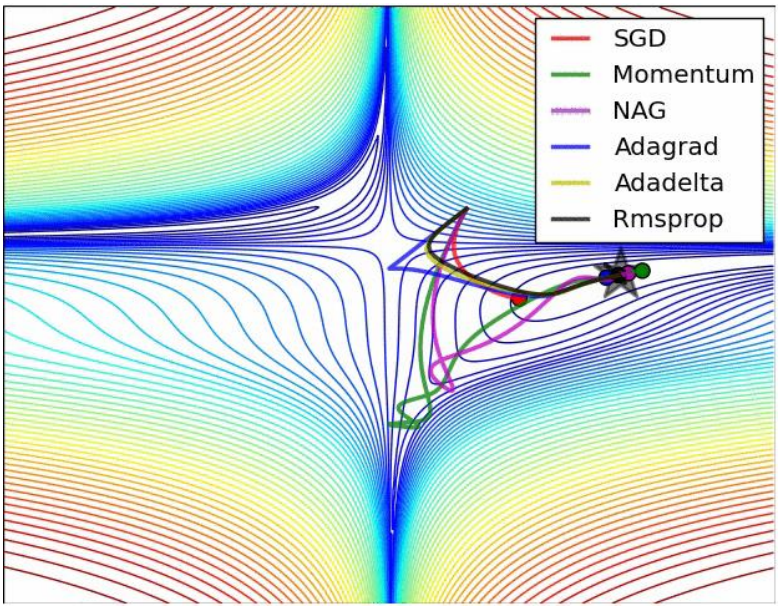


Рис.4.6. Результаты сравнения сходимости различных оптимизаторов.

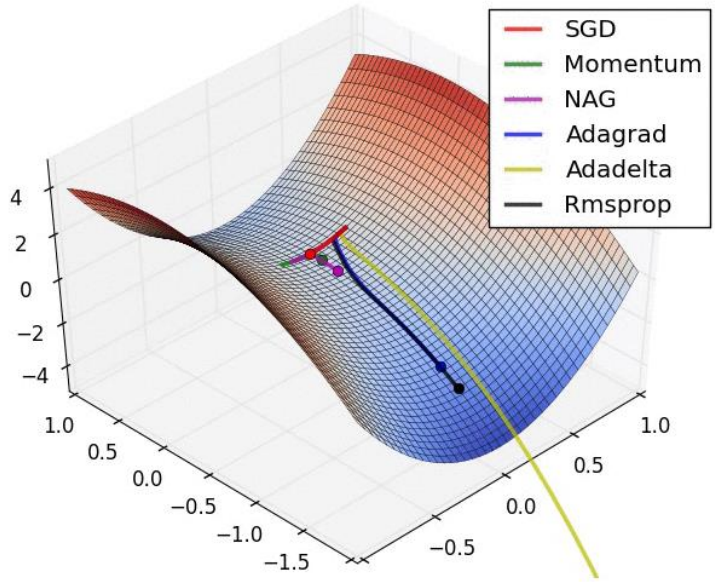


Рис.4.7. Результаты сравнения сходимости оптимизаторов седловой точки.

Рекомендации по выбору оптимизаторов при построении моделей.

В случае, когда данных мало, используют оптимизаторы Adagrad, Adadelta, RMSprop и Adam.

- Оптимизаторы Adadelta, RMSprop и Adam во многих случаях дают одинаковые результаты.
- Оптимизатор Adam отличается от RMSprop добавлением коррекции на смещения и импульса.
- В связи с этим Adam будет работать лучше, чем RMSprop.

Выводы

- Оптимизаторы в глубоком обучении необходимы, поскольку они корректируют параметры модели, чтобы минимизировать функцию потерь.
- В общем, выбор того, какой алгоритм оптимизации использовать, будет зависеть от конкретных характеристик проблемы, таких как размер набора данных и сложность модели.
- Важно тщательно рассмотреть плюсы и минусы каждого алгоритма и настроить все соответствующие гиперпараметры для достижения максимально возможной производительности.
- В целом, понимание роли оптимизаторов в глубоком обучении и различных доступных алгоритмов необходимо для создания и обучения эффективных моделей машинного обучения.

4.4. Реализации нейронной сети глубокого обучения для анализа урожайности сельскохозяйственных культур

В данном разделе мы достаточно подробно рассмотрим моделирования с применением нейронной сети для прогнозирования медианного значения прогнозируемой функции урожайности. Рассмотрим задачу будет ли цена на картофель, ячмень и корма ниже или выше среднего значения. Мы также изучим вопросы визуализации данных. Данные урожайности получены из районов Иссык-Кульского региона. Конкретно рассмотрим следующие вопросы. Изучение и обработка данных Создание и обучение нашей нейронной сети Визуализация потерь и точности Добавление регуляризации в нашу нейронную сеть. Работа с переобученными моделями.

Создадим нейронную сеть для создания модели и прогнозирования можно начинать с простого написания кода в пару строк.

В этом разделе мы рассмотрим также детали проектирования нейронной сети, использования пакета Keras, чтобы предсказать, будут ли урожайность выше или ниже ее медианного значения.

Программное обеспечение. Среда программного обеспечения состоит из блокнота Jupyter (в дальнейшем блокнот) системы Anaconda и должна быть настроена в среде, в которой установлены пакеты keras, tensorflow, pandas, scikit-learn и matplotlib . Ранее мы ввели некоторые интуитивное понимание нейронных сетей и того, как они работают, включая некоторые мелкие детали, такие как переобучение и методы по их устранению.

Набор данных, который мы будем использовать отражает урожайность сельскохозяйственных культур, по Иссык-Кульской области, картофеля, ячменя и различных кормовых культур. Количество входных функций для задачи прогнозирования сокращено до 17. Основными показателями служат урожайность, площадь засева и медианное значение, которая принимает значение true или false в зависимости от того будет ли урожайность выше или ниже медианного значения.

Изучение и обработка данных. Прежде чем мы закодируем какой-либо алгоритм машинного обучения, первое, что нам нужно сделать, это поместить наши данные в формат, который будет нужен алгоритму. В частности, нам необходимо:

Загрузить файл CSV (значения, разделенные запятыми) и преобразуем их в массивы. Массивы - это формат данных, который может обрабатывать наш алгоритм. Следующим шагом мы разделим наш набор данных на входные функции (которые мы называем x) и метку (которую мы называем y).

Масштабируем данные, мы называем это нормализацией, так, чтобы входные функции имели одинаковые порядки величины.

Разделим наш набор данных на обучающий набор, набор проверки и набор тестов. Мы должны были загрузить пакет pandas в свою среду. Нам нужно будет сообщить нашему блокноту, что мы будем использовать этот пакет, импортировав его. Введите следующий код и нажмите Alt-Enter на клавиатуре:

import pandas as pd

Это просто означает, что если я хочу сослаться на код в пакете pandas, мы его будем называть его именем pd. Затем мы читаем CSV-файл, выполнив следующую строку кода:

df = pd.read_csv ('harvest_full.csv')

Эта строка кода означает, что мы прочитаем csv-файл 'harvest_full.csv' (который должен находиться в том же каталоге, что и ваш блокнот) и сохраним его в переменной df. Если мы хотим узнать, что находится в df, просто введем df в серое поле и нажмем Alt-Enter:

df

Наш блокнот должен выглядеть так Листинг 5.1. :

Листинг 4.1. База данных harvest_full.csv.

```
import pandas as pd

df = pd.read_csv('harvest_full.csv')

df.head()
```

Структура базы данных имеет следующий вид.

	harvest	sunny days	rainy_days	total_1__pestisid	full_watering	cultivation
0	8450	7	5	856	2	1
1	9600	6	8	1262	2	0
2	11250	7	5	920	2	1
3	9550	7	5	756	1	0
4	14260	8	5	1145	2	1

Продолжение листинга 4.1.

foliar_application	root_ application	use_minerals	total_2_pestidsid	aboveMedian_harvest
3	8	0	548	1
3	6	1	460	1
3	6	1	608	1
3	7	1	642	0
4	9	1	836	1

База данных содержит следующие информации об урожайности картофеля за фиксированный год. Урожайность кг с гектара(harvest), количество активных солнечных дней в месяц (sunny days) и количество дождливых дней (rainy_days), весенняя обработка почвы пестицидом (total_1_pestidsid), полив -количество раз(full_watering),механическая обработка почвы(cultivation) ,листовая подкормка(foliar_application) ,корневая подкормка(root_application),использование минеральных удобрений(use_minerals),осенняя обработка почвы пестидцидом(total_2_pestidsid) и зависимая прогнозируемая переменная о медианном значении урожайности (aboveMedian_harvest). Таким образом будем изучать бинарную задачу классификации, будет ли урожайность выше среднего. Данная задача относится к задаче классификации. Построим нейронную сеть прогноза урожайности для данной бинарной задачи классификации.

В нашем последнем столбце у нас есть функция aboveMedian_harvest , которую мы хотели бы предсказать:

Урожайность выше медианы или нет? (1 - да и 0 - нет)

После создания базы данных их преобразуем в массивы для машинной обработки.Создаем фрейм

dataset = df.values

Чтобы преобразовать наш фрейм данных в массив, мы просто сохраняем значения df (путем доступа к df.values) в переменную dataset . Посмотрим, что находится внутри этой переменной «набор данных», введя «dataset» в поле в записной книжки и запустим ячейку (Alt-Enter). Теперь наш набор данных преобразуем в Фреймворке pandas. Чтобы преобразовать его в массив, просто получим доступ к его значениям:

Листинг 4.2. Просмотр данных базы

```
dataset = df.values
```

```
dataset
```

```
array([[ 8450,     7,     5, ...,     0,   548,     1],
       [ 9600,     6,     8, ...,     1,   460,     1],
       [11250,     7,     5, ...,     1,   608,     1],
       ...,
       [ 9042,     7,     9, ...,     2,   252,     1],
       [ 9717,     5,     6, ...,     0,   240,     0],
       [ 9937,     5,     6, ...,     0,   276,     0]], dtype=int64)
```

Преобразование нашего фрейма данных в массив. Теперь мы разделим наш набор данных на входные объекты (X) и объект, который мы хотим предсказать (Y). Чтобы сделать это разделение, мы просто назначаем первые 10 столбцов нашего массива переменной с именем X, а последний столбец нашего массива - переменной с именем Y. Код для выполнения первого назначения следующий:

X = dataset[:,0:10]

Данный код можно объяснить следующим образом, то есть объясним, что находится внутри квадратных скобок. Все, что находится до запятой, относится к строкам массива, а все, что после запятой, относится к столбцам массивов.

Поскольку мы не разделяем строки, мы ставим ":" перед запятой. Это означает взять все строки в наборе данных и поместить их в X.

Мы хотим извлечь первые 10 столбцов, поэтому «0:10» после запятой означает, что нужно взять столбцы от 0 до 9 и поместить их в X (мы не включаем столбец 10). Наши столбцы начинаются с индекса 0, поэтому первые 10 столбцов на самом деле являются столбцами от 0 до 9.

Затем мы присваиваем последний столбец нашего массива Y:

Y = dataset[:,10]

Таким образом мы разделили наш набор данных на входные функции (X) и метку того, что мы хотим предсказать (Y).

Следующий шаг обработки базы данных, это нормировка данных. Нормализация наших данных очень важна, поскольку мы хотим, чтобы входные функции были одного порядка, чтобы облегчить наше обучение. Мы будем использовать масштабизатор - min-max из scikit-learn, который масштабирует наши данные между 0 и 1. Мы должны убедиться, что масштаб входных функций не должны имеет различные высокие дисперсии, например от среднего. В нашей базе данных такие характеристики, как урожайность растения, измеряются тысячами кг за гектар, а обработка пестицидами колеблется до 1000 гр за квадратный метр. Точно так же оценка общего полива или количество обработки почвы варьируется от 0 до 5, а количество активных солнечных дней или дождливых дней колеблется до 10 дней, культивация почвы, как правило, составляет 0, 1 или 2.

Такие данные затрудняют инициализацию нейронной сети, что вызывает некоторые практические проблемы точности модели. Один из способов масштабирования данных - использовать существующий пакет из scikit-learn.

Сначала нам нужно импортировать код, который мы хотим использовать:

from sklearn import preprocessing

Это говорит о том, что мы хотим использовать код в «предварительной обработке»- preprocessing в пакете sklearn. Затем мы используем функцию под названием min-max scaler, которая масштабирует набор данных так, чтобы все входные функции лежали между 0 и 1 включительно. Вот код

```
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
```

Теперь наш масштабированный набор данных хранится в массиве X_scale. Для просмотра данных , как выглядит 'X_scale', просто запустим ячейку: X_scale.

Наш блокнот Jupyter теперь должен выглядеть примерно так:

```
X = dataset[:,0:10] Y = dataset[:,10]
```

Листинг 4.3. Нормировка данных

```
from sklearn import preprocessing
```

```
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
```

```
X_scale
```

```
array([[0.0334198 , 0.66666667, 0.5       , ..., 0.5       , 0.        ,
        0.3864598 ],
       [0.03879502, 0.55555556, 0.875     , ..., 0.33333333, 0.33333333,
        0.32440056],
       [0.04650728, 0.66666667, 0.5       , ..., 0.33333333, 0.33333333,
        0.42877292],
       ...,
       [0.03618687, 0.66666667, 1.        , ..., 0.58333333, 0.66666667,
        0.17771509],
       [0.03934189, 0.44444444, 0.625     , ..., 0.25      , 0.        ,
        0.16925247],
       [0.04037019, 0.44444444, 0.625     , ..., 0.33333333, 0.        ,
        0.19464034]])
```

Теперь мы подошли к нашему последнему шагу в обработке данных, который состоит в том, чтобы разделить наш набор данных на обучающий набор, набор для проверки и набор тестов.

Мы будем использовать код из scikit-learn под названием «train_test_split», который, как следует из названия, разбивает наш набор данных на обучающий набор и тестовый набор. Сначала импортируем нужный нам код:

from sklearn.model_selection import train_test_split

Затем разделите набор данных следующим образом:

X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)

Это сообщает scikit-learn, что ваш размер val_and_test будет составлять 30% от общего набора данных. Код сохранит разделенные данные в первых четырех переменных слева от знака равенства, как предполагают имена переменных.

К сожалению, эта функция помогает нам только разделить наш набор данных на два. Поскольку нам нужен отдельный набор проверки и набор тестов, мы можем использовать ту же функцию, чтобы снова выполнить разделение на val_and_test:

X_val, X_test, Y_val, Y_test = train_test_split (X_val_and_test, Y_val_and_test, test_size = 0,5)

Приведенный выше код разделит размер val_and_test поровну на набор проверки и набор тестов.

Таким образом, теперь у нас есть шесть переменных для наших наборов данных, которые мы будем использовать:

X_train (10 входных функций, 70% от полного набора данных)

X_val (10 входных функций, 15% от полного набора данных)

X_test (10 входных функций, 15% от полного набора данных)

Y_train (1 метка, 70% от полного набора данных)

Y_val (1 метка, 15% от полного набора данных)

Y_test (1 метка, 15% от полного набора данных)

Теперь мы посмотрим , как выглядят массивы для каждого из них (то есть какие они имеют размерности), просто запустим код.

print (X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)

Результаты на нашем блокноте Jupyter , все это выглядит следующим образом:

Листинг 4.4. Разделения данных на обучающее и тестирования.

```
from sklearn.model_selection import train_test_split

X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.

X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.

print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)
(1022, 10) (219, 10) (219, 10) (1022,) (219,) (219,)
```

Как видите, обучающий набор содержит 1022 точки данных, в то время как набор для проверки и тестирования имеет 219 точек данных. Переменные X имеют 10 входных функций, тогда как переменные Y могут прогнозировать только одну функцию.

Таким образом мы подготовили данные к построению нейронной сети. При обработке данных мы сделали следующие шаги:

Загрузили файл CSV (значения, разделенные запятыми) и преобразовали их в массивы.

Разделили наш набор данных на входные объекты и метку-прогнозируемую функцию меток. Масштабировали данные так, чтобы входные функции имели одинаковые порядки величины. Разделите наш набор данных на три: обучающий набор, набор проверки и набор тестов.

Создание и обучение нейронной сети

Теперь приступим к созданию нейронной сети. Обычно построение нейронной сети в машинном обучении состоит из двух этапов. Первый шаг - указать шаблон, построить архитектуру сети, а второй шаг - найти лучшие числа, параметры сети из данных для заполнения этого шаблона. Наш код с этого момента также будет выполнять эти два шага.

Первый шаг: настройка архитектуры

Первое, что нам нужно сделать, это настроить архитектуру. Давайте сначала подумаем, какую архитектуру нейронной сети мы хотим. Предположим, нам нужна эта нейронная сеть:

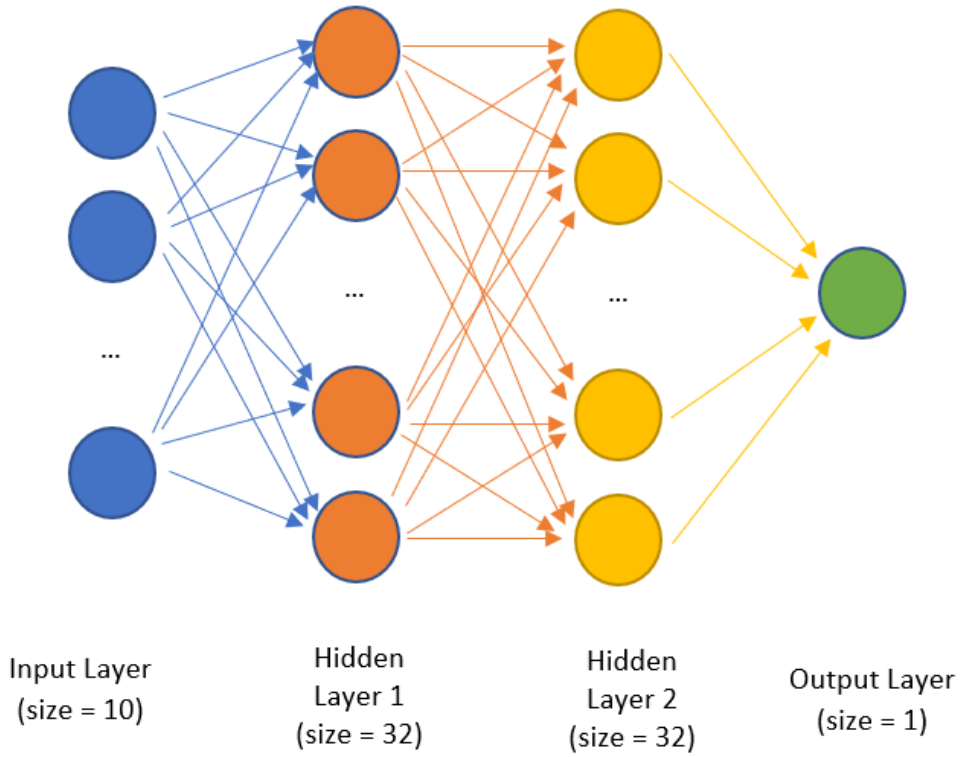


Рис.4.8. Архитектура нейронной сети, которую мы будем использовать для нашей задачи

На словах мы хотим иметь следующие слои:

Скрытый слой 1: 32 нейрона, активация ReLU

Скрытый слой 2: 32 нейрона, активация ReLU

Выходной слой: 1 нейрон, сигмовидная активация

Теперь нам нужно описать эту архитектуру Керасу. Мы будем использовать последовательную модель, что означает, что нам просто нужно последовательно описать вышеперечисленные слои.

Для начала импортируем необходимый код из Keras:

```
from keras.models import Sequential
from keras.layers import Dense
```

Затем мы указываем это в нашей последовательной модели Keras следующим образом:

```
model = Sequential([
 Dense(32, activation='relu', input_shape=(10,)),
 Dense(32, activation='relu'),
```

Dense(1, activation='sigmoid'),])

И вот так фрагмент кода, приведенный выше, определил нашу архитектуру! Приведенный выше код можно интерпретировать так:

model = Sequential([...])

Это говорит о том, что мы будем хранить нашу модель в переменной «модель», и мы будем описывать ее последовательно (слой за слоем) в квадратных скобках.

Dense(32, activation='relu', input_shape=(10,)),

У нас есть наш первый слой в виде плотного слоя с 32 нейронами, активацией ReLU и формой ввода 10, поскольку у нас есть 10 входных функций. Обратите внимание, что «Dense» относится к полностью подключенному слою, который мы и будем использовать.

Dense(32, activation='relu'),

Наш второй слой также представляет собой плотный слой с 32 нейронами, активация ReLU. Обратите внимание, что нам не нужно описывать входную форму, поскольку Keras может сделать вывод из выходных данных нашего первого слоя.

Dense(1, activation='sigmoid'),

Наш третий слой - это плотный слой с 1 нейроном, активация сигмовидной функцией.

И вот так мы написали архитектуру модели (шаблон) в коде!

Второй шаг: Для настройки модели необходимо параметры модели

После определения архитектуры нейронной сети, нам нужно найти для нее параметры. Прежде чем мы начнем обучение, мы должны настроить модель. Сначала мы определим , алгоритм для оптимизации, функцию потерь и метрики точности.

Для настройки модели с этими параметрами необходимо вызвать функцию model.compile:

model.compile (optimizer = 'sgd',
loss = 'binary_crossentropy',
metrics = [' precision '])

Мы помещаем следующие настройки в скобки после model.compile:

optimizer='sgd'

«sgd» относится к стохастическому градиентному спуску (здесь это относится к мини-пакетному градиентному спуску), который мы изучили в главе 3.Определим функцию потерь loss='binary_crossentropy'

Функция потерь для выходов, принимающих значения 1 или 0, называется бинарной кросс-энтропией. Метрика определяется как, accuracy

metrics=['accuracy']

Наконец, мы хотим отслеживать точность поверх функции потерь. Теперь обучим наш модель. Пишем код для обучения модели на данных:

hist = model.fit (X_train, Y_train,
batch_size = 32 , epochs = 100,
validation_data = (X_val, Y_val))

Функция fit называется «подгонкой », поскольку мы подбираем параметры к данным. Мы должны указать, на каких данных мы тренируемся, это X_train и Y_train . Затем мы указываем размер нашей мини-партии и как долго мы хотим ее тренировать (эпохи). Наконец, мы указываем, какие у нас данные проверки, чтобы модель сообщала нам, как мы поступаем с данными проверки в каждой точке. Эта функция выведет историю, которую мы сохраняем под переменной hist. Мы будем использовать эту переменную немного позже, когда перейдем к визуализации. Обучим модели на данных X_train и Y_train, которые мы разделили выше. В качестве batch_size = 32 , epochs = 100, а данные валидации из множества validation_data = (X_val, Y_val))

Теперь запустим Jupyter и посмотрим, процесс обучения модели. Выполнение программы должен выглядеть так:

Листинг 4.5. Обучение модели на данных X_train и Y_train.

```
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 6ms/step - loss: 0.7047 - accuracy: 0.5010 - val_loss: 0.6913 - val_accuracy: 0.5
342
Epoch 2/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6813 - accuracy: 0.6820 - val_loss: 0.6805 - val_accuracy: 0.6
393
Epoch 3/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6684 - accuracy: 0.6487 - val_loss: 0.6724 - val_accuracy: 0.6
256
Epoch 4/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6580 - accuracy: 0.6751 - val_loss: 0.6651 - val_accuracy: 0.6
347
Epoch 5/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6487 - accuracy: 0.6683 - val_loss: 0.6580 - val_accuracy: 0.6
804
Epoch 6/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6401 - accuracy: 0.6967 - val_loss: 0.6510 - val_accuracy: 0.7
078
Epoch 7/100
```

Таким образом, у нас мы увидим, что, уменьшается потеря и повышается точности с течением эпохи. На этом этапе мы можем поэкспериментировать с гиперпараметрами и архитектурой нейронной сети. Снова запуская процесс обучения, чтобы увидеть, как изменилось ваше обучение после настройки гиперпараметров.

В результате мы окончательно выберем модель, и сможем оценить ее на тестовом наборе. Чтобы определить точность нашего тестового набора, мы запускаем следующий фрагмент кода:

model.evaluate (X_test, Y_test) [1]

Причина, по которой у нас есть индекс 1 после функции model.evaluate, заключается в том, что функция возвращает потерю как первый элемент и точность как второй элемент. Чтобы вывести только точность, просто обратитесь ко второму элементу (который индексируется 1, поскольку первый элемент начинает свою индексацию с 0).

Из-за случайности того, как мы разделили набор данных, а также из-за инициализации весов, числа и график будут немного отличаться при каждом запуске нашей записной книжки. Тем не менее, вы должны получить точность теста от 88% до 95%.

Листинг 4.6. Проверка точности модели на тестовых данных

```
In [18]: model.evaluate(X_test, Y_test)[1]
         219/219 [==============================] - 0s 14us/step
Out[18]: 0.8858447523966227
```

Оценка на тестовом наборе получилось неплохо. Таким образом, с использованием нейронной технологии создано модель, которая решает бинарную задачу, будет ли урожайность выше медианного значения. Здесь архитектуру сети определяем с помощью модели Keras Sequential.

Мы указываем некоторые из наших настроек (оптимизатор, функцию потерь, показатели для отслеживания) с помощью model.compile.

Мы обучаем нашу модель и находим лучшие параметры для нашей архитектуры с обучающими данными с помощью model.fit.

Оцениваем нашу модель на тестовом наборе с помощью model.evaluate

Визуализация потерь и точности. Переобучении и некоторых методах регуляризации.

Что мы могли бы сделать, так это построить график потерь при обучении и потерь val по количеству прошедших эпох. Чтобы отобразить красивые графики, мы будем использовать пакет matplotlib. Как обычно, нам нужно импортировать код, который мы хотим использовать:

import matplotlib.pyplot as plt

Затем мы хотим визуализировать потерю обучения и потерю проверки. Для этого запустите этот фрагмент кода:

Листинг 4.7. Реализация модели

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Ошибка модели')
plt.ylabel('Ошибка')
plt.xlabel('Эпоха')
plt.legend(['Обучающие данные', 'Данные валидации'], loc='upper right')
plt.show()
```

Мы объясним каждую строку приведенного выше фрагмента кода. Первые две строки говорят, что мы хотим построить график потерь и val_loss. В третьей строке указывается заголовок этого графика «Модельные потери». Четвертая и пятая строки говорят нам, какие оси y и x должны быть помечены соответственно. Шестая строка включает легенду для нашего графика, и расположение легенды будет в верхнем правом углу. А седьмая строка указывает блокноту Jupyter отображать график.

Результат блокнота Jupyter должен выглядеть примерно так:

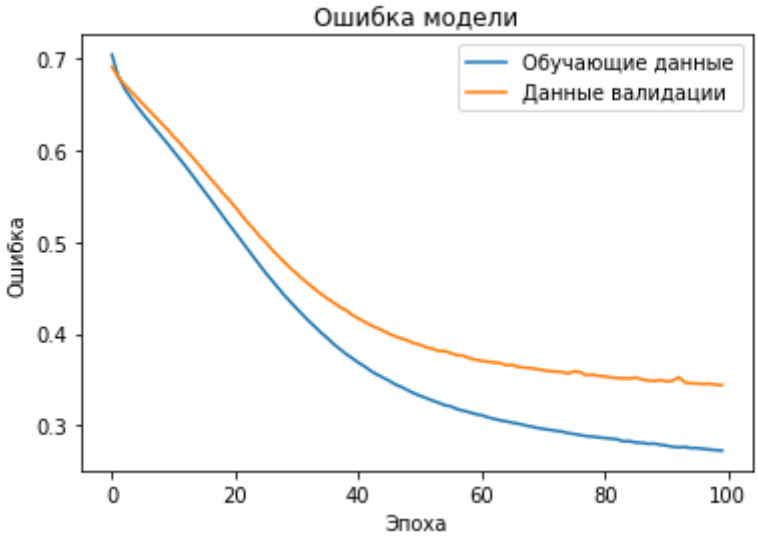


Рис. 4.9. График потери модели

Мы можем сделать то же самое для построения графика точности обучения и точности проверки с помощью кода, как и выше. У вас должен получиться график, который выглядит примерно так:

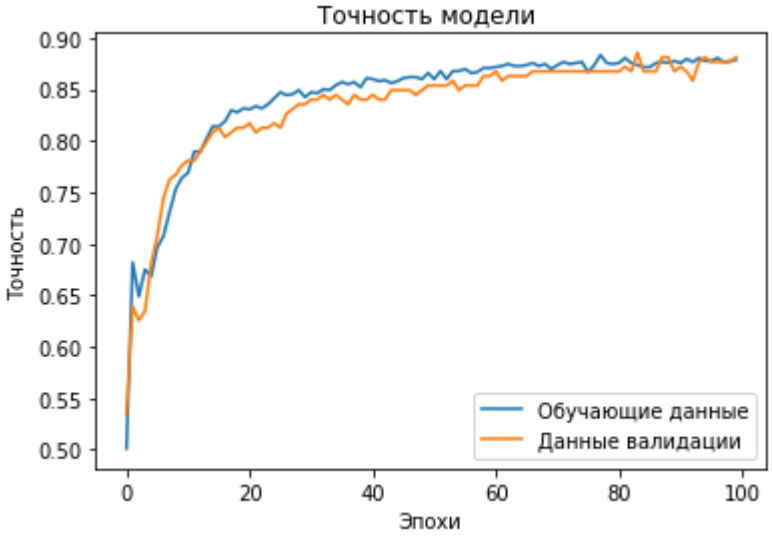


Рис. 4.10. График точности модели для набора для обучения и проверки

Поскольку улучшения в нашей модели для обучающего набора выглядят в некоторой степени совпадающими с улучшениями для набора для проверки, не похоже, что переобучение - огромная проблема для нашей модели.

мы используем matplotlib для визуализации потери / точности обучения и проверки с течением времени, чтобы увидеть, есть ли переоснащение в нашей модели.

Добавление регуляризации в нейронную сеть. Чтобы ввести регуляризацию в нашу нейронную сеть, давайте сформулируем ее с помощью нейронной сети, которая будет плохо соответствовать нашему обучающему набору. Мы назовем эту модель 2.

Листинг 4.8. Реализация модели регуляризации

```
model_2 = Sequential([
    Dense(1000, activation='relu', input_shape=(10,)),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1, activation='sigmoid'),
])
 model_2.compile (optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = [' precision '])
hist_2 = model_2.fit (X_train, Y_train,
          batch_size = 32 , epochs = 100,
          validation_data = (X_val, Y_val))
```

Если мы запустим этот код и построим графики потерь для hist_2, используя приведенный ниже код (обратите внимание, что код тот же, за исключением того, что мы используем.

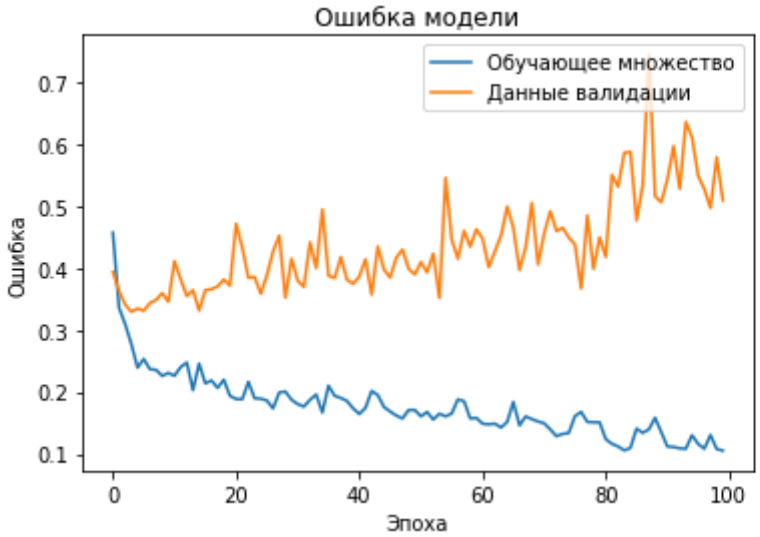


Рис. 4.11. Кривые потерь для модели с переобучением

Это явный признак переобучения. Потери в обучении уменьшаются, но потери при проверке намного превышают потери в обучении и увеличиваются (после точки перегиба эпохи 20). Если мы построим график точности, используя приведенный ниже код:

Мы также можем видеть более четкое расхождение между точностью обучения и точности проверки при обучении модели.

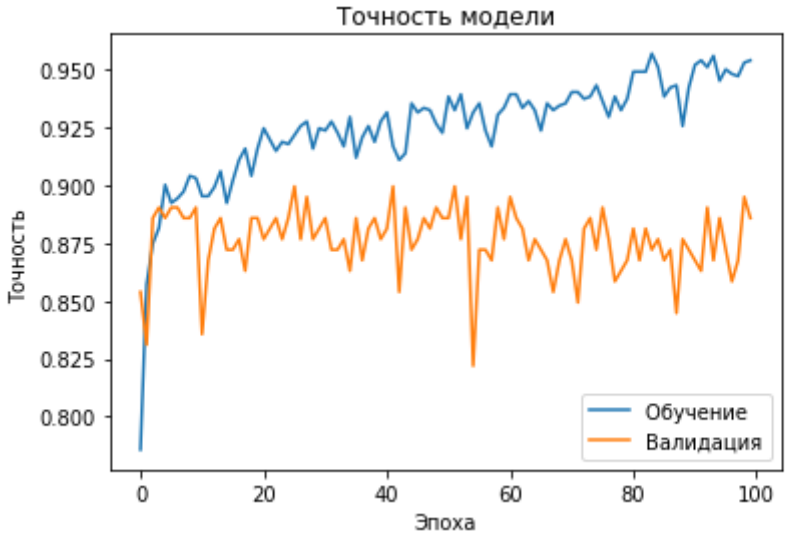


Рис. 4.12. Точность обучения и проверки нашей модели переобучения

Теперь давайте попробуем некоторые из наших стратегий, чтобы уменьшить чрезмерную подгонку (помимо возврата нашей архитектуры к нашей первой модели). Из трех здесь мы включим регуляризацию L2 и отсев. Причина, по которой мы не добавляем здесь раннюю остановку, заключается в том, что после того, как мы использовали первые две стратегии, потеря проверки не принимает U-образную форму, которую мы видим выше, и поэтому ранняя остановка не будет столь же эффективной. Во-первых, давайте импортируем код, который нам нужен для регуляризации и отключения L2:

from keras.layers import Dropout
from keras import Regularizers

Затем мы определяем нашу третью модель следующим образом:

Листинг 4.9. Реализация модели регуляризации в L^2

```
model_3 = Sequential([
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01), input_shape=(10,)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1, activation='sigmoid', kernel_regularizer=regularizers.l2(0.01)),
])
```

Можете ли вы заметить различия между Model 3 и Model 2? Есть два основных отличия:

Различие 1 : чтобы добавить регуляризацию L2, обратите внимание, что мы добавили немного дополнительного кода в каждый из наших плотных слоев, например:

kernel_regularizer=regularizers.l2(0.01)

Это говорит Keras включить квадраты значений этих параметров в нашу общую функцию потерь и взвесить их на 0,01 в функции потерь.

Отличие 2 : чтобы добавить Dropout, мы добавили новый слой, подобный этому:

Dropout(0.3),

Это означает, что нейроны в предыдущем слое с вероятностью 0,3 выпадут во время обучения. Давайте скомпилируем его и запустим с теми же параметрами, что и наша Модель 2 (переобученная): Листинг 4.10. Реализация модели с добавлением нового слоя

```
model_3.compile (optimizer = 'adam',
                 loss = 'binary_crossentropy',
                 metrics = [' precision '])
hist_3 = model_3.fit (X_train, Y_train,
           batch_size = 32 , epochs = 100,
           validation_data = (X_val, Y_val))
```

А теперь построим графики потерь и точности. Вы заметите, что вначале потери намного выше, и это потому, что мы изменили нашу функцию потерь. Чтобы построить так, чтобы окно увеличивалось в диапазоне от 0 до 1,2 для потерь, мы добавляем дополнительную строку кода (plt.ylim) при построении.

Листинг 4.11. Реализация ошибки модели

```
plt.plot(hist_3.history['loss'])
plt.plot(hist_3.history['val_loss'])
plt.title('Ошибка модели')
plt.ylabel('Ошибка')
plt.xlabel('Эпоха')
plt.legend(['Обуающие данные', 'Данные проверки'], loc='upper right')
plt.ylim(top=1.2, bottom=0)
plt.show()
```

У нас получится график потерь, который выглядит так:

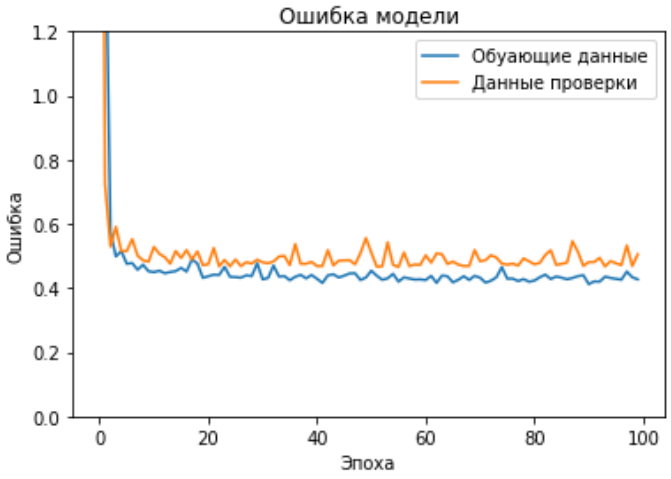


Рис.4.13. Уточненная ошибка модели

Вы можете видеть, что потеря валидации намного больше соответствует нашей потере обучения. Давайте построим график точности с помощью аналогичного фрагмента кода:

Листинг 4.12. Точность модели. Улучшенный вариант.

```
plt.plot(hist_3.history['accuracy'])
plt.plot(hist_3.history['val_accuracy'])
plt.title('Точность модели')
plt.ylabel('Точность')
plt.xlabel('Эпоха')
plt.legend(['Обучающие данные', 'Данные валидации'], loc='lower right')
plt.show()
```

И у нас получится такой сюжет:

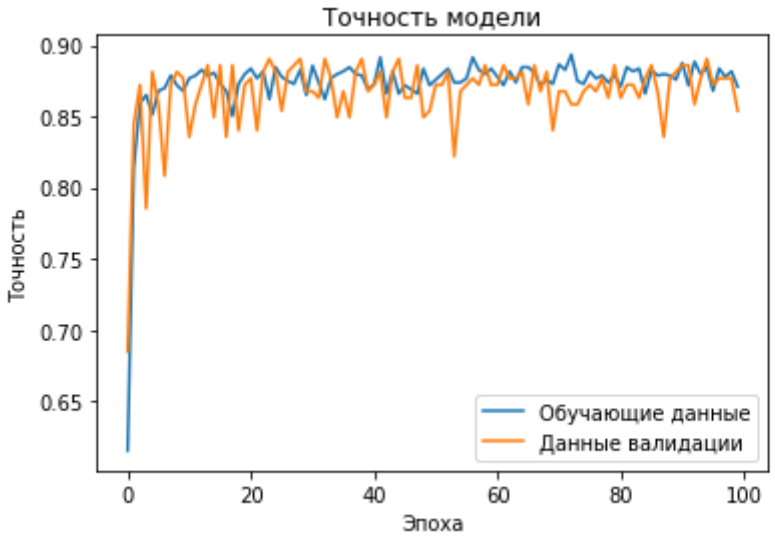


Рис.4.14. Улучшенный вариант регуляризации

По сравнению с нашей моделью в Модели 3, мы значительно уменьшили переобучение. Таким образом мы применили методы регуляризации для решения проблемы с переобучения модели т.е. чтобы уменьшить переобучение обучающей выборке.

Отметим, что очень важно чтобы уменьшить переобучение и справиться с переобучением в моделях, мы можем закодировать в нашей модели следующие стратегии, каждая из которых содержит примерно одну строку кода:

- L2 Regularization
- Dropout

Если мы визуализируем training / validation loss , training / accuracy loss мы увидим, что эти дополнения помогли справиться с переобучением.

Таким образом, мы рассмотрели технологию построения нейронной сети с применением библиотек Python. На основе Keras мы реализовали нейронную сеть, которая строить различные модели. Рассмотрели следующие вопросы

- Изучили и обработали реальные данные, заданные в виде .csv файла
- Создали и обучили нейронную сеть
- Визуализировали потерю и точность моделей
- Для переобученных моделей использовали регуляризацию в нейронную сеть

Таким образом мы построили полноценную нейронную сеть, которую можно реализовать для прикладных задач. Для эксперимента моделей с различными архитектурами необходимо только рассмотреть просто вопрос, замены разных слоев и изменения разных гиперпараметров. Мы убедились, что Keras действительно значительно упростил создание наших нейронных сетей, и мы продолжим использовать его для более сложных приложений в области компьютерного зрения и обработки естественного языка.

4.5. Интеграция сверточных нейронных сетей с алгоритмами машинного обучения

В данном разделе рассмотрим многоклассовую задачу классификации и сравним результаты, полученные нейронными сетями и алгоритмами машинного обучения. Для реализации будем использовать сверточный слой Conv2D с функцией активации LeakyReLU,BatchNormalization,

MaxPooling2D, слой для перехода из CNN в полносвязанный слой - Flatten,плотный слой-Dense,метод регуляризации или слой дезактивации нейронов - Dropout. Мы также загружаем множество необходимых пакетов для обучения модели. И наконец мы загружаем классификаторы метода опорных вектров и градиентного бустинга-xgboost для интеграции сверточных нейронных сетей с алгоритмами машинного обучения и сравним результаты метода CNN и алгоритмами машинного обучения. Для реализации необходимо загружать пакеты для реализации CNN. У нас имеются два каталога содержащие базу данных.

Листинг 4.14. Каталоги данных для создания модели

```
import pickle
import cv2
import os
from os import listdir
from sklearn.preprocessing import LabelBinarizer,MultiLabelBinarizer
from sklearn.model_selection import train_test_split
```

```
root_dir = r'D:/New Plant Diseases Dataset(Augmented)'

os.chdir(root_dir)
listdir()
```

```
['train', 'valid']
```

В этих каталогах содержатся база данных болезни сельскохозяйственных растений. Введем размеры измененного изображения и количество классов изображений, используемых для обучения модели. Мы используем функцию преобразования, чтобы изменить размер изображения до размера «DEFAULT_IMAGE_SIZE», который мы определим ниже.

DEFAULT_IMAGE_SIZE = tuple((256, 256))
N_IMAGES = 100
data_dir = os.path.join(root_dir,'train')
Загружаем все данные и определяем классы растений

```
print("Количество классов изображений  ...")
plant_disease_folder_list = listdir(data_dir)
print(len(plant_disease_folder_list))
```

```
Количество классов изображений  ...
37
```

Таким образом, у нас имеется 37 классов, типов болезней растений. Загрузим их. Загрузка данных обучения и проверки. База данных изображений ...

Листинг 4.15. Процесс загрузки данных болезни растений

```
База данных изображений ...
Processing  Apple___Apple_scab ...
Processing  Apple___Black_rot ...
Processing  Apple___Cedar_apple_rust ...
Processing  Apple___healthy ...
Processing  Blueberry___healthy ...
Processing  Cherry_(including_sour)___healthy ...
Processing  Cherry_(including_sour)___Powdery_mildew ...
Processing  Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot ...
Processing  Corn_(maize)___Common_rust_ ...
Processing  Corn_(maize)___healthy ...
Processing  Corn_(maize)___Northern_Leaf_Blight ...
Processing  Grape___Black_rot ...
Processing  Grape___Esca_(Black_Measles) ...
Processing  Grape___healthy ...
Processing  Grape___Leaf_blight_(Isariopsis_Leaf_Spot) ...
Processing  Peach___Bacterial_spot ...
Processing  Peach___healthy ...
Processing  Pepper_bell___Bacterial_spot ...
Processing  Pepper_bell___healthy ...
Processing  Potato___Early_blight ...
Processing  Potato___healthy ...
Processing  Potato___Late_blight ...
Processing  Raspberry___healthy ...
Processing  Soybean___healthy ...
Processing  Squash___Powdery_mildew ...
Processing  Strawberry___healthy ...
Processing  Strawberry___Leaf_scorch ...
Processing  Tomato___Bacterial_spot ...
Processing  Tomato___Early_blight ...
Processing  Tomato___healthy ...
Processing  Tomato___Late_blight ...
Processing  Tomato___Leaf_Mold ...
Processing  Tomato___Septoria_leaf_spot ...
Processing  Tomato___Spider_mites Two-spotted_spider_mite ...
Processing  Tomato___Target_Spot ...
Processing  Tomato___Tomato_mosaic_virus ...
Processing  Tomato___Tomato_Yellow_Leaf_Curl_Virus ...
```

Все изображения успешно загружены. База данных проверки имеют аналогичные названия. Всего изображений обучения и проверки:

print(len(image_list)) 7400

Преобразуем загруженные данные тренировочного изображения в массив numpy и проверяем количество загруженных изображений для обучения

Листинг 4.16. Определения количества изображений

```
np_image_list = np.array(image_list, dtype=np.float16) / 255.0
image_len = len(image_list)
print(f"Total number of images: {image_len}")
```

```
Total number of images: 7400
```

Каждому классу теперь присвоим метки. Это называется горячим кодированием. Изучим метки/классы в обучающем наборе данных.

Листинг 4.17. Определения количества классов

```
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)

# pickle.dump(label_binarizer,open('plant_disease_label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)

print("Total number of classes: ", n_classes)
```

```
Total number of classes:  37
```

Теперь данные обучения увеличим и разделим полученные набор данных. При этом будем использовать функцию ImageDataGenerator для увеличения данных путем выполнения различных операций с тренировочными изображениями. Вот программный код данных операций.

Листинг 4.18. Программный код увеличения данных

```
augment = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
                             height_shift_range=0.1, shear_range=0.2,
                             zoom_range=0.2, horizontal_flip=True,
                             fill_mode="nearest")
```

Разделение данных на обучающие и тестовые наборы для целей проверки.

Листинг 4.19. Процесс разделения данных

```
x_train, x_test, y_train, y_test = train_test_split(np_image_list,
                    image_labels, test_size=0.2, random_state = 42)
print('Successfully split data into TRAIN & TEST')
```

Successfully split data into TRAIN & TEST

Теперь построим модель. Определим гиперпараметры модели *классификации болезней растений.*

```
EPOCHS = 10
STEPS = 100
LR = 1e-3
BATCH_SIZE = 32
WIDTH = 256
HEIGHT = 256
DEPTH = 3
```

Далее создаем полносвязанный нейронную сеть и на основе ее построим модель CNN для многоклассовой классификации. Обучим модель. Модель обучения состоит из инициализации оптимизатора Adam с параметрами скорости обучения и затухания. Также выбираем тип потерь и метрики для модели и компилируем ее для обучения. Вот процесс инициализации оптимизатора, компиляции и обучение модели.

Листинг 4.21. Обучение модели нейронной сети

```
opt = Adam(learning_rate=LR, decay=LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
print("Training CNN...")
history = model.fit(augment.flow(x_train, y_train, batch_size=BATCH_SIZE),
                                 validation_data=(x_test, y_test),
                                 epochs=20,
                                 verbose=1)

scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
Training CNN...
Epoch 1/20
185/185 [==============================] - 347s 2s/step - loss: 0.4338 - accuracy: 0.1115 - val_loss: 0.1889 - val_accuracy: 0.
0304
Epoch 2/20
185/185 [==============================] - 353s 2s/step - loss: 0.1061 - accuracy: 0.3723 - val_loss: 0.1355 - val_accuracy: 0.
0939
```

Epoch 1/20

185/185 [==============================] - 347s 2s/step - loss: 0.4338 - accuracy: 0.1115 - val_loss: 0.1889 - val_accuracy: 0.0304

……

Epoch 20/20

185/185 [==============================] - 347s 2s/step - loss: 0.0159 - accuracy: 0.9375 - val_loss: 0.0979 - val_accuracy: 0.5986

47/47 [==============================] - 17s 363ms/step - loss: 0.0979 - accuracy: 0.5986

Test Accuracy: 59.864866733551025

Оценка модели Сравнение точности и потерь путем построения графика для обучения и проверки. Вот график точности обучения и проверки.

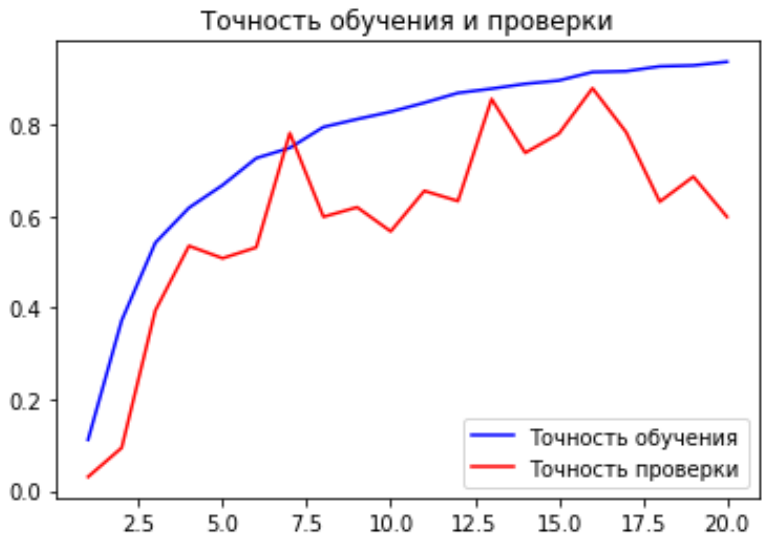


Рис.4.15. График точности модели на обучающих и тестовых множествах

Аналогично ошибка модели при обучении и проверке.

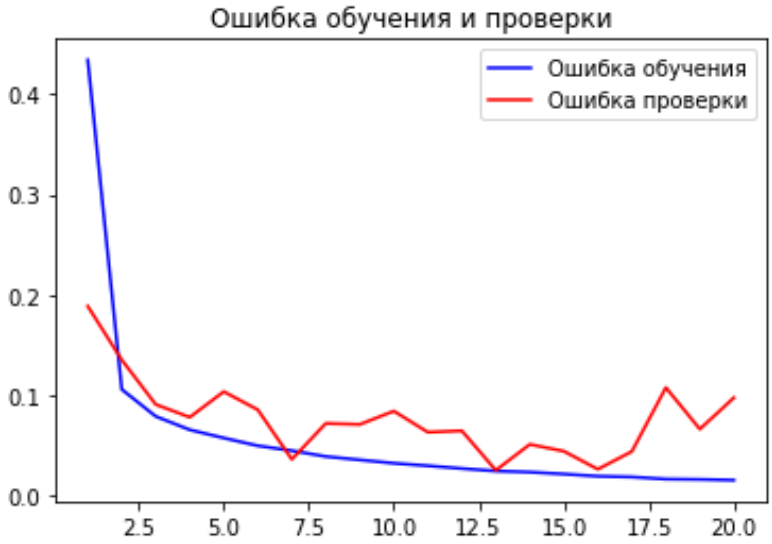


Рис.4.16. График поведения ошибки модели на обучающих и тестовых множествах

Оценим сначала точность модели с использованием метода «оценить.

Листинг 4.22. Оценка точности обученной модели

```
print("[INFO] Вычисление точности модели")
scores = model.evaluate(x_test, y_test)
print(f"Точность при тестировании: {scores[1]*100}")
```

[INFO] Вычисление точности модели

47/47 [==============================] - 16s 343ms/step - loss: 0.0979 - accuracy: 0.5986

Точность при тестировании: 59.864866733551025

Листинг 4.23. Проверка модели на тестовых множествах

```
x_train_predict = intermediate_layer_model.predict(x_train)
x_test_predict = intermediate_layer_model.predict(x_test)
print(x_test_predict.shape)
```

(1480, 1024)

Интеграция CNN с алгоритмом машинного обучения метода опорных векторов

Для интеграции и сравнительного анализа методов глубокого обучения и машинного обучения рассмотрим сначала машину опорных векторов. Пишем код интеграция CNN с машиной опорных векторов.

Листинг 4.24. Интеграция CNN с методом опорных векторов. Обучение модели

```
from sklearn.svm import SVC
svm = SVC(kernel='rbf')
svm.fit(x_train_predict,np.argmax(y_train,axis=1))
print('SVM Fit Complete')
```

SVM Fit Complete

Листинг 4.25. Точность метода опорных векторов на обучающих данных

```
svm.score(x_train_predict,np.argmax(y_train,axis=1))
```

Точность метода опорных векторов:
Out[27]:
0.9839527027027027
Точность метода SVM равно 97%

Листинг 4.26. Точность метода на тестовых данных

```
svm.score(x_test_predict,np.argmax(y_test,axis=1))
```

Out[28]: 0.9722972972972973

Листинг 4.27. Сохранение модели опорных векторов

```
pickle.dump(svm,open('svms.pkl', 'wb'))
```

```
Pred_labels = svm.predict(x_test_predict)
Pred_labels = pd.DataFrame(Pred_labels,index =None)
Pred_labels.head()
```

	0
0	10
1	14
2	34
3	1
4	19

Интеграция CNN с градиентным бустингом XGBoost.
Интеграция модели CNN с алгоритмом Extreme Gradient Boost.

Листинг 4.28. Обучение модели XGBoost

```
import xgboost as xgb
xb = xgb.XGBClassifier(use_label_encoder=False)
xb.fit(x_train_predict,np.argmax(y_train,axis=1))
print('XGBoost Fit Complete')
```

XGBoost Fit Complete

Точность данного метода 100%.

Листинг 4.29. Точность модели XGBoost на обучающих данных

```
xb.score(x_train_predict,np.argmax(y_train,axis=1))
```

Out[31]:1.0

На тестовом множестве дает следующую точность

Листинг 4.30. Точность модели на тестовых данных

```
xb.score(x_test_predict,np.argmax(y_test,axis=1))
```

Out[32]: 0.9837837837837838

Таким образом, мы получили результаты прогноза выше 97%.

4.6. Обобщенные архитектуры сверточных нейронных сетей. Трансферная обучение.

Сначала рассмотрим основные сверточные нейронные сети, которые являются базовыми при использовании трансферного обучения, реализацию которого приступим ниже. Сначала рассмотрим архитектуры основных сверточных нейронных сетей.

Структура сети AlexNet. AlexNet — сверточная нейронная сеть, которая оказала большое влияние на развитие машинного обучения, в особенности — на алгоритмы компьютерного зрения. Сеть с большим отрывом выиграла конкурс по распознаванию изображений ImageNet LSVRC-2012 в 2012 году (с количеством ошибок 15,3% против 26,2% у второго места). Архитектура AlexNet схожа с созданной Yann LeCum сетью LeNet. Однако у AlexNet больше фильтров на слое и вложенных сверточных слоев. Сеть включает в себя свертки, максимальное объединение, дропаут, аугментацию данных, функции активаций ReLU и стохастический градиентный спуск.

Модель AlexNet состоит из пяти слоев свертки и трех слоев объединения, в том числе трех полносвязных слоев. Структура AlexNet аналогична структуре LeNet, но для соответствия крупномасштабному набору данных ImageNet используется больше слоев свертки и больше пространства параметров. Модель AlexNet добавляет функцию активации ReLU после каждой свертки, что решает проблему исчезновения градиента сигмоиды и ускоряет сходимость. Техника случайного отбрасывания (выпадение) используется для выборочного игнорирования одного нейрона в обучении, чтобы избежать переобучения

модели (улучшение данных также используется для предотвращения переобучения). Для повышения точности добавлен нормализованный слой LRN (Local Response Normalization). Перекрывающийся максимальный пул, т. е.z связано с размером шага s . z > s позволяет избежать усредненного эффекта объединения средних значений.

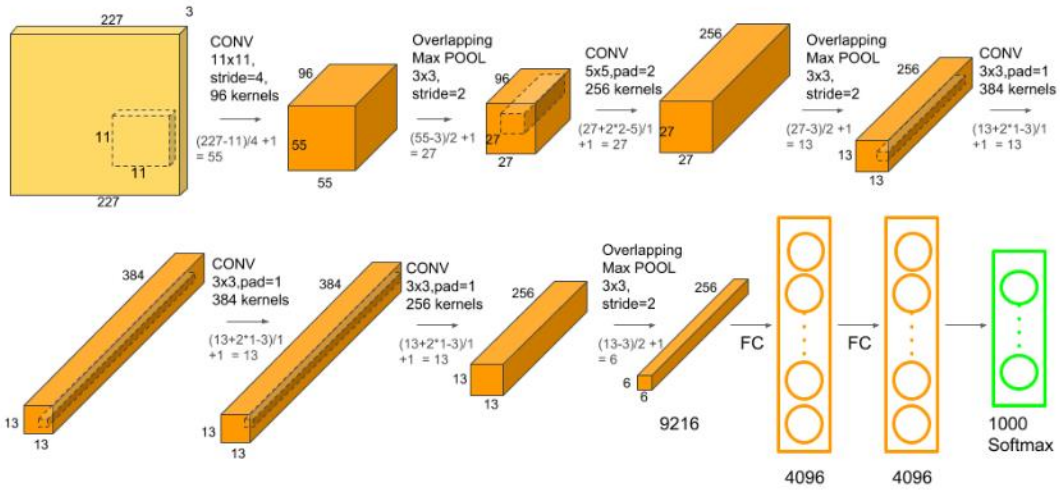


Рис.4.17. Общая структура нейронной сети AlexNet

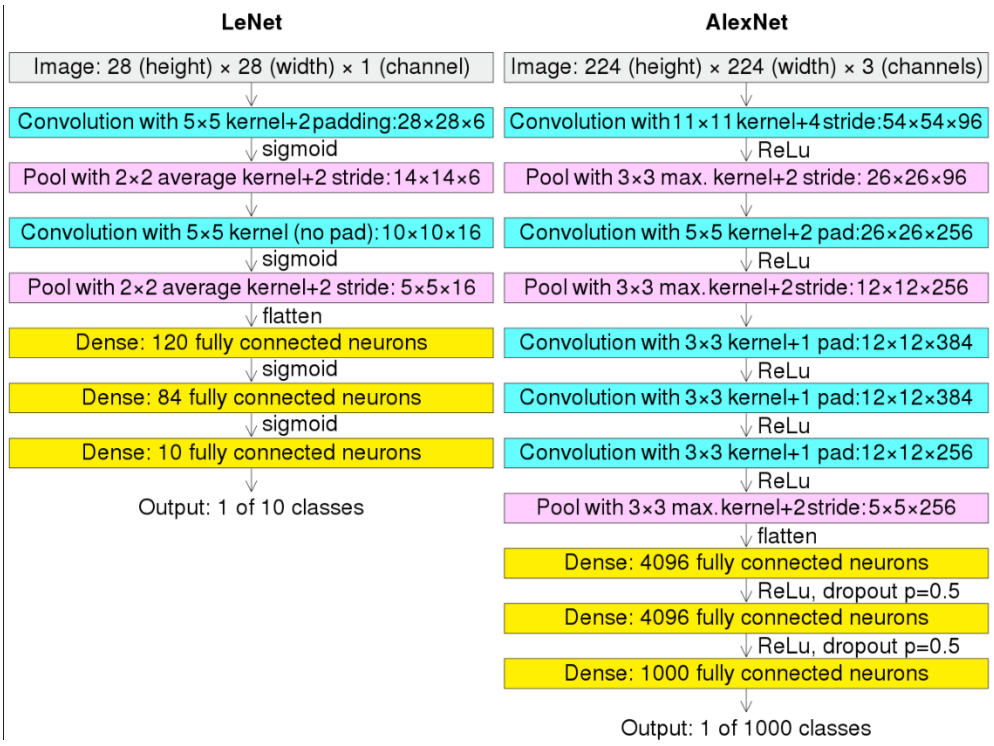


Рис.4.18. Общая структура AlexNet для различных классов задач

AlexNet успешно использует ReLU в качестве функции активации CNN, которая подтверждает, что ее эффект превышает сигмовидную в более глубоких сетях, и успешно решает проблему градиентной дисперсии сигмовидной в более глубоких сетях. Во время обучения отсев используется для случайного игнорирования некоторых нейронов во избежание переобучения модели. Обычно он используется на уровне полного соединения. Отсев не используется в прогнозировании, то есть отсев составляет 1 максимальное перекрытие объединения (размер шага меньше ядра свертки) используется в CNN. Ранее CNN обычно использовала объединение средних значений, а использование максимального объединения позволяет избежать нечеткого эффекта объединения средних значений. В то же время эффект перекрытия может улучшить богатство функций. Слой LRN (Local Response Normalization) предлагается для создания механизма конкуренции активности локальных нейронов,

Структура сети VGG-16. VGG-16 имеет в общей сложности 16 слоев, в том числе 13 сверточных слоев и 3 полносвязных слоя. Слои разделены максимальным пулом, а блоки активации всех скрытых слоев используют функцию ReLU. Используя слой пула в качестве границы, VGG16 имеет 6 блочных структур, и количество каналов в каждой блочной структуре одинаково. И слой свертки, и слой полного соединения имеют весовые коэффициенты, а уровень объединения не включает вес. Для нейронной сети свертки VGG16 слой свертки и слой пула отвечают за извлечение признаков, а последние три полных слоя соединения отвечают за выполнение задачи классификации. VGG использует слой свертки из нескольких ядер свертки меньшего размера (3 ядра).3) заменить больший слой свертки одного ядра свертки. С одной стороны, это может уменьшить параметры; с другой стороны, это эквивалентно более нелинейному отображению, которое может увеличить способность сети к подгонке/выражению. Все уровни свертки представляют собой ядра свертки 3 3, а ядро пула уровня пула — 2 2. VGG16 — это большая сеть со 138 миллионами параметров. Поэтому его главный недостаток заключается в том, что количество признаков, которые необходимо обучить, очень велико.

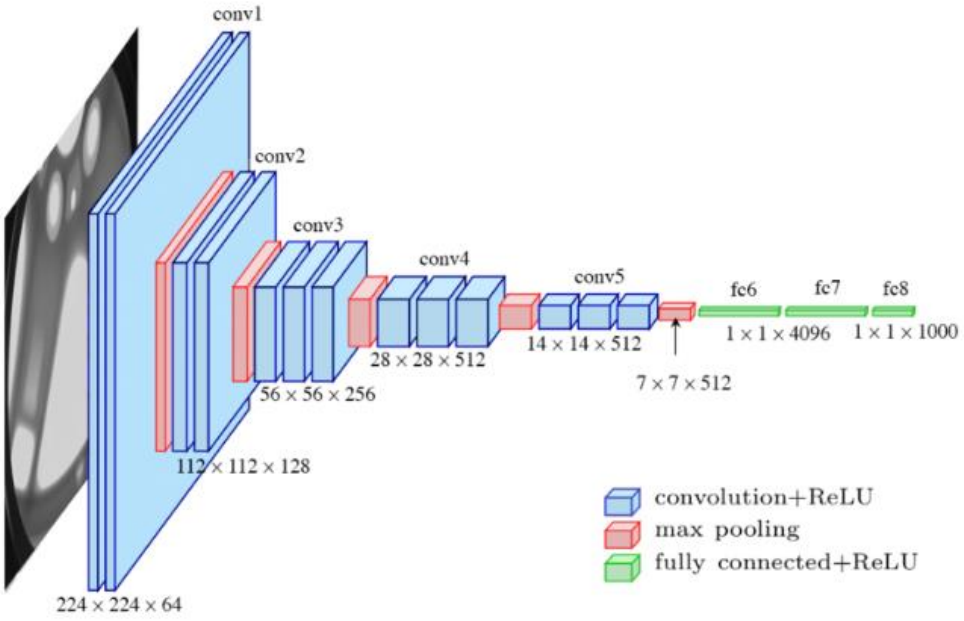


Рис.4.19. Общая структура архитектура VGG 16 для 1000 классов

Структура сети Resnet-50

В сверточной нейронной сети каждый слой нейронов связан только с левым и правым соседними нейронами, то есть входные данные поступают от предыдущего слоя и передаются на следующий слой. Таким образом, связь между соседними слоями не только ограничивает разнообразие сети, но и легко приводит к углублению количества слоев сети и сложности обучения модели. Чтобы решить эту проблему, сеть ResNet решает проблему снижения эффективности обучения глубокой сети за счет межуровневого соединения и подгонки члена ошибки. Модуль остаточного соединения является основной архитектурой сети ResNet, а отображение идентичности является ее основной идеей.

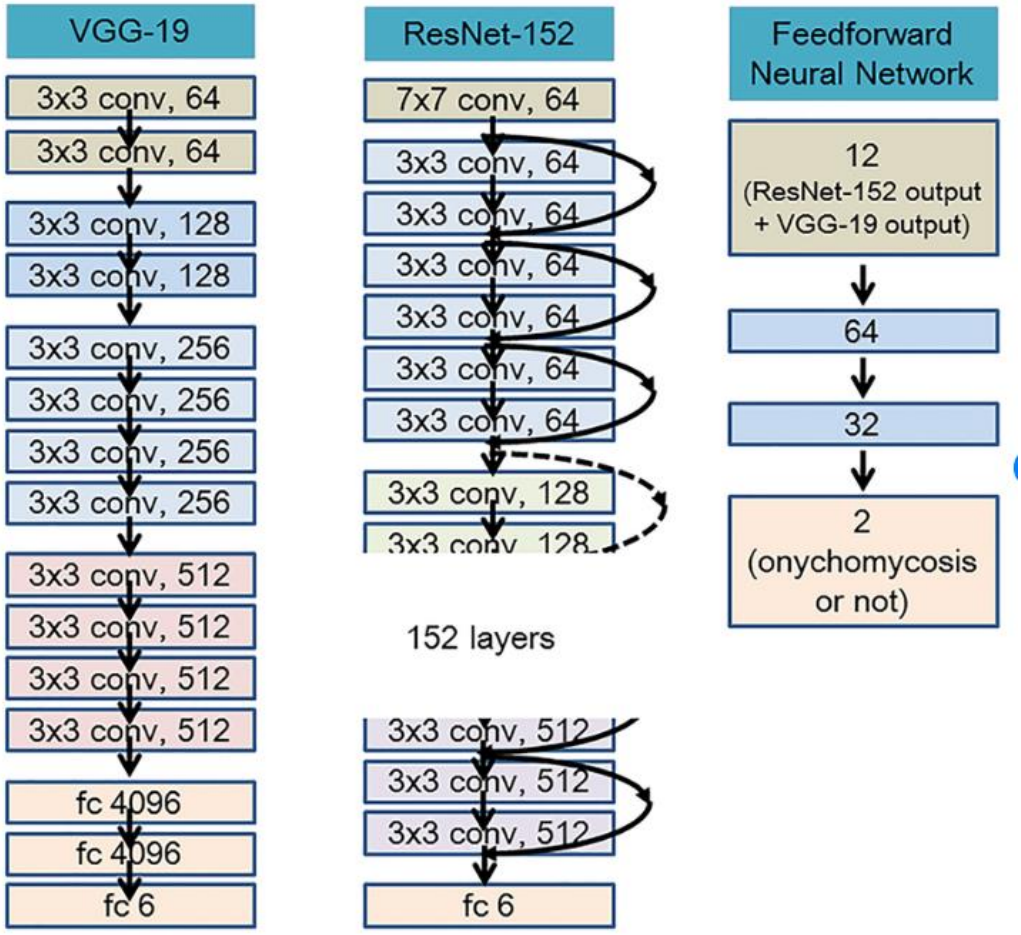


Рис.4.20. Общая архитектура сети VGG 19 и вариантов ResNet.

В процессе прямого распространения сеть без остаточного модуля получает окончательный результат вывода сети после свертки, объединения, и другие операции. ResNet использует глобальный уровень объединения средних значений. Используя остаточный модуль, можно обучить остаточную сеть из 152 слоев. ResNet-50 в основном использует свертку 3×3. ResNet-50 имеет два основных блока, один из которых является блоком идентификации. Входные и выходные размеры одинаковы, поэтому несколько блоков могут быть соединены последовательно, еще одним базовым блоком является Conv Block. Размеры входа и выхода разные, поэтому их нельзя соединить последовательно. Его функция состоит в изменении размерности собственного вектора. еще один базовый блок — Conv Block. Размеры входа и выхода разные, поэтому их нельзя соединить последовательно. Его функция состоит в изменении размерности собственного вектора. еще один базовый блок — Conv Block. Размеры входа и выхода разные, поэтому их нельзя соединить последовательно. Его функция состоит в изменении размерности собственного вектора.

Трансферное обучение.

Трансферное обучение обычно относится к процессу, в котором модель, обученная одной проблеме, каким-то образом используется для решения второй связанной проблемы. Например, знания, полученные при обучении распознаванию болезней растений, можно применить при попытке распознать конкретные болезни. В глубоком обучении трансферное обучение — это метод, при котором модель нейронной сети сначала обучается, например на больших данных ImageNet, и применяется на аналогичной рассматриваемой задаче. Трансферное обучение имеет то преимущество, что сокращает время обучения модели обучения и может привести к меньшей ошибке обобщения. Трансферное обучение - это метод машинного обучения, который использует существующие знания для решения проблем в разных, но связанных областях. Его цель - завершить передачу знаний между смежными областями. Для сверточных нейронных сетей трансферное обучение заключается в успешном применении «знаний», полученных на определенных наборах данных, в новых областях.

Обычно существует два метода применения трансферного обучения к сверточным нейронным сетям. Один из них заключается в использовании модели предварительной подготовки с обучающим весом для получения признаков, которые будут использоваться в новой задаче, то есть модель сети предварительной подготовки используется в качестве экстрактора признаков. В это время выходные данные сети извлекают интересующие функции в части перед последним полным слоем соединения. Другой — использовать новый набор данных для обучения сети точной настройке веса сети. В этом случае количество узлов выходного слоя необходимо изменить, чтобы оно соответствовало количеству категорий в новой задаче. Кроме того, в обоих случаях входные данные должны соответствовать входному размеру предобучаемой сети. Конкретный метод трансферного обучения зависит от разницы в размерах между целевым набором данных и исходным набором данных и сходства между ними. Если целевой набор данных очень мал и они похожи, во избежание переобучения в качестве средства извлечения признаков обычно используется предобучающая модель, напротив, применяется тонкая настройка.

Программная реализация трансферное обучения-VGG 16. Рассмотрим конкретный пример на VGG-16. Данная сеть, это сверточная нейронная сеть, как мы выше упомянули и состоит из 16 слоев. Сначала загружаем предварительно обученную версию сети, обученную на более чем миллионе изображений из базы данных ImageNet. Предварительно обученная сеть может классифицировать изображения по 1000 категориям объектов, таким как клавиатура, мышь, карандаш и многие животные. В результате сеть изучила богатые представления

функций для широкого спектра изображений. Сеть имеет входной размер изображения 224 на 224.

Для реализации VGG-16 предварительно определить необходимый пакет программ и директории базы данных. Далее загружается основной пакет программ для сети VGG 16.Сохраним размер изображения как (224,224), можно увеличить размер изображений для получения точных результатов.

IMAGE_SHAPE = [224, 224] batch_size=32

Также важно загрузка весовых коэффициентов VGG16, без верхнего слоя. Эти веса обучаются на наборе данных Imagenet. После загрузки входного изображение, определение входного изображения и весовых коэффициентов для исключения обучения начальных слоев сети VGG 16. Весь данный процесс приведена ниже

```
input_shape = (64,64,3) as required by VGG
vgg = VGG16(input_shape = (224,224,3), weights = 'imagenet', include_top = False)
```

Это исключит начальные слои из этапа обучения, поскольку они уже обучены.

Листинг 4.32. Определения классов, функции активации, оценки ошибки и точности модели, оптимизатора и компиляция сети VGG 16

```
x = Dense(num_classes, activation = 'softmax')(x)
model = Model(inputs = vgg.input, outputs = x)
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

Здесь добавлены выходные слои с функцией softmax, так как это проблема классификации с несколькими метками.

Генератор данных изображения. Класс ImageDataGenerator позволяет случайным образом поворачивать изображения на любой градус от 0 до 360, предоставляя целочисленное значение в аргументе rotate_range.

Листинг 4.33. Технологии увеличения изображений и определения класса

```
from keras.preprocessing.image import ImageDataGenerator
trdata = ImageDataGenerator()
train_data_gen = trdata.flow_from_directory(directory="Train",
                                            target_size=(224,224),shuffle=False, class_mode='categorical')
tsdata = ImageDataGenerator()
test_data_gen = tsdata.flow_from_directory(directory="Test",
                                           target_size=(224,224),shuffle=False, class_mode='categorical')
```

Found 385 images belonging to 3 classes.

Found 138 images belonging to 3 classes.

Обучение модели с эпохой =15

Обучение модели для 15 эпох

accuracy: 1.0000 - val_loss: 7.0003 - val_accuracy: 0.8768

Теперь сделаем прогноз на тестовом наборе.

Листинг 4.35. Создание матрицы прогнозов модели

```
Y_pred = model.predict(test_data_gen, test_data_gen.samples / batch_size)
val_preds = np.argmax(Y_pred, axis=1)
import sklearn.metrics as metrics
val_trues =test_data_gen.classes
from sklearn.metrics import classification_report
print(classification_report(val_trues, val_preds))
```

Вот матрица прогнозов

n	precition	recall	f1-score	support
0	0.98	0.78	0.87	59
1	0.91	0.98	0.95	53
2	0.68	0.88	0.77	26
accuracy			0.88	138
macro avg	0.86	0.88	0.86	138
weighted avg	0.90	0.88	0.88	138

precision recall f1-score support
0 0.98 0.78 0.87 59
1 0.91 0.98 0.95 53
2 0.68 0.88 0.77 26
accuracy 0.88 138
macro avg 0.86 0.88 0.86 138
weighted avg 0.90 0.88 0.88 138

Матрица путаницы строится следующим образом:

Листинг 4.36. Создание матрицы путаницы -погрешностей модели

```
Y_pred = model.predict(test_data_gen, test_data_gen.samples / batch_size)
val_preds = np.argmax(Y_pred, axis=1)
import sklearn.metrics as metrics
val_trues =test_data_gen.classes
cm = metrics.confusion_matrix(val_trues, val_preds)
cm
```

Out[7]:
array([[46, 3, 10],
[0, 52, 1],
[1, 2, 23]], dtype=int64)

Сохраняем модель для дальнейшего использования.

```
keras_file="Model.h5"
tf.keras.models.save_model(model,keras_file)
```

Листинг 4.37. Тестирование модели

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np
img_path = 'Test/Apple/apple fruit175.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
preds=model.predict(x)
```

Результат будет Apple. Используя VGG 16 и трансферное обучение для классификации изображений мы создали модель для классификации фруктов. Трансферное обучение является гибким, позволяя использовать предварительно обученные модели непосредственно в качестве предварительной обработки извлечения признаков и интегрировать их в совершенно новые модели. Keras обеспечивает удобный доступ ко многим высокопроизводительным моделям VGG, Inception и ResNet.

Реализация трансферного обучения для распознавания болезней растений с использованием архитектуры RESNET-9

Мы рассмотрим важный раздел компьютерного зрения задачу прогнозирования с применением сверточных нейронных сетей и прогаммирование на Pytorch. Программирование на Pytorch в настоящее время широко используется для решения задач классификации изображений. Будем изучать задачу классификации болезни сельскохозяйственных растений. Для построения модели мы будем использовать архитектуру CNN, которая очень хорошо подходить для задач распознавания изображений. Рассматривать будем многоклассовую задачу классификации. Результатом будет модель, которая будет прогнозировать различные болезни, которые распространены по республике. В качестве архитектуры нейронной сети возьмем Reznet-9. Использование ResNet это остаточной сети глубокое обучение.(Полный код приложения приведена в Приложение 4.25)

После первой архитектуры на основе CNN (AlexNet), победившей в конкурсе ImageNet 2012, каждая последующая архитектура-победитель использует больше слоев в глубокой нейронной сети для снижения частоты ошибок. Это работает для меньшего количества слоев, но когда мы увеличиваем

количество слоев, возникает общая проблема в глубоком обучении, связанная с так называемым исчезающим/взрывающимся градиентом. Это приводит к тому, что градиент становится равным 0 или слишком большим. Таким образом, когда мы увеличиваем количество слоев, частота ошибок при обучении и тестировании также увеличивается.

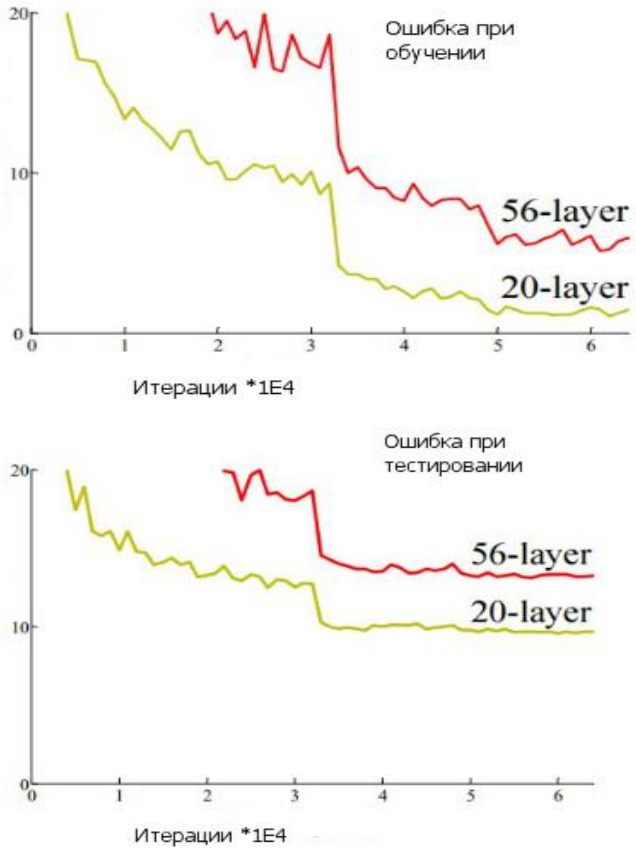


Рис.4.21. Результаты реализации архитектуры ResNet 9 с различными нейронными слоями

На рисунках показано ошибка при обучении и тестировании модели соответственно. На приведенном выше графике, видно, что 56-уровневая CNN дает больше ошибок как для обучающего, так и для тестового набора данных, чем 20-уровневая архитектура CNN. После дополнительного анализа частоты ошибок авторы пришли к выводу, что это вызвано исчезающим/взрывающимся градиентом. ResNet, предложенная в 2015 году исследователями Microsoft Research, представила новую архитектуру под названием Residual Network. По оси x отложена количество ошибок в масштабе 1E4.

4.7. Разработка модели для прогнозирования урожайности и болезней растений с применением Pytorch и ее развертывания для создания искусственного интеллекта

Этот набор данных создается с помощью автономного дополнения исходного набора данных. Исходный набор данных PlantVillage. Этот набор данных состоит из около 87 тысяча RGB-изображений здоровых и больных листьев сельскохозяйственных культур, которые подразделяются на 38 различных классов. Общий набор данных разделен на обучающий и проверочный наборы в соотношении 80/20 с сохранением структуры каталогов. Новый каталог, содержащий 33 тестовых изображения, создается позже для целей прогнозирования.

Наша цель ясна и проста. Нам нужно построить модель, которая может классифицировать здоровые и больные листья урожая, а также, если у урожая есть какое-либо заболевание, предсказать, какое это заболевание.

Импорт необходимых библиотек. Инсталлируем из библиотеки Pytorch пакет torchsummary необходимый модуль работы с изображениями и для печати сводки модели в стиле keras, хорошо отформатированного информации о модели и красиво выглядящего, поскольку Pytorch изначально не поддерживает это. Реализация данного приложения полностью приведена в приложении 4.2.Опишем кратко ее реализацию. Для реализации задачи загружаем необходимы пакеты и программы. Для изучения данных, загружаем изображения из каталога локального компьютера с 37 классами болезней растений. Уникальные растения:

['Apple', 'Blueberry', 'Cherry_(including_sour)', 'Corn_(maize)', 'Grape', 'Peach', 'Pepper_bell', 'Potato', 'Raspberry', 'Soybean', 'Squash', 'Strawberry', 'Tomato']

Количество уникальных растений всего 13, количество заболеваний 26.

Итак, у нас есть изображения листьев 14 растений, и, исключая здоровые листья, у нас есть 26 типов изображений, которые показывают конкретное заболевание конкретного растения.

Листинг 4.38. Болезни растений и их количество

	Количество изображений
Apple___Apple_scab	2016
Apple___Black_rot	1987
Apple___Cedar_apple_rust	1760
Apple___healthy	2008
Blueberry___healthy	1816
Cherry_(including_sour)___healthy	1826
Cherry_(including_sour)___Powdery_mildew	1683
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	1642
Corn_(maize)___Common_rust_	1907
Corn_(maize)___healthy	1859
Corn_(maize)___Northern_Leaf_Blight	1908
Grape___Black_rot	1888
Grape___Esca_(Black_Measles)	1920
Grape___healthy	1692
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1722
Peach___Bacterial_spot	1838
Peach___healthy	1728
Pepper_bell___Bacterial_spot	1913
Pepper_bell___healthy	1988
Potato___Early_blight	1939
Potato___healthy	1824
Potato___Late_blight	1939
Raspberry___healthy	1781
Soybean___healthy	2022
Squash___Powdery_mildew	1736
Strawberry___healthy	1824
Strawberry___Leaf_scorch	1774
Tomato___Bacterial_spot	1702
Tomato___Early_blight	1920
Tomato___healthy	1926
Tomato___Late_blight	1851
Tomato___Leaf_Mold	1882
Tomato___Septoria_leaf_spot	1745
Tomato___Spider_mites Two-spotted_spider_mite	1741
Tomato___Target_Spot	1827
Tomato___Tomato_mosaic_virus	1790
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1961

Визуализация вышеуказанной информации на графике. Построение количества изображений, доступных для каждого заболевания.

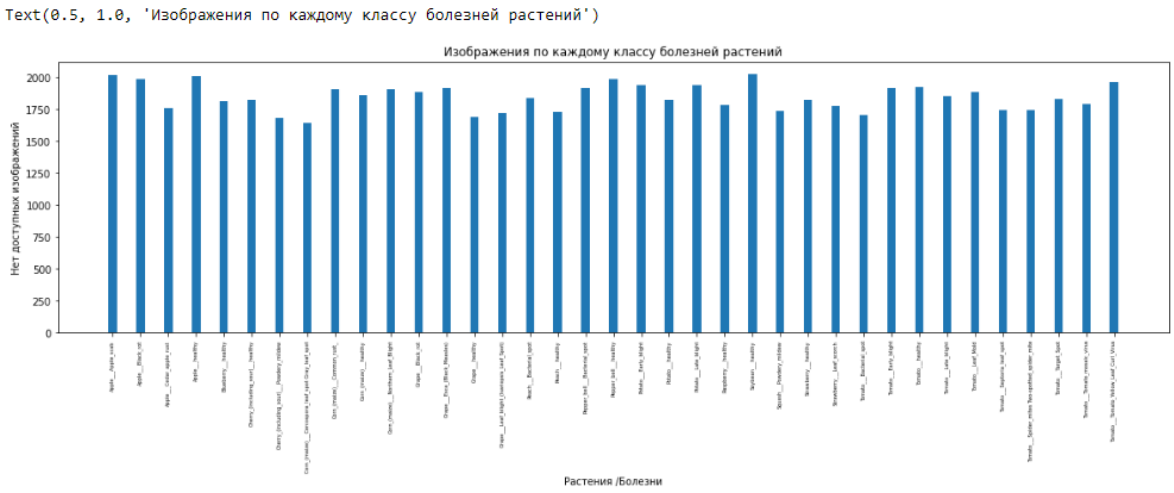


Рис.4.22. Гистограмма болезней растений относительно их количества.

Мы видим, что набор данных почти сбалансирован для всех классов, поэтому мы можем перейти к процессу прогнозирования. Сначала мы получим изображения доступные для обучения модели:

```
n_train = 0
for value in nums.values():
    n_train += value
print(f"Всего  {n_train} изображений для обучения")
```

There are 70295 images for training

Подготовим данные для обучения. Для этого все наборы данных проверки и обучения преобразуем в тензоры. Вот код:

```
train = ImageFolder(train_dir, transform=transforms.ToTensor())
valid = ImageFolder(valid_dir, transform=transforms.ToTensor())
```

torchvision.datasets — это класс, который помогает загружать все распространенные и известные наборы данных. Это также помогает в загрузке пользовательских наборов данных. Мы использовал подкласс torchvision.datasets.ImageFolder, который помогает загружать данные изображения, когда данные расположены следующим образом:

Форма изображения

```
img, label = train[0]
print(img.shape, label)
```

```
torch.Size([3, 256, 256]) 0
```

Мы можем видеть форму (3, 256 256) изображения, где 3 — количество каналов (RGB), а 256 x 256 — ширина и высота изображения.
Общее количество классов в наборе данных обучения

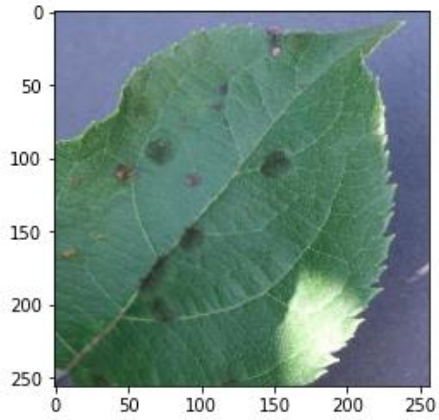
```
# Общее количество классов в в наборе данных обучения
len(train.classes)
```

```
37
```

Проверим некоторые изображения из обучающего набора данных.

```
show_image(*train[5])
```

```
Label :Apple___Apple_scab(0)
```




```
show_image(*train[44000])
```

```
Label :Soybean___healthy(23)
```

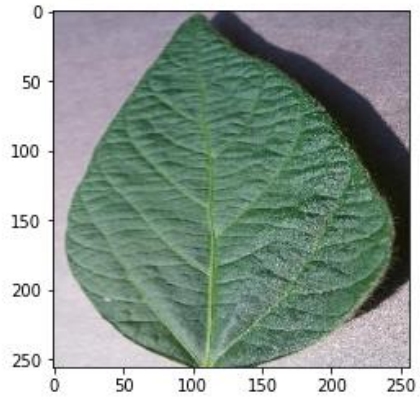


Рис.4.23. Изображение больных и здоровых растений

Установка размера партии

batch_size = 32

batch_size — это общее количество изображений, переданных в качестве входных данных одновременно при прямом распространении CNN. По сути, размер пакета определяет количество образцов, которые будут распространяться по сети. Например, предположим, что у вас есть 1050 обучающих выборок, и вы хотите установить размер партии равным 100. Алгоритм берет первые 100 выборок (с 1-й по 100-ю) из набора обучающих данных и обучает сеть. Затем он берет вторые 100 выборок (со 101-й по 200-ю) и снова обучает сеть. Мы можем продолжать выполнять эту процедуру, пока не распространим все выборки по сети.

DataLoaders для обучения и проверки

train_dl = DataLoader(train, batch_size, shuffle=**True**, num_workers=2, pin_memory=**True**)

valid_dl = DataLoader(valid, batch_size, num_workers=2, pin_memory=**True**)

DataLoader является подклассом, происходящим от torch.utils.data. Это помогает в загрузке больших и потребляющих память наборов данных. Он принимает batch_size, который обозначает количество образцов, содержащихся в каждом сгенерированном пакете.

Установка shuffle=True перемешивает набор данных. Это полезно, чтобы партии между эпохами не были похожи друг на друга. Это в конечном итоге сделает нашу модель более надежной.

num_workers, обозначает количество процессов, которые параллельно генерируют пакеты. Если у вас больше ядер в вашем процессоре, вы можете установить его на количество ядер в вашем процессоре. Поскольку Kaggle предоставляет 2-ядерный процессор, я установил его на 2.

Вспомогательная функция для отображения набора обучающих экземпляров

Изображения для первой партии обучения

Рис.4.24. База данных классов растений

Увеличение данных. Технологии увеличение данных широко используется в задачах распознавания изображений. Данная технология применяется для случаев, когда набор данных сложно собрать в очень больших случаях. Ниже на

рисунке представлены несколько вариантов преобразования исходного изображения вокруг главной оси.

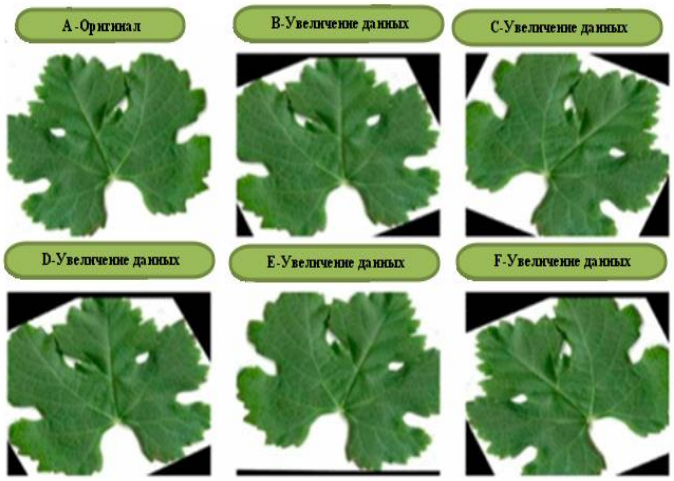


Рис.4.25. Методы увеличения данных

Моделирование. При работе с набором данных изображений рекомендуется использовать графический процессор вместо ЦП, поскольку ЦП являются обобщенными для общего назначения, а графические процессоры оптимизированы для обучения моделей глубокого обучения, поскольку они могут обрабатывать несколько вычислений одновременно. У них большое количество ядер, что позволяет лучше вычислять несколько параллельных процессов. Кроме того, вычисления в глубоком обучении должны обрабатывать огромные объемы данных — это делает пропускную способность памяти графического процессора наиболее подходящей. Чтобы беспрепятственно использовать графический процессор, если он доступен, мы определяем пару вспомогательных функций («get_default_device» и «to_device») и вспомогательный класс «DeviceDataLoader» для перемещения нашей модели и данных на графический процессор по мере необходимости.

Построение архитектуры модели. Будем использовать архитектуру нейронной сети ResNet 9, который стал одним из крупнейших прорывов в области компьютерного зрения для распознавания изображений

В ResNets, в отличие от традиционных нейронных сетей, каждый слой передает данные следующему слою, мы используем сеть с остаточными блоками, каждый слой передает данные следующему слою и непосредственно слоям, находящимся на расстоянии 2–3 слоев, чтобы избежать переобучения (ситуация, когда потери при проверке перестают уменьшаться в какой-то момент, а затем продолжают увеличиваться, в то время как потери при обучении все еще уменьшаются. Это также помогает предотвратить проблему исчезающего

градиента и позволяет нам обучать глубокие нейронные сети. Вот простой остаточный блок:

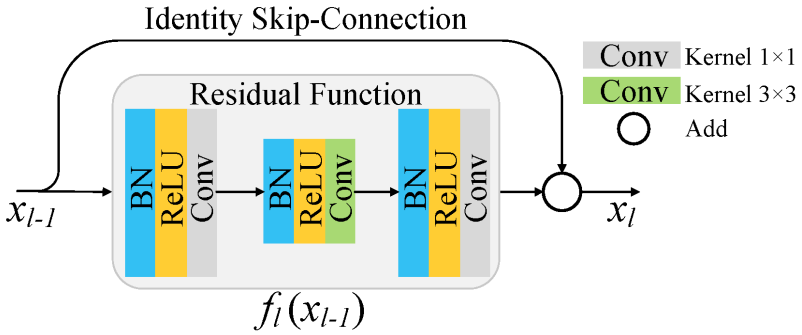


Рис.4.26. Архитектура нейронной сети ResNet 9.

Реализация кода остаточного блока. ReLU можно применять до или после добавления ввода. Теперь определим класс ImageClassificationBase со следующими функциями:

training_step - чтобы выяснить, насколько «неправильно» работает модель после этапа обучения или проверки. Мы используем эту функцию не только как показатель точности, который, вероятно, не будет дифференцируемым (это означает, t быть определено, что необходимо для улучшения модели в процессе обучения).

Краткий обзор документации PyTorch, которая дает функцию стоимости: cross_entropy.

validation_step - Поскольку показатель точности нельзя использовать при обучении модели, это не значит, что его не следует реализовывать! Точность в этом случае будет измеряться порогом и учитываться, если разница между прогнозом модели и фактической меткой ниже этого порога.

validation_epoch_end - мы хотим отслеживать потери/точности проверки и потери обучения после каждой эпохи, и каждый раз, когда мы это делаем, мы должны убедиться, что градиент не отслеживается.

epoch_end - Мы также хотим распечатать потери/точности проверки, потери обучения и скорость обучения, потому что мы используем планировщик скорости обучения (который будет изменять скорость обучения после каждой партии обучения) после каждой эпохи.

Мы также определяем функцию «точность», которая вычисляет общую точность модели для всей партии выходных данных, чтобы мы могли использовать ее в качестве метрики в «fit_one_cycle».

Для расчета ошибки и точности модели создается код базового классификатора для генерации и вычисления ошибки и точности нашей модели.

Определение окончательной архитектуры модели. Для построения архитектуры определяем полносвязанный блок ConvBlock на основе BatchNormalization и архитектуру ResNet 9 с параметром ImageClassificationBase, которое мы определили выше пишем следующий код

Теперь определяем объект модели и переносим его в устройство, с которым работаем...

Получение красиво отформатированной сводки нашей модели (как в Keras). Pytorch не поддерживает его изначально. Необходимо при обучении модели установить библиотеку torchsummary

Обучение модели. Прежде чем мы обучим модель, давайте определяется функция полезности, функцию «оценить», которая будет выполнять фазу проверки, и функцию «fit_one_cycle», которая будет выполнять весь процесс обучения. В fit_one_cycle мы использовали некоторые приемы:

Планирование скорости обучения: вместо использования фиксированной скорости обучения мы будем использовать планировщик скорости обучения, который будет изменять скорость обучения после каждой партии обучения. Существует много стратегий изменения скорости обучения во время обучения, и мы будем использовать одну из них, которая называется «Политика скорости обучения за один цикл»*, которая включает в себя начало с низкой скорости обучения, постепенно увеличивая ее от партии к партии до высокая скорость обучения примерно для 30% эпох, затем постепенно снижается до очень низкого значения для оставшихся эпох.

Затухание веса: мы также используем затухание веса, которое представляет собой метод регуляризации, который предотвращает слишком большое увеличение веса за счет добавления дополнительного члена к функции потерь.

Градиентная обрезка: помимо весов слоев и результатов, также полезно ограничить значения градиентов небольшим диапазоном, чтобы предотвратить нежелательные изменения параметров из-за больших значений градиента. Эта простая, но эффективная техника называется отсечением градиента.

Давайте проверим наши потери и точность проверки

```
%%time
history = [evaluate(model, valid_dl)]
history
CPU times: user 44 s, sys: 3.28 s, total: 47.3 s
Wall time: 1min 32s
Out[34]:
[{'val_loss': tensor(3.6397, device='cuda:0'), 'val_accuracy': tensor(0.0191)}]
```

Поскольку веса инициализируются случайным образом, точность приближается к 0,019 (то есть вероятность получения правильного ответа составляет 1,9%, или можно сказать, что модель выбирает класс случайным образом). Теперь объявите некоторые гиперпараметры для обучения модели. Мы можем изменить его, если результат неудовлетворителен.

```
epochs = 2
max_lr = 0.01
grad_clip = 0.1
weight_decay = 1e-4
opt_func = torch.optim.Adam
```

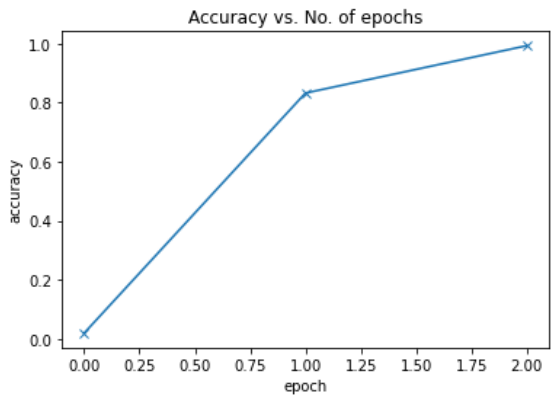
Приступим к обучению нашей модели....

Для запуска следующей ячейки может потребоваться от 15 до 45 минут в зависимости от вашего графического процессора. В kaggle (P100 GPU) это заняло около 20 минут Wall Time.

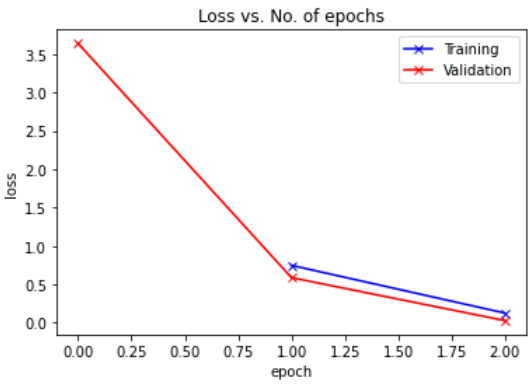
```
%%time
history += fit_OneCycle(epochs, max_lr, model, train_dl, valid_dl,
   grad_clip=grad_clip,
   weight_decay=1e-4,
   opt_func=opt_func)
Epoch [0], last_lr: 0.00812, train_loss: 0.7466, val_loss: 0.5865, val_acc: 0.8319
Epoch [1], last_lr: 0.00000, train_loss: 0.1248, val_loss: 0.0269, val_acc: 0.9923
CPU times: user 11min 16s, sys: 7min 13s, total: 18min 30s
Wall time: 19min 53s
```

Мы получили точность 99,2 %. Построим теперь график точности модели.

```
plot_accuracies(history)
```




```
plot_losses(history)
```



```
plot_lrs(history)
```

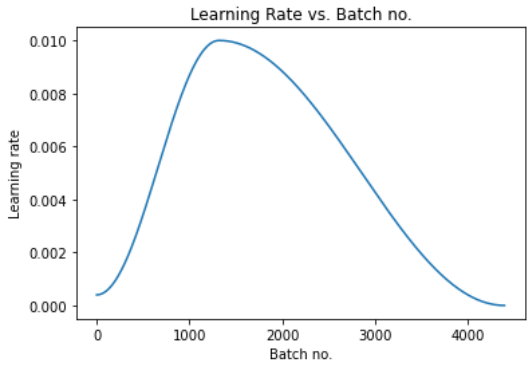


Рис.4.27. Результаты точности и ошибки моделей и ее скорость обучения

Тестирование модели на тестовых данных. У нас есть только 33 изображения в тестовых данных, поэтому давайте проверим модель на всех изображениях.

Прогноз первого изображения:

```
img, label = test[0]
plt.imshow(img.permute(1, 2, 0))
print('Label:', test_images[0], ', Predicted:', predict_image(img, model))
```

Label: AppleCedarRust1.JPG , Predicted: Apple___Cedar_apple_rust

Получение всех прогнозов с фактическими метками. Данные тестовых изображений и их правильные прогнозы.

Листинг 4.39. Результаты тестирования болезни растений.

```
for i, (img, label) in enumerate(test):
    print('Label:', test_images[i], ', Predicted:', predict_image(img, model))
```

```
Label: AppleCedarRust1.JPG , Predicted: Apple___Cedar_apple_rust
Label: AppleCedarRust2.JPG , Predicted: Apple___Cedar_apple_rust
Label: AppleCedarRust3.JPG , Predicted: Apple___Cedar_apple_rust
Label: AppleCedarRust4.JPG , Predicted: Apple___Cedar_apple_rust
Label: AppleScab1.JPG , Predicted: Apple___Apple_scab
Label: AppleScab2.JPG , Predicted: Apple___Apple_scab
Label: AppleScab3.JPG , Predicted: Apple___Apple_scab
Label: CornCommonRust1.JPG , Predicted: Corn_(maize)___Common_rust_
Label: CornCommonRust2.JPG , Predicted: Corn_(maize)___Common_rust_
Label: CornCommonRust3.JPG , Predicted: Corn_(maize)___Common_rust_
```

```
Label: PotatoEarlyBlight1.JPG , Predicted: Potato___Early_blight
Label: PotatoEarlyBlight2.JPG , Predicted: Potato___Early_blight
Label: PotatoEarlyBlight3.JPG , Predicted: Potato___Early_blight
Label: PotatoEarlyBlight4.JPG , Predicted: Potato___Early_blight
Label: PotatoEarlyBlight5.JPG , Predicted: Potato___Early_blight
Label: PotatoHealthy1.JPG , Predicted: Potato___healthy
Label: PotatoHealthy2.JPG , Predicted: Potato___healthy
Label: TomatoEarlyBlight1.JPG , Predicted: Tomato___Early_blight
Label: TomatoEarlyBlight2.JPG , Predicted: Tomato___Early_blight
Label: TomatoEarlyBlight3.JPG , Predicted: Tomato___Early_blight
Label: TomatoEarlyBlight4.JPG , Predicted: Tomato___Early_blight
Label: TomatoEarlyBlight5.JPG , Predicted: Tomato___Early_blight
Label: TomatoEarlyBlight6.JPG , Predicted: Tomato___Early_blight
Label: TomatoHealthy1.JPG , Predicted: Tomato___healthy
Label: TomatoHealthy2.JPG , Predicted: Tomato___healthy
Label: TomatoHealthy3.JPG , Predicted: Tomato___healthy
Label: TomatoHealthy4.JPG , Predicted: Tomato___healthy
Label: TomatoYellowCurlVirus1.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
Label: TomatoYellowCurlVirus2.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
Label: TomatoYellowCurlVirus3.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
Label: TomatoYellowCurlVirus4.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
Label: TomatoYellowCurlVirus5.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
Label: TomatoYellowCurlVirus6.JPG , Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus
```

Мы видим, что модель идеально предсказала все тестовые изображения

Теперь охраненим модели. Есть несколько способов сохранить модель в Pytorch, ниже приведены два наиболее распространенных способа.

1.Сохранить/загрузить state_dict

При сохранении модели для вывода необходимо сохранить только изученные параметры обученной модели. Сохранение state_dict модели с помощью функции torch.save() даст вам максимальную гибкость для последующего восстановления модели, поэтому это рекомендуемый метод для сохранения моделей. Распространенным соглашением PyTorch является сохранение моделей с использованием расширения файла .pt или .pth.

Помните, что вы должны вызвать model.eval(), чтобы установить слои отсева и пакетной нормализации в режим оценки перед выполнением логического вывода. Несоблюдение этого правила приведет к противоречивым результатам вывода.

Сохранение в рабочий каталог kaggle.

PATH = './plant-disease-model.pth'

torch.save(model.state_dict(), PATH)

Сохранить/загрузить всю модель Этот процесс сохранения/загрузки использует наиболее интуитивно понятный синтаксис и требует наименьшего количества кода. Сохранение модели таким образом сохранит весь модуль с помощью модуля pickle Python. Недостатком этого подхода является то, что сериализованные данные привязаны к определенным классам и точной структуре каталогов, используемой при сохранении модели. Причина этого в том, что pickle не сохраняет сам класс модели. Вместо этого он сохраняет путь к файлу, содержащему класс, который используется во время загрузки.

Из-за этого ваш код может по-разному ломаться при использовании в других проектах или после рефакторинга.

Сохранение всей модели в рабочий каталог

PATH = './plant-disease-model-complete.pth'

torch.save(model, PATH)

4.7. Практическая реализация трансферного обучения глубоких нейронных сетей с технологиями компьютерного зрения для задач садоводства в Иссык-Кульской области.

Различные болезни фруктовых деревьев в последние годы сильно распространились по регионам Кыргызской Республики, представляя серьезную угрозу урожайности многих сельскохозяйственных культур. Яблоки и груши как фруктовые деревья, распространены в Иссык-Кульском регионе и являются основным видом фруктов для этого региона. В связи с этим остро, стоит вопрос сохранения этих фруктовых садов. В последние годы из-за сильно распространения вирусных заболеваний, как бактериальный ожог погибли многие сады, принося миллионные убытки сельхозпроизводителям.

Поддержание здоровья растений является одной из важнейших задач для получения ожидаемого и высокого урожая. Технологии глубокого обучения позволяет проводить диагностику болезней растений автоматически на основе обработки их изображений игнорируя привлечения экспертов на садоводческие участки и на поля, чтобы проверить, заражены ли растения болезнями.

Очевидно, что последствия распространения болезней сельскохозяйственных культур могут сильно повлиять на продовольственную безопасность всей страны. В последние годы аномальные случаи, связанные с изменениями климата для данного региона, сильно повлияли на годовые доходы многих фермеров и сельхозпроизводителей. Для решения этих проблем глубокое обучение как передовая область искусственного интеллекта, должен способствовать автоматизировать прогностический анализ, его применение в цифровом сельском хозяйстве. Данная проблема тесно связано и остается областью исследований, требующей дальнейшего развития для решения различных проблем, связанных с прогнозированием болезней сельскохозяйственных культур.

В данном главе исследуется применение различных сверточных нейронных сетей CNN, ResNet , VGG 16,19, Inception, Mobile Net, Alex Net и современный метод прогнозирования EfficientVS2 для распознавания наиболее распространенных болезней садовых культур для данного региона. Для повышения производительности и тестирования модели данные дополнительно

обучены на основе трансферного обучения. Сбор данных больных и здоровых изображений, на примере груши осуществлялся в уязвимых регионах, выбор, которых осуществлялся с учетом изменения климата.

В последние годы в связи с изменениями климата резко возросло, также возросло число болезней среди сельскохозяйственных растений в зависимости от температурного режима. Аномально жаркие и холодные температурные режимы установленный в летний и зимний периоды, частые весенние заморозки, выветривание почвы и нехватка воды увеличило количество неурожаев для регионов страны, которые повлияли на экономику страны в целом. Изменения климата повлияло и на все этапы земледелия, принося большие убытки сельхозпроизводителям. В связи с этим необходимо менять и методы управления аграрных секторов во всех регионах КР. Фермерам, приходится адаптироваться к новым климатическим условиям. Болезни садовых культур и их заразность могут существенно повлиять на нормальный рост всех плантаций многих садов. Например, только в Иссык-Кульской области КР многие плантации грушевых садов были разорены из-за этой проблемы одна за другой.

Причиной является болезнь, бактериологический ожог груш определенного сорта. В результате научные диагностические меры будут иметь решающее значение, чтобы избежать неправильного использования методов лечения, чрезмерного использования пестицидов, их остатков, что может привести к значительному снижению урожайности сельскохозяйственных культур. В последние годы CNN стало наиболее часто используемым методом для классификации болезней растений и вредителей. Извлечения признаков классификационной сети CNN состоит из каскадного слоя свертки + слоя объединения, за которым следует слой полного соединения (или средний слой объединения) + структура softmax для классификации. Существуют сверточные нейронные сети для классификации болезней и вредителей растений с помощью глубокого обучения с технологиями компьютерного зрения, включая AlexNet [39], GoogleLeNet [40], VGGNet [41], ResNet [42], Inception V4 [43], DenseNets [44], MobileNet [45] и SqueezeNet [46], которые обучались на больших данных. На практике в реальных садовых угодьях, изображения листьев груши обычно имеют сложный фон и содержат шум из-за освещения, перекрытия листочков и перекрытия стебля. Все эти факторы затрудняют выделение признаков пораженных участков, что приводит к низкой точности распознавания. Чтобы решить эту проблему, в этой статье применяется трансферная обучение и технологии Pytorch для распознавания болезней листьев груши, основанная на глубоком обучении.

В целом задача прогнозирование болезней фруктовых деревьев представляют с собой сложную и взаимосвязанную задачу, которая требует различные технические и экспериментальные навыки. Традиционный подход предусматривает участие специалиста для данной отрасли, который путем тщательного анализа поверхности листвы ставит диагноз. Данный подход является затратным и требует много средств.

В диссертационной работе с помощью методов глубокого обучения с технологиями компьютерного зрения создаются прогностические модели, которые извлекают полезную информацию из большого количества сельскохозяйственных данных. Модели прогнозирования, основы обработки изображений и модели прогнозирования, зависят от данных о погоде, основанные на различных типах данных, структура которых могут быть разными.

Несколько современных и качественные исследования по распознаванию болезней груш рассмотрели Fenu, Malloci, [156-159], с помощью ансамблевых методов.

Используя нейронные сети глубокого обучения VGG16, Inception V3, ResNet50 и ResNet101и трансферного обучения разработана схема «сеть DL + разрешение», которую использовали для анализа влияющих факторов и распознавания заболеваний. В данной статье экспериментальные результаты показали, что результат прямо пропорционально точности распознавания болезней и времени обучения, точность распознавания болезней груши реализованное на Pytorch составил 93.94%, 79.79% на данных обучения и проверки соответственно. На тестирующих данных мы имеем 78.71% точности. На обновленной и дополненной с собственными данными болезней груши открытой системы Plant Village точность CNN модели составил 97,5% и 94% соответственно

Сбор данных и Dataset. Сбор набора данных, болезни растений является дорогостоящим и трудоемким процессом для экспертов по данной области. В связи этим ограничиваются количество данных, которая является одной из основных проблем достижения точности задач распознавания. В настоящее время трансферное обучение является широко обсуждаемым и проверенным методом решения данной. При этом обычно модель сосредоточена на одном или двух конкретных наборах данных. Сбор данных для данного исследования осуществлялся на садовых участках, где выращиваются груши и яблони. Изображения, которые получаются из реальных условий (изображения обычно присылаются фермерами) получаются в увеличенном, уменьшенном размере, с поворотом на определенный угол, сдвиги. В связи с этим в работе используются

технологии глубокого обучения – аугментации данных. Широко использовалось также метод увеличения данных. Идентификация и анализ, основных влияющих факторов также играет очень важную роль в выявлении болезней растений. Используя нейронные сети глубокого обучения, в том числе VGG16, InceptionV3, ResNet50, Alex Net и Efficient Net, которые мы использовали для анализа основных признаков при распознавании заболеваний груши.

На рисунках 4.28-4.29 представлены изображения необработанные здоровые и больные листья груш, полученные от фермеров из различных регионов.

Рис. 4.28. Больные изображения листьев груши, полученные от фермеров.

.

Рис. 4.29. Здоровые листья груши

Метод увеличение данных. Для решения проблемы дисбаланса набора данных в данной статье рассматривается различные методы увеличения данных, которые существенно влияют на качество моделей. Увеличение данных — это метод, предназначенный для уменьшения переобучения будущей модели, при котором размер набора данных увеличивается за счет создания новых выборок из обучающего набора путем выполнения нескольких различных преобразований. В нашем случае размер изображений изменяется до 256 × 256 пикселей, и мы ее считаем ее как оптимизацию модели с прогнозированием на этих уменьшенных изображениях. В этом исследовании мы применяли стандартные методы: поворот изображения, вращение, сдвиг, масштабирование, перемещение, отражение и изменение цвета, чтобы увеличить размер набора данных почти вдвое по сравнению с исходным размером набора данных. Набор обучающих данных содержит изображения, которые в основном выровнены по вертикали. При прогнозировании мы используем различные технологии изображения листьев под разными углами. В методах увеличения данных использовался библиотеки глубокого обучения Keras на Python. Операции изменения ширины и высоты, вырезания, масштабирования, горизонтального поворота, яркости и заполнения выполнялись для изображений обычного класса. Степень поворота изображения была установлена случайным образом от 0 до 90.

Количество образцов для каждого класса было уравнено. Такое равное распределение позволяет использовать все данные вместо выбора случайных данных в процессе обучения. Ожидается, что такая ситуация повысит точность обучения и положительно повлияет на результаты классификации. Различные операции увеличения изображения показаны ниже на рис. 4.30-.4.33.

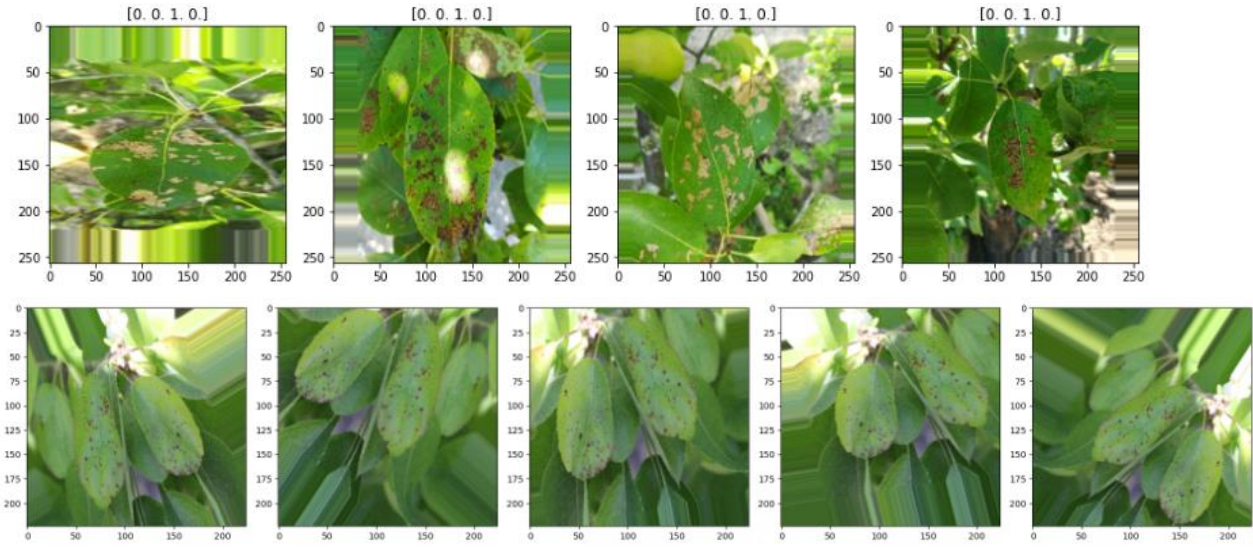


Рис. 4.30. Случайный поворот изображений.

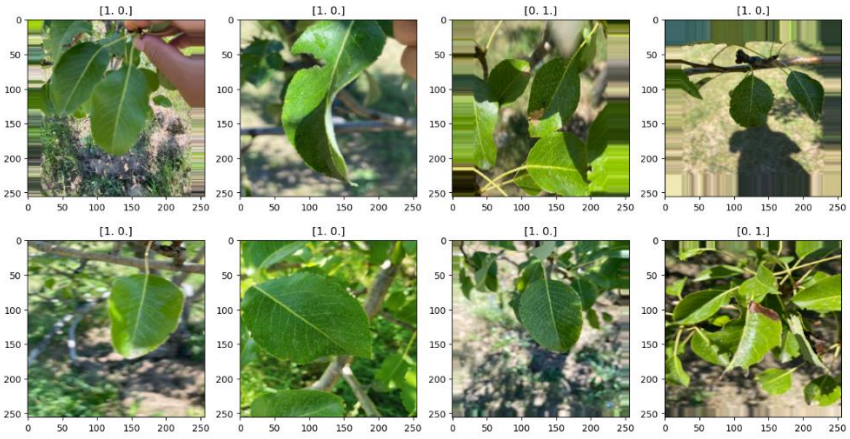


Рис. 4.31. Увеличение и уменьшение масштаба изображений

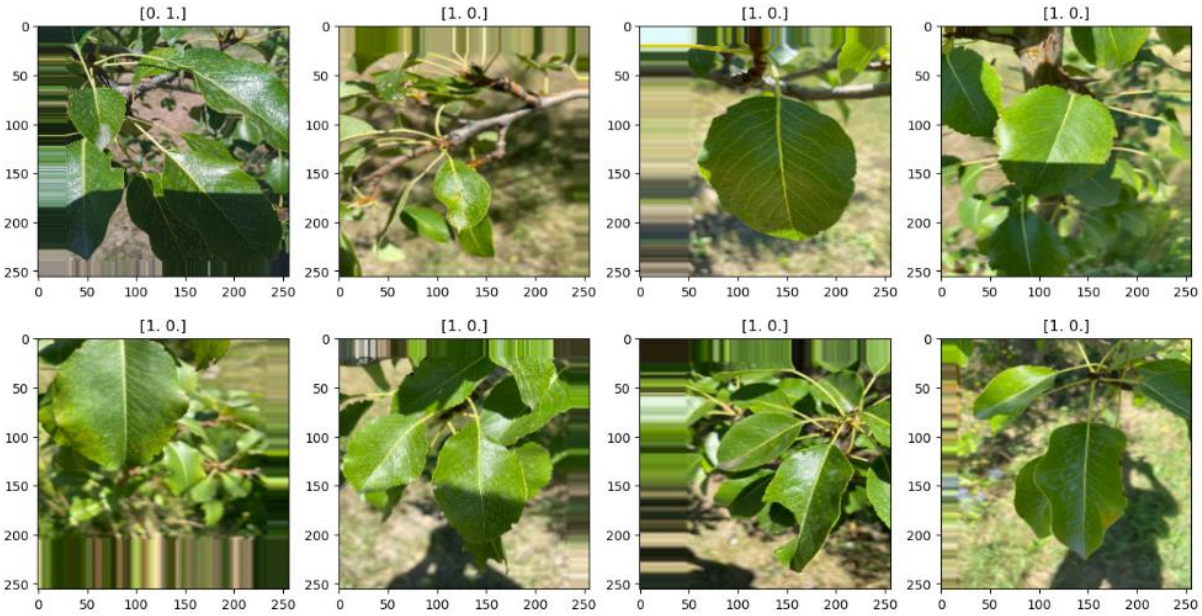


Рис. 4.32. Смещение изображений

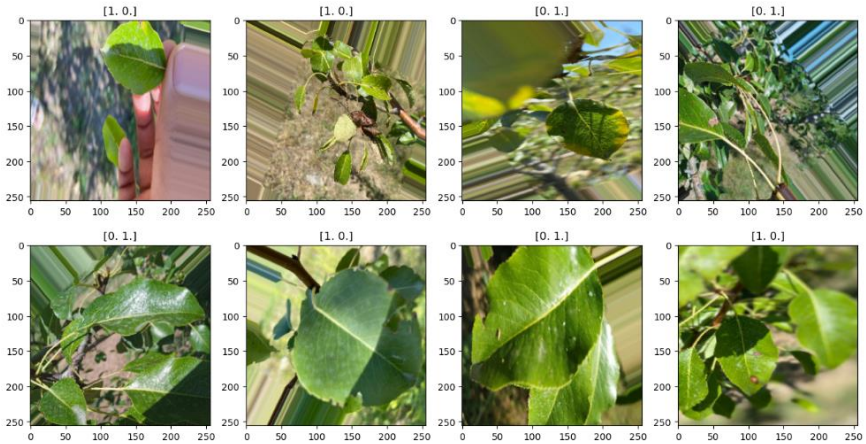


Рис. 4.33. Объединение нескольких методов увеличения данных

4.7.1. Архитектуры сверточных нейронных сетей. Модель EfficientNet

Представленная в 2019 году командой исследователей Google AI, EfficientNet стала идеальной архитектурой для решения многих сложных задач, включая распознавание объектов, сегментацию изображений и даже языковую обработку. Его успех обусловлен способностью сбалансировать два важнейших фактора глубокого обучения: эффективность вычислений и производительность модели.Традиционные модели глубокого обучения часто подразумевают компромисс между точностью и потреблением ресурсов. EfficientNet решает эту проблему, представляя новый подход, называемый «составным масштабированием».

Систематически масштабируя размеры модели (ширину, глубину и разрешение) принципиальным образом, EfficientNet достигает беспрецедентного уровня эффективности без ущерба для точности. Этот метод позволяет модели достичь оптимального баланса, делая ее адаптируемой к различным вычислительным бюджетам и возможностям оборудования.

EfficientNet — это архитектура сверточной нейронной сети и метод масштабирования, который равномерно масштабирует все измерения глубины/ширины/разрешения с использованием составного коэффициента. В отличие от традиционной практики произвольного масштабирования этих факторов, метод масштабирования EfficientNet равномерно масштабирует ширину, глубину и разрешение сети с помощью набора фиксированных коэффициентов масштабирования. Например, если мы хотим использовать несколько раз увеличить вычислительных ресурсов , например на 2^N, то мы можем просто увеличить глубину сети на α^N, ширина на β^N, и размер изображения на γ^N, где являются постоянными коэффициентами, определяемыми путем поиска по небольшой сетке исходной небольшой модели. EfficientNet использует составной коэффициент принципиальным образом равномерно масштабировать ширину, глубину и разрешение сети. EfficientNet использует метод, называемый составным коэффициентом, для простого, но эффективного масштабирования моделей. Вместо случайного увеличения ширины, глубины или разрешения составное масштабирование равномерно масштабирует каждое измерение с помощью определенного фиксированного набора коэффициентов масштабирования.

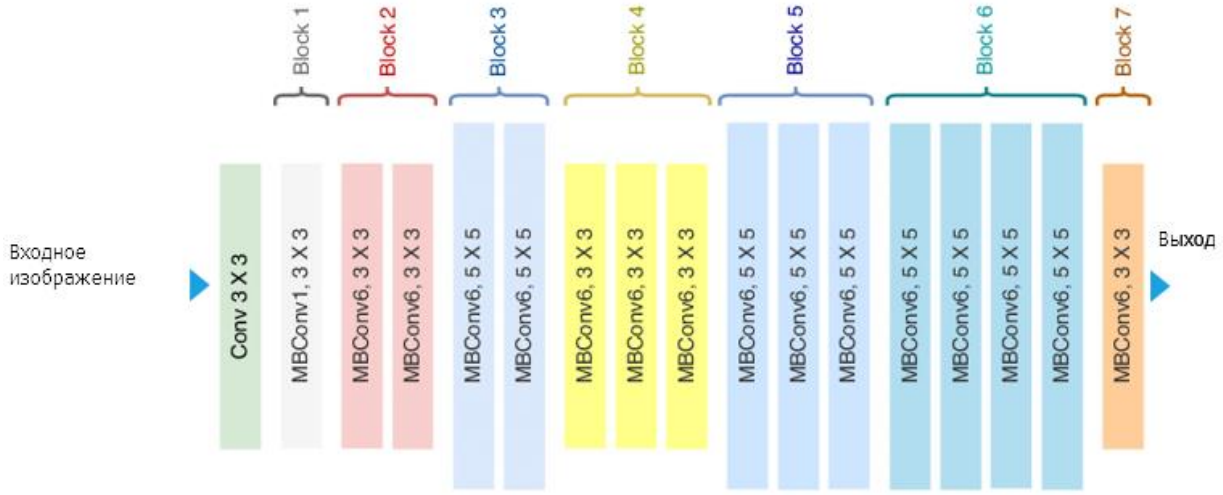


Рис. 4.34. Сверточная нейронная сеть EfficientNet

EfficientNet — это сверточная нейронная сеть, построенная на концепции «сложного масштабирования». Эта концепция касается давнего компромисса между размером модели, точностью и вычислительной эффективностью. Идея составного масштабирования заключается в масштабировании трех основных измерений нейронной сети: ширины, глубины и разрешения.

Ширина: масштабирование ширины относится к количеству каналов в каждом слое нейронной сети. Увеличивая ширину, модель может отображать более сложные закономерности и особенности, что приводит к повышению точности. И наоборот, уменьшение ширины приводит к созданию более легкой модели, подходящей для сред с низким уровнем ресурсов.

Глубина: масштабирование глубины относится к общему количеству слоев в сети. Более глубокие модели могут отображать более сложные представления данных, но они также требуют больше вычислительных ресурсов. С другой стороны, более мелкие модели эффективны в вычислительном отношении, но могут пожертвовать точностью.

Разрешение: Масштабирование разрешения включает в себя настройку размера входного изображения. Изображения с более высоким разрешением предоставляют более подробную информацию, что потенциально приводит к повышению производительности. Однако они также требуют больше памяти и вычислительной мощности. С другой стороны, изображения с более низким разрешением потребляют меньше ресурсов, но могут привести к потере мелких деталей.

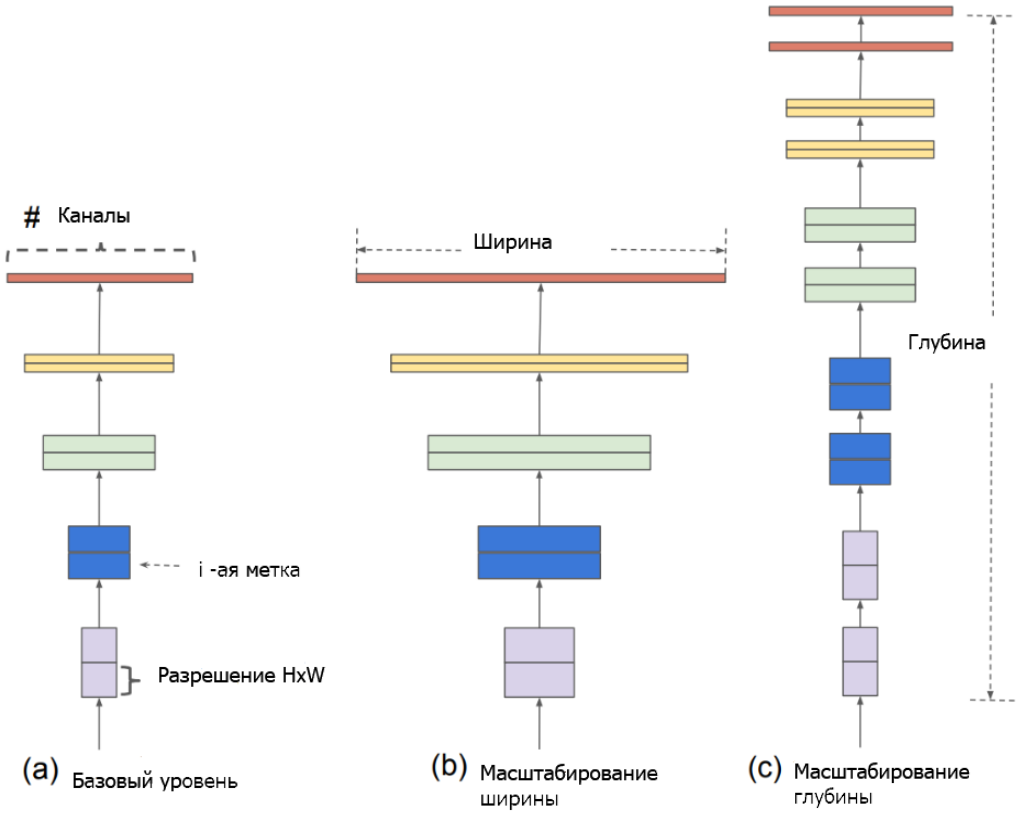


Рис. 4.35. Каналы, ширина и глубина сверточной нейронной сети EfficientNet.

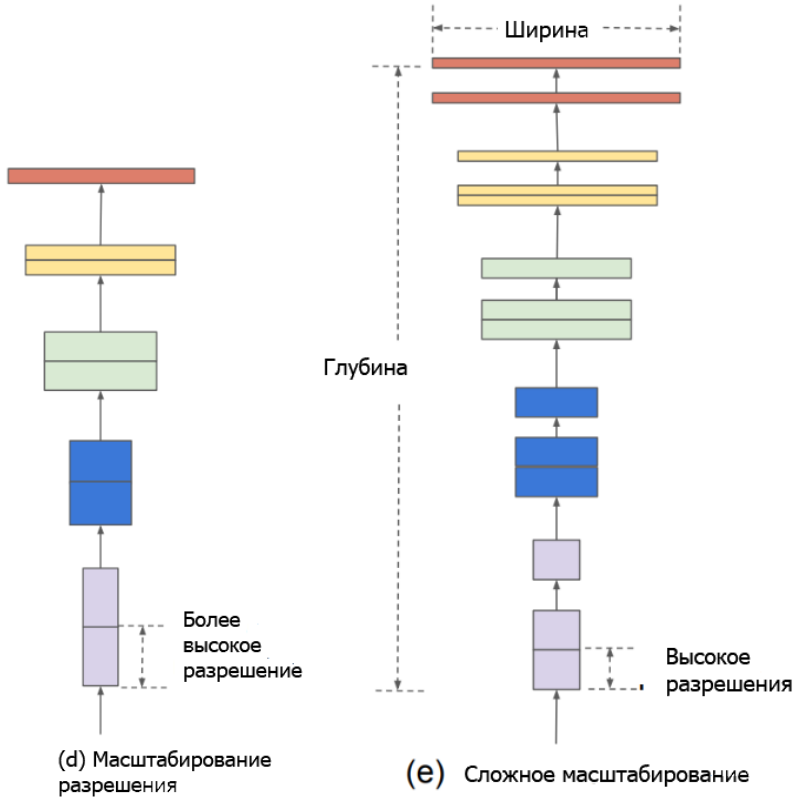


Рис. 4.36. Простое и сложное масштабирование сверточной нейронной сети EfficientNet.

Метод составного масштабирования оправдан интуицией, согласно которой, если входное изображение больше, то сети требуется больше слоев для увеличения рецептивного поля и больше каналов для захвата более мелкозернистых структур на большем изображении. Базовая сеть EfficientNet-B0 основана на инвертированных остаточных блоках MobileNetV2 в дополнение к блокам сжатия и возбуждения.

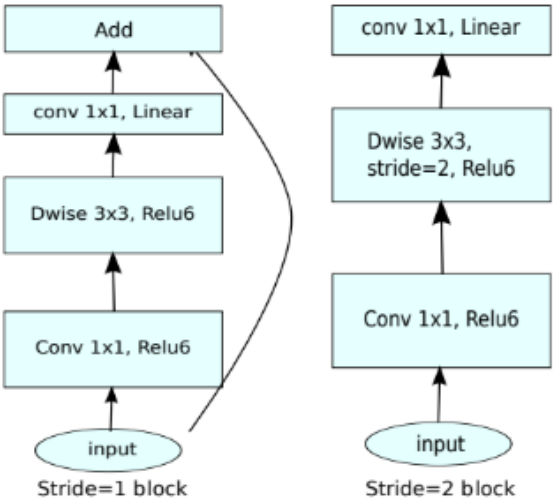


Рис. 4.37. Остаточные блоки сверточной нейронной сети EfficientNet

В данной работе для обучения моделей использовано наиболее часто используемые архитектуры сверточных нейронных сетей, включая Rez Net50, Inception, MobileNet и VGG16,19 и AlexNet, EfficientNetV2S и задача заключалась оценить точность этих глубоких моделей.

BGG19. Это сверточная нейронная сеть, разработанная Visual Geometric Group (VGG) в Оксфордском университете [41] в 2014 году. Точность модели достигла 92,7% точности теста Top-5 в ImageNet, который представляет собой базу данных, содержащую более 14 миллионов изображений. принадлежность к 1000 категориям объектов. Новизна архитектуры VGG заключалась в простоте использования более глубокого уровня с фильтрами меньшего размера для классификации изображений. Первая версия, VGG16, состоит из 16 слоев, а вторая версия, названная VGG19, состоит из 19 слоев. Обе архитектуры принимают на вход изображение размером 224×224 с тремя цветовыми каналами.

Inception. Это сеть, разработанная [43] в 2014 году и предназначенная для повышения производительности классификации изображений при сохранении оптимального использования вычислительных ресурсов.

Mobile Net. Идея [45], аналогичная сети Inception, MobileNet представляет собой сверточную архитектуру, созданную для достижения высоких результатов

за счет снижения вычислительной сложности, необходимой для сверточных слоев.

4.7.2. Трансферное обучение

Благодаря достойной эффективности глубокого обучения в последние годы распознавание болезней растений значительно улучшилось. Трансферное обучение — это метод, используемый в машинном обучении для повышения производительности моделей, несмотря на относительную нехватку данных. В этой статье мы предлагаем другую технологию трансферного обучения, обеспечивающую высокую эффективность универсального распознавания болезней растений основанное на нескольких наборах данных о заболеваниях растений. Трансферного обучения отличается от ныне популярной по следующим факторам. Для предварительного обучения модели используется сбор собственных данных, набор данных, относящийся к болезням груш по четырём классам, включая здоровые листья растений. Совместное использование открытых данных, например с Plant Village и данных, полученных самостоятельно. В нашем случае наряду с естественным увеличением данных и классов, для обучения модели применяется метод основанное на технологиях Pytorch. Еще одним из наборов данных, для сравнения наших результатов было использовано открытый набор данных DiaMosPlant. Таким образом все полученные модели для прогнозирования, являются результатами использования совместного использования нескольких наборов данных. Все модели основаны на различных CNN технологиях.

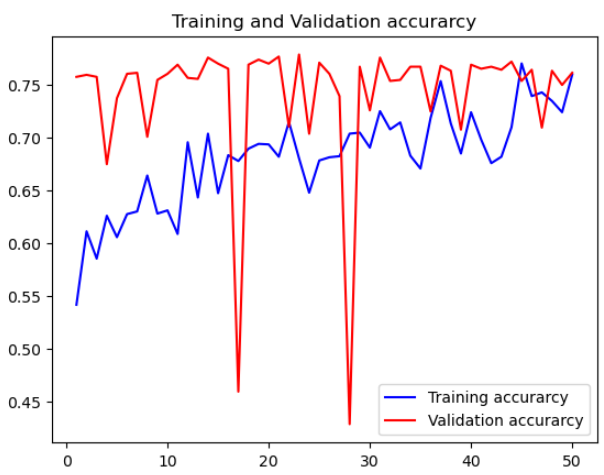
Полученные результаты в настоящей работе основаны на архитектурах сверточных нейронных сетей, как Resnet50, VGG16, InceptionV3, MobileNetV2, VGG19, AlexNet и EfficientNet, которые были использованы для построения моделей для прогнозирования болезней груш. Следует отметить, что использование трансферного -предварительного, обучения модели на наборе данных, связанных с растениями, с большим количеством изображений и классов (крупномасштабный), а также с широким разнообразием изображений ImageNet является необходимым условием для построения успешной-требуемой модели. Следовательно, поиск подходящего исходного набора данных, для построения улучшенных моделей имеет важное значение для распознавания болезней растений.

В диссертации подход к трансферному обучению с использованием четырех предварительно обученных архитектур CNN, включая Xception, ResNet50, EfficientNetB4 и MobileNet-V2, был принят для обнаружения

ржавчины на грушах. При трансферном обучении модель может использовать ранее полученные знания из других задач для решения новой проблемы. Поскольку при трансферном обучении модели предварительно уже обучаются на большом наборе данных, по сравнению с обучением модели с нуля требуется меньше данных, а также это может сэкономить время обучения и повысить производительность модели.

Ниже полученные оценки показывают, что предлагаемые методы трансферного обучения с технологиями глубокого обучения могут улучшить общую производительность классификации болезней груши. При использовании технологий распознавания на Pytorch результаты показали на 93.94% на обучающем множестве данных и 79.79% на данных валидации. На обновленной базе данных Plant Village прогнозирование болезни груш составил Train Accuracy: 0.97504325, Test Accuracy:0.9362568, Validation Accuracy: 0.94163530.

Ниже на рисунках приведены результаты реализации нейронных сверточных сетей ResNet, VGG16, Inception, CNN и Alex Net и EfficientNetV2S с помощью трансферного обучения.



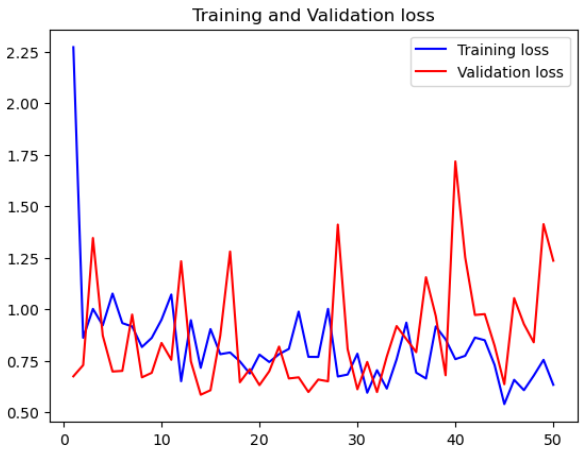


Рис.4.38. Трансферное обучение на основе модели Resnet50. Показатели ResNet следующие: Validation_loss: 0.5299 – Validation_accuracy: 0.7761

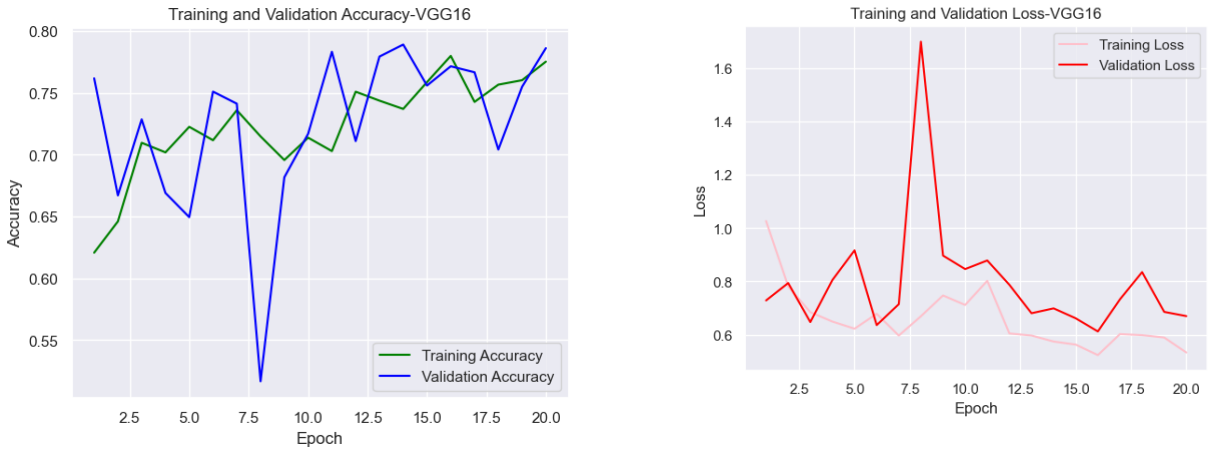


Рис.4.39. Трансферное обучение на основе модели VGG-16

Результаты трансферного обучения VGG116, следующие: validation_loss: 0.6699 – validation_accuracy: 0.7861

Трансферное обучение на основе модели InceptionV3:

Рис.4.40. Трансферное обучение на основе модели Inception.

Результаты трансферного обучения с помощью InceptionV3: train accuracy: 96%, valid accuracy: 93%

loss: 1.6773 valid loss: 4.2058

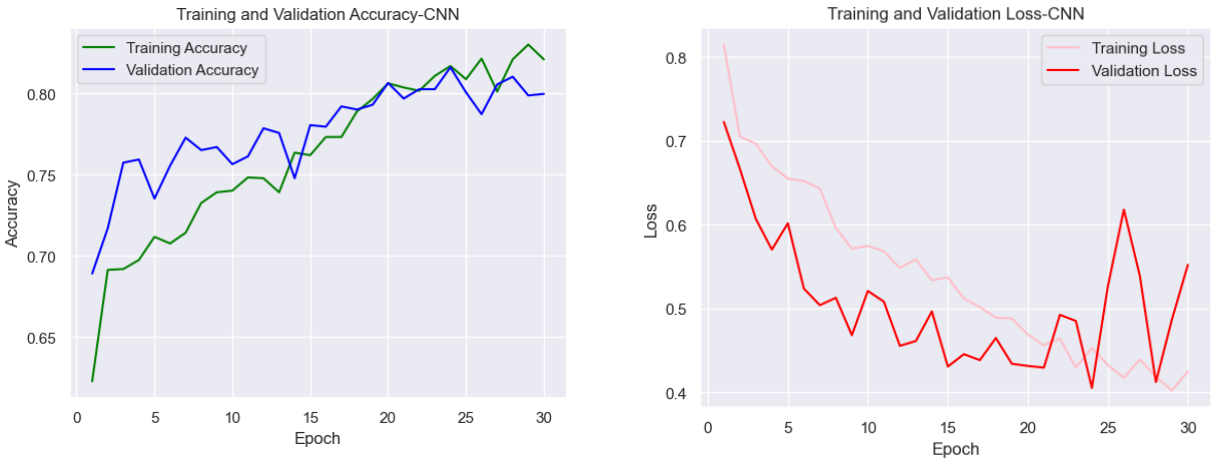


Рис.4.41. Результаты реализации CNN

Результаты CNN: validation_loss: 0.5520 - validation_accuracy: 0.7998

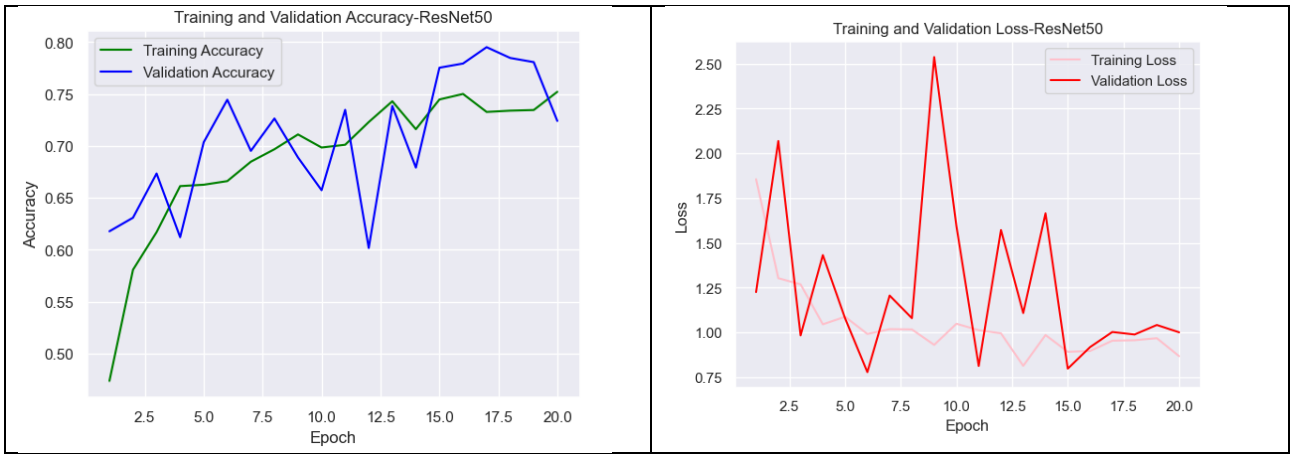


Рис. 4.42. Реализация ResNet50

loss: 0.8670- val_loss: 1.0004 accuracy: 0.7521 - val_accuracy: 0.7240

Реализация AlexNet на Pytorch:

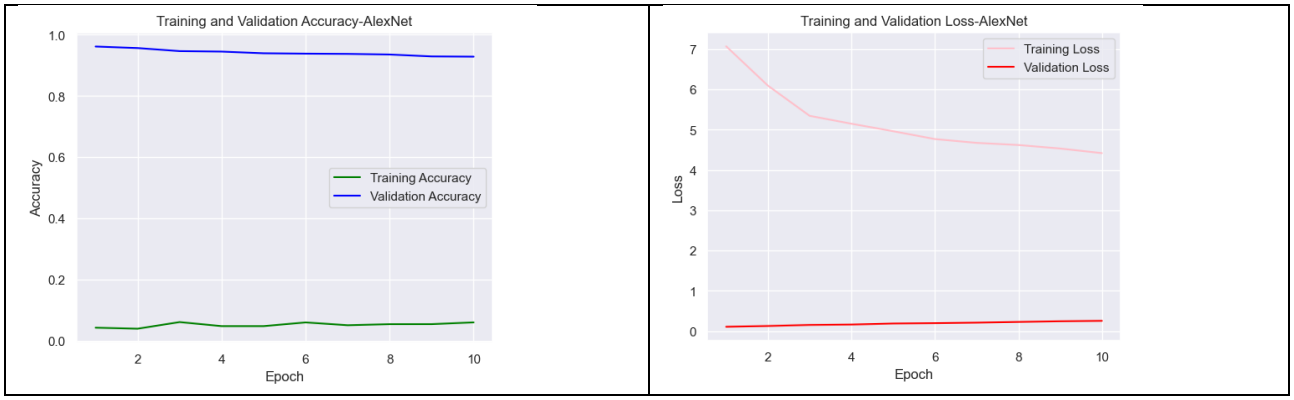


Рис. 4.43. Реализация AlexNet

Реализация на Pytorch AlexNet: loss: 4.4199 - val_loss: 0.2562 val_accuracy: 92%.

Для сравнения набором данных DiaMOS состоящих из 3006 изображений груш при разделении данных: length of train size :1788 length of validation size :767 length of test size :1218 результаты ошибки специальной схемой CNN реализованное с помощью Pytorch, показали следующий результат ошибки модели при обучении и валидации:

Достигнутые точности:

Train Accuracy: 0.9211409395973155

Test Accuracy: 0.7871396895787139

Validation Accuracy: 0.7679269882659713/

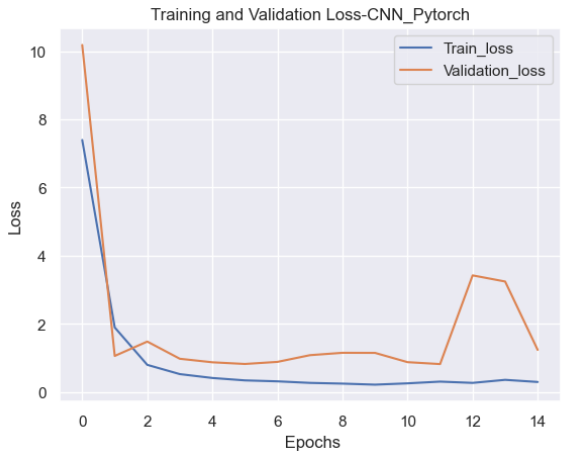


Рис. 4.44. Ошибка модели на обучающем и проверочных данных CNN реализованное на Pytorch

Реализация модели EfficientNetV2S реализованное для наших данных дали результаты

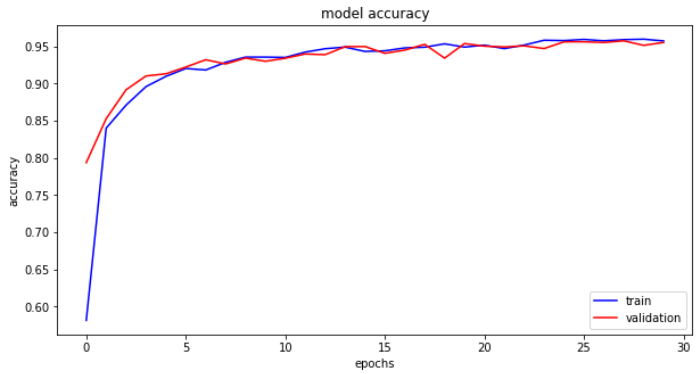


Рис. 4.45. Точность модели на обучающем и проверочных данных EfficientNetV2S

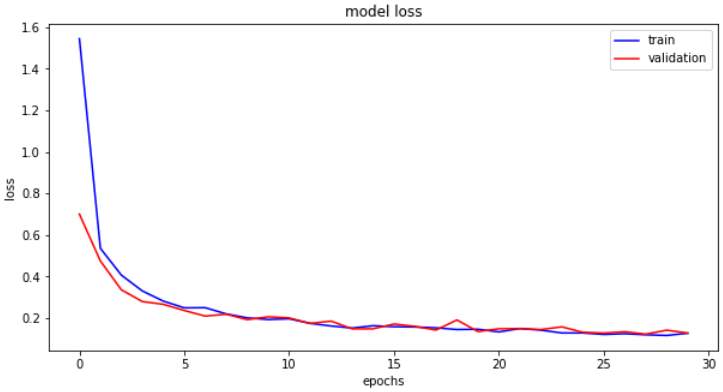


Рис. 4.46. Ошибка модели на обучающем и проверочных данных EfficientNetV2S

Всего эпох 30, использовался оптимизатор Адама, скорость обучения lr: 0.0010.

```
Epoch 30/30
200/200 [==============================] - 146s 730ms/step - loss: 0.1268 - accuracy: 0.9573 - val_loss: 0.1278 - val_accuracy:
0.9553 - lr: 0.0010
```

Точность модели достигала 95.84% , при этом ошибка модели loss=0.1254

```
1900/1900 [==============================] - 731s 384ms/step - loss: 0.1130 - accuracy: 0.9606
Train: accuracy = 0.960589  ;  loss = 0.112972
550/550 [==============================] - 73s 132ms/step - loss: 0.1278 - accuracy: 0.9553
Validation: accuracy = 0.955327  ;  loss = 0.127811
297/297 [==============================] - 40s 135ms/step - loss: 0.1254 - accuracy: 0.9584
Test: accuracy = 0.958421  ;  loss = 0.125422
```

Интеграция CNN моделей с алгоритмами машинного обучения, например с помощью алгоритма SVM на обучающем множестве получилось точность 0.9, на тестовом множестве 0.88, в то время CNN с алгоритмом Extreme Gradient Boost (XGBoost) получилось результат на обучающем множестве 1, а на тестовом точность 0.97.

Компьютер, использованный в этом исследовании, имел графический процессор GPU Ryzen 1080, который мог поддерживать размер пакета до 32; количество эпох моделях в основном были стабильны в течение всех протестированных номеров эпох. Была ранняя остановка при обучении моделей, чтобы обеспечить стабильность в диапазоне номеров эпох.

Размер изображения: изображения меньшего размера не использовались из-за небольшого размера ржавчины; использовался размер изображения (256×256), который могло поддерживать наше оборудование; глубина модели: глубина сети фиксирована для предварительно обученных моделей.

Картина прогнозирования болезней груш существенно меняется, когда в качестве обучения базы данных используется, дополненная база данных Plant

Village. С помощью технологии Pytorch была создана модель plant_disease_model_pear_full.pth для прогнозирования 33 класса болезней сельскохозяйственных растений, включая 4 болезни груш, с общим параметром нейронной CNN сети 52,589,249. Обучение нейронной сети осуществлялся с помощью Ryzen 1080 с GPU процессором. В качестве набора данных использовано открытая база данных Plant Village, DiaMosPlant и собственная база данных с общим объемом данных 57311 изображений из них проверки модели 14615 и валидации 23212 изображений, а для тестирования модели выделено 14615 изображений. Точность при обучении валидации и тестировании в данном случае имеет следующие показатели:

Train Accuracy: 0.97504325, Test Accuracy:0.9362568, Validation Accuracy: 0.94163530

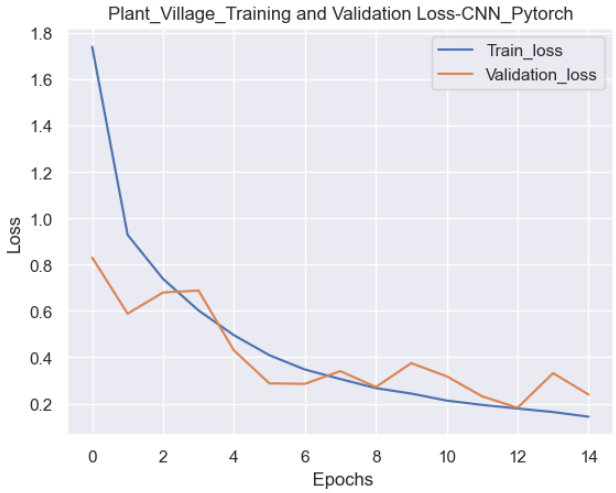


Рисунок. 4.47. Ошибка модели на обучающем и проверочных данных CNN реализованное на Pytorch с данными Plant Village

Полученные наши результаты сравниваются с наиболее тесно связанной литературой, так как эксперименты проводились в разных условиях с разными наборами данных. Известно, что среди них наиболее популярных является набор данных Plant Village.

Для трансферного обучения было выбрано InceptionV3, потому что у него меньше параметров по сравнению с другими моделями. Меньшее количество параметров приводит к сокращению времени на обучение модели. Поскольку мы собрали количество изображений с соответствующими метками болезней груши, недостаточно поддерживать обучение с нуля. Затем мы используем управляемое трансферное обучение в этой работе. Многие исследования показывают, что трансферное обучение дает хорошие результаты, не требуя большого количества образцов. Трансферное обучение, как мы уже упоминали, представляет собой

повторное использование предварительно обученной модели для построения новой, более продвинутой модели.

Таким образом использование трансферного обучения, в построении моделей дает определенные преимущества, потому что она может обучать глубокие нейронные сети с меньшим количеством данных, а также дает хорошую точность по сравнению обучения CNN с нуля.

При известном тяжести болезней растений во многих случаях процесс отдельной реализация моделей болезнь–урожайность с применением современных алгоритмов машинного обучения, согласно таблицам 4.7.1. - 4.7.4 дают хороший желаемый результат. Во всех приведенных ниже таблицах приведены результаты урожайности культур с ниже средней тяжестью болезни растений определенные выше полученные алгоритмами машинного обучения.

Таблица 4.7.1. Результаты оценок моделей, полученных алгоритмами машинного обучения

№	Estimates/ML Algorithm	R^2	MAE	MSE	RMSE	MAX	MAPE in %
1	Linear Regression	0,01	2	8	3	13	11,06
2	Decision Tree Regression	-0,61	3	12	4	16	13,49
3	Stochostic Gradient Descent Regression	0	2	8	3	13	11,12
4	K – Nearest Neighbour (n_neighbors=5)	0,03	2	8	3	12	10,58
5	SVR	0,10	2	9	3	13	10,12
6	Gradient Boosting Regression	0,12	2	7	3	12	10,14
7	Random Forest Regression	0,11	2	7	12	12	10,19

Таблица 4.7.2. Результат прогнозирования с алгоритмом градиентного бустинга

```
XGBoost's Accuracy is:  0.9886363636363636
              precision    recall  f1-score   support

     alfalfa       1.00      0.96      0.98       106
       apple       1.00      1.00      1.00        13
      barley       1.00      1.00      1.00       104
        corn       1.00      0.97      0.99        39
        pear       1.00      1.00      1.00        29
      potato       0.97      1.00      0.98       149

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

Таблица 4.7.3. Результат прогнозирования с алгоритмом случайный лес

```
RF's Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

     alfalfa       0.97      0.99      0.98       106
       apple       1.00      1.00      1.00        13
      barley       1.00      1.00      1.00       104
        corn       1.00      1.00      1.00        39
        pear       1.00      1.00      1.00        29
      potato       0.99      0.98      0.99       149

    accuracy                           0.99       440
   macro avg       0.99      1.00      0.99       440
weighted avg       0.99      0.99      0.99       440
```

Для анализа голосующего алгоритма реализованы следующие ансамбли алгоритмов и результаты VotingRegressor дали следующие результаты:

Таблица 4.7.4. Результаты ансамблевого метода

№	Estimates/ML Algorithm	R^2	MAE	MSE
1	RandomForestRegressor AdaBoostRegressor VotingRegressor	0.800329	1.981150	6.039916
2	AdaBoost GradientBoostingRegressor VotingRegressor	0.998058	1.9922521	6.11885
3	Random Forest и GradientBoostingRegressor VotingRegressor	0.89814	1.9646782	6.03540
4	Random Forest	0.96578	1.971011	5.98589

	AdaBoost и GradientBoostingRegressor VotingRegressor			

Теперь приведем сравнительный анализ точности моделей и результаты расчетов с другими алгоритмами машинного обучения.

```
Results of applying machine learning algorithms
Decision Tree --> 0.6704545454545454
Навье-Байес --> 0.6568181818181819
SVM --> 0.3409090909090909
Logistic Regression --> 0.5659090909090909
RF --> 0.990909090909091
XGBoost --> 0.9886363636363636
```

В данном случае, у нас есть победители случайный лес и XGBoost, которые с большой точностью создают модель урожайности сельскохозяйственных культур.

4.8. Разработка и проектирование искусственного интеллекта на основе фреймворков Python

На основе построенной данной модели построено обширное веб приложение с использованием платформы Фреймворка Flask, которая завоевала большую популярность построения веб систем для многих прикладных задач. Веб система содержит всю информацию о построенной нашей модели. Она основано на искусственном интеллекте прогнозирования, какую культуру сажать фермеру, на основе вносимых удобрений, состава почвы и погодные условия по месту расположения фермерского хозяйства. Система содержит веб страницы прогнозирования об использовании удобрений. Важной страницей веб системы является использования элемента искусственного интеллекта компьютерного зрения по распознаванию болезней растений и рекомендации по ее лечению. Ниже приведены основные страницы системы. Полная версия программы построения веб системы приведено в Приложении 4.3. диссертации. Приведем основные страницы веб приложения. Главная страница сайта (index.html) содержит общее меню, страницу О нас , общие рекомендации и футер с информацией разработчика.

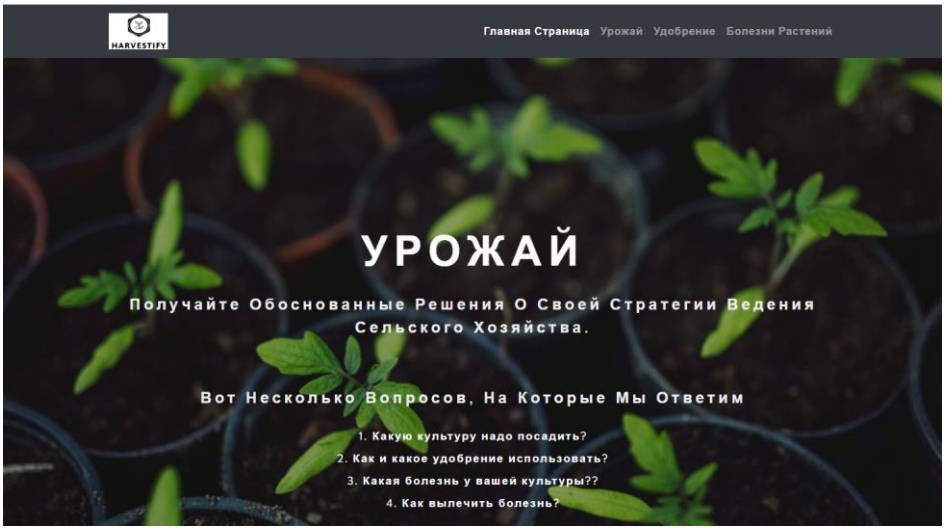
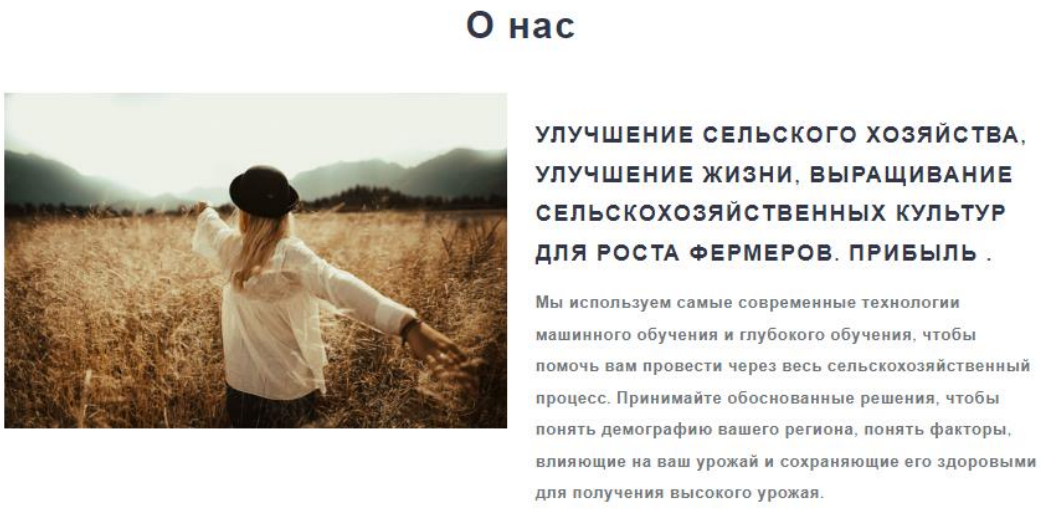


Рис.4.48. Главная страница веб системы с элементами искусственного интеллекта



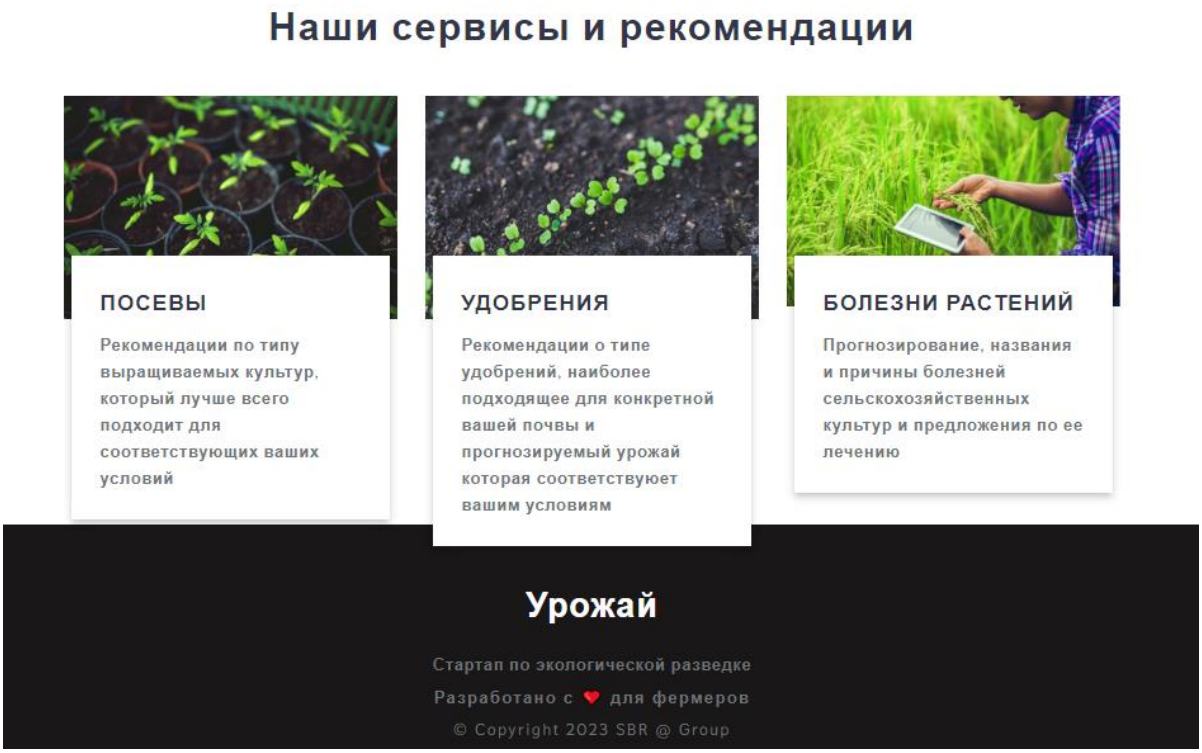


Рис.4.49. Главная страница веб-приложения

Далее на странице (crop.html) приложения делается прогнозирование о рекомендации фермеру по выращиванию сельскохозяйственной культуры на своем участке с учетом использования удобрений, почвенных характеристик, погодные условия и район расположения земельного участка.

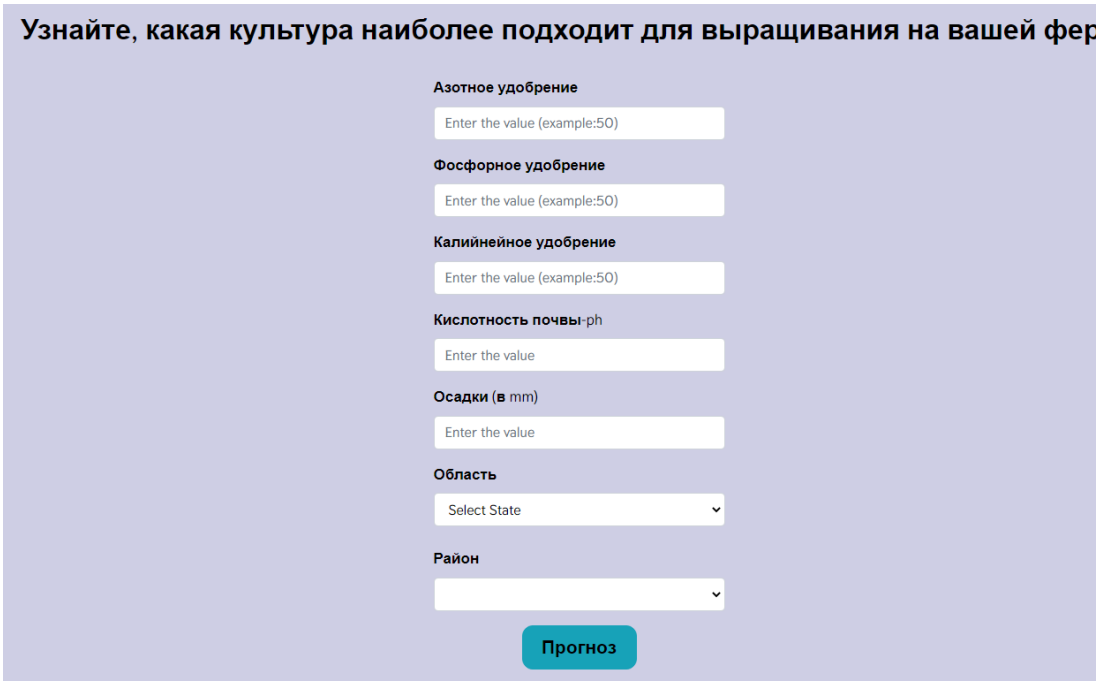


Рис.4.50. Страница прогноза рекомендаций по выращиванию растений

На странице crop-result.html на основе внесенных удобрений прогнозируется урожайность.

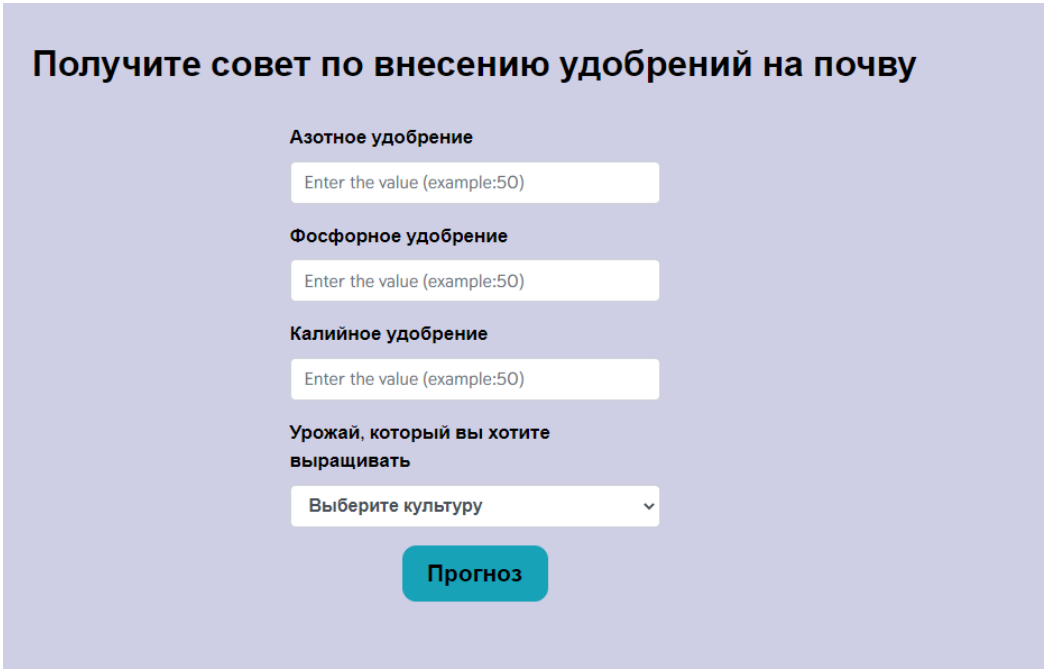


Рис.4.51. Страница прогноза рекомендаций по внесению удобрений

На странице disease.html прогнозируется болезни растений.

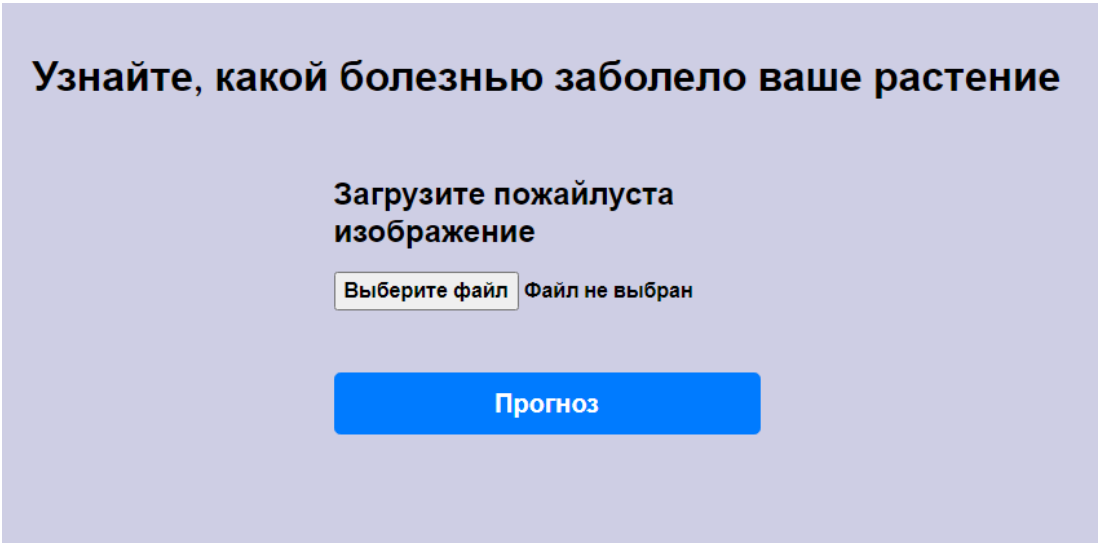


Рис.4.52. Страница прогноза, болезни растений

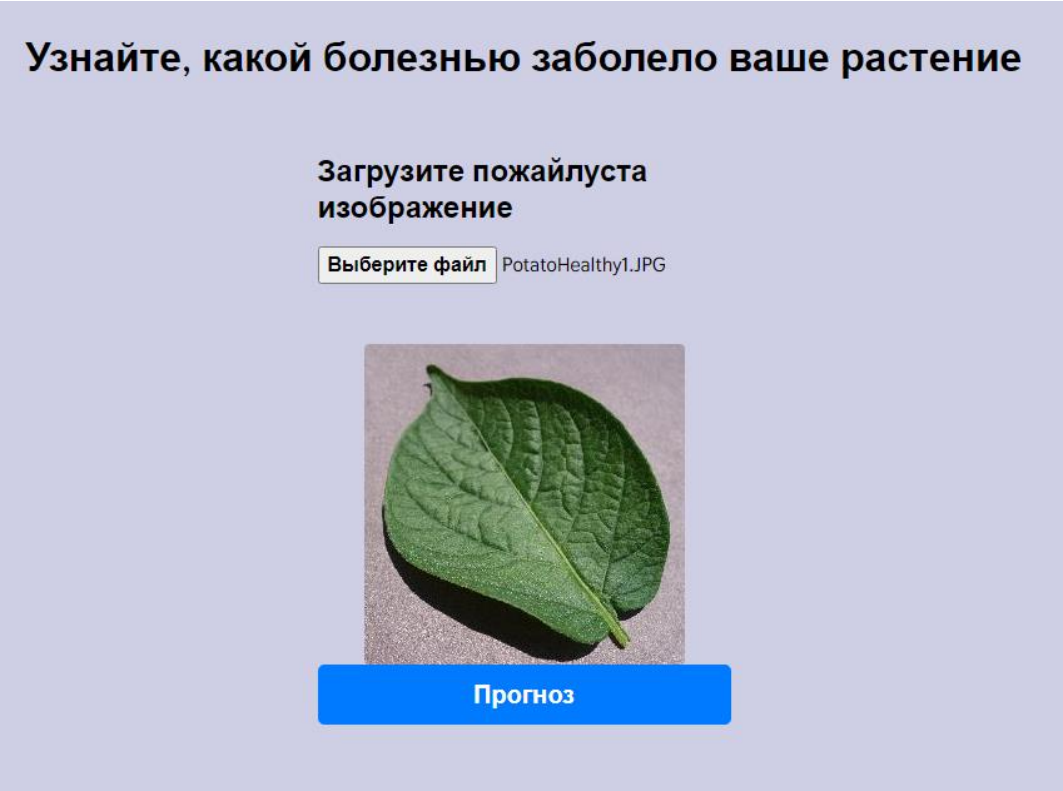


Рис.4.53. Загрузка данных для прогнозирования болезни растений

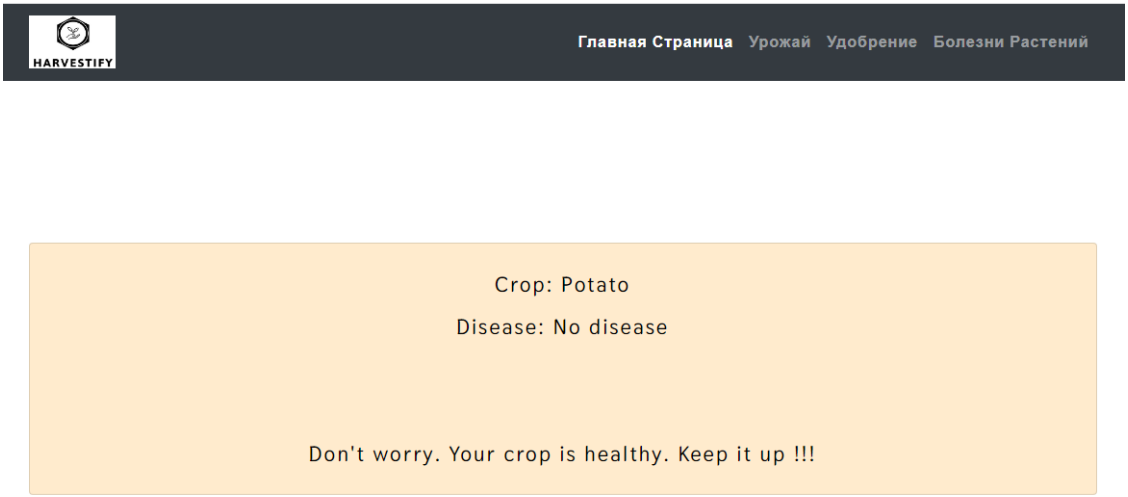


Рис.4.54. Информация прогнозирования болезни растений

Рассмотрим еще один пример на прогнозирования.

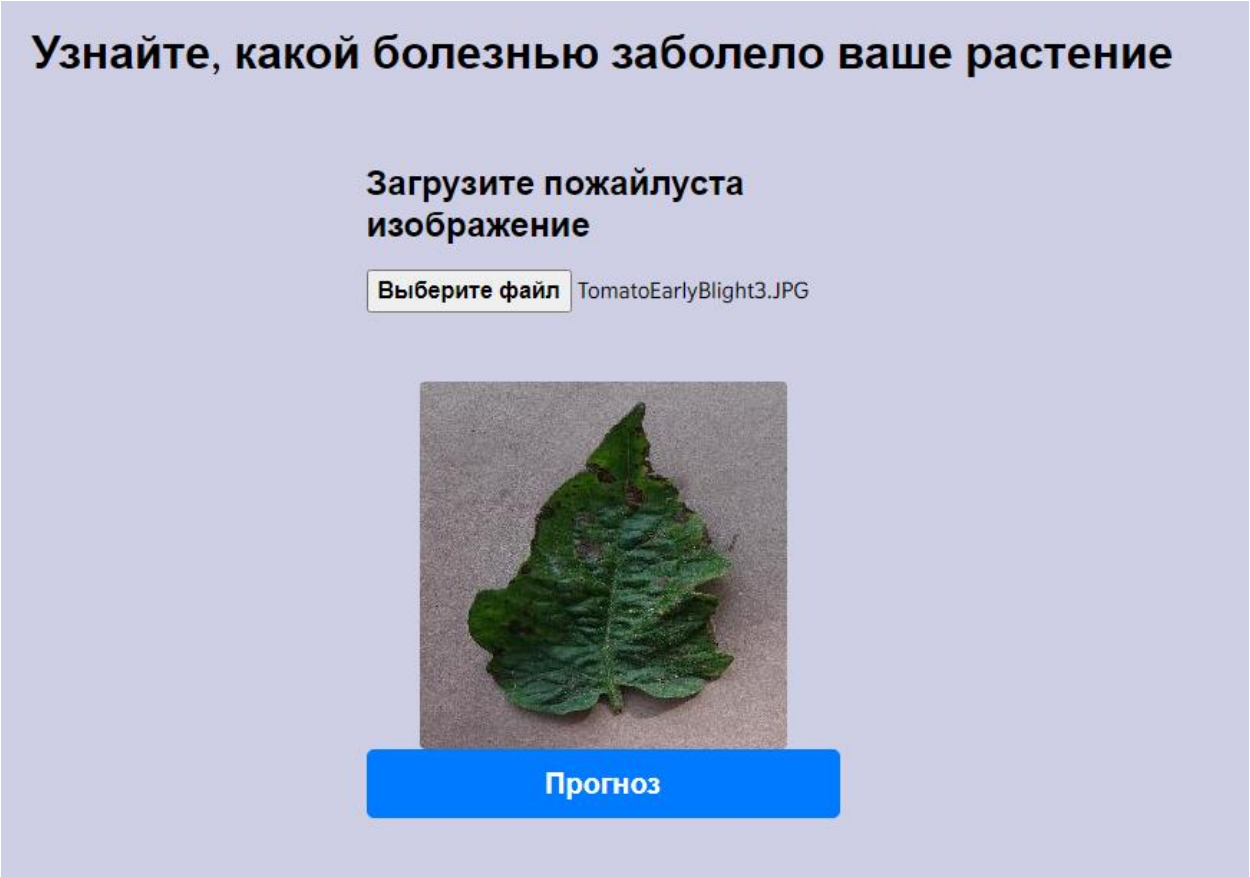


Рис.4.55. Пример для прогнозирования болезни растений

Вот результаты прогноза:

Культура: помидор

Болезнь: ранний упадок

Причина заболевания:

1. Ранняя гниль может быть вызвана двумя различными близкородственными грибами, Alternaria tomatophila и Alternaria solani.

2. Alternaria tomatophila более вирулентна для томатов, чем A. solani, поэтому в регионах, где обнаружена A. tomatophila, она является основной причиной ранней гнили на томатах. Однако, если A.tomatophila отсутствует, A.solani вызовет раннюю гниль на помидорах.

Как предотвратить/вылечить заболевание

1. Используйте семена, свободные от патогенов, или собирайте семена только с растений, свободных от болезней.

2. Избавьтесь от томатов и родственных культур как минимум на два года.

3. Контролируйте чувствительные сорняки, такие как черный паслен и мохнатый паслен, а также растения томатов-добровольцев на протяжении всего севооборота.

4. Правильно вносите удобрения для поддержания активного роста растений. В частности, не переусердствуйте с калием и поддерживайте адекватный уровень как азота, так и фосфора.

5. Избегайте работы с растениями, когда они мокрые от дождя, полива или росы.

6. Используйте капельное орошение вместо дождевателя, чтобы листва оставалась сухим.

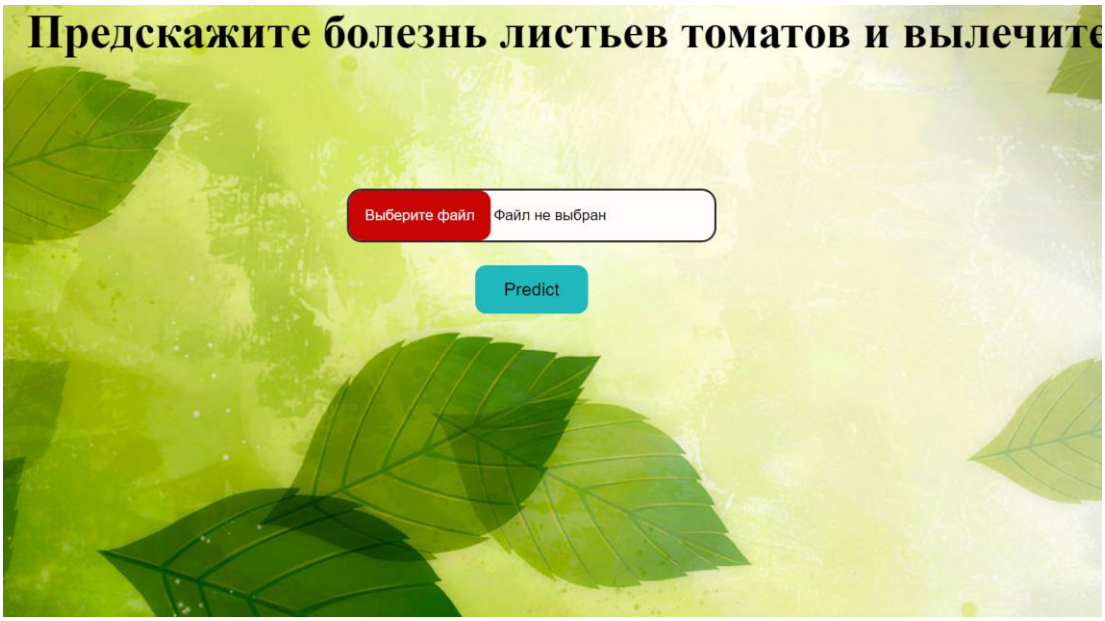


Рис.4.56. Пример искусственного интеллекта для прогнозирования болезни томатов

Загружаем изображение болезни томатов

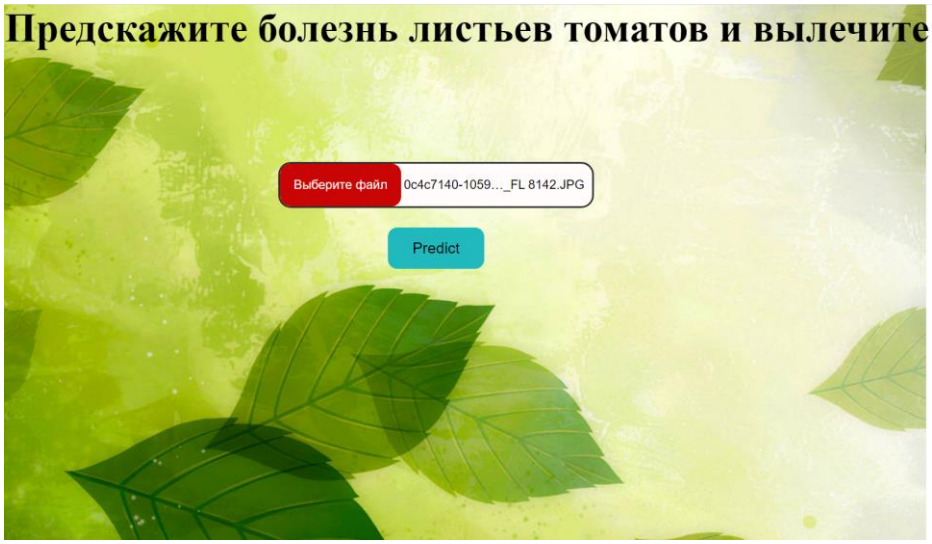


Рис.4.57. Загрузка данных в систему

Далее с помощью прогнозирования определяем, процесс диагностирования болезни растений.

Рис.4.58. Выявление болезни томата с помощью модели искусственного интеллекта и рекомендации по лечению.

Точно также определяется другие болезни растений из тестового меножества данных.

Рис.4.59. Тестирование болезни томата с помощью модели искусственного интеллекта и рекомендации по лечению.

Рис.4.60. Определение болезни томата с помощью прогнозирования на тестовых данных

Вот интерфейс другого искусственного интеллекта для определения болезни хлопчатника. Отметим, что модель обучалось на открытых данных, отражающе наиболее ярких болезней хлопчатника. В настоящее время данное направление данное является перспективным направлением для развития легкой промышленности КР.

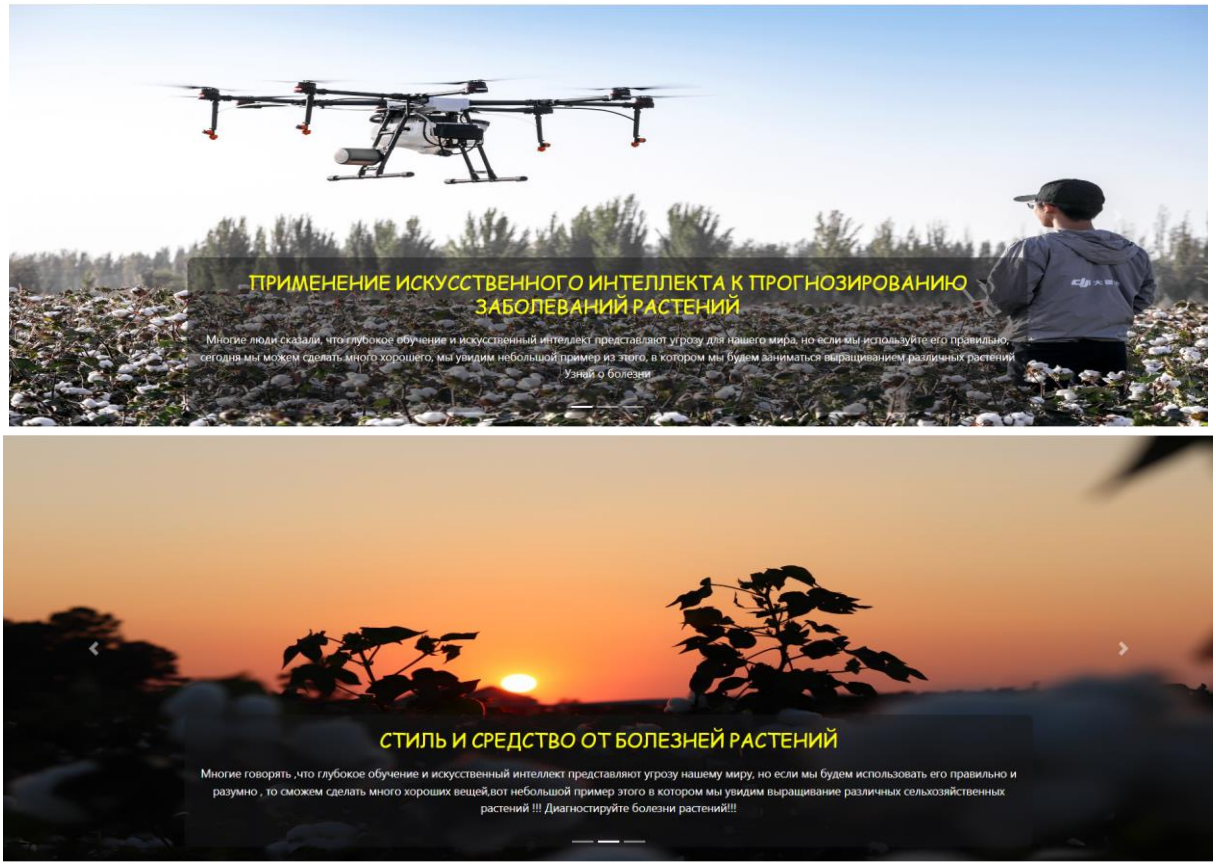


Рис.4.61. Главная страница искусственного интеллекта определения болезней хлопчатника

Рис.4.62. Вся страница искусственного интеллекта по прогнозированию болезни хлопчатника

Настоящий искусственный интеллект является новым направлением в распознавании болезней хлопчатников, которая в настоящее время возрождается на юге республики и является весьма актуальной темой для

развития легкой промышленности КР. Весь процесс обучения данной модели осуществлялось на Colaboratory Google с использованием GPU процессора.

ВЫВОДЫ К ГЛАВЕ 4

В данной главе диссертации исследован современный подход прогнозирования задач сельского хозяйства на основе глубокого обучения и технологии компьютерного зрения. Рассмотрены также прогнозирования урожайности с использованием нейронных сетей различной архитектуры. Подробно рассмотрены методы обучения моделей с применением метода регуляризации. Отдельно было рассмотрена технологии интеграции алгоритмов машинного обучения и современного подхода в распознавании изображений сверточные нейронные сети. Методы CNN занимают особое место в прогнозировании многих задач сельского хозяйства. Одним из них является метод распознавания болезней растений, которое изучалось в данной главе диссертации является трансферное обучения моделей с применением компьютерного зрения. В данном направлении рассмотрены уже обученные модели на больших данных ImageNet. В частности для трансферного обучения рассмотрены модели VGG16 ResNet для прогнозирования болезней растений основанное на платформе данных PlantVillage. Определено, что для определения болезней растений использованное с различными архитектурами нейронных сетей и основанные на глубоком обучении с технологиями компьютерного зрения ResNets значительно лучше работают для классификации изображений, когда некоторые параметры настраиваются и применяются такие методы, как планирование скорости обучения, отсечение градиента и уменьшение веса. Модель способна идеально предсказать каждое изображение в тестовом наборе без каких-либо ошибок. Важной проблемой, является использования Фреймворков Python для построения веб систем , которые используют уже обученные модели для задач прогнозирования. В данном направлении построены системы искусственного интеллекта по определению урожайности и рекомендации по использованию пестицидов в сельском хозяйстве. Построены искусственные интеллекты по распознаванию болезней растений для широкого круга задач по рекомендации лечения болезней растений по результатам прогнозирования. Модели для прогнозирования задач распознавания болезней многих растений проверены на реальных тестовых данных Иссык-Кульской области.

Как основной инструмент глубокого обучения изучен процесс построения моделей с нуля и использования трансферного обучения на базе данных

различных объемов данных. Модели были получены с использованием CNN, VGG16, VGG 19, InceptionV3 и Resnet50 и Alex Net для новых данных. Исследуется проблема переобучаемых моделей на основе технологий регуляризации. Получены различные модели для определенного вида растений, а именно для груш.

В данной главе диссертации было изучено применение различных сверточных нейронных сетей для обучения модели для повышения эффективности классификации болезней растений в наборе данных, собранном в полевых условиях. Для разработки моделей на основе CNN были сравнены четыре современные архитектуры CNN, Resnet50, MobileNetV2, InceptionV3? VGG16 и VGG19. Классификаторы обучены с помощью метода трансферного обучения с использованием методов увеличения данных.

Модели ResNet50 показали, что являются одними из наиболее точных и часто используемых моделей для обнаружения болезней растений.

Среди всех моделей высокой эффективности показали модели ResNet и EfficientNet . Результаты показали, что ResNet50 была второй лучшей моделью в обнаружении ржавчины, с несколько меньшей точностью, чем EfficientNet. Таким образом, можно сделать вывод, что EfficientNet и ResNet являются двумя наиболее сильными предварительно обученными моделями CNN для обнаружения болезней и вредителей растений. Надо лишь отметить, что гиперпараметры в данных моделях , такие как оптимизаторы, скорость обучения и размер пакета, являются сильно определяющими компонентами производительности моделей, где соответствующий выбор этих компонентов при обучении модели гарантирует высочайшую точность.

Для реализации моделей было использовано AMD Ryzen 7 5800X 8-Core Processor , 3.80 GHz с оперативной памятью32ГГб. CNN модели реализованы на базе GPU NVIDIA GeForce RTX 3090. Модели обучались и тестировался на наборе данных, который были собраны непосредственно на полях грушевого сада, пометив четыре основные вида заболевания. Чтобы получить репрезентативный набор данных и обучить более надежную модель реальным условиям применения, изображения были сделаны в разные промежутки времени с разным освещением, сходством заболеваний, масштабированием, углами, сложным фоном и несколькими листьями. Были отобраны лучшие модели для построения трансферного обучения.

Результаты показали, что лучшими обученными сверхточными нейронными сетями являются CNN с помощью технологий Pytorch, который дала ощутимый результат с использованием разные открытые наборы данных при обучении, валидации и тестирования модели данных. Получены CNN

модели с различными нейросетями как Resnet50, VGG16,19, MobileNetV2, InceptionV3 и для переобученных моделей был применен метод регуляризации и технологии увеличения данных. Применение вычислений с использованием специальной нейросети CNN на Pytorch с использованием GPU процессоров улучшила общую производительность реализуемых алгоритмов.

В заключение предлагаемое исследование обеспечивает применение трансферного обучения и применение технологий Pytorch в классификации болезней листьев, принятой для идентификации болезней груши. Производительность в данной статье вышеперечисленных моделей была оценена с помощью изображений, собранных в полевых условиях, количество, которых в дальнейшем необходимо увеличить. Вычислительная сложность во всех моделях преодолевается с помощью GPU процессоров и обучение моделей осуществлялся на Pytorch. При известном тяжести заболевания растений, модели которых мы создали выше, важно исследовать категории урожайности. В данной работе создан модель урожайности, болезнь-урожайность. В нашем случае в моделях урожайности присутствует категория болезнь, как категориальная переменная. Полученные результаты применения машинного обучения результаты приведены в таблицах 1-4. Для дальнейшего дополнения к данному исследованию необходимо применить методы глубокого обучения с различными входами и выходами, отражающие климатические параметры.

ЗАКЛЮЧЕНИЕ

В диссертационной работе проведено обширное исследование применения различных алгоритмов и методов машинного обучения и методов глубокого обучения для широкого круга сельскохозяйственных задач. В настоящее время роль искусственного интеллекта в прикладных исследованиях занимает особое место. Одним из архиважных задач для нашей страны, и вообще в мире, является продовольственная безопасность. В диссертационной работе построены модели и задачи прогнозирования урожайности сельскохозяйственных культур на основе машинного и глубокого обучения. Ключевым фактором для повышения урожайности кроме, погодных условий и технологий выращивания является борьба с болезнями растений. Распознавания и классификации болезней сельскохозяйственных растений является основой получения желаемого урожая фермеров. В данной диссертации построены различные модели прогнозирования урожайности сельскохозяйственных культур на основе элементов искусственного интеллекта. Использованы также методы классификации и распознавания на основе компьютерного зрения, которые могут использоваться для обнаружения болезней растений, а также для помощи фермерам в автоматическом обнаружении многих видов болезней . В этом направлении применяется различные архитектуры нейронных сетей в глубоком обучении с применением технологий компьютерного зрения для задач болезни растений. Кроме того, были обобщены несколько методов/сопоставлений для распознавания симптомов заболевания. Здесь имеет место развитие технологий глубокого обучения в последние годы для выявления болезней листьев растений. Разработанные модели по болезням растений и прогнозы по урожайности с развёртыванием в веб системы будут полезным инструментом для ученых, занимающихся выявлением болезней растений, а также окажет неоценимую помощь многим фермерам непосредственно занимающиеся растениеводством. Также, проводится сравнительное исследование между методами машинного и глубокого обучения. Несмотря на то, что в последние годы был отмечен значительный заметный прогресс, все еще остаются некоторые проблемы в исследованиях, которые необходимо устранить и внедрить эффективные методы обнаружения болезней растений.

Предложены новые технологии, а именно технологии машинного обучения, глубокое обучение и распознавания образов компьютерное зрение для широкого класса практических задач сельского хозяйства и фермерских хозяйств. Разработаны модели и веб системы на базе глубокого машинного

обучения и нейронных сетей с элементами искусственного интеллекта для прогнозирования задач сельского хозяйства.

В целях развития государственной программы по искусственному интеллекту и цифровизации сельского хозяйства в диссертационной работе выполнены работы по созданию моделей и прогнозирования многих задач сельского хозяйства. Полученные результаты, в главах 3 и 4, на основе новых методов машинного и глубокого обучения с технологиями компьютерного зрения для различных сельскохозяйственных задач играют существенную роль для экономики страны. Основные результаты диссертации опубликованы в зарубежных журналах Web of Science и Scopus, а также в местных изданиях входящих в список ВАК КР.

Многочисленные расчеты и модели были обучены с использованием интеллектуальных систем Keras и TensorFlow. Пакеты программ выполненные в диссертации были реализованы в системе Anaconda –Jupyter Notebook с использованием языка Python и приведены в виде приложений. Искусственный интеллект по распознаванию болезней сельскохозяйственных растений созданный на базе обученных моделей в диссертации созданы на Фреймворках Flask и Django.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Romero, J.R. Roncallo, P.F. Akkiraju, P.CPonzoni, . I. Echenique, V.C. &Carballido J.A. Using classification algorithms for predicting durum wheat yield in the province of Buenos Aires Comput. Electron. Agric., 96 (2013), pp. 173-179, 10.1016/j.compag.2013.05.006
2. Paul, M., Vishwakarma, S.K., Verma, A., 2015. Analysis of soil behaviour and prediction of crop yield using data mining approach. In: 2015 International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, pp. 766–771. https://doi.org/10.1109/CICN.2015.156.
3. Jeong, J.P. Resop, N.D. Mueller, D.H. Fleisher, K. Yun, E.E. Butler, S.H. Kim
4. Random forests for global and regional crop yield predictions.PLoS ONE, 11 (6) (2016), 10.1371/journal.pone.0156571
5. Gandhi, N., Petkar, O., Armstrong, L.J., Tripathy, A.K., 2016. Rice crop yield prediction in India using support vector machines. In: 2016 13th International Joint Conference on Computer Science and Software Engineering, JCSSE 2016. https://doi.org/10.1109/JCSSE.2016.7748856.
6. Gandhi, N., Armstrong, L., 2016. Applying data mining techniques to predict yield of rice in humid subtropical climatic zone of India. In: Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016, 1901–1906. Retrieved from https://ieeexplore.ieee.org/abstract/document/7724597/.
7. Sujatha, R., Isakki, P., 2016. A study on crop yield forecasting using classification techniques. In: 2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE 2016. https://doi.org/10.1109/ICCTIDE.2016.7725357.
8. Cheng, H. L. Damerow, Y. Sun, M. Blanke
9. Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks J. Imag., 3 (1) (2017), p. 6, 10.3390/jimaging3010006
10. Bargoti, S. J.P. Underwood Image segmentation for fruit detection and yield estimation in apple orchards J. Field Rob., 34 (6) (2017), pp. 1039-1060, 10.1002/rob.21699
11. Shekoofa, Y. Emam, N. Shekoufa, M. Ebrahimi, E. Ebrahimie
12. Determining the most important physiological and agronomic traits contributing to maize grain yield through machine learning algorithms: a new avenue in intelligent agriculture. PLoS ONE, 9 (5) (2014), Article e97288, 10.1371/journal.pone.0097288
13. Brown, G. (2017). "Ensemble learning" in Encyclopedia of Machine Learning and Data Mining. Ed. Sammut, K., Webb, G.I. (Boston, MA: Springer, USA), 393–402.

14. Breiman, L. (1996). Packing predictors. Max. To study. 24(2), 123–140. DOI: 10.1007 / BF00058655

15. CrossRef Full text | Google Scholar

16. Breiman, L. (2001). Random forests. Max. To study. 45(1), 5–32. DOI: 10.1023 / A: 1010933404324Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785–794).

17. You, J., Li, X., Low, M., Lobell, D., and Ermon, S. (2017). "Deep gaussian process for crop yield prediction based on remote sensing data," in Thirty-First AAAI Conference on Artificial Intelligence (San Francisco, CA), 4559–4566.

18. Lee SH, Chan CS, Mayo SJ, Remagnino P. How deep learning extracts and learns leaf features for plant classification. Pattern Recogn. 2017;71:1–13. Article Google Scholar

19. Tsaftaris SA, Minervini M, Scharr H. Machine learning for plant phenotyping needs image processing. Trends Plant Sci. 2016;21(12):989–91. Article CAS PubMed Google Scholar

20. Fuentes A, Yoon S, Park DS. Deep learning-based techniques for plant diseases recognition in real-field scenarios. In: Advanced concepts for intelligent vision systems. Cham: Springer; 2020.Google Scholar

21. Yang D, Li S, Peng Z, Wang P, Wang J, Yang H. MF-CNN: traffic flow prediction using convolutional neural network and multi-features fusion. IEICE Trans Inf Syst. 2019;102(8):1526–36.Article Google Scholar

22. Sundararajan SK, Sankaragomathi B, Priya DS. Deep belief cnn feature representation based content based image retrieval for medical images. J Med Syst. 2019;43(6):1–9.Article Google Scholar

23. Melnyk P, You Z, Li K. A high-performance CNN method for offline handwritten chinese character recognition and visualization. Soft Comput. 2019;24:7977–87.Article Google Scholar

24. Li J, Mi Y, Li G, Ju Z. CNN-based facial expression recognition from annotated rgb-d images for human–robot interaction. Int J Humanoid Robot. 2019;16(04):504–5.Article Google Scholar

25. Kumar S, Singh SK. Occluded thermal face recognition using bag of CNN(BoCNN). IEEE Signal Process Lett. 2020;27:975,Article Google Scholar

26. Wang X. Deep learning in object recognition, detection, and segmentation. Found Trends Signal Process. 2016;8(4):217–382.Article CAS Google Scholar

27. Boulent J, Foucher S, Théau J, St-Charles PL. Convolutional neural networks for the automatic identification of plant diseases. Front Plant Sci. 2019;10:941.Article PubMed PubMed Central Google Scholar

28. Kumar S, Kaur R. Plant disease detection using image processing—a review. Int J Comput Appl. 2015;124(2):6–9.Google Scholar

29. Martineau M, Conte D, Raveaux R, Arnault I, Munier D, Venturini G. A survey on image-based insect classification. Pattern Recogn. 2016;65:273–84.Article Google Scholar

30. Jayme GAB, Luciano VK, Bernardo HV, Rodrigo VC, Katia LN, Claudia VG, et al. Annotated plant pathology databases for image-based detection and recognition of diseases. IEEE Latin Am Trans. 2018;16(6):1749–57.Article Google Scholar

31. Kaur S, Pandey S, Goel S. Plants disease identification and classification through leaf images: a survey. Arch Comput Methods Eng. 2018;26(4):1–24.CAS Google Scholar

32. Shekhawat RS, Sinha A. Review of image processing approaches for detecting plant diseases. IET Image Process. 2020;14(8):1427–39.Article Google Scholar

33. Hinton GE, Salakhutdinov R. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–7.Article CAS PubMed Google Scholar

34. Liu W, Wang Z, Liu X, et al. A survey of deep neural network architectures and their applications. Neurocomputing. 2017;234:11–26.Article Google Scholar

35. Fergus R. Deep learning methods for vision. CVPR 2012 Tutorial; 2012.

36. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828.Article PubMed Google Scholar

37. Boureau YL, Le Roux N, Bach F, Ponce J, Lecun Y. [IEEE 2011 IEEE international conference on computer vision (ICCV)—Barcelona, Spain (2011.11.6–2011.11.13)] 2011 international conference on computer vision—ask the locals: multi-way local pooling for image recognition; 2011. p. 2651–8.

38. Zeiler MD, Fergus R. Stochastic pooling for regularization of deep convolutional neural networks. Eprint Arxiv. arXiv:1301.3557. 2013.

39. TensorFlow. https://www.tensorflow.org/.Torch/PyTorch. https://pytorch.org/.

40. Caffe. http://caffe.berkeleyvision.org/.

41. Theano. http://deeplearning.net/software/theano/.

42. Krizhenvshky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional networks. In: Proceedings of the conference neural information processing systems (NIPS), Lake Tahoe, NV, USA, 3–8 December; 2012. p. 1097–105.

43. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the 2015 IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 7–12 June; 2015. p. 1–9.

44. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv. arXiv:1409.1556. 2014.

45. Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. arXiv. arXiv:1611.05431. 2017.

46. Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence. 2016.

47. Huang G, Lrj Z, Maaten LVD, et al. Densely connected convolutional networks. In: IEEE conference on computer vision and pattern recognition. 2017. p. 2261–9.

48. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv. arXiv:1704.04861. 2017.

49. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50 × fewer parameters and < 0.5 MB model size. arXiv. arXiv:1602.07360. 2016.

50. Priyadharshini RA, Arivazhagan S, Arun M, Mirnalini A. Maize leaf disease classification using deep convolutional neural networks. Neural Comput Appl. 2019;31(12):8887–95.Article Google Scholar

51. Wen J, Shi Y, Zhou X, Xue Y. Crop disease classification on inadequate low-resolution target images. Sensors. 2020;20(16):4601.Article PubMed Central Google Scholar

52. Thangaraj R, Anandamurugan S, Kaliappan VK. Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. J Plant Dis Prot. 2020. https://doi.org/10.1007/s41348-020-00403-0.Article Google Scholar

53. Atila M, Uar M, Akyol K, Uar E. Plant leaf disease classification using efficientnet deep learning model. Ecol Inform. 2021;61:101182.

54. Article Google Scholar

55. Sabrol H, Kumar S. Recent studies of image and soft computing techniques for plant disease recognition and classification. Int J Comput Appl. 2015;126(1):44–55.Google Scholar

56. Yalcin H, Razavi S. Plant classification using convolutional neural networks. In: 2016 5th international conference on agro-geoinformatics (agro-geoinformatics). New York: IEEE; 2016.

57. Fuentes A, Lee J, Lee Y, Yoon S, Park DS. Anomaly detection of plant diseases and insects using convolutional neural networks. In: ELSEVIER conference ISEM 2017—The International Society for Ecological Modelling Global Conference, 2017. 2017.

58. Hasan MJ, Mahbub S, Alom MS, Nasim MA. Rice disease identification and classification by integrating support vector machine with deep convolutional

neural network. In: 2019 1st international conference on advances in science, engineering and robotics technology (ICASERT). 2019.

59. Thenmozhi K, Reddy US. Crop pest classification based on deep convolutional neural network and transfer learning. Comput Electron Agric. 2019;164:104906.Article Google Scholar Fang T, Chen P, Zhang J, Wang B. Crop leaf disease grade identification based on an improved convolutional neural network. J Electron Imaging. 2020;29(1):1.Article Google Scholar

60. Nagasubramanian K, Jones S, Singh AK, Sarkar S, Singh A, Ganapathysubramanian B. Plant disease identification using explainable 3D deep learning on hyperspectral images. Plant Methods. 2019;15(1):1–10.Article Google Scholar

61. Picon A, Seitz M, Alvarez-Gila A, Mohnke P, Echazarra J. Crop conditional convolutional neural networks for massive multi-crop plant disease classification over cell phone acquired images taken on real field conditions. Comput Electron Agric. 2019;167:105093.Article Google Scholar

62. Tianjiao C, Wei D, Juan Z, Chengjun X, Rujing W, Wancai L, et al. Intelligent identification system of disease and insect pests based on deep learning. China Plant Prot Guide. 2019;039(004):26–34.Google Scholar

63. Dechant C, Wiesner-Hanks T, Chen S, Stewart EL, Yosinski J, Gore MA, et al. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. Phytopathology. 2017;107:1426–32.Article PubMed Google Scholar

64. Wiesner-Hanks T, Wu H, Stewart E, Dechant C, Nelson RJ. Millimeter-level plant disease detection from aerial photographs via deep learning and crowdsourced data. Front Plant Sci. 2019;10:1550.Article PubMed PubMed Central Google Scholar

65. Shougang R, Fuwei J, Xingjian G, Peishen Y, Wei X, Huanliang X. Deconvolution-guided tomato leaf disease identification and lesion segmentation model. J Agric Eng. 2020;36(12):186–95.Google Scholar

66. Fujita E, Kawasaki Y, Uga H, Kagiwada S, Iyatomi H. Basic investigation on a robust and practical plant diagnostic system. In: IEEE international conference on machine learning & applications. New York: IEEE; 2016.

67. Mohanty SP, Hughes DP, Salathé M. Using deep learning for image-based plant disease detection. Front Plant Sci. 2016;7:1419. https://doi.org/10.3389/fpls.2016.01419.Article PubMed PubMed Central Google Scholar

68. Brahimi M, Arsenovic M, Laraba S, Sladojevic S, Boukhalfa K, Moussaoui A. Deep learning for plant diseases: detection and saliency map visualisation. In: Zhou J, Chen F, editors. Human and machine learning. Cham: Springer International Publishing; 2018. p. 93–117.

69. Chapter Google Scholar

70. Barbedo JG. Plant disease identification from individual lesions and spots using deep learning. Biosyst Eng. 2019;180:96–107.Article Google Scholar
71. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017;39(6):1137–49.Article PubMed Google Scholar
72. Liu W, Anguelov D, Erhan D, Szegedy C, Berg AC. SSD: Single shot MultiBox detector. In: European conference on computer vision. Cham: Springer International Publishing; 2016.
73. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
74. Redmon J, Farhadi A. Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 6517–25.
75. Redmon J, Farhadi A. Yolov3: an incremental improvement. arXiv preprint. arXiv:1804.02767. 2018.
76. Fuentes A, Yoon S, Kim SC, Park DS. A robust deep-learning-based detector for real-time tomato plant diseases and pests detection. Sensors. 2017;17(9):2022.Article PubMed Central Google Scholar
77. Ozguven MM, Adem K. Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms. Phys A Statal Mech Appl. 2019;535(2019):122537.Article Google Scholar
78. Zhou G, Zhang W, Chen A, He M, Ma X. Rapid detection of rice disease based on FCM-KM and faster R-CNN fusion. IEEE Access. 2019;7:143190–206. https://doi.org/10.1109/ACCESS.2019.2943454.Article Google Scholar
79. Xie X, Ma Y, Liu B, He J, Wang H. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. Front Plant Sci. 2020;11:751.Article PubMed PubMed Central Google Scholar
80. Singh D, Jain N, Jain P, Kayal P, Kumawat S, Batra N. Plantdoc: a dataset for visual plant disease detection. In: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. 2019.
81. Sun J, Yang Y, He X, Wu X. Northern maize leaf blight detection under complex field environment based on deep learning. IEEE Access. 2020;8:33679–88. https://doi.org/10.1109/ACCESS.2020.2973658.Article Google Scholar
82. Bhatt PV, Sarangi S, Pappula S. Detection of diseases and pests on images captured in uncontrolled conditions from tea plantations. In: Proc. SPIE 11008, autonomous air and ground sensing systems for agricultural optimization and phenotyping IV; 2019. p. 1100808. https://doi.org/10.1117/12.2518868.
83. Zhang B, Zhang M, Chen Y. Crop pest identification based on spatial pyramid pooling and deep convolution neural network. Trans Chin Soc Agric Eng. 2019;35(19):209–15.Google Scholar

84. Ramcharan A, McCloskey P, Baranowski K, Mbilinyi N, Mrisho L, Ndalahwa M, Legg J, Hughes D. A mobile-based deep learning model for cassava disease diagnosis. Front Plant Sci.
2019;10:272. https://doi.org/10.3389/fpls.2019.00272.Article PubMed PubMed Central Google Scholar

85. Selvaraj G, Vergara A, Ruiz H, Safari N, Elayabalan S, Ocimati W, Blomme G. AI-powered banana diseases and pest detection. Plant Methods.
2019. https://doi.org/10.1186/s13007-019-0475-.Article PubMed PubMed Central Google Scholar

86. Tian Y, Yang G, Wang Z, Li E, Liang Z. Detection of apple lesions in orchards based on deep learning methods of CycleGAN and YOLOV3-dense. J Sens.
2019. https://doi.org/10.1155/2019/7630926.Article Google Scholar

87. Zheng Y, Kong J, Jin X, Wang X, Zuo M. CropDeep: the crop vision dataset for deep-learning-based classification and detection in precision agriculture. Sensors. 2019;19:1058. https://doi.org/10.3390/s19051058Article PubMed Central Google Scholar

88. Arsenovic M, Karanovic M, Sladojevic S, Anderla A, Stefanović D. Solving current limitations of deep learning based approaches for plant disease detection. Symmetry.
2019;11:21. https://doi.org/10.3390/sym11070939.Article Google Scholar

89. Fuentes AF, Yoon S, Lee J, Park DS. High-performance deep neural network-based tomato plant diseases and pests diagnosis system with refinement filter bank. Front Plant Sci.
2018;9:1162. https://doi.org/10.3389/fpls.2018.01162.Article PubMed PubMed Central Google Scholar

90. Jiang P, Chen Y, Liu B, He D, Liang C. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. IEEE Access.
2019. https://doi.org/10.1109/ACCESS.2019.2914929.Article PubMed PubMed Central Google Scholar

91. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell. 2015;39(4):640–51.Google Scholar

92. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: 2017 IEEE international conference on computer vision (ICCV). New York: IEEE; 2017.

93. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Berlin: Springer; 2015. p. 234–41. https://doi.org/10.1007/978-3-319-24574-4_28.

94. Badrinarayanan V, Kendall A, Cipolla R. Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell. 2019;39(12):2481–95. Article Google Scholar

95. Wang Z, Zhang S. Segmentation of corn leaf disease based on fully convolution neural network. Acad J Comput Inf Sci. 2018;1:9–18.Google Scholar

96. Wang X, Wang Z, Zhang S. Segmenting crop disease leaf image by modified fully-convolutional networks. In: Huang DS, Bevilacqua V, Premaratne P, editors. Intelligent computing theories and application. ICIC 2019, vol. 11643. Lecture Notes in Computer Science. Cham: Springer;
2019. https://doi.org/10.1007/978-3-030-26763-_62.Chapter Google Scholar

97. Lin K, Gong L, Huang Y, Liu C, Pan J. Deep learning-based segmentation and quantification of cucumber powdery mildew using convolutional neural network. Front Plant Sci. 2019;10:155.Article PubMed PubMed Central Google Scholar

98. Kerkech M, Hafiane A, Canals R. Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach. Comput Electron Agric. 2020;174:105446.Article Google Scholar

99. Stewart EL, Wiesner-Hanks T, Kaczmar N, Dechant C, Gore MA. Quantitative phenotyping of northern leaf blight in UAV images using deep learning. Remote Sens. 2019;11(19):2209.Article Google Scholar

100. Wang Q, Qi F, Sun M, Qu J, Xue J. Identification of tomato disease types and detection of infected areas based on deep convolutional neural networks and object detection techniques. Comput Intell Neurosci.
2019. https://doi.org/10.1155/2019/9142753.Article PubMed PubMed Central Google Scholar

101. Hughes DP, Salathe M. An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. Comput Sci. 2015.

102. Shah JP, Prajapati HB, Dabhi VK. A survey on detection and classification of rice plant diseases. In: IEEE international conference on current trends in advanced computing. New York: IEEE; 2016.

103. Prajapati HB, Shah JP, Dabhi VK. Detection and classification of rice plant diseases. Intell Decis Technol. 2017;11(3):1–17.Google Scholar

104. Barbedo JGA, Koenigkan LV, Halfeld-Vieira BA, Costa RV, Nechet KL, Godoy CV, Junior ML, Patricio FR, Talamini V, Chitarra LG, Oliveira SAS. Annotated plant pathology databases for image-based detection and recognition of diseases. IEEE Latin Am Trans. 2018;16(6):1749–57.Article Google Scholar

105. Brahimi M, Arsenovic M, Laraba S, Sladojevic S, Boukhalfa K, Moussaoui A. Deep learning for plant diseases: detection and saliency map visualisation. In: Zhou J, Chen F, editors. Human and machine learning. Human–

computer interaction series. Cham: Springer; 2018. https://doi.org/10.1007/978-3-319-90403-0_6.

106. Tyr WH, Stewart EL, Nicholas K, Chad DC, Harvey W, Nelson RJ, et al. Image set for deep learning: field images of maize annotated with disease symptoms. BMC Res Notes. 2018;11(1):440.Article Google Scholar

107. Thapa R, Snavely N, Belongie S, Khan A. The plant pathology 2020 challenge dataset to classify foliar disease of apples. arXiv preprint. arXiv:2004.11958. 2020.

108. Wu X, Zhan C, Lai YK, Cheng MM, Yang J. IP102: a large-scale benchmark dataset for insect pest recognition. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR). New York: IEEE; 2019.

109. Huang M-L, Chuang TC. A database of eight common tomato pest images. Mendeley Data. 2020. https://doi.org/10.17632/s62zm6djd2.1.

110. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Proceedings of the 2014 conference on advances in neural information processing systems 27. Montreal: Curran Associates, Inc.; 2014. p. 2672–80.

111. Pu Y, Gan Z, Henao R, et al. Variational autoencoder for deep learning of images, labels and captions [EB/OL]. 2016–09–28. arxiv:1609.08976.

112. Oppenheim D, Shani G, Erlich O, Tsror L. Using deep learning for image-based potato tuber disease detection. Phytopathology. 2018;109(6):1083–7.Article Google Scholar

113. Too EC, Yujian L, Njuki S, Yingchun L. A comparative study of fine-tuning deep learning models for plant disease identification. Comput Electron Agric. 2018;161:272–9.Article Google Scholar

114. Chen J, Chen J, Zhang D, Sun Y, Nanehkaran YA. Using deep transfer learning for image-based plant disease identification. Comput Electron Agric. 2020;173:105393.

115. Zhang S, Huang W, Zhang C. Three-channel convolutional neural networks for vegetable leaf disease recognition. Cogn Syst Res. 2018;53:31–41. https://doi.org/10.1016/j.cogsys.2018.04.006.

116. Liu B, Ding Z, Tian L, He D, Li S, Wang H. Grape leaf disease identification using improved deep convolutional neural networks. Front Plant Sci. 2020;11:1082. https://doi.org/10.3389/fpls.2020.01082. Article PubMed PubMed Central Google Scholar

117. Karthik R, Hariharan M, Anand S, et al. Attention embedded residual CNN for disease detection in tomato leaves. Appl Soft Comput J. 2020;86:105933. Article Google Scholar

118. Guan W, Yu S, Jianxin W. Automatic image-based plant disease severity estimation using deep learning. Comput Intell Neurosci. 2017;2017:2917536.

119. Barbedo JGA. Factors influencing the use of deep learning for plant disease recognition. Biosyst Eng. 2018;172:84–91.

120. Barbedo JGA. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Comput Electron Agric. 2018;153:46–53.

121. Nawaz MA, Khan T, Mudassar R, Kausar M, Ahmad J. Plant disease detection using internet of thing (IOT). Int J Adv Comput Sci Appl. 2020. https://doi.org/10.14569/IJACSA.2020.0110162.

122. Сабитов Б.Р. Проектирование базы данных бизнес-процессов агропромышленного комплекса в среде Embarcadero ER Studio [Текст] / Сабитов Б.Р. // Приволжский Научный Вестник, № 11(63)-2016.Стр.17-21 , Режим доступа:

123. https://cyberleninka.ru/article/n/proektirovanie-bazy-dannyh-obrazovatelnyh-protsessov-v-srede-embarcadero-er-studio

124. Сабитов Б.Р. Разработка сетевой информационной системы «Успеваемость » в среде Embarcadero Rad Studio XE10 Seattle с применением LiveBinding технологий [Текст] / Сабитов Б.Р. ,Алмазбекова З., Сейтбеков А.// Приволжский Научный Вестник, № 1(65)-2017.Стр.31-40, Режим доступа: https://cyberleninka.ru/article/n/razrabotka-setevoy-informatsionnoy-sistemy-uspevaemost-v-srede-embarcadero-rad-studio-xe-10-seattle-s-primeneniem-livebinding-tehnologiy

125. Б.Р. Сабитов А. Сейтбеков, У.Т. Керимов, Б.Д. Давлятова. Математическая модель оптимального распределения инвестиционного вложения между отраслями , Журнал : Экономика и предпринимательство, № 9 (ч.3),стр.611-613, 2017 г. Режим доступа: http://www.intereconom.com/archive/373.html

126. Сабитов Б.Р. Problems of expert system devolepment of the investment design in agro-indastrial [Текст] / Sabitov B.R., Choroev K.Ch., Seitbekov A., Kerimov U.T complex of KyrgyzstanЭкономика и предпринимательство. 2018. № 12 (101). С. 414-417. Режим доступа:https://www.elibrary.ru/item.asp?id=36722118

127. Сабитов Б.Р. Моделирование структурных диспропорций экономики КР [Текст] / Б.И. Бийбосунов, Б. Сабитов, К. Чороев, Б. Давлятова // Фундаментальные исследования. – Пенза, 2019. - № 7. – с. 21 - 26. – Режим доступа: https://www.elibrary.ru/item.asp?id=39164245

128. Применение технологий машинного обучения в прикладных задачах, Сабитов Б.Р. и др., Вестник КНУ, специальный выпуск. 2019. http://lib.knu.kg/files/2019/vknu2019spec.pdf

129. Численный метод построения моделей для прогнозирования экономических показателей сельского хозяйства. Бийбосунов Б.И.,Сабитов Б.Р.,Алмасбекова З.,Эсенаманова Г.К., Электронный Журнал |Дневник науки |

www.dnevniknauki.ru | 2020 № 2| СМИ ЭЛ № ФС 77-68405 ISSN 2541-8327 , Режим доступа: https://www.elibrary.ru/item.asp?id=42512576

130. Сабитов Б.Р. Технологии использования глубокого обучения для задач сельского хозяйства. Журнал «Современны проблемы механики» выпуск №43(1), 2021 г. https://www.elibrary.ru/item.asp?id=48227011

131. Сабитов Б.Р. Построение моделей и прогнозирование с применением алгоритмов машинного обучения задач сельского хозяйства. Сабитов Ч.Б., Алмазбекова З., Сабитов Б.Р. Журнал «Современны проблемы механики» выпуск №45(3), 2021 г. https://www.elibrary.ru/item.asp?id=48227011

132. Сабитов Б.Р. Сравнительный анализ прогнозирование нелинейных моделей с применением алгоритмов дерево решений и ближайших соседей машинного обучения. Сабитов Б.Р., Омуралиева Б.Б., 2021, Вестник научный журнал Кыргызского государственного университета имени И.Арабаева специальный выпуск 2 часть 2021г

133. Сабитов Б.Р. Технологии построения нелинейных моделей с применением дерево решений Сабитов Б.Р., Омуралиева Б.Б., 2021, Вестник научный журнал Кыргызского государственного университета имени И.Арабаева специальный выпуск 2 часть 2021г

134. Сабитов Б.Р. Технологии визуализации характеристик АПК с примененеием библиотеки Python. Сабитов Б.Р., Мусуралиева Д.Э. и др. Вестник научный журнал Кыргызского государственного университета имени И.Арабаева специальный выпуск 2 часть 2021г Вестник научный журнал Кыргызского государственного университета имени И.Арабаева специальный выпуск 2 часть 2021г стр.157 -162,

135. Сабитов Б.Р. Использование алгоритмов машинного обучения для построения линейных моделей задач сельского хозяйства. Сабитов Б.Р., Мусуралиева Д.Э. и др Вестник КГУ им.И.Арабаева, специальный выпуск 2 часть, 171 -176, Бишкек , 2021

136. Применение регрессионного анализа для прогнозирования урожайности культур в сельском хозяйстве с применением машинного обучения. Вестник научный журнал Кыргызского государственного университета имени И.Арабаева специальный выпуск 2 часть 2021г.

137. Сабитов Б.Р. Технологии визуализации характеристик АПК с примененеием библиотеки Python, Бийбосунов Б.И., Сабитов Б.Р., Мусуралиев а Д.Э. и др. Вестник КГУ им.И.Арабаева, специальный выпуск 2 часть, 124 - 129,Бишкек

138. Сабитов Б.Р.Анализ данных характеристик АПК с применением библиотек Pandas Сабитов Б.Р., Эсенаманова Г.К. и др. Вестник КГУ им.И.Арабаева, специальный выпуск 2 часть,135 -140, Бишкек

139. Прогнозирования оттока фермеров с применением машинного обучения, Сабитов Б.Р., Эсенаманова Г.К. и др. Вестник КГУ им.И.Арабаева, специальный выпуск 2 часть,141 -144, Бишкек

140. Сабитов Б.Р., Сейтказиева Н.С., Кубанычбекова А.К. Технологии интеграции машинного обучения с веб приложениями.Журнал “Современные проблемы механики/ гидрогазодинамика, геомеханика, геотехнологии и информатика” Выпуск 47 (1), 2022 г., с. 38-46, https://www.elibrary.ru/item.asp?id=50062452

141. Сабитов Б.Р., Сейтказиева Н.С., Осмонов Э.Т., Калтаев Б.Э Идентификация болезней растений с применением нейронных сетей журнал “Современные проблемы механики/ гидрогазодинамика, геомеханика, геотехнологии и информатика” Выпуск 47 (1), 2022 г., с. 72-82,https://www.elibrary.ru/item.asp?id=50062455

142. Сабитов Б.Р., Орозобекова А.К., Жадилов Б.М., Сейтбеков А., Шамырова Д.Р., Шеримбекова Э.Б. Визуализация данных урожайности сельскохозяйственных культур с применением python технологий вестник КГУСТА № 1(75), Бишкек, 2022, с.82-86.DOI:10.35803/1694-5298.2022.1.82-86 , https://www.elibrary.ru/item.asp?id=48339898

143. Сабитов Б.Р.,Орозобекова А.К., Жадилов Б.М., Сейтбеков А., Шамырова Д.Р., Шеримбекова Э.Б. Применение системы FbProphet на базе технологий машинного обучения при прогнозировании задач АПК, Вестник КГУСТА № 1(75), Бишкек, 2022, с.87-94. https://www.elibrary.ru/item.asp?id=48339899

144. Сабитов Б.Р., Сейтказиева Н.С., Картанова А.Дж. Методы компьютерного зрения в задачах прогнозирования болезней растений с использованием трансферного обучения. [Текст] // Журнал Института машиноведения и автоматики НАН КР «Проблемы автоматики и управления» № 3 (45), Бишкек, 2022 – 12 с. https://www.elibrary.ru/item.asp?id=50020291

145. Сабитов Б.Р., Сейтказиева Н.С., Картанова А.Дж. Идентификация болезней томатов на основе многоклассовой классификации. [Текст] // Журнал Института машиноведения и автоматики НАН КР «Проблемы автоматики и управления» № 3 (45), Бишкек, 2022 – 11 с. https://www.elibrary.ru/item.asp?id=50020292

146. Сабитов и др. «Моделирование и прогнозирование задач сельского хозяйства на основе машинного обучения». Труды Международной научно-практической конференции. «Научно-технологическое развитие АПК для целей устойчивого развития». Бишкек. 2022 г.

147. Baratbek Sabitov et all. Deep learning Methods for Recognition of Orchard Crops /Baratbek Sabitov1, Saltanat Biibsunova2, Altyn Kashkaroeva3, Bolotbek Biibosunov.IJCSNS .International Journal of Computer Science and Network Security.VOL. 22 No 10,October 2022/

https://doi.org/10.22937/IJCSNS.2022.22.10.33,
http://paper.ijcsns.org/07_book/202210/20221033.pdf

148. Сабитов Б.Р., Сейтказиева Н.С., Кашкароева А.А.Построение сверточной нейронной сети для прогнозирования болезней кукурузы. Журнал Наука и Новые технологии. Бишкек. 2022 ,№ 6 ,http://www.science-journal.kg/ru/journal/1/about

149.Сабитов Б.Р., Сейтказиева Н.С., Кашкароева А.А.Бинарная задача классификации болезни растений с применением технологий глубокого обучения. Журнал Наука и Новые технологии Бишкек,№ 6 , http://www.science-journal.kg/ru/journal/1/about 2022

150.Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. arXiv. arXiv:1611.05431. 2017.

151.Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence. 2016.

152. Huang G, Lrj Z, Maaten LVD, et al. Densely connected convolutional networks. In: IEEE conference on computer vision and pattern recognition. 2017. p. 2261–9.

153. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv. arXiv:1704.04861. 2017.

154. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50 × fewer parameters and < 0.5 MB model size. arXiv. arXiv:1602.07360. 2016.

155. Priyadharshini RA, Arivazhagan S, Arun M, Mirnalini A. Maize leaf disease classification using deep convolutional neural networks. Neural Comput Appl. 2019;31(12):8887–95.Article Google Scholar

156. Fenu, G.; Malloci, F.M. Using Multi-Output Learning to Diagnose Plant Disease and Stress Severity. Complexity 2021, 2021, 18. [Google Scholar] [CrossRef]

157. Fenu, G.; Malloci, F.M. Forecasting Plant and Crop Disease: An Explorative Study on Current Algorithms. Big Data Cogn. Comput. 2021, 5, 2. [Google Scholar] [CrossRef]

158. FenuG, Malloci F.M.. Classification of Pear Leaf Diseases Based on Ensemble Convolutional Neural Networks. AgriEngineering 2023, 5(1), 141-152; [Google Scholar] [CrossRef]

159. Fenu, G.; Malloci, F.M. An Application of Machine Learning Technique in Forecasting Crop Disease. In Proceedings of the 2019 3rd International Conference on Big Data Research, Cergy-Pontoise, France, 20–22 November 2019; pp. 76–82. [Google Scholar] [CrossRef]

160. Brahimi, M.; Boukhalfa, K.; Moussaoui, A. Deep learning for tomato diseases: Classification and symptoms visualization. Appl. Artif. Intell. 2017, 31, 299–315. [Google Scholar] [CrossRef]
161. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. Comput. Electron. Agric. 2018, 147, 70–90. [Google Scholar] [CrossRef][Green Version]
162. Li, Y.; Wang, H.; Dang, L.M.; Sadeghi-Niaraki, A.; Moon, H. Crop pest recognition in natural scenes using convolutional neural networks. Comput. Electron. Agric. 2020, 169, 105174. [Google Scholar] [CrossRef]
163. Bereciartua-Pérez, A.; Gómez, L.; Picón, A.; Navarra-Mestre, R.; Klukas, C.; Eggers, T. Insect counting through deep learning-based density maps estimation. Comput. Electron. Agric. 2022, 197, 106933. [Google Scholar] [CrossRef]
164. Wang, A.; Zhang, W.; Wei, X. A review on weed detection using ground-based machine vision and image processing techniques. Comput. Electron. Agric. 2019, 158, 226–240. [Google Scholar] [CrossRef]
165. Nevavuori, P.; Narra, N.; Lipping, T. Crop yield prediction with deep convolutional neural networks. Comput. Electron. Agric. 2019, 163, 104859. [Google Scholar] [CrossRef]
166. Albarrak, K.; Gulzar, Y.; Hamid, Y.; Mehmood, A.; Soomro, A.B. A Deep Learning-Based Model for Date Fruit Classification. Sustainability 2022, 14, 6339. [Google Scholar] [CrossRef]
167. Hamid, Y.; Wani, S.; Soomro, A.B.; Alwan, A.A.; Gulzar, Y. Smart Seed Classification System based on MobileNetV2 Architecture. In Proceedings of the 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 25–27 January 2022; pp. 217–222. [Google Scholar] [CrossRef]
168. Sladojevic, S.; Arsenovic, M.; Anderla, A.; Culibrk, D.; Stefanovic, D. Deep neural networks based recognition of plant diseases by leaf image classification. Comput. Intell. Neurosci. 2016, 2016, 3289801. [Google Scholar] [CrossRef] [PubMed][Green Version]
169. Lu, Y.; Yi, S.; Zeng, N.; Liu, Y.; Zhang, Y. Identification of rice diseases using deep convolutional neural networks. Neurocomputing 2017, 267, 378–384. [Google Scholar] [CrossRef]
170. .Liu, B.; Zhang, Y.; He, D.; Li, Y. Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry 2017, 10, 11. [Google Scholar] [CrossRef][Green Version]

171. Ferentinos, K.P. Deep learning models for plant disease detection and diagnosis. Comput. Electron. Agric. 2018, 145, 311–318. [Google Scholar] [CrossRef]
172. Too, E.C.; Yujian, L.; Njuki, S.; Yingchun, L. A comparative study of fine-tuning deep learning models for plant disease identification. Comput. Electron. Agric. 2019, 161, 272–279. [Google Scholar] [CrossRef]
173. Waheed, A.; Goyal, M.; Gupta, D.; Khanna, A.; Hassanien, A.E.; Pandey, H.M. An optimized dense convolutional neural network model for disease recognition and classification in corn leaf. Comput. Electron. Agric. 2020, 175, 105456. [Google Scholar] [CrossRef]
174. Ramcharan, A.; McCloskey, P.; Baranowski, K.; Mbilinyi, N.; Mrisho, L.; Ndalahwa, M.; Legg, J.; Hughes, D.P. A mobile-based deep learning model for cassava disease diagnosis. Front. Plant Sci. 2019, 2019, 272. [Google Scholar] [CrossRef][Green Version]
175. Javierto, D.P.P.; Martin, J.D.Z.; Villaverde, J.F. Robusta Coffee Leaf Detection based on YOLOv3- MobileNetv2 model. In Proceedings of the 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Manila, Philippines, 28–30 November 2021; pp. 1–6. [Google Scholar] [CrossRef]
176. Hassan, S.M.; Maji, A.K.; Jasiński, M.; Leonowicz, Z.; Jasińska, E. Identification of plant-leaf diseases using CNN and transfer-learning approach. Electronics 2021, 10, 1388. [Google Scholar] [CrossRef]
177. Vallabhajosyula, S.; Sistla, V.; Kolli, V.K.K. Transfer learning-based deep ensemble neural network for plant leaf disease detection. J. Plant Dis. Prot. 2022, 129, 545–558. [Google Scholar] [CrossRef]
178. Sutaji, D.; Yıldız, O. LEMOXINET: Lite ensemble MobileNetV2 and Xception models to predict plant disease. Ecol. Inform. 2022, 70, 101698. [Google Scholar] [CrossRef]
179. Kaur, N. Plant leaf disease detection using ensemble classification and feature extraction. Turk. J. Comput. Math. Educ. (TURCOMAT) 2021, 12, 2339–2352. [Google Scholar]
180. Chen, J.; Zeb, A.; Nanehkaran, Y.; Zhang, D. Stacking ensemble model of deep learning for plant disease recognition. J. Ambient. Intell. Humaniz. Comput. 2022, 1–14. [Google Scholar] [CrossRef]
181. Biibosunov B, Sabitov B., Biibosunova S, Sheishenov J, Zhusupkeldiev Sh, Mamadalieva Zh.

Machine learning for crop yield forecasting. CYBERNETICS AND PHYSICS / Volume 12, 2023, Number 3, http://lib.physcon.ru/doc?id=8c3f4d244777

182. Sabitov B.R., Kartanova A.D., Talant uulu K., Seitkazieva N.S., Dyikanova A., Orozobekova A.K. MODELING AND FORECASTING TASKS OF AGRICULTURE BASED ON MACHINE LEARNING // E3S Web Conf., 380 (2023) 01026, P.15. DOI: https://doi.org/10.1051/e3sconf/202338001026
https://www.scopus.com/authid/detail.uri?authorId=58261397400