Казахский национальный исследовательский технический
университет имени К.И. Сатпаева
Институт автоматики и информационныхтехнологий

УДК004.421.2 На правах рукописи

Мамбетов Нурбол Адилович

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На соискание академической степени магистра

Название диссертации Исследование и разработка геоинформационной службы (системы) для оказания информационной помощи фермерам

Направление подготовки 7M06101 – Software Engineering

Научный руководитель
Доктор Ph.D, ассоц.-
профессор
_____Мукажанов Н.К
«__» _____ 2023 г.

Рецензент,
Доктор Ph.D, ассистент
профессор
_____Т.Т. Чинибаева
«__» _____ 2023 г.

Нормоконтроль
доктор Ph.D, ассоц.-
профессор
_____ А.Т.Ахмедиярова
«__» _____ 2023 г.

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. физ.-мат. наук, ассоц.-проф
_____ А.Н.Молдагулова
«__»_____ 2023 г.

Алматы 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Казахский национальный исследовательский технический
университет имени К.И. Сатпаева

Институт автоматики и информационных технологий

Кафедра Программная инженерия

Специальность: 7М06101 – SoftwareEngineering

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. физ.-мат. наук, ассоц.-
профессор
_____ А.Н.Молдагулова
«___ » _____ 2023г.

ЗАДАНИЕ
на выполнение магистерской диссертации

магистранту, обучающемуся Мамбетову Нурболу Адиловичу

Тема диссертации: «Исследование и разработка геоинформационной службы (системы) для оказания информационной помощи фермерам»

Срок сдачи законченной диссертации «___»_____

Исходные данные к магистерской диссертации. Дан анализ моделей и методов системы геоинформационных систем. Поставленные цели и задачи диссертационной работы обуславливают комплексность методологии исследования, основанной на системном, процессно-ориентированном подходе к изучению различных аспектов управления.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации: а) определение требований – анализ потребностей, целей, задач и назначения разработки; б) исследование особенностей современных геоинформационных систем; в) обоснование возможности и необходимости использования ГИС подхода в системе сельского хозяйства для повышения эффективности их функционирования; г)

проектирование механизмов построения и использования систем рекомендации и прогнозирования урожайности.

Рекомендуемая основная литература 1. Цветков В.Я. Основы геоинформатики. – М.: Издательство Лань, 2023. 2. Ковин Р.Г, Н.Г.Марков. Методы и подходы создания ГИС приложений. – Томский Политехнический Университет, 2012. 3. Курлович Д.М, Н.В.Жуковская.ГИС технологии. – Минск: БГУ, 2020.

ГРАФИК

подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Раздел 1. Теоретические основы геоинформационных систем для фермерных хозяйств	15.10.2022 г.	Выполнено
Раздел 2. Методы и средства системы рекомендации и прогнозирования для геоинформационных систем	18.01.2023 г.	Выполнено
Раздел 3. Построение геоинформационной службы рекомендации и прогнозирования	25.05.2023 г.	Выполнено

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант, (уч. степень, звание)	Сроки	Подпись
Нормоконтроль	А.Т.Ахмедиярова, доктор Ph.D, ассоц. профессор кафедры Програмной инженерии		
Антиплагиат	Б. Аубакир, ассистент кафедры Програмной инженерии		
Проверка ПО	Н.К. Мукажанов, ассоц.-профессор кафедры		

	Програмной инженерии		

Дата выдачи задания "____"_____ 2023 г.

Заведующий кафедрой _____ А.Н.Молдагулова

Научный руководитель _____ Н.К.Мукажанов

Задание принял к исполнению магистрант _____ Н.А.Мамбетов

Дата "_____"_____ 2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сельское хозяйство, с момента его изобретения и зарождения, было основным видом деятельности каждой культуры и цивилизации на протяжении всей истории человечества. Это не только огромный аспект растущей экономики, но и крайне важно для нашего выживания. Это также важнейший сектор для экономики нашей страны и будущего человечества. Это также способствует увеличению доли занятых. Поскольку время идет, потребность в производстве возрастает в геометрической прогрессии.

Чтобы производить продукцию в массовом количестве, люди используют технологию крайне неправильным образом. Новые виды гибридные сорта выводятся изо дня в день. Однако эти сорта не обеспечивают необходимого содержания в качестве урожая, полученного естественным путем. Эти неестественные методы портят почву. Все это приводит к дальнейшему ущербу окружающей среде. Большинство из этих неестественных методов обычно используются для того, чтобы избежать потерь.

Машинное обучение играет все более важную роль в сельском хозяйстве, предоставляя новые возможности для повышения эффективности и оптимизации процессов. Вот некоторые области, где машинное обучение применяется в сельском хозяйстве:

1. Предсказание урожайности: Машинное обучение может использоваться для анализа различных данных, таких как погодные условия, почвенные характеристики, сорта культур и исторические данные урожайности, с целью предсказания урожайности. Это позволяет сельским хозяйственным предприятиям принимать более информированные решения о планировании посевов, управлении ресурсами и прогнозировании производства.
2. Оптимизация использования ресурсов: Машинное обучение может помочь оптимизировать использование ресурсов в сельском хозяйстве, таких как вода, удобрения и пестициды. Алгоритмы машинного обучения могут анализировать данные о почве, погоде и растениях, чтобы определить оптимальные дозировки удобрений и пестицидов, а также оптимальные методы полива.
3. Защита растений: Машинное обучение может быть использовано для определения и прогнозирования заболеваний и вредителей растений. Алгоритмы машинного обучения могут обрабатывать данные о симптомах, погодных условиях и географическом распространении, чтобы предсказать и предотвратить возникновение болезней и насекомых, что позволяет сельским хозяйственным предприятиям принимать своевременные меры по защите растений.
4. Автоматическое сортировка и классификация продукции: Машинное обучение может быть применено для автоматической сортировки и классификации сельскохозяйственной продукции на основе различных критериев, таких как размер, цвет, качество и зрелость. Это позволяет ускорить и автоматизировать процесс обработки и упаковки продукции, а также повысить точность и единообразие классификации.

5. Прогнозирование спроса на продукцию: Машинное обучение может использоваться для анализа данных о рынке, тенденциях потребления и социальных медиа, чтобы предсказывать спрос на сельскохозяйственную продукцию. Это позволяет сельскохозяйственным предприятиям планировать производство и адаптировать ассортимент продукции в соответствии с рыночными требованиями.

Актуальность обосновывается в том что данная система предоставляет информацию о свойствах почвы, климатических условиях местности для фермеров. Получение полной информации о местности, участков помогает фермерам точно определить и принять решение над каким видом сельским хозяйством заниматься.

Целью исследования является исследование и разработка геоинформационной системы для предоставления информации по характеристикам выбранного участка.

Задачи исследования:
-Анализ предметной области
-Исследование разработки геоинформационной системы для представления информации рекомендательного характера
-Исследование методов машинного обучения для применения сельского хозяйство (для предсказания урожайности)
-Разработка системы
-Тестирование системы и получение результатов

*Предмет исследования:*геоинформационные системы, рекомендательные системы, машинное обучение.
*Методы исследования:*методы классификации данных, ассоциативные правил, модели машинного обучения и др.
*Объект исследования:*данные по свойствам почв, ГИС библиотеки с открытым исходным кодам, формирование предсказании по урожайности.
Апробация результатов исследования. Основные положения и результаты диссертационного исследования были успешно доложены и обсуждены на международной научно-практической конференции «Сатпаевские чтения – 2022. Тренды современных научных исследований» (г.Алматы, 2022г) ; международная научно-практическая конференция «I Куандыковские чтения» (г.Алматы, 2023 г).

1 Анализ предметной области

Анализ предметной области сельского хозяйства и машинного обучения включает в себя ряд ключевых аспектов:
1. Данные урожайности: Для прогнозирования урожайности необходимо иметь доступ к историческим данным, которые описывают урожайность различных культур в разных условиях. Эти данные могут включать информацию о почве, погоде, использовании удобрений и т. д. Собранные данные могут быть использованы для обучения моделей машинного обучения и разработки прогностических моделей.
2. Сенсорные данные и датчики: Современные технологии в сельском хозяйстве позволяют собирать различные данные с помощью датчиков, таких как данные о влажности почвы, температуре, освещении и т. д. Эти данные могут быть использованы для мониторинга условий роста растений и определения оптимальных параметров для достижения максимальной урожайности.
3. Модели машинного обучения: Применение алгоритмов машинного обучения, таких как регрессия, случайные леса, нейронные сети и др., может помочь в построении прогностических моделей для предсказания урожайности на основе исторических данных и других факторов. Эти модели могут учесть влияние различных переменных, таких как погода, тип почвы, использование удобрений и др., для прогнозирования будущей урожайности с высокой точностью.
4. Оптимизация решений: Машинное обучение также может применяться для оптимизации решений в сельском хозяйстве. Например, модели машинного обучения могут помочь определить оптимальные моменты для посева, полива, удобрения и сбора урожая, основываясь на анализе данных о погоде и состоянии почвы. Это может помочь фермерам оптимизировать использование ресурсов и повысить эффективность своей работы.
5. Предсказание болезней и вредителей: Машинное обучение также может применяться для предсказания и контроля болезней растений и вредителей. Анализ данных о симптомах болезней и вредителях, погодных условиях и других факторах может помочь в создании моделей, способных предсказывать риск возникновения и распространения болезней и вредителей, что позволит фермерам принимать своевременные меры для их предотвращения и контроля.

1.1Основные понятия

Ниже приведены основные понятия которые используются в исследовательской работе.
ГИС (геоинформационная система) — это компьютерная система, предназначенная для сбора, хранения, анализа, управления и отображения

географической информации. Она объединяет географические данные (данные, связанные с местоположением на Земле) с информационными технологиями, чтобы помочь пользователям исследовать и принимать решения, связанные с пространственными аспектами предметной области[2].
ИС - Информационная система (ИС) - это система, которая собирает, обрабатывает, хранит и передает информацию для поддержки различных видов деятельности в организации или внутри определенного контекста. Она объединяет людей, данные, процессы и технологии в единое целое, чтобы обеспечить эффективное функционирование организации и принятие обоснованных решений.
Машинное обучение, МО (Machine Learning) - это подраздел искусственного интеллекта, который изучает алгоритмы и статистические модели, которые позволяют компьютерным системам обучаться и делать предсказания или принимать решения на основе данных, без явного программирования.
Набор данных (dataset) представляет собой собранный и структурированный набор информации, который используется для обучения моделей машинного обучения или проведения анализа данных. Набор данных состоит из наблюдений, которые могут быть представлены в виде таблицы или матрицы, где каждая строка представляет отдельное наблюдение, а каждый столбец - признак или переменную, описывающую наблюдение.
Веб-приложение (web application) - это программное обеспечение, которое разрабатывается для использования через веб-браузер. Оно доступно для пользователей через интернет и не требует установки на компьютер или мобильное устройство[1].

1.2 Методы исследования

Геоинформационные системы (ГИС) и машинное обучение (МО) тесно связаны друг с другом и предоставляют возможности для проведения различных исследований. Ниже приведены несколько методов исследования, которые объединяют ГИС и МО:
1. Классификация исходя из геопространственных данных: ГИС и МО используются для классификации геопространственных данных, таких как спутниковые изображения или лазерные сканирования, с целью выделения и идентификации различных объектов и покрытий земли. Методы машинного обучения, такие как классификация на основе случайного леса, метод опорных векторов или глубокие нейронные сети, могут быть применены для этой цели.
2. Прогнозирование и моделирование: ГИС и МО используются для прогнозирования и моделирования различных географических явлений. Например, можно использовать методы машинного обучения для разработки моделей прогнозирования погоды или распространения загрязнений воздуха на основе геопространственных данных[4].

3. Кластеризация и ассоциативные анализы: Машинное обучение применяется для кластеризации геопространственных данных, чтобы выделить схожие области или группы объектов.
4. Рекомендательные системы: ГИС и МО используются для создания рекомендательных систем, которые предлагают персонализированные рекомендации, основанные на географическом контексте.
Современные разработки в области геоинформационных систем и машинного обучения открывают широкие перспективы для проведения исследований, связанных с пространственными данными[2,3].

Объекты исследования. В качестве объектов исследования выступают такие средства как набор данных (dataset), open source библиотека leaflet, алгоритмы обучения для классификации данных ML и язык программирование python, фреймворк flask.
Обзор исследований почвы с использованием машинного обучения и рекомендательных систем включает в себя следующие аспекты:
1. Применение машинного обучения для классификации почв: исследовательская работа будет фокусироваться на использовании методов машинного обучения, таких как алгоритмы классификации (например, SVM, Random Forest, нейронные сети и др.), для автоматической классификации типов почв на основе различных характеристик и признаков.
2. Предсказание свойств почвы: Исследования будет направляться на разработку моделей машинного обучения для предсказания различных свойств почвы, таких как содержание органического вещества, pH-значение, содержание питательных веществ и другие. Модели машинного обучения будет использоваться для анализа данных, собранных из различных источников.
3. Оптимизация применения удобрений: Рекомендательная система будет применена для оптимизации применения удобрений в сельском хозяйстве, учитывая различные факторы, такие как тип почвы, климатические условия, культуры и требования растений. Это позволяет сельскохозяйственным предприятиям и фермерам эффективно использовать ресурсы и снизить отрицательное воздействие на окружающую среду.
4. Интеграция геопространственных данных: Обзор исследований также включает анализ интеграции геопространственных данных с данными о почве и использование методов машинного обучения для анализа пространственных паттернов и связей между почвенными характеристиками и географическими факторами, такими как рельеф, водные ресурсы, ландшафтные типы и др.
5. Разработка инструментов и платформ: Система охватывает разработку инструментов и платформ, основанных на машинном обучении и рекомендательных системах, которые позволяют исследователям, агрономам и сельскохозяйственным предприятиям эффективно анализировать и интерпретировать данные почвы и получать рекомендации для оптимального использования почвенных ресурсов.

1.2Сферы применения геоинформационных систем

Геоинформационные системы (ГИС) имеют широкий спектр применения в различных областях. Вот некоторые из них:

1. Географическое планирование и управление территориями: ГИС используются для анализа и планирования использования земель, разработки генеральных планов городов, определения оптимальных местоположений объектов инфраструктуры (дороги, железные дороги, водопроводы и т.д.) и принятия решений по управлению территорией.
2. Картография и навигация: ГИС используются для создания карт различных масштабов и типов, включая топографические карты, тематические карты (например, карты погоды, экологические карты) и цифровые карты для навигационных систем. ГИС также позволяют выполнять геокодирование (преобразование адресов в координаты) и маршрутизацию (построение оптимальных маршрутов).
3. Управление ресурсами и окружающей средой: ГИС используются для управления природными ресурсами, такими как вода, леса, земельные участки и минеральные ресурсы. Они помогают оптимизировать использование и охрану ресурсов, предсказывать изменения окружающей среды и анализировать экологические воздействия различных действий.
4. Сельское хозяйство и лесное хозяйство: ГИС используются в сельском хозяйстве и лесном хозяйстве для планирования посевов, управления урожайностью, контроля состояния почвы, определения оптимальных мест для выращивания определенных культур и управления лесными ресурсами.
5. Геология и геофизика: ГИС применяются в геологических и геофизических исследованиях для анализа и визуализации геологической информации, планирования разработки месторождений полезных ископаемых, определения геологических рисков и моделирования геологических процессов[9,10].
6. Городское планирование и развитие: ГИС используются для анализа городской инфраструктуры, прогнозирования роста городов, разработки мастер-планов и мониторинга развития городских территорий.
7. Геодезия и кадастр: ГИС применяются для создания и управления кадастровыми данными, а также для выполнения геодезических измерений, маркировки границ и определения координат точек.
8. Транспортное планирование и логистика: ГИС используются для анализа и оптимизации транспортных сетей, прогнозирования потоков движения, планирования маршрутов и определения оптимального размещения объектов логистики.

Это только некоторые из множества сфер применения геоинформационных систем. Благодаря своей способности анализировать, визуализировать и прогнозировать географические данные, ГИС играют важную роль в принятии решений и оптимизации процессов в различных отраслях.

Прогноз урожайности сельскохозяйственных культур с использованием машинного обучения возможен и может быть достигнут через различные подходы и модели. Однако для создания точного прогноза требуется доступ к большому объему данных о растениях, условиях выращивания, климатических факторах и других соответствующих параметрах. Важно отметить, что точность прогноза урожайности зависит от доступности и качества входных данных, а также от выбранной модели и ее настройки. При разработке моделей прогнозирования урожайности необходимо учесть различные факторы, включая климатические условия, внештатные события (например, засухи или наводнения), тип почвы, сорта растений и применяемые методы возделывания[13].

Таким образом, прогноз урожайности с использованием машинного обучения является перспективным направлением, но требует комплексного подхода, адекватных данных и согласования с экспертами сельскохозяйственной отрасли для достижения наилучших результатов.

1.3Классификация геоинформационных систем для сельского хозяйства

Классификация геоинформационных систем (ГИС) в сельскохозяйственной сфере представляет большой интерес. Ниже приведены основные виды классификации ГИС в сельском хозяйстве:

1. ГИС для управления земельными ресурсами: рассматриваются ГИС, применяемые для управления земельными ресурсами в сельском хозяйстве. ИС включать в себя ГИС для планирования посевных площадей, учета использования земли, мониторинга почвенного плодородия и определения оптимальных мест для выращивания определенных культур.
2. ГИС для мониторинга и управления урожайными площадями: рассматриваются ГИС, используемые для мониторинга и управления урожайными площадями. Включает в себя ГИС для определения оптимального полива, контроля за заболеваниями растений, прогнозирования урожайности, оценки рисков и принятия решений на основе данных о почве, климате и других факторах.

Объектом исследования геоинформационных систем (ГИС) являются пространственные данные и их анализ. ГИС представляет собой инструмент, который использует компьютерную технологию для сбора, хранения, анализа и визуализации географической информации. ГИС позволяет работать с различными типами данных, такими как карты, снимки со спутников, аэрофотоснимки, статистические данные и другие пространственные наборы данных.Объектом исследования ГИС может быть широкий спектр задач, связанных с пространственным анализом и управлением географической информацией. Некоторые из них включают:

1. Анализ местоположения: Исследования, направленные на определение наиболее подходящего местоположения для конкретных целей, таких как выбор места для нового предприятия или определение оптимального расположения точек обслуживания.
2. Управление ресурсами: Анализ и оптимизация использования природных ресурсов, таких как земля, вода и леса. ГИС может помочь в планировании использования земли, оценке потенциала для различных видов сельского хозяйства, определении уровня устойчивости и принятии решений по управлению ресурсами.
3. Картография и визуализация: Создание карт и визуализация географической информации. ГИС позволяет составлять и отображать картографические продукты с высокой степенью детализации и точности.
4. Пространственный анализ: Исследование пространственных паттернов, взаимосвязей и воздействия на основе географической информации. Это может включать анализ транспортных сетей, распределения населения, изменений в использовании земли и др.
5. Решение проблем аварийных ситуаций и планирование мер чрезвычайного положения: ГИС используется для управления и анализа информации в случае природных и технологических катастроф, планирования эвакуации, определения уязвимых зон и т.д.
Исследование в области ГИС включает разработку методологий, алгоритмов и программного обеспечения для обработки и анализа географической информации. Основная цель заключается в понимании пространственных паттернов, связей и явлений для принятия информированных решений в различных сферах деятельности.

1.4Аналогичные системы

ArcGIS, QGIS и EOS - это географические информационные системы (ГИС), которые предоставляют возможности для работы с географическими данными и пространственным анализом. Они предоставляют функциональность, которая позволяет пользователям создавать, редактировать, анализировать и визуализировать географические данные.
ArcGIS: ArcGIS (Geographic Information System) - это комплексное программное обеспечение, разработанное компанией Esri, предназначенное для работы с географической информацией и решения задач пространственного анализа.
ArcGIS предоставляет пользователю мощные инструменты для создания, редактирования, анализа и визуализации географических данных. Оно включает в себя набор инструментов и функций для работы с различными типами геоданных, такими как векторные данные (точки, линии, полигоны), растровые данные (изображения), географические базы данных и другие.

С помощью ArcGIS пользователь может создавать карты, выполнять анализ пространственных данных, моделировать и прогнозировать различные явления, проводить геостатистический анализ, оптимизировать маршруты и многое другое. Он также обеспечивает возможность интеграции с другими системами и базами данных, что делает его мощным инструментом для работы с географической информацией в различных отраслях, таких как география, геология, геодезия, экология, транспорт, градостроительство и другие.

ArcGIS предоставляет графический интерфейс пользователя, что делает его доступным даже для тех, кто не имеет глубоких знаний программирования. Однако, для более сложных задач и автоматизации процессов, ArcGIS также поддерживает использование скриптования на языке Python, что позволяет пользователю создавать собственные сценарии и автоматизировать задачи.

ArcGIS широко используется в различных областях, включая географическое исследование, управление ресурсами, геопространственное планирование, картографию, анализ экологических систем, геологию, градостроительство, транспортное планирование и др. Оно является одним из наиболее распространенных и мощных инструментов ГИС в мире.

QGIS: QGIS (Quantum GIS) - QGIS (Quantum GIS) - это бесплатная и открытая географическая информационная система (ГИС), разработанная сообществом разработчиков. QGIS предоставляет возможности для создания, редактирования, анализа и визуализации географических данных.

Основные особенности QGIS включают:

1. Работа с различными типами географических данных: QGIS поддерживает широкий спектр форматов данных, включая векторные данные (точки, линии, полигоны), растровые данные (изображения), базы данных и другие. Это позволяет пользователям импортировать, экспортировать и работать с различными источниками географических данных.
2. Визуализация и стилизация: QGIS предоставляет мощные инструменты для создания карт и визуализации географических данных. Пользователи могут настраивать стили, цвета, символы и многое другое для отображения данных в удобном и информативном виде.
3. Пространственный анализ: QGIS обеспечивает возможности для проведения пространственного анализа данных, включая операции пересечения, объединения, буферизации, измерения расстояний и другие. Это позволяет пользователям извлекать информацию из данных и проводить различные аналитические задачи.
4. Поддержка плагинов: QGIS имеет расширяемую архитектуру, которая позволяет добавлять функциональность с помощью плагинов. Существует широкий выбор плагинов, разработанных сообществом пользователей, которые добавляют новые инструменты и возможности к стандартной установке QGIS.
5. Мультиплатформенность: QGIS доступен для различных операционных систем, включая Windows, macOS и Linux, что делает его доступным для широкой аудитории пользователей.

QGIS активно развивается сообществом разработчиков и пользователей, и регулярно выпускает обновления и новые версии с улучшениями и новыми функциями. Он является популярным инструментом для работы с географической информацией в различных областях, включая научные исследования, географическое планирование, управление природными ресурсами и многие другие..

EOS - онлайн агро платформа, помогает фермерам отслеживать состояние посевов по данным с последних спутниковых снимков и хранить всю информацию в одном месте, тем самым значительно экономить время и деньги.

Все эти системы предлагают возможности для работы с географическими данными, однако они имеют разные функциональности, лицензии и специфические особенности. Выбор системы будет зависеть от потребностей, доступности данных и предпочтений в использовании программного обеспечения[14].

Определение задачи исследования

Целью нашего исследования является исследование и разработка геоинформационной системы для предоставления информации по характеристикам выбранного участка.

В данный момент в стране не разработана предсказательная система по аграрному сектору. В связи с этой мы решили спроектировать и разработать адаптированную под условия нашей страны гео информационную систему который будет прогнозировать, предсказывать урожайности агрокультур, предупреждать о болезни растений. В рамках исследований будем использовать методы классификации, анализа МО.

Задачи исследования:

-Анализ предметной области

-Исследование разработки геоинформационной системы для представления информации рекомендательного характера

-Исследование методов машинного обучения для применения сельского хозяйство (для предсказания урожайности)

-Разработка системы

-Тестирование системы и получение результатов

Ожидаемые результаты исследования:

1. Разработка модели или алгоритма: результатом исследования будет разработка новой модели или алгоритма, который позволяет эффективно анализировать гео данные с использованием методов машинного обучения, таких как: Decision Tree, Random Forest, SVM, XGBoost.

2. Улучшение точности и эффективности: ожидаемым результатом исследования является улучшение точности и эффективности анализа геопространственных данных с помощью методов машинного обучения.

3. Предсказания и прогнозы: результат исследования должен позволить предсказывать урожайность культуры для выбранных участков почвы, определять вид болезни или предотвратит их.

4. Рекомендации и принятие решений: создание рекомендаций и поддержка принятия решений на основе анализа геопространственных данных.

2 Разработка ГИС

2.1 Архитектура системы

Онлайн агро платформа - это веб-сервис или приложение, предназначенное для поддержки и оптимизации различных аспектов сельского хозяйства и связанных с ним операций. Такие платформы обычно предлагают набор инструментов, функций и ресурсов, которые помогают фермерам и другим участникам аграрной отрасли в их ежедневной работе.Онлайн агро платформы обладают потенциалом для повышения эффективности, улучшения принятия решений и оптимизации сельскохозяйственных операций. Они позволяют фермерам получать доступ к новым технологиям, данным и знаниям, что может способствовать повышению урожайности, снижению затрат и улучшению устойчивости аграрных систем.

Архитектура информационной системы (ИС) - это описание структуры, компонентов, взаимодействия и организации информационной системы. Она определяет, как компоненты ИС взаимодействуют друг с другом и с внешними системами, а также как они обрабатывают, хранят и передают данные.

В зависимости от конкретного контекста и требований, архитектура информационной системы может использовать различные архитектурные стили, такие как клиент-сервер, микросервисы, SOA (архитектура, основанная на службах) и др. Эти стили определяют основные принципы организации и взаимодействия компонентов системы. В нашей работе была выбрана клиент серверная архитектура, так как такая система легко расширяется, модифицируется на основе веб компонентов.

Клиент-серверная архитектура (Client-Server Architecture) - это распределенная архитектура, в которой функциональность системы разделена на две основные части: клиентскую и серверную.

Клиентская часть представляет собой интерфейс пользователя или приложение, которое отправляет запросы на сервер и получает ответы от него[14].

Серверная часть представляет собой централизованную систему, которая обрабатывает запросы от клиентов и предоставляет им запрошенные данные или услуги.

Преимущества клиент-серверной архитектуры:

1. Масштабируемость: Клиент-серверная архитектура обеспечивает возможность горизонтального и вертикального масштабирования. Сервер может быть настроен для обработки большого количества запросов от клиентов путем добавления вычислительных ресурсов или использования кластеров серверов.
2. Централизованное управление: Центральный сервер управляет ресурсами и данными, что облегчает управление и обновление системы. Это позволяет более эффективно контролировать доступ, обновлять программное обеспечение и вносить изменения без необходимости вносить изменения на каждом клиентском устройстве.

3. Улучшенная безопасность: Клиент-серверная архитектура обеспечивает более прочную защиту данных и контроль доступа к ресурсам. Сервер может предоставлять авторизацию и аутентификацию пользователей, шифрование данных и механизмы контроля доступа, чтобы обеспечить безопасность системы и предотвратить несанкционированный доступ.
4. Распределенная обработка: Клиентские устройства могут выполнять легкие задачи, в то время как более сложные вычисления и обработку данных можно делегировать серверу. Это позволяет эффективно использовать вычислительные ресурсы, сокращает нагрузку на клиентские устройства и повышает производительность системы.
5. Гибкость и модульность: Клиент-серверная архитектура позволяет легко добавлять, изменять или модифицировать клиентские и серверные компоненты независимо друг от друга. Это позволяет гибко настраивать и обновлять систему, а также упрощает разделение задач и ответственностей между клиентской и серверной сторонами.
6. Доступность и надежность: Клиент-серверная архитектура обеспечивает высокую доступность системы. Если один сервер недоступен или выходит из строя, клиенты могут переключиться на другой доступный сервер.
Клиент-серверная архитектура широко применяется во многих областях, включая сетевые приложения, веб-сайты, облачные сервисы и мобильные приложения.

На рисунке 2.1.1 приведена архитектура системы.

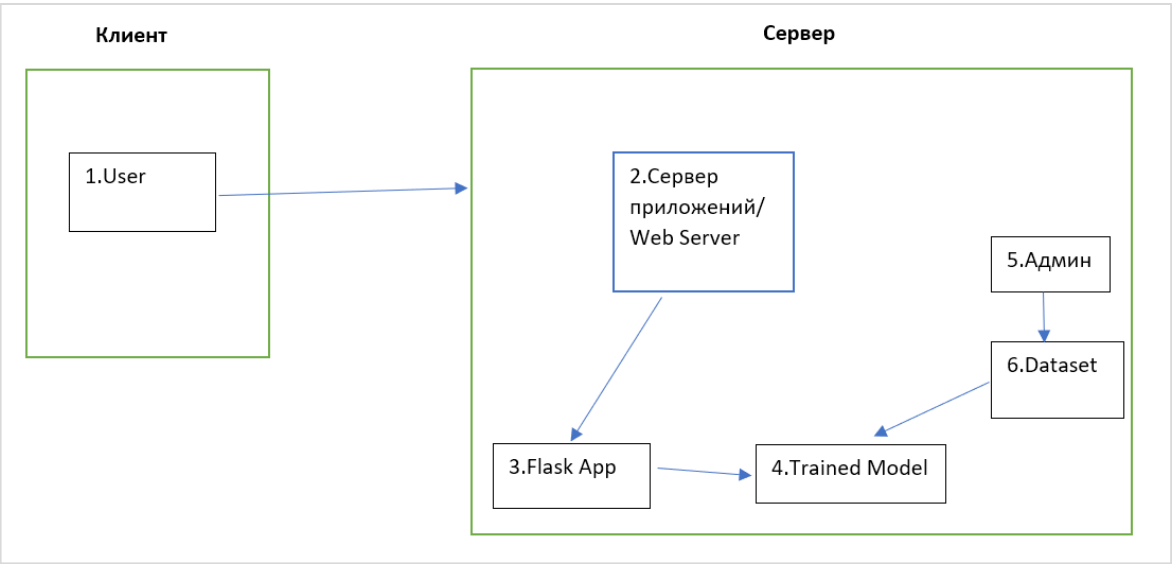

Рисунок 2.1.1. Компоненты клиент серверной архитектуры

1. User - документы в виде html страниц, предоставляемый доступ к функциональности системы через веб браузер, используя элементы HTML разметки (кнопки, поля ввода, ссылки, таблицы, и т.д)

2. Сервер приложений, веб сервер - получает запросы от клиента, обрабатывает с помощью веб приложения и отправляет ответ обратно к клиенту.
3. Веб приложение будет разработан на фреймворке Flask, используемый язык программирования python.
4. Trained model - обученный модель, который по параметрам почвы прогнозирует урожайность, определяет болезни по снимкам растений.
5. Админ панель - функциональность администратора системы, можно загрузить или обновлять наборы данных, обучить снова, менять настройки системы.
6. Dataset - наборы данных, хранятся в виде файлов csv, json, xls.

2.2 Функционал системы

Рекомендация по урожайности культур. На основе указанных параметров почвы, окружающей среды система будет рекомендовать самые подходящие культуры для посева к выбранному участку.В качестве параметров будем указывать количество азота, фосфора, калия, уровень ph (рисунок 2.2.1). А значения окружающей среды такие как температура, влажность, атмосферное давление будем определять с открытого сервиса данных openweathermap.org.

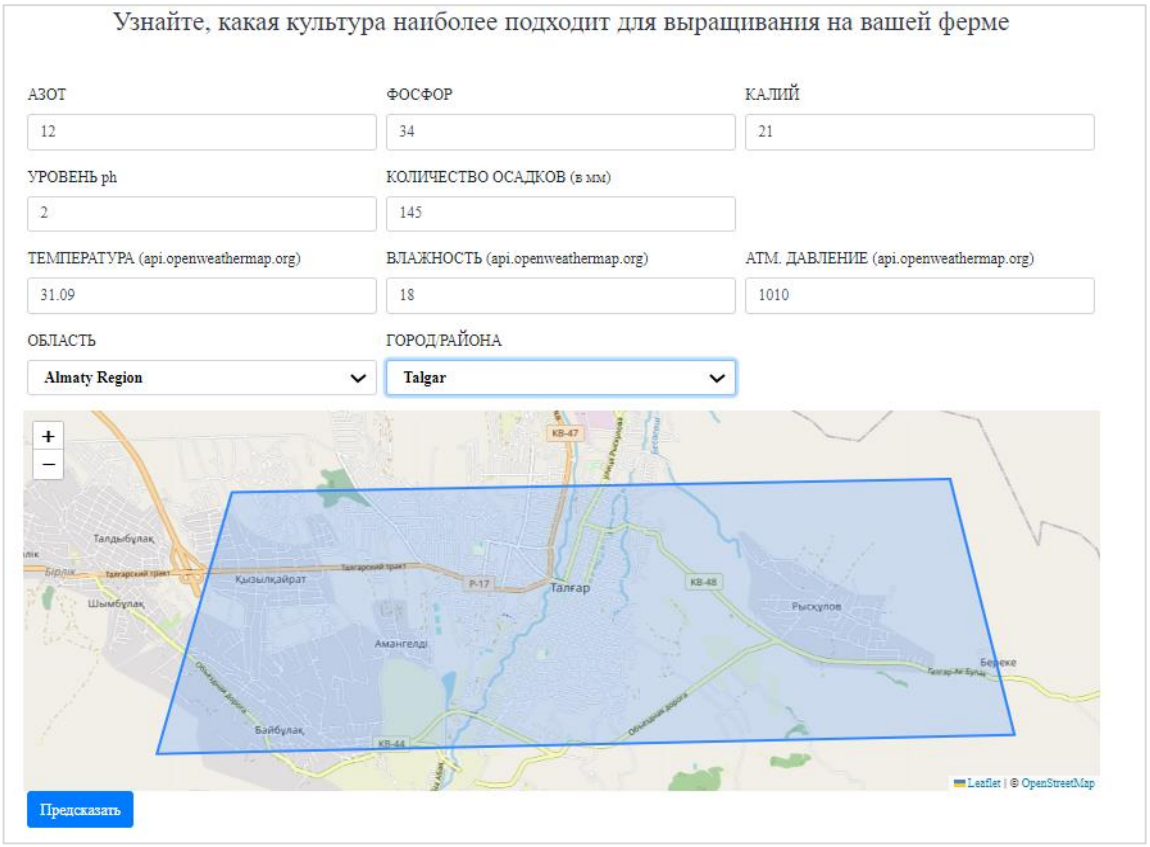

Рисунок 2.2.1. Рекомендация по сельским культурам

Рекомендация по выбору удобрения. Сервис помогает определить самые оптимальные значения объем удобрении азота, фосфора и калия для выбранной культуры. Если например пользователь указал в избытке азота, то система будет рекомендовать уменьшить содержание азота и с каким способом можно это делать.

*Определение болезни растений и рекомендации по способу лечения.*Система с помощью машинного обучения обучается различать здоровые листья с зараженными. И когда пользователь будет загружать снимки листья, система будет определять болезни и как можно вылечить их.

2.3 Применения МО для предсказания

Машинное обучение (ML) имеет дело с задачами, когда связь между входными и выходными переменными неизвестна или ее трудно получить. Термин “обучение” здесь означает автоматическое получение структурных описаний из примеров того, что описывается. В отличие от традиционных статистических методов, ML не делает предположений о правильной структуре модели данных, которая описывает данные. Эта характеристика очень полезна для моделирования сложного нелинейного поведения, такого как функция прогнозирования урожайности сельскохозяйственных культур. Методы ML наиболее успешно применяется для прогнозирования урожайности сельскохозяйственных культур. Контролируемый алгоритм обучения состоит из целевой переменной результата target (или зависимая переменная), которая должна быть предсказана на основе заданного набора предикторов (независимых переменных). Используя этот набор переменных, мы генерируем функцию, которая сопоставляет входные данные с желаемыми выходными данными. Процесс обучения продолжается до тех пор, пока модель не достигнет желаемого уровня точности обучающих данных.

Машинное обучение - это подраздел искусственного интеллекта, который занимается разработкой и применением алгоритмов и моделей, позволяющих компьютерам "обучаться" на основе данных и опыта, а затем использовать полученные знания для принятия решений, делания прогнозов или выполнения задач без явного программирования.

Вместо того чтобы напрямую программировать компьютер на выполнение конкретной задачи, в машинном обучении мы предоставляем компьютеру данные и позволяем ему самостоятельно "научиться" и выявить закономерности и общие законы в этих данных. Это позволяет машине делать предсказания или принимать решения на основе новых данных, которые она ранее не видела.

Машинное обучение включает в себя несколько основных подходов и типов моделей:
1. Обучение с учителем (Supervised Learning): В этом подходе модель обучается на размеченных данных, где каждый пример данных имеет соответствующую метку или выходные значения. Модель стремится найти зависимость между входными данными и соответствующими выходными значениями, чтобы в дальнейшем предсказывать выходные значения для новых наблюдений. Примеры алгоритмов обучения с учителем включают линейную регрессию, логистическую регрессию, метод опорных векторов (SVM), случайные леса и нейронные сети.
2. Обучение без учителя (Unsupervised Learning): В этом подходе модель обучается на неразмеченных данных, где нет заранее определенных выходных значений. Целью модели является выявление скрытых паттернов, группировка данных или понижение размерности. К примеру, метод кластеризации позволяет группировать данные на основе их сходства, а методы снижения размерности, такие как метод главных компонент (PCA), позволяют уменьшить размерность данных, сохраняя при этом наиболее важные аспекты.
3. Обучение с подкреплением (Reinforcement Learning): В этом подходе модель обучается через взаимодействие с окружающей средой. Модель принимает решения и выполняет действия, а затем получает награду или штраф в зависимости от результата этих действий. Цель модели состоит в максимизации полученной награды. Обучение с подкреплением широко применяется в области робототехники, игрового программирования и оптимального управления.
Машинное обучение имеет широкий спектр применений, включая обработку естественного языка, компьютерное зрение, рекомендательные системы, анализ данных, медицинскую диагностику, финансовый анализ, прогнозирование и многое другое. Оно является важным инструментом для извлечения ценной информации и автоматизации сложных задач на основе данных.
Что такое искусственная нейронная сеть?Слои глубокого обучения представляют собой узлы искусственной нейронной сети (ИНС), которые работают как нейроны человеческого мозга. Узлы могут представлять собой комбинацию аппаратного и программного обеспечения. Каждый уровень в алгоритме глубокого обучения состоит из узлов ИНС. Каждый узел или искусственный нейрон соединяется с другим и имеет связанный с ним номер значения и пороговый номер. Узел отправляет номер своего значения в качестве входных данных узлу следующего слоя при активации. Он активируется, только если его выход превышает указанное пороговое значение. В противном случае никакие данные не передаются.Компьютерное зрение – это реальное применение глубокого обучения. Точно так же, как искусственный интеллект позволяет компьютерам думать, компьютерное зрение позволяет им видеть, наблюдать и реагировать.

Глубокое обучение – это часть машинного обучения. Алгоритмы глубокого обучения можно рассматривать как непростую и математически сложную эволюцию алгоритмов машинного обучения.
Машинное обучение и искусственный. Термины «машинное обучение» и «искусственный интеллект» (ИИ) могут использоваться взаимозаменяемо, это не одно и то же. Искусственный интеллект – это общий термин для различных стратегий и методов, используемых для того, чтобы сделать машины более похожими на людей. ИИ включает в себя все, от умных помощников, таких как Alexa, до роботов-пылесосов и беспилотных автомобилей. Машинное обучение – одна из многих других ветвей искусственного интеллекта. Хотя машинное обучение – это ИИ, все действия ИИ нельзя назвать машинным обучением.
Машинное обучение и наука о данных. Машинное обучение и наука о данных – это не одно и то же. Наука о данных – это область исследования, в которой используется научный подход для извлечения смысла и понимания из данных. Специалисты по работе с данными используют ряд инструментов для анализа данных, и машинное обучение является одним из таких инструментов. Специалисты по работе с данными понимают общую картину данных, таких как бизнес-модель, предметная область и сбор данных, в то время как машинное обучение — это вычислительный процесс, который работает только с необработанными данными.
Преимущества и недостатки машинного обучения
Преимущества моделей машинного обучения:

1. определение тенденций и закономерностей данных, которые может упустить человек;
2. работа без вмешательства человека после настройки; например, машинное обучение в ПО для кибербезопасности может постоянно отслеживать и выявлять нарушения в сетевом трафике без участия администратора;
3. результаты могут стать более точными с течением времени;
4. обработка различных форматов данных в динамических, больших объемов и сложных сред данных.

Недостатки моделей машинного обучения:

1. исходная подготовка является дорогостоящим и трудоемким процессом; трудоемкая реализация в отсутствии достаточного объема данных;
2. ресурсоемкий процесс, требующий больших первоначальных инвестиций, если оборудование устанавливается собственными силами;
3. без помощи специалиста может быть сложно правильно интерпретировать результаты и устранить неопределенность.

Пример разработки модели предсказания урожайности с помощью машинного обучения на языке Python может включает следующие шаги:

1. Подготовка данных: загружаем и предобработаем данные об урожайности и факторах, включая удаление пропущенных значений, нормализацию или стандартизацию данных.
2. Разделение данных: разделяем данные на обучающую и тестовую выборки. Обычно используется разделение в соотношении 70:30 или 80:20, где 70% или 80% данных используются для обучения модели, а остальные данные используются для оценки ее производительности.
3. Выбор модели: выбираем подходящую модель машинного обучения для предсказания урожайности. Некоторые из популярных моделей, которые могут использоваться, включают линейную регрессию, случайный лес, градиентный бустинг и нейронные сети.
4. Обучение модели: обучаем выбранную модель на обучающих данных. Для этого используем методы обучения, предоставляемые библиотеками машинного обучения, такими как scikit-learn или TensorFlow.
5. Оценка модели: оцениваем производительность модели на тестовой выборке. Используем метрики оценки регрессии, такие как средняя абсолютная ошибка (MAE) или коэффициент детерминации (R2), для определения точности предсказаний модели.
6. Тюнинг модели: если необходимо, проведем тюнинг модели, варьируя гиперпараметры и проверяя их влияние на производительность модели. Это может быть сделано с использованием методов перекрестной проверки или оптимизации гиперпараметров.
7. Прогнозирование урожайности: после обучения и оценки модели используем ее для прогнозирования урожайности на новых наборах факторов или входных данных.

2.3.1 Необходимые информации, параметры для определения урожайности

Для определения урожайности с использованием машинного обучения требуется следующая информация и параметры:
1. Данные о урожайности: данные о прошлых урожаях для различных полей или участков. Включают в себя информацию о типе культуры, урожайности в конкретные годы.
2. Агрономические данные: данные о почвенных свойствах, включая плотность почвы, содержание органического вещества, pH и т.д. Также будет полезно учесть данные о климатических условиях, таких как осадки, температура и влажность.
3. Географические данные: данные для каждого участка или поля, включая координаты, высоту над уровнем моря и другие характеристики местоположения.

4. Входные параметры для модели: набор входных параметров, которые будут использоваться для предсказания урожайности. Эти параметры включают данные о почве, климате, географическом положении, агрономических практиках, используемых удобрениях и пестицидах и других факторах, которые могут влиять на урожайность.
5. Модель машинного обучения: выбираем модель машинного обучения, которая будет использоваться для предсказания урожайности. Это может быть регрессионная модель, случайный лес, нейронная сеть или другой алгоритм, который лучше всего подходит для нашего набора данных и целей.
6. Обучающий набор данных: набор данных, включающий исторические данные урожайности и соответствующие входные параметры. Разделяется на тренировочный и тестовый наборы для оценки производительности модели.
7. Обучение и оценка модели: Обучаем модель на обучающем наборе данных и оцениваем ее производительность на тестовом наборе. Это включает процесс подбора оптимальных параметров модели, таких как гиперпараметры, и проведение кросс-валидации для проверки ее обобщающей способности.
После успешного обучения модели мы будем использовать ее для предсказания урожайности на новых участках или полей, используя соответствующие входные параметры. Это поможет нам принять информированные решения относительно агрономических практик, планирования урожаев и оптимизации сельскохозяйственных операций.

2.3.2 Обзор набора данных

Для разработки модели машинного обучения, которая предсказывает урожайность, нам понадобится набор данных, который содержит информацию о урожайности и соответствующие входные параметры. Важно иметь достаточное количество данных для обучения модели. Чем больше данных у нас есть, тем лучше будет производительность модели. Также необходимо обратить внимание на качество данных, проверить их на наличие выбросов, пропущенных значений или ошибок, а также сбалансированность данных для различных классов урожайности.
Набор данных можно собрать самостоятельно путем сбора информации на месте или использовать уже существующие открытые наборы данных, предоставляемые аграрными организациями, исследовательскими учреждениями или государственными агентствами, которые собирают данные о сельском хозяйстве и урожайности. Набор данных (dataset) - это коллекция структурированных или неструктурированных данных, которые связаны между собой и образуют информацию, представленную в определенном формате. Набор данных может включать числовые значения, текст, изображения, аудио или другие типы информации.

Набор данных используется в контексте анализа данных и машинного обучения, где данные используются для обучения моделей, выявления паттернов, прогнозирования и принятия решений. Он может быть представлен в виде таблицы, матрицы, списка или другой структуры данных, в которой каждая строка представляет отдельный элемент данных, а столбцы содержат различные атрибуты или характеристики этих данных.
Набор данных может быть получен из различных источников, таких как опросы, эксперименты, сенсоры, базы данных и интернет. Он может быть предварительно обработан, очищен и преобразован для удаления выбросов, заполнения пропущенных значений или изменения формата данных для удобства использования в конкретных задачах анализа или моделирования.
Важными характеристиками набора данных являются его размер, разнообразие, качество, доступность и актуальность. Выбор правильного набора данных является важным шагом в анализе данных и машинном обучении, поскольку качество и репрезентативность данных существенно влияют на результаты и надежность моделей и выводов, полученных из анализа.
В качестве источника данных было решено использовать датасет с сайта Kaggle. На рисунке 2.3.2.1 приведены основные характеристики нашего набора данных. Далее получаем основные характеристики набора данных (рисунок 2.3.2.2).

```
In [2]: PATH = 'dataset/dataset.csv'
        df = pd.read_csv(PATH)

In [3]: df.head()
Out[3]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Рисунок 2.3.2.1. Набор данных для анализа

Поля данных:
N - коэффициент содержания азота в почве
P - коэффициент содержания фосфора в почве
K - коэффициент содержания калия в почве
temperature - температура в градусах Цельсия
humidity - относительная влажность в %
ph - значение ph почвы
rainfall - количество осадков в мм
label – наименование культуры сельского хозяйства

```
n [7]: df['label'].unique()
ut[7]: array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
              'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
              'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
              'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
             dtype=object)

n [9]: df['label'].value_counts()
ut[9]: rice          100
       maize         100
       jute          100
       cotton        100
       coconut       100
       papaya        100
       orange        100
       apple         100
       muskmelon     100
       watermelon    100
       grapes        100
```

Рисунок 2.3.2.2. Характеристика набора данных

2.3.3 Построение модели МО

Разделяем наш набор данных на тренировочную и тестовую, как показана на рисунке 2.3.3.1.

```
# Разделение на обучающие и тестовые данные

from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.2,random_state =2)
```

Рисунок 2.3.3.1. Разделение на тестовый и тренировочный датасет

2.4Классификатор Decision Tree

Классификатор Decision Tree (дерево решений) является мощным алгоритмом машинного обучения, который применяется для задач классификации и регрессии. Он основан на представлении данных в виде иерархической структуры в виде дерева, где каждый узел представляет признаки данных, каждое ребро представляет возможные значения признаков, а каждый лист дерева представляет класс или значение, прогнозируемое моделью. Принцип работы алгоритма Decision Tree приведена ниже на рисунке 2.4.1.

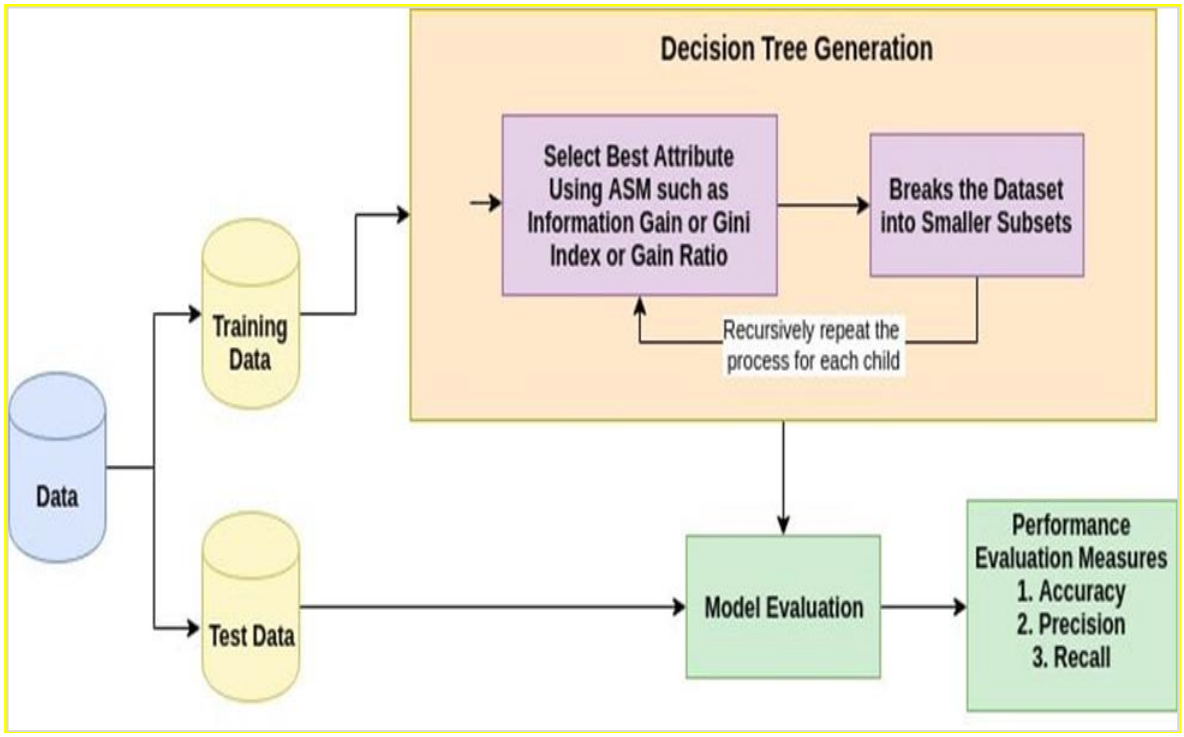

Рисунок 2.4.1. Принцип работы классификатора Decision Tree

Процесс построения дерева решений начинается с корневого узла, который представляет собой весь набор данных. Затем на каждом узле дерева производится разбиение данных на основе определенного признака. Каждое разбиение создает новый узел, и этот процесс повторяется рекурсивно до достижения критериев остановки. В каждом узле дерева решений принимается решение на основе значения одного из признаков. Этот признак выбирается таким образом, чтобы максимизировать разделение классов и минимизировать неоднородность в каждом поддереве. В итоге, каждый лист дерева соответствует определенному классу или метке.

Преимущества деревьев решений включают:

- Простоту интерпретации и понимания результатов.
- Способность обрабатывать как числовые, так и категориальные данные.
- Устойчивость к выбросам и шуму в данных.

Однако, деревья решений также имеют некоторые ограничения:

- Склонность к переобучению при сложных структурах данных.
- Неэффективность в работе с большими объемами данных.
- Чувствительность к небольшим изменениям в данных, что может привести к разным структурам дерева.

На рисунке 2.4.2 приведена результат работы алгоритма DecisionTree.

```
]: from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
```

```
DecisionTrees's Accuracy is:  90.0
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.59      1.00      0.74        16
    chickpea       1.00      1.00      1.00        21
     coconut       0.91      1.00      0.95        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.74      0.93      0.83        28
 kidneybeans       0.00      0.00      0.00        14
      lentil       0.68      1.00      0.81        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       0.00      0.00      0.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
```

Рисунок 2.4.2. Результат выполнения классификатора Decision Tree

2.5 Классификатор Gaussian Naive Bayes

Классификатор Gaussian Naive Bayes (Гауссов наивный Байес) - это вероятностный алгоритм классификации, основанный на принципе наивного Байесовского классификатора и предположении о нормальном распределении (гауссовом распределении) данных.

Наивный Байесовский классификатор основан на принципе Байеса и предполагает условную независимость между признаками, при условии известного значения целевой переменной. В случае Гауссова наивного Байеса предполагается, что каждый признак имеет нормальное (гауссово) распределение.

Процесс обучения Гауссова наивного Байеса заключается в оценке параметров нормального распределения для каждого класса, используя обучающий набор данных. Это включает вычисление среднего значения и дисперсии каждого признака для каждого класса.

При выполнении классификации с помощью Гауссова наивного Байеса используется принцип максимального правдоподобия. Для нового наблюдения вычисляется вероятность принадлежности к каждому классу, используя оцененные параметры распределения и предположение о независимости

признаков. Класс с наибольшей вероятностью становится предсказанием модели.
Преимущества Гауссова наивного Байеса включают:
- Эффективность в обучении и предсказании, особенно на больших наборах данных.
- Хорошая производительность в случаях, когда независимость признаков достаточно близка к реальной модели данных.
- Способность обрабатывать числовые признаки.
Однако, Гауссов наивный Байес имеет некоторые ограничения:
- Предположение о независимости признаков может быть неправильным для некоторых задач, особенно если признаки сильно коррелируют.
- Не учитывает взаимодействия между признаками.
- Необходимость предварительной обработки данных, чтобы они соответствовали предположению о гауссовом распределении.
Гауссов наивный Байес широко используется для классификации текстовых данных и других задач, где предположение о независимости признаков может
Этим классификатором и воспользуемся.Результат работы алгоритма приведена на рисунке 2.5.1.

```
from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain,Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
Naive Bayes's Accuracy is:  0.990909090909091
```

Рисунок 2.5.1. Результат выполнения классификатора Gaussian Naive Bayes

2.6Классификатор Support Vector Machine (SVM)

Support Vector Machine (SVM) - это алгоритм машинного обучения, который используется для задач классификации и регрессии. SVM строит гиперплоскость или несколько гиперплоскостей в многомерном пространстве, которые разделяют данные разных классов.

Основные концепции и принципы работы классификатора Support Vector Machine:

1. Разделяющая гиперплоскость: SVM стремится найти гиперплоскость, которая максимально разделяет данные разных классов. Гиперплоскость - это многомерная аналогия линии или плоскости. В случае двух классов, SVM строит гиперплоскость, которая максимально отделяет точки одного класса от точек другого класса.
2. Максимизация зазора: SVM стремится максимизировать зазор между разделяющей гиперплоскостью и ближайшими точками обучающего набора. Зазор - это расстояние между гиперплоскостью и ближайшими точками разных классов. Максимизация зазора обеспечивает лучшую обобщающую способность и повышает точность классификации новых данных.
3. Функция потерь и опорные векторы: SVM использует функцию потерь, такую как функция Хинджа (Hinge loss), для оценки ошибки классификации. Опорные векторы - это точки обучающего набора, которые находятся на или близко к границе разделяющей гиперплоскости. Они играют важную роль в определении и построении гиперплоскости.
4. Ядерные функции: SVM может использовать ядерные функции, такие как полиномиальные, радиально-базисные функции (RBF) или сигмоидальные функции, чтобы преобразовать данные в более высокое размерное пространство. Это позволяет SVM эффективно работать с нелинейными разделяющими поверхностями.
5. Регуляризация: SVM также включает регуляризацию, чтобы уменьшить переобучение и повысить обобщающую способность модели. Регуляризация помогает контролировать важность опорных векторов и гладкость разделяющей гиперплоскости.

SVM широко применяется в различных областях, таких как классификация текстов, распознавание образов, биомедицинская диагностика, финансовый анализ и другие, благодаря своей способности работать с разделяющими поверхностями и высокой обобщающей способности.Результат работы алгоритма SVM приведена на рисунке 2.6.1.

```
from sklearn.svm import SVC

SVM = SVC(gamma='auto')

SVM.fit(Xtrain,Ytrain)

predicted_values = SVM.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('SVM')
print("SVM's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
SVM's Accuracy is:  0.10681818181818181
              precision    recall  f1-score   support

       apple       1.00      0.23      0.38        13
      banana       1.00      0.24      0.38        17
```

Рисунок 2.6.1. Результат выполнения классификатора SVM

2.7 Классификатор Логистическая регрессия

Логистическая регрессия - это алгоритм машинного обучения, который используется для задач бинарной классификации, то есть разделения данных на два класса. Однако ее также можно расширить для многоклассовой классификации.

Основные концепции и принципы работы классификатора Логистическая регрессия:

1. Логистическая функция: Логистическая регрессия использует логистическую функцию (также называемую сигмоидной функцией) для моделирования вероятности принадлежности экземпляра к одному из классов. Логистическая функция принимает входные значения и возвращает вероятность, ограниченную от 0 до 1.
2. Линейная комбинация: Логистическая регрессия использует линейную комбинацию входных признаков с соответствующими весами, которые оценивают вклад каждого признака в классификацию. Затем эта линейная комбинация передается через логистическую функцию для получения вероятности.
3. Обучение: Процесс обучения логистической регрессии заключается в настройке весов модели таким образом, чтобы минимизировать ошибку классификации. Для этого используется метод оптимизации, например, градиентный спуск, который обновляет веса в направлении, противоположном градиенту функции потерь.

4. Функция потерь: Логистическая регрессия использует логистическую функцию потерь (также известную как бинарная кросс-энтропия) для оценки ошибки классификации. Функция потерь измеряет расхождение между предсказанными вероятностями и фактическими метками классов.
5. Регуляризация: Логистическая регрессия также включает регуляризацию, чтобы снизить переобучение модели и повысить ее обобщающую способность. Регуляризация контролирует сложность модели, добавляя штраф к большим значениям весов.
На рисунке 2.7.1 показана результат работы Логистической регрессии.

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
Logistic Regression's Accuracy is:  0.9522727272727273
```

Рисунок 2.7.1. Результат выполнения Логистической регрессии

2.8 Классификатор Random Forest

Одним из популярных методов машинного обучения, используемых для прогнозирования урожайности, является метод случайного леса (Random Forest). Этот метод строит ансамбль решающих деревьев, которые обучаются на исторических данных урожайности и соответствующих факторах. Затем модель может использоваться для предсказания урожайности на основе новых данных.Random Forest (Случайный лес) - это ансамблевый метод машинного обучения, который комбинирует несколько решающих деревьев для выполнения задач классификации и регрессии. Он получил свое название от того, что каждое дерево строится независимо и случайным образом на основе случайной выборки данных.
Основные концепции и принципы работы классификатора Random Forest:
1. Бутстрэп выборка: При построении каждого дерева Random Forest из обучающего набора данных создается случайная подвыборка с возвращением. Это означает, что некоторые экземпляры данных могут быть выбраны несколько раз, а некоторые - ни разу. Этот процесс называется бутстрэп выборкой.

2. Случайные признаки: При разделении каждого узла дерева случайно выбирается только некоторое подмножество признаков из всего набора признаков. Это помогает увеличить разнообразие деревьев и уменьшить корреляцию между ними.
3. Множественные деревья: Random Forest состоит из множества деревьев, каждое из которых обучается независимо. Каждое дерево делает прогноз, и финальное решение определяется голосованием или усреднением прогнозов от всех деревьев.
4. Бэггинг: Random Forest использует технику бэггинга (bootstrap aggregating), которая комбинирует прогнозы нескольких деревьев для повышения обобщающей способности модели. Бэггинг помогает уменьшить переобучение и повысить стабильность модели.
5. Важность признаков: Random Forest позволяет оценить важность признаков, основываясь на том, как часто признаки используются для разделения узлов деревьев. Это может быть полезно для определения наиболее информативных признаков в задаче.
Random Forest обладает рядом преимуществ, включая способность обрабатывать большие наборы данных, устойчивость к выбросам и способность к обобщению на новые данные. Он часто применяется в различных областях, включая медицину, финансы, маркетинг и другие, для решения задач классификации.
В рамках данного исследования был изучен и применен алгоритм Random Forest, который является мощным инструментом машинного обучения для задач классификации и регрессии. Random Forest представляет собой ансамбль решающих деревьев, где каждое дерево обучается на случайно выбранных подмножествах признаков и данных.
Алгоритм Random Forest обладает рядом преимуществ, таких как высокая точность предсказаний, устойчивость к переобучению и способность обрабатывать большие объемы данных. Он также позволяет оценить важность признаков, что помогает выявить наиболее информативные факторы для классификации или регрессии.
В ходе исследования было проведено обучение и тестирование модели Random Forest на реальных наборах данных, что позволило получить значимые результаты. Было показано, что Random Forest демонстрирует высокую точность классификации и регрессии, превосходящую другие алгоритмы машинного обучения.
Таким образом, на основании проведенного исследования можно сделать вывод о том, что Random Forest является эффективным и надежным инструментом для решения задач классификации и регрессии. Его применение может быть ценным в различных областях, таких как медицина, финансы, биология и другие, где необходима точная и устойчивая модель предсказаний.
Однако, необходимо учитывать, что Random Forest не лишен некоторых ограничений, включая вычислительную сложность для больших данных и потребность в настройке гиперпараметров. Поэтому дальнейшие исследования

будут направлены на оптимизацию и адаптацию алгоритма Random Forest для конкретных задач и контекстов применения.
В целом, Random Forest является важным и полезным инструментом машинного обучения, способным обеспечить высокую точность и надежность предсказаний. Его применение может иметь значимый вклад в различных областях и открыть новые возможности для исследований и практического применения машинного обучения. Результат работы алгоритма Random Forest приведен на рисунке 2.8.1.

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(Xtrain,Ytrain)

predicted_values = RF.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('RF')
print("RF's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
RF's Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
```

Рисунок 2.8.1. Результат выполнения Random Forest

2.9Классификатор XGBoost

XGBoost (eXtreme Gradient Boosting) - это высокоэффективный и мощный алгоритм машинного обучения, который применяется для задач классификации, регрессии и ранжирования. Он является разновидностью градиентного бустинга и широко используется во многих соревнованиях по анализу данных и промышленных приложениях. XGBoost основан на

градиентном бустинге и является одним из наиболее мощных и популярных алгоритмов в сфере анализа данных и соревнований по машинному обучению.
Основные концепции и принципы работы классификатора XGBoost:
1. Градиентный бустинг: XGBoost базируется на градиентном бустинге, который является методом ансамблирования моделей. Градиентный бустинг строит последовательность слабых моделей (обычно деревьев решений) и постепенно улучшает их, минимизируя ошибку предсказания на каждом шаге.
2. Объединение моделей: XGBoost объединяет предсказания нескольких деревьев решений в одно композитное предсказание. Каждое дерево обучается на остатках предыдущих деревьев, чтобы сконцентрироваться на тех образцах, где модель делает ошибки.
3. Градиентный спуск: XGBoost использует градиентный спуск для настройки параметров модели. Он оптимизирует функцию потерь, минимизируя разницу между фактическими значениями и предсказаниями модели.
4. Регуляризация: XGBoost включает регуляризацию для предотвращения переобучения и улучшения обобщающей способности модели. Регуляризация может включать L1- и L2-регуляризацию, а также штрафы за сложность модели и количество использованных деревьев.
5. Отбор признаков: XGBoost автоматически выполняет отбор наиболее информативных признаков, используя их важность, рассчитанную на основе их вклада в улучшение функции потерь.
XGBoost может использоваться как для задач классификации, так и для задач регрессии, а также для решения других задач, таких как ранжирование и ранжирование с перестановками. Он является гибким и мощным инструментом, который позволяет достичь высокой точности и производительности в решении разнообразных задач машинного обучения.XGBoost предлагает оптимизированные алгоритмы и структуры данных, которые обеспечивают высокую скорость работы и эффективное использование памяти.
XGBoost обладает рядом преимуществ, включая высокую точность предсказания, возможность обработки больших объемов данных, эффективность вычислений и поддержку распараллеливания. Он широко применяется в различных областях, включая финансы, рекламу, рекомендательные системы и многие другие. На рисунке 2.9.1 показаны результат алгоритма XGBoost.

```
import xgboost as xgb
XB = xgb.XGBClassifier(eval_metric='mlogloss')
XB.fit(Xtrain,Ytrain)

predicted_values = XB.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)

model.append('XGBoost')
print("XGBoost's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
XGBoost's Accuracy is:  0.9931818181818182
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
```

Рисунок 2.9.1. Результат выполнения классификатора XGBoost

XGBoost является мощным и эффективным алгоритмом машинного обучения, который предоставляет высокую точность предсказаний и может быть успешно применен для решения широкого спектра задач классификации и регрессии.

2.10 Сравнительный анализ методологии классификаторов

На основе полученных результатов будем сформировать диаграмму сравнения результатов как показано на рисунке 2.10.1.

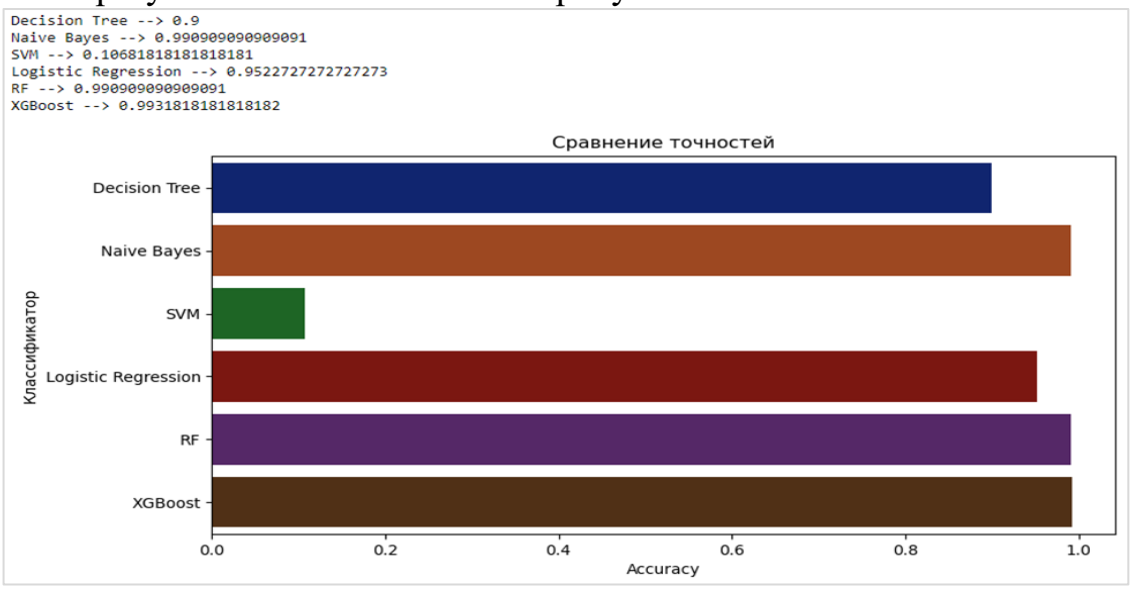

Рисунок 2.10.1. Сводный отчет результатов классификаторов

Как мы можем видеть из вышеперечисленных результатов, классификаторы Naive Bayes, Random Forest и XGBoost показали хорошие результаты, тогда как остальные оценки менее удовлетворительные.

2.11 Пример рекомендации модели (прогноз)

На рисунке 2.11.1 показаны результаты проверки моедели.

```
# XGBoost Prediction
mydata = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = XB.predict(mydata)
print(prediction)
```

```
['coffee']
```

```
# Random Forest Prediction
mydata = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(mydata)
print(prediction)
```

```
['coffee']
```

```
# NaiveBayes Prediction
mydata = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = NaiveBayes.predict(mydata)
print(prediction)
```

```
['coffee']
```

Рисунок 2.11.1. Прогноз модели

3Разработка веб сервиса для рекомендации и прогнозирования урожая

3.1 Сервис рекомендации урожая

Было решено выбрать в качестве метода машинного обучения Random Forest на основе сравнения результатов сравнения с другими методами. Теперь для составления веб приложения нам требуется обучить модель и сохранить, чтобы в дальнейшем было возможность обратиться к нему за рекомендациями.
Алгоритм разработки сервиса:

- обучить модель и сохранить в виде файла
- разработать веб приложение и интегрировать с обученной моделью
- разработать веб интерфейс с возможностью выбора участка на карте

В качестве инструментарий для обучения модели будем использовать Jupiter Notebook, а чтобы сохранить используем модуль pickle. Чтобы выбрать участок решено использовать открытую библиотеку картографирования leaflet.js.
Pickle - это модуль в языке программирования Python, который используется для сериализации (преобразования объекта в поток байтов) и десериализации (восстановления объекта из потока байтов) объектов Python. Он позволяет сохранять объекты в файл или передавать их по сети, а затем восстанавливать их в исходное состояние. Важно отметить, что модуль pickle может быть уязвим для безопасности, поэтому при работе с непроверенными данными из ненадежных источников необходимо быть осторожным, чтобы избежать возможных атак на безопасность.
Python - это высокоуровневый, интерпретируемый, общего назначения язык программирования. Он был разработан в конце 1980-х годов Гвидо ван Россумом и с тех пор стал одним из наиболее популярных языков программирования.
Ключевые особенности Python:
1. Простота чтения и написания кода: Python имеет простой и понятный синтаксис, который делает код на нем легким для чтения и написания. Он поддерживает читаемость кода, используя отступы вместо фигурных скобок или ключевых слов для обозначения блоков кода. Это делает Python привлекательным языком для начинающих программистов.
2. Многофункциональность: Python является многофункциональным языком, который поддерживает различные стили программирования, включая объектно-ориентированное программирование (ООП), процедурное программирование и функциональное программирование. Он также предоставляет обширную стандартную библиотеку, которая содержит множество полезных модулей и инструментов для различных задач.
3. Поддержка кросс-платформенности: Python является кросс-платформенным языком, что означает, что он может работать на разных операционных системах, таких как Windows, macOS и Linux. Это делает его удобным выбором для разработки переносимых приложений.

4. Большое сообщество и экосистема: Python имеет активное сообщество разработчиков, которое предлагает множество библиотек, модулей и фреймворков для различных областей разработки, таких как наука о данных, веб-разработка, машинное обучение, искусственный интеллект и другие. Некоторые из популярных библиотек включают NumPy, Pandas, Matplotlib, TensorFlow и Django.
5. Интеграция с другими языками: Python имеет возможность интеграции с другими языками программирования, такими как C/C++, Java и другими. Это позволяет использовать Python для разработки определенных компонентов приложений или для расширения функциональности с использованием других языков.
Python широко применяется во многих областях, включая веб-разработку, научные исследования, анализ данных, машинное обучение, автоматизацию задач, разработку игр и многое другое.
Jupyter Notebook (ранее известный как IPython Notebook) - это интерактивная среда разработки, которая позволяет создавать и выполнять код, объединяя его с текстом, изображениями, формулами и визуализацией результатов. Jupyter Notebook предоставляет удобный интерфейс для работы с кодом на различных языках программирования, включая Python, R, Julia и другие.
Ключевые особенности Jupyter Notebook:
1. Интерактивная среда: Jupyter Notebook предоставляет интерактивную среду, где код может быть написан и выполняться по ячейкам. Вы можете выполнить отдельные ячейки кода, что позволяет пошагово анализировать результаты и проводить эксперименты. Это удобно для разработки, тестирования и отладки кода.
2. Объединение кода и документации: Jupyter Notebook позволяет встраивать код вместе с различными типами контента, такими как текст, изображения, формулы и графики. Это делает его отличным инструментом для создания интерактивных отчетов, презентаций, обучающих материалов и научных исследований.
3. Поддержка различных языков программирования: Jupyter Notebook поддерживает несколько языков программирования, включая Python, R, Julia и другие. Вы можете создавать ячейки с кодом на разных языках в рамках одного ноутбука.
4. Визуализация результатов: Jupyter Notebook обеспечивает возможность визуализации данных и результатов вычислений прямо в ноутбуке. Вы можете строить графики, создавать таблицы, отображать изображения и прочие визуальные элементы для анализа и представления данных.
5. Гибкость и расширяемость: Jupyter Notebook позволяет расширять его функциональность с помощью плагинов и расширений. Можно установить дополнительные расширения, которые предоставляют дополнительные возможности, такие как автоматическое сохранение, отладка кода, поддержка Git и многое другое.

Jupyter Notebook является популярным инструментом среди данных ученых, программистов и исследователей для выполнения интерактивного программирования, проведения анализа данных, создания отчетов и обмена знаниями.

Leaflet.js - это открытая JavaScript библиотека, которая используется для создания интерактивных карт и веб-картографических приложений. Она предоставляет простой и эффективный способ внедрения картографической функциональности в веб-приложения.

Некоторые ключевые особенности и возможности Leaflet.js:

1. Легковесность и простота использования: Leaflet.js является легковесной библиотекой, которая имеет небольшой размер и простой в освоении API. Она предлагает простые методы и функции для создания и настройки карт, добавления слоев, маркеров, полигонов и других географических элементов.
2. Кросс-платформенность: Библиотека Leaflet.js работает на различных платформах и устройствах, включая компьютеры, планшеты и мобильные устройства. Она поддерживает множество веб-браузеров, таких как Chrome, Firefox, Safari, Edge и другие.
3. Интерактивность: Leaflet.js предоставляет возможности для взаимодействия с картами и географическими элементами. Вы можете добавлять пользовательские действия, такие как щелчок мыши или перемещение, для обработки событий на карте. Это позволяет создавать интерактивные картографические приложения с функциями масштабирования, перемещения и выбора объектов на карте.
4. Настраиваемость и расширяемость: Leaflet.js предлагает широкий выбор настраиваемых опций и расширений, которые позволяют создавать уникальные и индивидуальные картографические приложения. Вы можете настроить внешний вид и стиль карты, добавить пользовательские значки, слои и панели управления.
5. Интеграция с другими сервисами: Leaflet.js позволяет интегрировать карты с различными сервисами и источниками данных, такими как OpenStreetMap, Mapbox, Google Maps, Bing Maps и другие. Вы можете использовать собственные данные или воспользоваться открытыми географическими данными для отображения информации на карте.

Благодаря своей гибкости и простоте использования, Leaflet.js является популярным инструментом для создания интерактивных картографических приложений веб-разработки.

В качестве платформы для backend будем использовать фреймворк Flask.

Flask - это легкий фреймворк веб-приложений для языка программирования Python. Он предоставляет простую и эффективную основу для создания веб-приложений, API и других веб-сервисов.

Ключевые особенности Flask:

1. Простота использования: Flask разработан с упором на простоту и минималистичность. Он имеет небольшое API и простую структуру, что делает его легким для изучения и начала работы.

2. Маршрутизация и представления: Flask предлагает мощный механизм маршрутизации, который позволяет связать URL-адреса с функциями-обработчиками, называемыми представлениями (views). Представления обрабатывают запросы, выполняют нужные операции и возвращают ответы.
3. Встроенная поддержка шаблонов: Flask включает в себя встроенную поддержку шаблонов Jinja2, которая позволяет создавать динамические HTML-страницы и другие типы контента, вставляя переменные, циклы, условия и другие элементы в шаблон.
4. Расширяемость: Flask поддерживает большое количество расширений (extensions), которые предоставляют дополнительные возможности и интеграцию с другими библиотеками и инструментами. Например, есть расширения для работы с базами данных, аутентификации, обработки форм и многое другое.
5. Встроенная отладка и разработка: Flask предлагает удобные инструменты для отладки и разработки веб-приложений. Он включает в себя встроенный сервер разработки, автоматическую перезагрузку при изменении кода и подробные сообщения об ошибках.
6. Гибкость: Flask позволяет разработчикам создавать веб-приложения любого размера и сложности. Он не навязывает жестких правил и позволяет выбирать подходящие инструменты и архитектуру в соответствии с требованиями проекта.
Flask является одним из самых популярных фреймворков для разработки веб-приложений на языке Python благодаря своей простоте, гибкости и активному сообществу разработчиков. На рисунке 3.1.1 показан сохранение модели с помощью библиотеки pickle.

```
In [ ]: import pickle
        # указываем путь где будем хранить наш обученный модель
        RF_pkl_filename = '../models/RandomForest.pkl'
        # открываем для записи
        RF_Model_pkl = open(RF_pkl_filename, 'wb')
        pickle.dump(RF, RF_Model_pkl)
        # сохраняем и закрываем ссылку
        RF_Model_pkl.close()
```

Рисунок 3.1.1. Сохранение модели

Рассмотрим взаимодействие flask и нашей моделю. На рисунке 3.1.2 показан обработка запроса с flask.Как видно на рисунке 3.1.2 мы объявляем маршрут “urojai” для обработки запроса от клиента, который будет будет доступна только для POST запросов.

После получения POST параметров передаем их значения в наш модель crop_recommendation_model который и будет рекомендовать какую лучше культуру можно вырастить (Рисунок 3.1.3).
На рисунке 3.1.4 показан пользовательский интерфейс для взаимодействия с системой.

```python
@ app.route('/urojai', methods=['POST'])
def crop_prediction():
    title = 'SE Disser – Рекомендация по культуре'

    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])
        t = float(request.form['t'])
        h = float(request.form['h'])

        try:
            data = np.array([[N, P, K, t, h, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)
        except:
            return render_template('try_again.html', title=title)
```

Рисунок 3.1.2. Обработка запроса пользователя и предсказание с помощью модели

```python
# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))
```

Рисунок 3.1.3. Восстановление модели с pkl файла

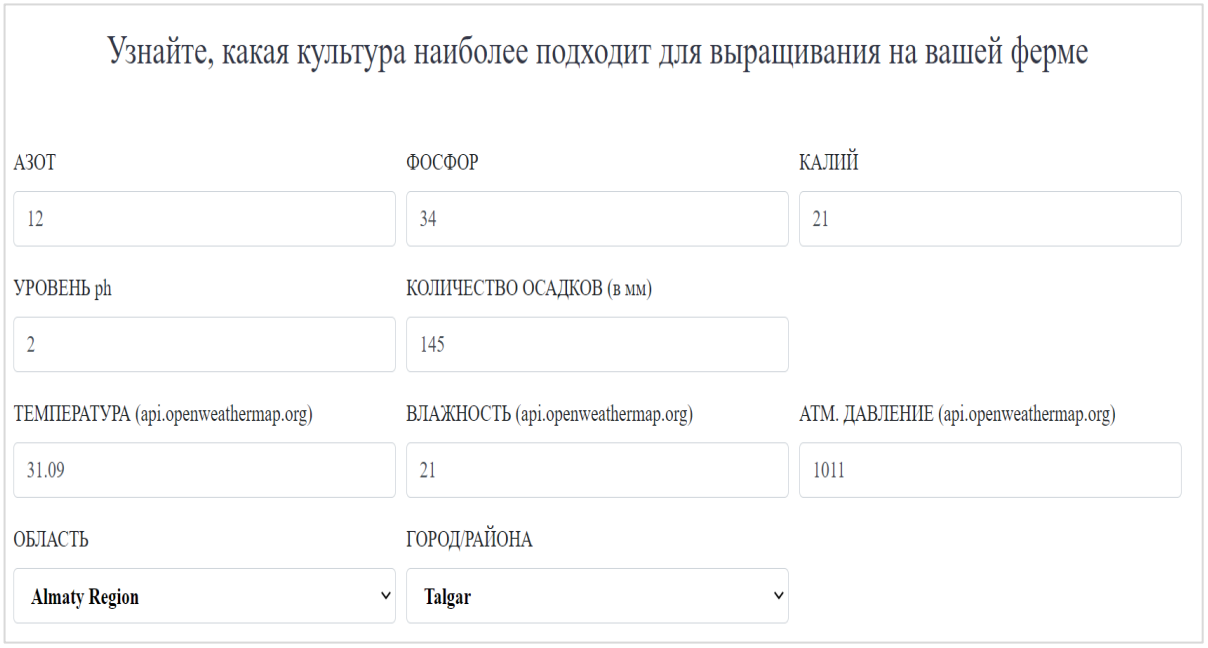

Рисунок 3.1.4. Форма ввода параметров

С помощью карты leaflet.jsможно выбрать определенный участок как показана на рисунке 3.1.5.

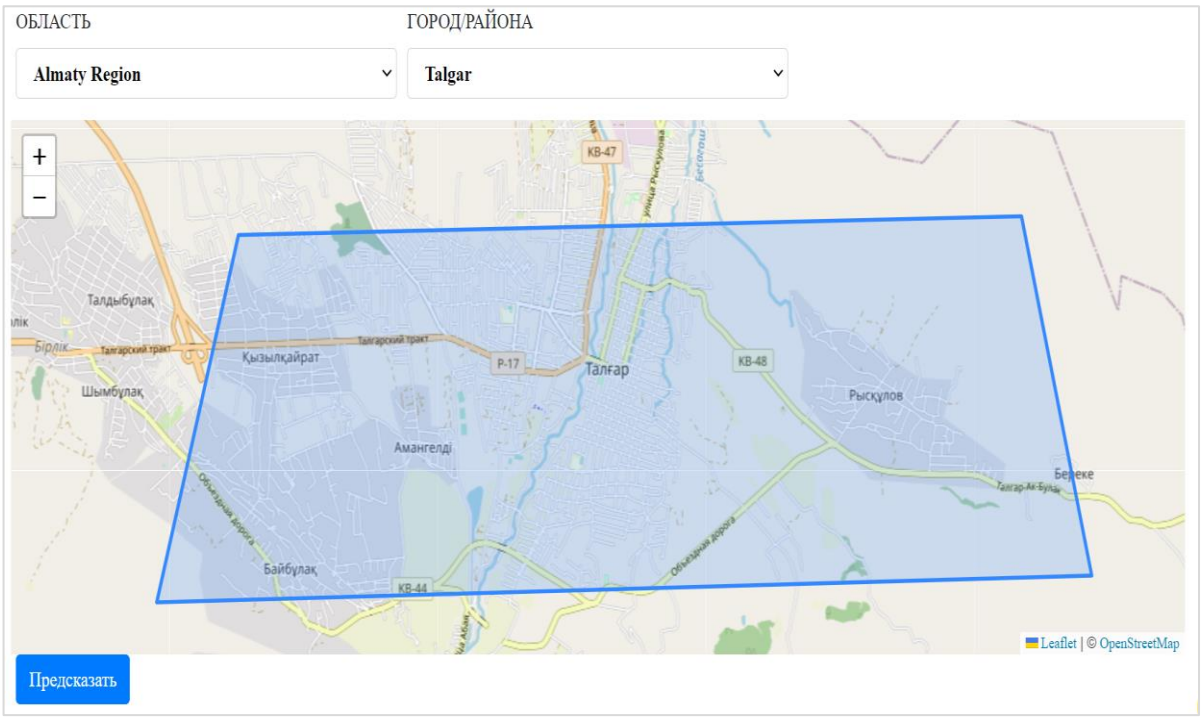

Рисунок 3.1.5. Выбор местности на карте

На основе введенных данных получаем результаты. Результаты системы показаны на рисунке 3.1.6.

Рисунок 3.1.6. Получение рекомендации системы

3.2 Сервис определения болезни растения

Для определения болезни растения будем обучать модель на распознавание снимков между заболевшими и здоровыми листьями культур. Набор данных взято с репозиторий https://github.com/spMohanty/PlantVillage-Dataset. Этот набор данных состоит примерно из 87 тыс. rgb-изображений здоровых и больных листьев сельскохозяйственных культур, которые разделены на 38 различных классов. Общий набор данных делится в соотношении 80/20 на обучающий и проверочный наборы с сохранением структуры каталогов. Позже для целей прогнозирования будет создан новый каталог, содержащий 33 тестовых изображения, как показана на рисунке 3.2.1.

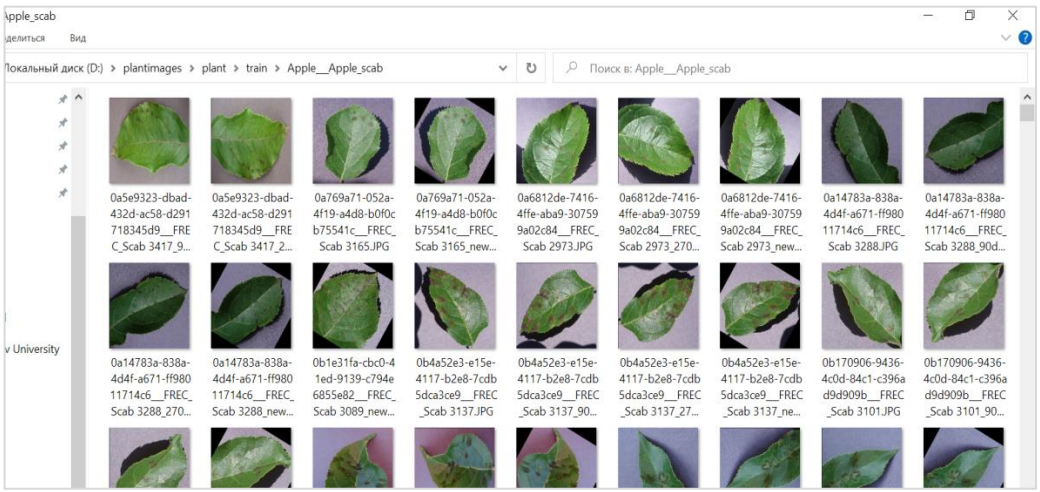

Рисунок 3.2.1 Набор данных

Нашей целью является построить модель, которая может классифицировать здоровые и больные листья урожая, а также, если у урожая есть какое-либо заболевание, предсказать, какое именно. На рисунке 3.2.2 показаны список болезней растений.

```
data_dir = "D://plantimages//plant"
train_dir = data_dir + "//train"
valid_dir = data_dir + "//valid"
diseases = os.listdir(train_dir)
```

```
# печатаем названия болезней растения
print(diseases)
```

```
['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust', 'Apple___healthy', 'Blueberry___healthy', 'Cherry_(incl
uding_sour)___healthy', 'Cherry_(including_sour)___Powdery_mildew', 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot', 'Corn
_(maize)___Common_rust_', 'Corn_(maize)___healthy', 'Corn_(maize)___Northern_Leaf_Blight', 'Grape___Black_rot', 'Grape___Esca_
(Black_Measles)', 'Grape___healthy', 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Orange___Haunglongbing_(Citrus_greening)',
'Peach___Bacterial_spot', 'Peach___healthy', 'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy', 'Potato___Early_bligh
t', 'Potato___healthy', 'Potato___Late_blight', 'Raspberry___healthy', 'Soybean___healthy', 'Squash___Powdery_mildew', 'Strawbe
rry___healthy', 'Strawberry___Leaf_scorch', 'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy', 'Tomato___L
ate_blight', 'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite', 'Tomato___Ta
rget_Spot', 'Tomato___Tomato_mosaic_virus', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus']
```

Рисунок 3.2.2. Список болезней растения

```
# названия уникальных растения в наборе данных
print(f"Названия (уникальные): \n{plants}")
```

```
Названия (уникальные):
['Apple', 'Blueberry', 'Cherry_(including_sour)', 'Corn_(maize)', 'Grape', 'Oran
rry', 'Soybean', 'Squash', 'Strawberry', 'Tomato']
```

```
# уникальное количество растении
print("уникальное количество растении: {}".format(len(plants)))
```

```
уникальное количество растении: 14
```

```
# количество болезней
print("Количество болезней растений: {}".format(NumberOfDiseases))
```

```
Количество болезней растений: 26
```

Рисунок 3.2.3. Информация о количествах болезней

Итак, у нас есть изображения листьев 14 растений, и, исключая здоровые листья, у нас есть 26 типов изображений, которые показывают конкретное заболевание конкретного растения.

3.2.1 Моделирование

Рекомендуется использовать GPU вместо центрального процессора при работе с набором данных изображений, поскольку центральные процессоры обобщены для общего назначения, а графические процессоры оптимизированы для обучения моделей глубокого обучения, поскольку они могут обрабатывать

несколько вычислений одновременно. Они имеют большое количество ядер, что позволяет лучше выполнять вычисления нескольких параллельных процессов. Кроме того, вычисления в режиме глубокого обучения требуют обработки огромных объемов данных — это делает пропускную способность памяти графического процессора наиболее подходящей. Чтобы беспрепятственно использовать графический процессор, если таковой доступен, мы определяем пару вспомогательных функций (get_default_device & to_device) и вспомогательный класс DeviceDataLoader для перемещения нашей модели и данных в графический процессор по мере необходимости. Вспомогательные функции приведены на рисунке 3.2.1.1.

```
# для перемещения данных в графический процессор (если доступно)
def get_default_device():
    """Выберите GPU, если доступно, иначе CPU"""
    if torch.cuda.is_available:
        return torch.device("cuda")
    else:
        return torch.device("cpu")

# for moving data to device (CPU or GPU)
def to_device(data, device):
    """Переместить тензор(ы) на выбранное устройство"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

# Переместите тензор (тензоры) в выбранное положение для загрузки в устрой
class DeviceDataLoader():
    """Оберните загрузчик данных для перемещения данных на устройство"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Выдает пакет данных после перемещения его на устройство"""
        for b in self.dl:
            yield to_device(b, self.device)
```

Рисунок 3.2.1.1. Вспомогательные функции

Мы будем использовать ResNet, которые стали одним из главных достижений в области компьютерного зрения с момента их появления в 2015 году.

Основная идея ResNet заключается в использовании так называемых "skip connections" или "residual connections" (пропускающих соединений или остаточных соединений). Это соединения, которые пропускают один или несколько слоев нейронной сети, позволяя информации изначально "проходить мимо" этих слоев.

Пропускающие соединения в ResNet решают проблему деградации точности при увеличении глубины нейронной сети. Обычно с увеличением глубины сети

происходит ухудшение результатов, так как градиенты становятся очень маленькими и исчезают в процессе обратного распространения ошибки. Пропускающие соединения позволяют сохранять и использовать информацию из предыдущих слоев, что упрощает обучение и помогает преодолеть проблему деградации.Прежде чем мы обучим модель, определим служебную функцию - функцию оценки, которая будет выполнять этап 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 серии тренировок. Существует множество стратегий для изменения скорости обучения во время обучения, и та, которую мы будем использовать, называется “Политика скорости обучения за один цикл”, которая предполагает начинать с низкой скорости обучения, постепенно увеличивая ее от партии к партии до высокой скорости обучения примерно в течение 30% периодов, затем постепенно уменьшаем его до очень низкого значения для оставшихся эпох.

Уменьшение веса: Мы также используем уменьшение веса, которое является методом регуляризации, который предотвращает слишком большие веса путем добавления дополнительного члена к функции потерь.

Обрезка градиента: Помимо весов слоев и выходных данных, также полезно ограничить значения градиентов небольшим диапазоном, чтобы предотвратить нежелательные изменения параметров из-за больших значений градиента. Этот простой, но эффективный прием называется градиентной обрезкой.

Мы также запишем скорость обучения, используемую для каждой партии.

```
def fit_OneCycle(epochs, max_lr, model, train_loader, val_loader, weight_decay=0,
                 grad_clip=None, opt_func=torch.optim.SGD):
    torch.cuda.empty_cache()
    history = []

    optimizer = opt_func(model.parameters(), max_lr, weight_decay=weight_decay)
    # scheduler for one cycle learning rate
    sched = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr, epochs=epochs, steps_per_epoch=l


    for epoch in range(epochs):
        # Training
        model.train()
        train_losses = []
        lrs = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()

            # gradient clipping
            if grad_clip:
                nn.utils.clip_grad_value_(model.parameters(), grad_clip)
```

Рисунок 3.2.1.2. Функция fit_one_cycle

При сохранении модели для логического вывода необходимо сохранить только изученные параметры обученной модели. Сохранение state_dict модели с помощью функции torch.save() даст нам максимальную гибкость для последующего восстановления модели, вот почему это рекомендуемый метод сохранения моделей.

Распространенным соглашением PyTorch является сохранение моделей с использованием расширения файла .pt или .pth. Сохраняем модель как показана на рисунке 3.2.1.3.

```
# сохраняем обученный модель
PATH = './plant-disease-model.pth'
torch.save(model.state_dict(), PATH)
```

Рисунок 3.2.1.3. Сохранение модели

На рисунке 3.2.1.4 показан интерфейс определения болезни растения.

Рисунок 3.2.1.4. Интерфейс для предсказания болезни растений

На рисунке 3.2.1.5 показаны результат определения болезни растений.

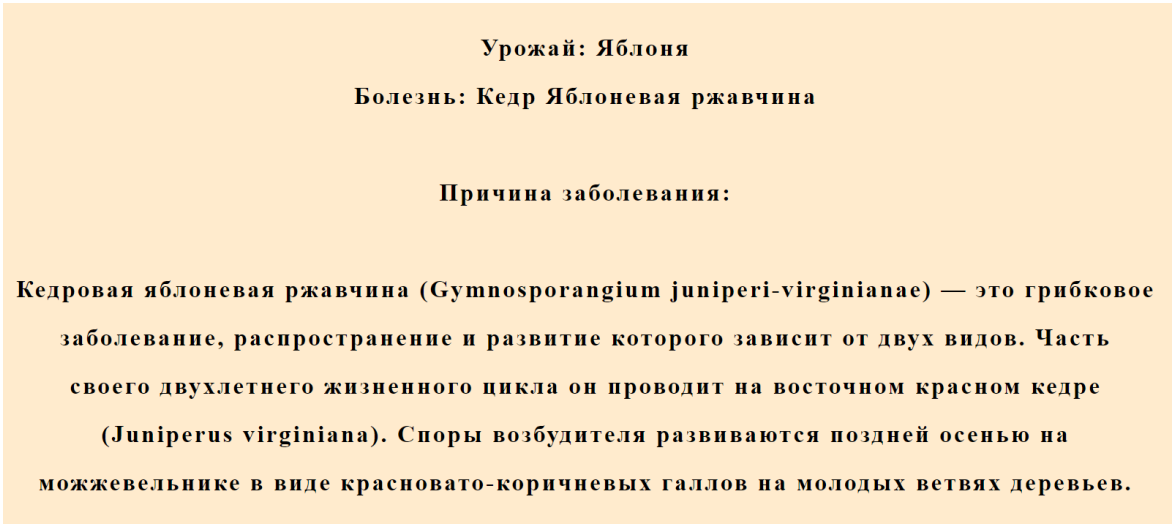

Рисунок 3.2.1.5. Предсказание модели о болезни

ResNet значительно лучше подходят для классификации изображений, когда изменяются некоторые параметры 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.

Заключение

В работе были изложены возможности алгоритмов машинного для выявления подходящих культур к выбранным участкам посева. Были обучены модели к шести алгоритмам чтобы сравнить и выбрать наиболее лучшие из них. Самыми производительными алгоритмами оказались XGBoost, Random Forest, Naive Bayes.
Подобраны оптимальные значения гиперпараметров для этих моделей и определены наиболее значимые переменные для классификации каждого типа почв. Точность методики на валидационной выборке составила 90%.
Перспективные высокие технологии в сельском хозяйстве движутся в будущее семимильными шагами. Они предлагают существенную помощь фермерам в их усилиях по оптимизации затрат, упрощению управления сельским хозяйством и повышению производительности. Повышение урожайности, а также снижение затрат на техническое обслуживание помогают повысить рентабельность.

В заключение, сельское хозяйство активно внедряет высокие технологии для совершенствования процессов производства и увеличения эффективности. Несколько перспективных технологий, которые оказывают значительное влияние на сельское хозяйство, включают интернет вещей (IoT) и датчики, беспилотные автоматизированные системы (дроны), автоматизацию и робототехнику, анализ данных и машинное обучение, вертикальное фермерство и гидропонику, а также блокчейн технологии.Эти технологии предлагают фермерам и другим участникам сельскохозяйственного сектора возможности оптимизации использования ресурсов, повышения производительности, улучшения качества продукции и устойчивости производства. Они позволяют собирать и анализировать данные, прогнозировать урожайность, автоматизировать задачи и оптимизировать процессы.

Преимущества внедрения высоких технологий в сельском хозяйстве включают повышение эффективности использования ресурсов, снижение затрат на производство, улучшение контроля над условиями выращивания, увеличение точности и надежности операций, а также улучшение качества продукции. Кроме того, эти технологии могут способствовать более устойчивому сельскому хозяйству, сокращению отходов и негативного воздействия на окружающую среду.
Однако при внедрении высоких технологий в сельском хозяйстве также возникают некоторые вызовы, такие как высокие затраты на оборудование и обучение персонала, необходимость обеспечения надежности и безопасности систем, а также преодоление технических и организационных преград.

В целом, перспективные высокие технологии играют важную роль в развитии сельского хозяйства, повышении его эффективности и устойчивости. При правильном внедрении и использовании эти технологии могут привести к значительному прогрессу в сельскохозяйственном секторе, обеспечивая более продуктивное и устойчивое производство пищевых продуктов для растущей мировой популяции.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дринча В.М. Информационные системы на службе сельского хозяйства. Экономика и финансы, 2014. – С.135.
2. Меденников В.И. Современное состояние информационно-управляющих систем поддержки принятия решений в АПК, 2015. – С. 23-45.
3. Гасликова И.Р., Гохберг Л.М. Информационные технологии в России. М.: ЦИСН, 2012. – С. 45-64.
4. Юрченко И.Ф. Информационные технологии обоснования мелиораций.М., 2020. – С. 34-85.
5. Кондрашова С.С. Информационные технологии в управлении: Учебное пособие. - К.: МАУП. 2013. – С .89-101.
6. Kangning Wei, Jinghua Huang, Shaohong Fu, A survey of e-commerce recommender systems, in: 2017 international conference on service systems and service manage- ment, IEEE, 2017, pp. 1–5
7. Shaha T. Al-Otaibi, Mourad Ykhlef. A survey of job recommender systems, Int. J. Phys. Sci, 2012. - С. 5127–5142.
8. J.Malinowski, T.Keim, O.Wendt, T.Weitzel. A bilateral recommendation approach, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06), 6, IEEE, 2006. С. 136-137.
10. A.Golec, E.Kahya. A fuzzy model for competency-based employee evaluation and selection, Comput. Ind. Eng, 2007. - С.145.
11. G.Sidorov, A.Gelbukh, H.Gomez-Adorno, D.Pinto. Similarity of Features in VectorSpace Model, Computacion y Sistemas 2014. – С. 491–504.
12. P.A. Longley, M.F. Goodchild, David J. Maguire. Geographic Information Systems and Science, 2015. – С. 345.
13. Paul Bolstad, Geographic Information Systems: Principles, Techniques, Management, and Applications, 2016.- С. 45.
14. Kang-Tsung Chang.Introduction to Geographic Information Systems, 2015.
15. M.Price. Mastering ArcGIS,SciPublish, 2017. - С. 243.
16. P.Bolstad. GIS Fundamentals: A First Text on Geographic Information Systems, XanEdu Publishing Inc, 2020. – С. 345.
17. M. Smith, Michael F, Goodchild, P.A.Longley. Geospatial Analysis: A Comprehensive Guide, Spatialanalysisonline Publish,2015. – С. 221.
18. L.Tateosian. Python for ArcGIS,2015.- С. 145.
19. M.Law, A.Collins. Getting to Know ArcGIS Pro ,EsRi, 2016.- С. 45.
20. M.Wegmann, B.Leutner,S.Dech.Remote Sensing and GIS for Ecologists: Using Open Source Software,Pelagic Publishing, 2019.-С. 25.
21.M.Neteler, H.Mitasova. Open Source GIS: A GRASS GIS Approach: Springer New York, NY, 2004.-С.12.

Приложение А

Листинг программы Сервиса рекомендации урожая

```
# Importing essential libraries and modules
from flask import Flask, render_template, request, Markup, jsonify, make_response
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
# ==============================================================
# ------------------------LOADING THE TRAINED MODELS -----------------------
# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple___Cedar_apple_rust',
'Apple___healthy',
                   'Blueberry___healthy',
'Cherry_(including_sour)___Powdery_mildew',
                   'Cherry_(including_sour)___healthy',
                   'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
                   'Corn_(maize)___Common_rust_',
                   'Corn_(maize)___Northern_Leaf_Blight',
                   'Corn_(maize)___healthy',
                   'Grape___Black_rot',
                   'Grape___Esca_(Black_Measles)',
                   'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
                   'Grape___healthy',
                   'Orange___Haunglongbing_(Citrus_greening)',
                   'Peach___Bacterial_spot',
                   'Peach___healthy',
                   'Pepper,_bell___Bacterial_spot',
                   'Pepper,_bell___healthy',
                   'Potato___Early_blight',
```

```
                'Potato___Late_blight',
                'Potato___healthy',
                'Raspberry___healthy',
                'Soybean___healthy',
                'Squash___Powdery_mildew',
                'Strawberry___Leaf_scorch',
                'Strawberry___healthy',
                'Tomato___Bacterial_spot',
                'Tomato___Early_blight',
                'Tomato___Late_blight',
                'Tomato___Leaf_Mold',
                'Tomato___Septoria_leaf_spot',
                'Tomato___Spider_mites Two-spotted_spider_mite',
                'Tomato___Target_Spot',
                'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
                'Tomato___Tomato_mosaic_virus',
                'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()
# Loading crop recommendation model
crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
open(crop_recommendation_model_path, 'rb'))
# =========================================================
# Custom functions for calculations
def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
:params: city_name
:return: temperature, humidity
"""
api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"
    complete_url = base_url + "appid=" + api_key + "&q=" +
city_name+"&units=metric"
    response = requests.get(complete_url)
    x = response.json()
    if x["cod"] != "404":
        y = x["main"]
        temperature = y["temp"] #round((y["temp"] - 273.15), 2)
```

```
        humidity = y["humidity"]
        pressure = y["pressure"]
        return temperature, humidity, pressure
    else:
return None
def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
:params: image
:return: prediction (string)
"""
transform = transforms.Compose([
transforms.Resize(256),
transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
img_t = transform(image)
img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
    yb = model(img_u)
# Pick index with highest probability
_, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    # Retrieve the class label
    return prediction

# ==============================================================
# ---------------------------------- FLASK APP -------------------------------------------
app = Flask(__name__)
# render home page
@ app.route('/')
def home():
title = 'SE Disser - Главная страница'
return render_template('index.html', title=title)
# render crop recommendation form page
@ app.route('/urojai')
def crop_recommend():
title = 'SE Disser - Рекомендация по культуре'
return render_template('crop.html', title=title)
# render fertilizer recommendation form page
@ app.route('/udobrenie')
def fertilizer_recommendation():
title = 'SE Disser - Предложение по удобрению'
```

```
return render_template('fertilizer.html', title=title)

# render disease prediction input page
# ========================================================
# RENDER PREDICTION PAGES
# render crop recommendation result page
@ app.route('/urojai', methods=['POST'])
def crop_prediction():
    title = 'SE Disser — Рекомендацияпокультуре'
    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])
        t = float(request.form['t'])
        h = float(request.form['h'])
try:
data = np.array([[N, P, K, t, h, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]
            return render_template('crop-result.html', prediction=final_prediction,
title=title)
except:
return render_template('try_again.html', title=title)
# render fertilizer recommendation result page
@ app.route('/udobrenie-predict', methods=['POST'])
def fert_recommend():
    title = 'SE Disser - Предложениепоудобрению'
    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])
    df = pd.read_csv('Data/fertilizer.csv')
    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
    pr = df[df['Crop'] == crop_name]['P'].iloc[0]
    kr = df[df['Crop'] == crop_name]['K'].iloc[0]
    n = nr - N
    p = pr - P
    k = kr - K
    temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
    max_value = temp[max(temp.keys())]
```

```
    if max_value == "N":
        if n < 0:
            key = 'NHigh'
        else:
            key = "Nlow"
    elif max_value == "P":
        if p < 0:
            key = 'PHigh'
        else:
            key = "Plow"
    else:
        if k < 0:
            key = 'KHigh'
        else:
            key = "Klow"
    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response,
title=title)
# render disease prediction result page
@app.route('/bolezn-predict', methods=['GET', 'POST'])
def disease_prediction():
    title = 'SE Disser — Обнаружениеболезней'
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
if not file:
return render_template('disease.html', title=title)
        try:
            img = file.read()
            prediction = predict_image(img)
            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
except:
            pass
return render_template('disease.html', title=title)
@app.route('/weather', methods=['POST'])
def get_weaher_data():
    city_name = request.form['city_name']
    data =  weather_fetch(city_name)
    return make_response(jsonify(data), 200)
    #return                                                    {temperature,humidity}
==========================================================
```

```
if __name__ == '__main__':
app.run(debug=True)
```