МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Слушатель Тарасов Матвей Андреевич

Москва, 2024

Содержание

Введение

Современные технологии беспилотных летательных аппаратов (БПЛА) становятся все более актуальными благодаря широкому спектру применений в различных областях, включая сельское хозяйство, мониторинг окружающей среды, транспорт и безопасность. Мультироторные беспилотники, обладающие высокой маневренностью и простотой в управлении, занимают особое место в этой сфере. Несмотря на их потенциал и преимущества, вопросы обеспечения безопасности и надежности остаются первоочередными. Совершение аварийных ситуаций может повлечь за собой серьезные последствия, поэтому разработка систем, способствующих их своевременному обнаружению и предотвращению, является крайне важной задачей.

В последние годы наблюдается рост интереса к интеграции различных датчиков и технологий для повышения уровня безопасности БПЛА. Одним из ключевых направлений в данной области является использование инерциальных датчиков, таких как акселерометры и гироскопы. Эти устройства позволяют улавливать параметры движения летательного аппарата, что открывает новые возможности для мониторинга его состояния и окружающей среды. Однако использование данных инерциальных датчиков требует разработки алгоритмов, способных эффективно обрабатывать и анализировать информацию для выявления возможных аварийных ситуаций.

В данной работе будет рассмотрен процесс разработки системы детектирования аварийных ситуаций БПЛА мультироторного типа в условиях отложенных данных с использованием машинного обучения на основе анализа информации, получаемой от инерциальных датчиков. В частности, будет проанализировано, как мониторинг вибрации может служить индикатором возникновения потенциально опасных состояний. Вибрация является одним из универсальных показателей, характеризующих работу механических систем, и несет в себе информацию о состоянии узлов и агрегатов летательного аппарата.

Понимание ее режимов и закономерностей позволяет обнаруживать отклонения, указывающие на возможные неисправности или непредвиденные внешние воздействия.

Ключевым моментом в исследовании будет являться реализация методов обработки сигналов, получаемых от инерциальных датчиков, с целью создания системы, способной реагировать на изменения в поведении БПЛА. Эта система будет включать в себя алгоритмы детектирования аномалий, основанные на машинном обучениии, которые позволят идентифицировать аварийные ситуации до того, как они приведут к критическим последствиям. Таким образом, работа не только внесет вклад в теорию и практику безопасности эксплуатации БПЛА, но и откроет новые горизонты для их применения в различных областях, включая экологический мониторинг, поисково-спасательные операции и др.

Таким образом, разработка системы детектирования аварийных ситуаций является важным шагом к повышению надежности и безопасности мультироторных БПЛА. Успешная реализация данного проекта может привести к значительным улучшениям как в области технологии управления БПЛА, так и в области обеспечения безопасности полетов.

1 Аналитическая часть

1.1 Постановка задачи

В условиях динамичного и потенциально опасного рабочего окружения мультироторные беспилотные летательные аппараты (БПЛА) подвержены различным рискам, которые включают механические неисправности, внешние воздействия и погодные условия. Для повышения безопасности и надежности необходимо реализовать эффективные методы и алгоритмы, позволяющие своевременно детектировать аварийные ситуации на основе анализа показаний инерциальных датчиков, таких как акселерометры и гироскопы.

В случае аварийной ситуации с БПЛА, первым этапом является детектирование проблемы. Это может быть как неисправность системы управления, потеря связи, отказ двигателя или батареи, поломка винтов так и другие неполадки. Важно, чтобы БПЛА был оснащен системами мониторинга, которые отслеживают основные параметры. Когда происходит сбой, система автоматически генерирует сигнал об аварии, и алгоритм БПЛА активирует аварийный режим, который включает в себя переключение на автоматическое управление, возврат в исходную точку или немедленную посадку. Все эти действия требуют быстрого реагирования, чтобы минимизировать последствия аварии.

Следующим этапом является поиск места для посадки. В отличие от пилотируемых воздушных судов, БПЛА часто не имеют большого радиуса маневрирования, особенно если отказ произошел на значительной высоте. Поэтому на данном этапе БПЛА, как правило, ориентируется на ближайшие безопасные участки для посадки. Современные беспилотники оснащаются навигационными системами и сенсорами, которые могут автоматически сканировать землю, выбирая подходящие места для экстренной посадки — открытые поля, площадки, крыши зданий или другие безопасные участки.

Третий этап — это сама посадка. На этом этапе беспилотник использует систему автоматического пилотирования для контролируемого сплошного снижения, при этом часто учитывается скорость ветра, особенности рельефа и расстояние до места посадки. Если посадка осуществляется на заранее определенную платформу, как это бывает при посадке на вертолетные площадки или специальные посадочные станции, система использует датчики точной навигации, такие как GPS и визуальные сенсоры, чтобы точно привести аппарат к месту. После посадки система автоматически прекращает работу двигателей и активирует процессы, обеспечивающие безопасность аппарата, а также передает данные о завершении миссии или аварии на наземную станцию для дальнейшего анализа и принятия решений.

В данной работе будет рассматриваться реализация первого этапа — детектирования аварийной ситуации.

Здесь первым шагом является обработка сигналов, получаемых от инерциальных датчиков. На данном этапе может быть важно использование фильтров для устранения шума, который может исказить данные. Могут применяться как низкочастотные, так и высокочастотные фильтры, в зависимости от целей анализа. Не менее важным является возможное использование алгоритмов сглаживания, таких как метод скользящего среднего или экспоненциальное сглаживание, для уменьшения колебаний в показаниях датчиков.

После обработки сигналов с датчиков нужно выявить ключевые параметры состояния БПЛА, которые могут свидетельствовать об аварийной ситуации. Если показатели изменяются внезапно, то это может сигнализировать о потерях управления или механических повреждениях.

Другим важным аспектом является анализ вибрации — изучение спектра вибрационных сигналов, для выявления аномальных частот или амплитуд, которые могут указать на неисправности в двигателях или других механизмах.

На рисунке 1 представлена блок-схема модели обнаружения отказов БПЛА, состоящая из пяти ключевых шагов. Процесс начинается с записи данных ИНС

(инерциальной навигационной системы) во время полета. Эти данные затем передаются на Arduino для анализа и обработки.

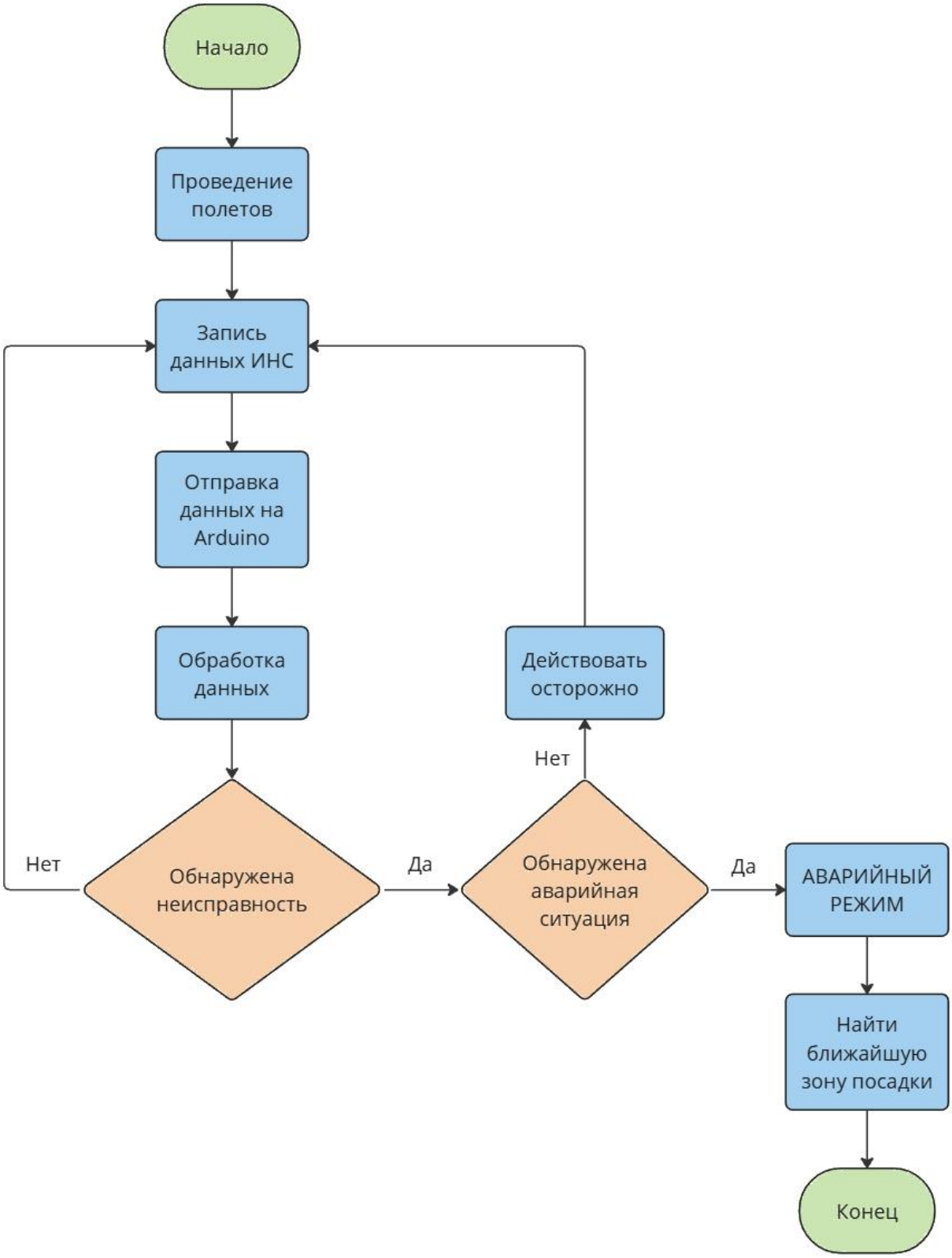

Рисунок 1. Блок-схема модели обнаружения отказов БПЛА.

Первый оранжевый ромб в схеме отвечает за выявление неисправностей в пропеллерах, в то время как второй оранжевый ромб предназначен для определения аварийной ситуации или общего отказа системы (например, невозможность выполнять базовые функции управления, что может привести к потере контроля над устройством.). Если пропеллеры функционируют исправно, процесс измерения вибрации повторяется до завершения полета. Однако, в случае обнаружения неисправности пропеллера без серьезных отказов, квадрокоптер продолжает полет, действуя с повышенной осторожностью.

Если же система фиксирует одновременно и неисправность пропеллера, и отказ, квадрокоптер переходит в аварийный режим, после чего немедленно инициируется поиск ближайшей безопасной зоны для посадки, чтобы предотвратить возможные катастрофические последствия.

В ходе данной работы разработано тестовое программное обеспечение, реализующее детектирование аварийной ситуации на базе автономных беспилотных летательных аппаратов (БПЛА) с использованием машинного обучения в условиях отложенных данных. Метод применяется к данным, получаемым с бортовых датчиков БПЛА (гироскопа и акселерометра), что позволяет эффективно обнаруживать физические неисправности, такие как поломка пропеллеров.

Рисунок 2. Поврежденные пропеллеры.

Основой для разработки системы послужил датасет, использованный в статье [1], посвященной методам обнаружения неисправностей БПЛА с помощью машинного обучения. Данный датасет был собран в ходе экспериментов на квадрокоптере с поврежденными пропеллерами, показанными на рисунке 2, в количестве 0, 1, 2, 3 и 4, что позволило смоделировать реальные условия эксплуатации БПЛА.

В ходе экспериментов были записаны показания множества датчиков, включающие данные о состоянии пропеллеров, информацию о скорости, ускорении и прочем. Эти данные использовались для создания моделей, способных предсказать вероятные неисправности и своевременно уведомить оператора. Была поставлена задача минимизировать количество ложных срабатываний и снизить вычислительные затраты.

Статья, на основе которой был разработан тестовый продукт, представляет собой описание подхода к обнаружению физических неисправностей на БПЛА, показанном на рисунке 3, с использованием машинного обучения. Модель, предложенная в статье, состоит из нескольких ключевых этапов, которые обеспечивают эффективную работу системы при ограниченных вычислительных ресурсах.

Рисунок 3. БПЛА, используемый для проведения эксперимента.

Целью тестового программного обеспечения, созданного в рамках данной работы, является извлечение признаков из показаний датчиков БПЛА (гироскопа и акселерометра) в процессе полетов квадрокоптера с различными степенями неисправности и построение моделей машинного обучения для детектирования аварийной ситуации в условиях отложенных данных. Программное обеспечение должно продемонстрировать способность к эффективному обнаружению неисправностей на основе предоставленных данных с датчиков.

Таким образом, данное исследование направлено на развитие и тестирование технологий, которые могут быть использованы для повышения надежности и безопасности БПЛА, а также для улучшения методов машинного обучения в условиях ограниченных вычислительных ресурсов.

1.2 Описание используемых методов

Для идентификации потенциальных аварийных ситуаций необходимо применение алгоритмов машинного обучения (МО) для анализа данных с акселерометров и гироскопов. МЛ-алгоритмы способны выявить сложные закономерности и аномалии, которые могут быть незаметны при традиционном анализе данных, и на основе этого принять решение о наличии угрозы безопасности. Рассмотрим несколько наиболее распространенных алгоритмов, которые могут быть использованы для детектирования аварийных ситуаций.

1.2.1 Метод опорных векторов (SVM - Support Vector Machine)

Метод опорных векторов (Support Vector Machines, SVM) [2] — это мощный алгоритм машинного обучения, используемый для решения задач классификации и регрессии. Ключевая идея SVM заключается в поиске оптимальной гиперплоскости, которая наилучшим образом разделяет данные на разные классы. Гиперплоскость – это многомерное обобщение линии, разделяющей пространство на две части, при этом каждая часть соответствует определенному классу.

Основная цель SVM – найти гиперплоскость с максимальным отступом (margin) от ближайших точек данных, называемых опорными векторами. Этот отступ является мерой "уверенности" классификации, и его максимизация позволяет повысить обобщающую способность модели.

Метод опорных векторов (SVM) особенно полезен для работы с задачами, где количество признаков значительно превышает число объектов. В таких случаях SVM демонстрирует свою способность эффективно находить границы между классами. Это связано с использованием линейных разделителей, адаптированных к высокой размерности.

Одним из ключевых преимуществ SVM является его устойчивость к переобучению. Благодаря концепции максимизации отступа (distance to the nearest point of any class), SVM находит лучшие гиперплоскости, минимизируя вероятность ошибки при классификации новых данных. Эта концепция обеспечивает высокую обобщающую способность, позволяя успешно применять SVM в широком спектре задач.

Помимо достоинств, данный метод имеет и недостатки, например, чувствительность к выбору ядра и гиперпараметров. Неправильный выбор ядра или гиперпараметров может негативно повлиять на производительность модели. Также SVM может быть вычислительно сложным, особенно для больших наборов данных, что делает его менее эффективным по сравнению с некоторыми другими алгоритмами (например, случайными лесами) на больших выборках.

Тем не менее метод опорных векторов — это мощный инструмент для классификации. Он показывает отличные результаты в разнообразных задачах классификации и регрессии.

1.2.2 Метод k-ближайших соседей (k-nearest neighbour)

Метод k-ближайших соседей (kNN) [3] — это популярный и интуитивно понятный алгоритм, который используется для классификации и регрессии в машинном обучении. Основная идея этого метода заключается в том, что

объекты, которые находятся близко друг к другу в пространстве признаков, вероятнее всего, принадлежат к одному классу.

Алгоритм можно применить как к задачам классификации, так и к задачам регрессии. В классификации объекту присваивается класс в зависимости от того, какой класс является наиболее распространённым среди k ближайших соседей. В регрессии алгоритм возвращает среднее значение от признаков ближайших соседей.

Метод kNN основывается на концепции расстояния между объектами. Он использует различные метрики, где одной из наиболее распространённой является евклидово расстояние. Эта метрика вычисляется по теореме Пифагора и показывает, насколько далека одна точка от другой. Однако стоит учитывать, что евклидово расстояние хорошо работает только в маломерных пространствах и не подходит для данных с большим количеством признаков.

Важным шагом перед применением алгоритма является нормализация данных. Поскольку разные признаки могут иметь различные диапазоны значений, нормализация позволяет привести их к сопоставимому виду.

При выборе значения для k важно помнить, что слишком маленькое значение может привести к переобучению, а слишком большое — к неэффективному моделированию.

Этот метод имеет как свои преимущества, так и недостатки. Простота реализации и отсутствие необходимости в построении сложной модели являются положительными аспектами. С другой стороны, алгоритм может значительно замедляться при большом количестве данных и требовать выбора подходящего значения k, от которого сильно зависит точность модели.

1.2.3 Многослойный перцептрон (MLP)

Многослойный перцептрон (MLP) [4] — это тип искусственной нейронной сети, который состоит из нескольких слоев нейронов, соединенных между собой.

Входной слой принимает входные данные, которые могут быть представлены в виде векторов чисел. Между входным и выходным слоями

располагаются один или несколько скрытых слоев. Каждый нейрон в скрытом слое принимает взвешенные суммы выходов нейронов из предыдущего слоя, применяет к ним нелинейную функцию активации и передает результат на следующий слой. Выходной слой выдает результат работы сети. Количество нейронов в выходном слое зависит от задачи.

MLP обучается на данных с помощью алгоритма обратного распространения ошибки (backpropagation). Суть алгоритма заключается в том, что сеть сравнивает свои предсказания с истинными значениями и корректирует веса связей между нейронами таким образом, чтобы уменьшить ошибку.

Нейрон — основной строительный блок MLP. Каждый нейрон принимает входные данные, применяет к ним линейную комбинацию весов и функции активации, и передает результат на следующий слой.

Веса — это числа, которые определяют силу связи между нейронами. Обучение MLP заключается в поиске оптимальных значений весов и смещений.

Функции активации — это нелинейные функции, которые применяются к результату линейной комбинации весов. Они позволяют MLP моделировать сложные зависимости между входными и выходными данными. Примеры функций активации: сигмоида, ReLU, tanh (гиперболический тангенс).

Преимущество MLP заключается в его универсальности. Многослойный перцептрон может быть использован для решения широкого спектра задач, включающих классификацию, регрессию, кластеризацию. Благодаря функциям активации, MLP способен моделировать сложные нелинейные зависимости в данных, а алгоритм обратного распространения ошибки позволяет эффективно обучать нейросеть на больших объемах данных.

Однако, MLP может быть сложным в настройке и обучении, особенно для больших и сложных задач. Он также может переобучиться на обучающих данных, что приведёт к плохой обобщающей способности на новых данных. Результат работы многослойного перцептрона сильно зависит от выбора оптимальной архитектуры (количество слоев, нейронов в каждом слое, функции активации), что может стать сложной задачей.

1.2.4 Логистическая регрессия

Логистическая регрессия [5] является одним из наиболее широко используемых методов в статистике и машинном обучении для решения задач классификации. Несмотря на свое название, логистическая регрессия используется не для регрессии, а для предсказания вероятностей, которые затем используются для классификации объектов в один из заранее определённых классов. Этот метод особенно полезен в задачах бинарной классификации, где объект должен быть отнесён к одному из двух классов, например, «положительный» или «отрицательный». Однако логистическая регрессия также может быть использована для многоклассовых задач с помощью расширений.

Основной принцип работы логистической регрессии заключается в том, чтобы предсказать вероятность принадлежности объекта к положительному классу. Эта вероятность рассчитывается с помощью специальной функции, называемой логистической функцией, или сигмоидой. Эта функция преобразует линейную комбинацию признаков объекта в значение, которое лежит в пределах от 0 до 1. Модель строится таким образом, чтобы найти значения коэффициентов (весов), которые позволяют минимизировать ошибку предсказания на основе обучающих данных.

Для нахождения оптимальных коэффициентов логистической регрессии используется минимизация функции стоимости, которая оценивает разницу между истинными метками классов и предсказанными значениями. Чем меньше эта разница, тем лучше модель. Чтобы минимизировать функцию стоимости, обычно применяется метод градиентного спуска. Это итеративный процесс, при котором на каждом шаге вычисляется, в каком направлении нужно изменить параметры модели, чтобы уменьшить ошибку. Обновление параметров происходит постепенно, с учётом того, как сильно они влияют на ошибку. Градиентный спуск продолжает обновлять параметры до тех пор, пока модель не сойдётся к оптимальным значениям.

Логистическая регрессия может быть расширена для многоклассовых задач с использованием подходов «один против всех» или «один против одного».

В первом случае для каждого класса строится отдельный бинарный классификатор, который решает, принадлежит ли объект к этому классу или нет. Затем объект классифицируется в тот класс, для которого вероятность будет максимальной.

В методе «один против одного» создаются классификаторы для каждой пары классов, и объект классифицируется на основе голосования этих классификаторов.

Также существует подход с многоклассовой логистической регрессией, который использует функцию активации Softmax и позволяет одновременно учитывать все классы при предсказании.

Преимущества логистической регрессии заключаются в её простоте и интерпретируемости, поскольку это линейная модель. Логистическая регрессия также требует минимальной настройки гиперпараметров, что делает её удобным инструментом для быстрого применения, особенно на небольших данных. Помимо этого, алгоритм имеет относительно низкие вычислительные затраты в сравнении с более сложными моделями, такими как нейронные сети или деревья решений.

Однако логистическая регрессия имеет и свои ограничения. Одним из основных недостатков является предположение о линейной разделимости классов, что означает, что она не будет эффективна, если данные нельзя разделить с помощью прямой или гиперплоскости. В таких случаях другие методы, например, случайные леса или нейронные сети, могут быть более подходящими. Также логистическая регрессия чувствительна к мультиколлинеарности, когда признаки сильно коррелируют друг с другом. Она также плохо справляется с несбалансированными данными, когда один из классов встречается гораздо реже другого. В таких случаях модель может склоняться к классификации всех объектов как принадлежащих к более частому классу.

В целом, логистическая регрессия является хорошим инструментом для решения задач классификации, но с некоторыми ограничениями по применению.

1.2.5 Наивный байесовский классификатор (NBC)

Наивный байесовский классификатор (NBC) [6] — это статистическая модель, используемая для классификации объектов на основе их признаков, с использованием теоремы Байеса. Основная идея модели заключается в том, что для предсказания класса объекта рассматриваются условные вероятности, при этом предполагается, что признаки объекта являются независимыми друг от друга. Это «наивное» предположение часто не соответствует реальности, но оно значительно упрощает расчет и делает модель очень эффективной.

Модель пытается вычислить вероятность того, что объект принадлежит к определенному классу, исходя из вероятностей признаков, которые этот объект имеет. Наивный байесовский классификатор обучается на основе статистики, собранной из обучающих данных, вычисляя вероятности для каждого признака в каждом классе.

Одним из важнейших преимуществ наивного байесовского классификатора является его простота и высокая эффективность при обработке больших объемов данных. Модель обучается быстро и может быть использована в реальных приложениях, где необходимо быстро получить результат. Она также хорошо работает, когда данные имеют большое количество признаков, поскольку несмотря на предположение о независимости признаков, модель часто даёт вполне приемлемые результаты в реальных задачах.

Однако у наивного байесовского классификатора есть и несколько недостатков. Основным из них является его предположение о независимости признаков. В реальных данных признаки часто бывают зависимыми, и это может привести к значительному снижению точности модели, особенно когда зависимости между признаками важны для классификации.

Другим недостатком является то, что модель чувствительна к выбросам и шуму в данных, так как она основывает свои предсказания на статистике, собранной из обучающего набора.

В целом, наивный байесовский классификатор представляет собой мощный инструмент для решения задач классификации, особенно в тех случаях, когда данные можно аппроксимировать гипотезой о независимости признаков или когда требуется быстрое обучение и предсказание.

1.2.6 Дерево решений (Decision tree)

Дерево решений (Decision tree) [7] — это один из наиболее популярных методов машинного обучения, используемый для классификации и регрессии. Оно представляет собой структуру, напоминающую дерево, где каждый узел содержит условие для одного из признаков, а ветви представляют собой возможные исходы этого условия. В конечных узлах дерева находятся решения или предсказания, которые могут быть метками классов для задачи классификации или значениями для задачи регрессии. Алгоритм строит дерево, разделяя данные на подмножества таким образом, чтобы каждый узел в дереве максимально разделял классы или минимизировал ошибку предсказания для задачи регрессии. Этот процесс продолжается до тех пор, пока не будет достигнут критерий остановки, который может быть задан в виде глубины дерева, минимального количества объектов в узле или другого условия.

Одним из главных достоинств деревьев решений является их простота и интерпретируемость. Строение дерева позволяет наглядно увидеть, какие признаки и какие пороги используются для классификации, что делает модель легко объяснимой. Деревья решений не требуют сложной предобработки данных, такой как нормализация или стандартизация, поскольку они могут работать с разными типами признаков, включая категориальные. Кроме того, они могут легко справляться с отсутствующими значениями, так как алгоритм может просто игнорировать недостающие признаки при построении решений.

Однако у деревьев решений есть и несколько значительных недостатков. Одним из них является склонность к переобучению. Дерево решений может стать слишком сложным, если оно будет продолжать разбиение данных до очень глубокой структуры. Это приведет к тому, что модель будет слишком точно соответствовать обучающим данным и плохо обобщать новые данные.

Таким образом, дерево решений — это мощный и интуитивно понятный метод, который отлично работает в ряде задач и обладает рядом преимуществ, таких как высокая интерпретируемость и гибкость.

1.2.7 Случайный лес (Random Forest)

Алгоритм случайного леса (Random Forest) [8, 9] представляет собой метод ансамблевого обучения, который использует множество решающих деревьев для повышения точности предсказаний. В процессе построения модели создается несколько решающих деревьев, каждое из которых обучается на случайной подвыборке данных, взятой с возвращением (bootstrapping). Это означает, что некоторые примеры данных могут использоваться несколько раз, а другие могут быть исключены из обучения каждого дерева. Кроме того, при построении каждого дерева выбирается случайное подмножество признаков, что способствует уменьшению корреляции между деревьями и улучшает общую производительность модели.

Когда требуется сделать предсказание, каждое дерево выдает свои результаты (голосование для классификации и среднее значение для регрессии), и итоговое решение принимается на основе этих результатов. Например, в задаче классификации выбирается класс, получивший наибольшее количество голосов от всех деревьев.

Среди ключевых преимуществ можно выделить высокую точность предсказаний, устойчивость к выбросам, способность обрабатывать данные с пропусками и различными типами признаков.

Однако у алгоритма есть и недостатки. Он требует значительных вычислительных ресурсов, а время обучения может быть дольше по сравнению

с простыми моделями. Алгоритм также склонен к переобучению на зашумленных данных.

В целом, метод случайного леса — это универсальный инструмент для решения задач классификации и регрессии. Его простота и эффективность делают его подходящим для обработки больших объемов данных и извлечения точных результатов.

1.2.8 CatBoost

CatBoost [10] — это один из самых популярных алгоритмов машинного обучения, разработанный для решения задач классификации и регрессии с использованием градиентного бустинга. CatBoost был разработан компанией Yandex. Он представляет собой эффективный инструмент для работы с табличными данными. Одной из ключевых особенностей CatBoost является его способность автоматически обрабатывать категориальные признаки, что делает его удобным инструментом для работы с реальными данными, где часто встречаются такие признаки, как текстовые метки, идентификаторы и другие категориальные переменные.

Алгоритм CatBoost использует метод градиентного бустинга, который заключается в поочередном построении множества слабых моделей (чаще всего решающих деревьев), каждая из которых исправляет ошибки предыдущей модели. На каждом шаге алгоритм добавляет дерево, которое минимизирует ошибку на предыдущем шаге, улучшая итоговую модель. Однако отличие CatBoost от других алгоритмов градиентного бустинга, таких как XGBoost и LightGBM, заключается в его особом подходе к обработке категориальных признаков. CatBoost преобразует категориальные признаки в числовые с помощью алгоритма, который использует статистическую информацию о категориях, избегая необходимости предварительного кодирования с помощью one-hot encoding или других методов. Это делает алгоритм более эффективным и точным, особенно когда происходит работа с большими наборами категориальных данных.

Кроме того, CatBoost обладает рядом уникальных особенностей, которые делают его более устойчивым и точным. Например, он применяет технику поочередного использования случайных перестановок данных для уменьшения переобучения, а также использует специальные методы для борьбы с дисбалансом данных. CatBoost также обладает высокой скоростью обучения. Алгоритм имеет встроенные методы для работы с пропущенными значениями и шумом в данных, что делает его ещё более удобным и устойчивым. Также CatBoost позволяет пользователю увидеть, какие признаки вносят наибольший вклад в предсказания.

Однако, несмотря на все достоинства, CatBoost также имеет некоторые недостатки. Одним из них является необходимость правильной настройки гиперпараметров для получения наилучших результатов. Также, несмотря на свою скорость, CatBoost может быть не так быстр, как некоторые другие алгоритмы, например, LightGBM.

В целом, CatBoost является мощным инструментом для работы с табличными данными, который демонстрирует высокую производительность и точность в самых различных задачах. Его уникальные особенности, такие как автоматическая обработка категориальных признаков и внутренняя оптимизация, делают его предпочтительным выбором для многих задач в реальной жизни.

1.2.9 AdaBoost

AdaBoost [11], или Adaptive Boosting, — это метод машинного обучения, который применяется для улучшения точности классификации, объединяя несколько слабых классификаторов в один более сильный. Основная идея заключается в том, чтобы построить серию классификаторов, каждый из которых исправляет ошибки предыдущего. Процесс начинается с того, что на первой итерации обучается слабый классификатор (чаще всего это решающее дерево небольшой глубины). Все примеры в обучающей выборке на первом шаге имеют одинаковые веса. После того как классификатор сделает свои предсказания, алгоритм

обращает внимание на те примеры, которые он классифицировал неверно. Модель увеличивает веса соответствующих примеров, чтобы на следующем шаге новый классификатор уделял этим примерам больше внимания. Таким образом, каждый последующий классификатор пытается исправить ошибки предыдущих, при этом вес каждого классификатора зависит от того, насколько хорошо он справился с задачей. В финале все эти слабые классификаторы объединяются, и результат предсказания делается с учетом их взвешенных голосов.

AdaBoost не требует большого количества гиперпараметров для настройки, что делает его простым в использовании. Важно также, что метод имеет встроенную защиту от переобучения, если правильно контролировать количество итераций. Помимо этого, он может работать с различными типами базовых классификаторов.

Однако, как и любой алгоритм, AdaBoost имеет свои недостатки. Одним из основных минусов является чувствительность к выбросам и шуму в данных. Когда в обучающей выборке присутствуют ошибки или необычные примеры, алгоритм будет стремиться их исправить, и это может привести к ухудшению качества работы модели. Также, несмотря на свою эффективность, AdaBoost может требовать значительных вычислительных ресурсов, особенно если количество итераций велико и данные очень большие. В таком случае время обучения может значительно увеличиться. Еще одной проблемой является его склонность к плохим результатам на сильно несбалансированных данных, где одна из категорий представлена значительно меньше другой.

В общем, AdaBoost — это мощный инструмент классификации, но его эффективность сильно зависит от качества данных и гиперпараметров.

1.2.10 Gradient Boosting

Gradient Boosting [12] — это метод машинного обучения, который используется для создания модели путем комбинации множества слабых классификаторов. В отличие от AdaBoost, который делает акцент на исправление

ошибок предыдущих классификаторов путем увеличения весов ошибочных примеров, в Gradient Boosting строится последовательность моделей, каждая из которых минимизирует ошибку через градиентный спуск.

Процесс обучения начинается с того, что модель делает предсказания на основе некоторого начального набора. Затем вычисляется ошибка, и новая модель обучается так, чтобы уменьшить эту ошибку. Модели обучаются не на всей выборке данных, а на ошибках, которые были сделаны предыдущими классификаторами. Этот подход отличается от AdaBoost, в котором внимание сосредотачивается на ошибках через увеличение весов неправильно классифицированных примеров. В Gradient Boosting создается новый классификатор, который ориентирован непосредственно на исправление ошибок предыдущих.

Алгоритм работает очень хорошо даже с небольшой настройкой гиперпараметров. Другим большим преимуществом является способность эффективно работать с несбалансированными данными.

Однако, несмотря на свои преимущества, Gradient Boosting имеет и несколько недостатков. Например, метод может быть достаточно медленным. Каждое новое дерево строится последовательно, что требует значительных вычислительных ресурсов и времени. Еще одним важным недостатком является его чувствительность к шуму и выбросам. Поскольку алгоритм ориентирован на минимизацию ошибок, он может начать переобучаться на выбросах в данных, что приводит к ухудшению качества модели.

Таким образом, Gradient Boosting является очень мощным инструментом для решения задач классификации и регрессии, и его высокая точность делает его одним из самых популярных методов. Однако его чувствительность к выбросам, необходимость в значительных вычислительных ресурсах и возможные сложности с настройкой гиперпараметров могут быть серьёзными ограничениями для выполнения некоторых задач.

1.2.11 XGBoost

XGBoost (Extreme Gradient Boosting) [13] — это усовершенствованная версия алгоритма Gradient Boosting, которая нацелена на повышение производительности и эффективности. XGBoost использует подход, основанный на градиентном бустинге, но включает множество улучшений. Как и другие методы бустинга, XGBoost строит ансамбль моделей, каждая из которых исправляет ошибки предыдущих.

Основное улучшение — это добавление регуляризации, которая помогает избежать переобучения модели. Регуляризация контролирует сложность модели, что делает её более стабильной и менее склонной к подстройке под шумные или аномальные данные. Это улучшение помогает значительно повысить обобщающую способность модели.

Кроме того, XGBoost оптимизирует процесс построения деревьев, используя более продвинутые методы, что делает обучение быстрее и более эффективным. В отличие от стандартного градиентного бустинга, который обучает деревья по одному, XGBoost использует процесс поэтапного построения деревьев, что ускоряет время обучения. Также алгоритм имеет встроенную функцию для параллельной обработки. Помимо этого, XGBoost поддерживает работу с пропущенными данными, что делает его удобным для работы с реальными данными. Еще одной важной особенностью является возможность ранней остановки, что позволяет избежать излишних итераций в случае, если модель перестает улучшаться, тем самым экономя время и ресурсы.

Одними из самых главных достоинств является его высокая точность и способность достигать отличных результатов на различных типах данных. Алгоритм может обрабатывать как числовые, так и категориальные данные, что делает его универсальным инструментом для решения широкого круга задач. XGBoost также поддерживает методы кросс-валидации.

В общем, XGBoost — это один из самых мощных и эффективных методов для решения задач классификации и регрессии. Он объединяет в себе сильные стороны градиентного бустинга, добавляя дополнительные возможности.

1.2.12 LightGBM

LightGBM (Light Gradient Boosting Machine) [14] — это библиотека для градиентного бустинга, разработанная Microsoft, которая предназначена для решения задач классификации и регрессии. Основное преимущество LightGBM заключается в способности работать с большими объемами данных и ускоренном обучении при сохранении высокой точности. Он использует улучшенные методы построения деревьев, которые делают обучение более эффективным и быстрым.

Главное отличие LightGBM от традиционного градиентного бустинга и других подобных методов заключается в использовании метода построения деревьев, называемого leaf-wise boosting. В отличие от метода, который строит деревья по уровням, как в XGBoost, LightGBM начинает разбиение с самых глубоких листьев, что позволяет дереву расти быстрее и с большей точностью.

Кроме того, LightGBM использует алгоритм гистограммного обучения, который агрегирует данные в гистограммы для ускорения процесса поиска оптимальных разбиений. Это позволяет значительно сократить количество операций и ускорить обучение, особенно при работе с большими объемами данных.

Также LightGBM имеет встроенную поддержку категориальных признаков, что позволяет эффективно работать с данными, содержащими такие признаки, без необходимости их преобразования в числовые или one-hot закодированные форматы. Это упрощает работу с данными и может привести к улучшению производительности модели, поскольку алгоритм может напрямую использовать категориальные признаки для построения разбиений.

В целом, LightGBM является мощным инструментом для решения задач классификации и регрессии, и часто применяется, когда требуется высокая скорость обучения и обработка больших данных.

1.2.13 Histogram-based Gradient Boosting

Histogram-based Gradient Boosting [15], или гистограммный градиентный бустинг, — это улучшенная версия классического метода градиентного бустинга, разработанная для повышения эффективности при работе с большими данными. Основная идея заключается в том, чтобы ускорить процесс построения деревьев решений, используя гистограммы для дискретизации признаков. Это позволяет значительно уменьшать вычислительные затраты, особенно когда набор данных содержит большое количество признаков и примеров.

Вместо того чтобы рассматривать все возможные пороги для разделения данных, как в стандартном градиентном бустинге, метод гистограммного бустинга агрегирует данные в несколько интервалов. Каждый признак делится на несколько равных интервалов, и затем для каждого интервала вычисляется сводная информация. Это не только ускоряет обучение, но и уменьшает использование памяти, так как хранятся только бины, а не полные данные признаков. Такие гистограммы значительно сокращают сложность вычислений при разбиении признаков, что важно, особенно когда речь идет о больших наборах данных.

Процесс обучения в гистограммном бустинге заключается в последовательном построении деревьев, каждое из которых стремится исправить ошибки предыдущего. В отличие от традиционного градиентного бустинга, где разбиение на основе признаков выполняется по всем возможным значениям, здесь выбираются наиболее информативные бины, что ускоряет обучение.

Достоинством также является поддержка обработки категориальных признаков. В отличие от многих других алгоритмов, гистограммный градиентный бустинг может работать с категориальными признаками напрямую, что упрощает процесс подготовки данных.

Его основным недостатком является ограниченная поддержка распределенных вычислений по сравнению с более мощными фреймворками, такими как XGBoost и LightGBM.

В целом, Histogram-based Gradient Boosting — это мощный и быстрый инструмент для решения задач классификации и регрессии на больших данных.

1.2.14 StackingClassifier

StackingClassifier [16] — это метод, который объединяет несколько моделей с целью улучшения точности предсказаний. В отличие от простых методов, таких как бэггинг или бустинг, где результат определяется на основе голосования или усреднения предсказаний нескольких моделей, здесь используется другой подход: предсказания моделей передаются в новую модель, которая учится на этих предсказаниях и генерирует итоговое предсказание.

Процесс обучения StackingClassifier состоит из двух этапов. На первом этапе обучаются несколько базовых классификаторов. Эти модели могут быть совершенно разными. Каждый из этих классификаторов работает независимо друг от друга, и выдает свое предсказание. После того как все базовые классификаторы обучены, их предсказания становятся входными данными для второй модели — мета-классификатора, который обучается на этих предсказаниях. Мета-классификатор, в свою очередь, пытается комбинировать предсказания базовых моделей и выдавать итоговое предсказание, которое, как правило, более точное, чем предсказания отдельных классификаторов.

Благодаря возможности сочетания различных типов моделей, StackingClassifier эффективно предотвращает переобучение, так как базовые модели компенсируют ошибки друг друга.

Тем не менее, у StackingClassifier есть и некоторые недостатки. Во-первых, этот метод может быть вычислительно дорогим. Требуется обучить несколько базовых моделей, а затем еще одну модель, которая будет работать на их предсказаниях. Этот процесс может занять значительное время и потребовать значительных вычислительных ресурсов.

Тем не менее, StackingClassifier представляет собой мощный инструмент, который может значительно повысить точность предсказаний, комбинируя несколько моделей. Однако для эффективного использования этого метода

требуется достаточно большое вычислительное время и правильная настройка мета-классификатора.

1.2.15 Voting Classifier

Voting Classifier [17] — это метод ансамблевого обучения, который объединяет несколько моделей для принятия окончательного решения о классификации.

Существует два типа объединения предсказаний моделей Voting Classifier: hard voting и soft voting.

При hard voting итоговое решение принимается на основе большинства голосов, то есть класс, который выбирает большее количество моделей, становится предсказанием итоговой модели.

При soft voting результат определяется через усреднение вероятностей, предсказанных моделями, и класс с наибольшей вероятностью становится окончательным решением. Soft voting, как правило, дает более точные результаты.

Основным достоинством Voting Classifier является способность уменьшать влияние ошибок, которые могут быть свойственны отдельным моделям. Это делает Voting Classifier менее чувствительным к переобучению, поскольку ансамбль моделей может лучше обобщать данные и справляться с шумом. Другим важным преимуществом Voting Classifier является его простота в применении. Этот метод не требует сложных алгоритмов или процедур для объединения моделей.

Однако, основным недостатком является то, что Voting Classifier требует много вычислительных ресурсов, так как нужно обучить несколько моделей, а затем комбинировать их предсказания. Для этого необходимо большее количество времени на обучение и большее количество памяти, чем при использовании одной модели.

Таким образом, Voting Classifier — это полезный метод для повышения точности классификации, особенно в случаях, когда есть несколько сильных моделей, которые могут взаимно дополнять друг друга.

2 Практическая часть

2.1 Загрузка и первичная обработка данных

Первым шагом в процессе разработки тестового программного обеспечения является загрузка с GitHub и первичная обработка данных для дальнейшего анализа и применения методов машинного обучения. Процесс начинается с загрузки архивированного файла, который содержит набор данных полетов БПЛА с пятью разными состояниями.

Первоначально данные загружаются с указанного URL с помощью библиотеки `requests`, которая отправляет запрос на скачивание файла в формате ZIP. Этот файл содержит несколько подкаталогов, каждый из которых включает в себя данные, относящиеся к различным экспериментам. После получения данных с интернета файл сохраняется в оперативной памяти с использованием библиотеки `io`, и далее архив распаковывается с помощью `zipfile.ZipFile`. Извлеченные файлы сохраняются в указанной директории, что позволяет приступить к дальнейшей обработке.

На следующем этапе создается папка `ProcessedDataset`. Далее начинается обработка каждого подкаталога в структуре исходных данных. Каждый подкаталог представляет собой отдельный эксперимент, и данные в этих папках разделены по классам.

Для каждого из таких подкаталогов создается свой новый каталог с названиями, соответствующими количеству неисправных винтов — «0», «1», «2», «3», и «4», в папке `ProcessedDataset`, где будут храниться обработанные данные. Внутри каждого подкаталога осуществляется дальнейшая работа с файлами. Каждый файл в этих папках представляет собой набор данных, сохраненных в формате JSON Lines (JSONL). Каждый такой файл содержит строки данных, представленных в формате JSONL, что позволяет загружать их построчно для дальнейшей обработки.

После открытия и чтения содержимого файлов JSONL, данные каждой строки парсятся в Python-объект с использованием библиотеки `json`. Полученные данные собираются в список, который затем преобразуется в структуру данных `Pandas DataFrame`.

На этом этапе данные, содержащие значения угловых скоростей гироскопа и ускорений акселерометра, подвергаются дополнительному разбиению. Для каждого элемента в колонках, содержащих данные гироскопа (`gyro_rad`) и акселерометра (`accelerometer_m_s2`), создаются новые колонки, в которые распаковываются значения по осям X, Y и Z. После этого ненужные колонки удаляются, включая оригинальные данные о гироскопе и акселерометре, а также дополнительные метки времени и флаги, такие как `accelerometer_clipping`. Это позволяет оставить только наиболее важную информацию для анализа.

Затем выполняется переименование оставшихся колонок для улучшения читаемости и соответствия стандартам, принятым в модели. Колонка времени (`timestamp`) переименовывается в `time`, а также изменяются имена других столбцов, связанных с интеграцией данных гироскопа и акселерометра. После всех преобразований данные сохраняются в формате CSV для последующего использования.

Процесс обработки повторяется для всех файлов в подкаталоге, при этом каждому файлу присваивается уникальное имя, соответствующее его порядковому номеру и классу. После завершения обработки всех файлов одного подкаталога все CSV-файлы с данными для каждого класса собираются в единую таблицу. Эти данные объединяются с помощью функции `concat` из библиотеки `Pandas`, что позволяет создать единый файл с полными данными для каждого класса. Этот файл сохраняется в соответствующем подкаталоге своего класса.

Для проверки корректности выполнения обработки данных, загружается один из итоговых файлов, и выводятся первые несколько строк, как показано на рисунке 5. Это позволяет убедиться в том, что данные были успешно обработаны

и теперь готовы к использованию в дальнейшем анализе и обучении моделей машинного обучения.

[4]:

	time	gInt	aInt	gx	gy	gz	ax	ay	az
0	947975062	4998	4998	0.008529	0.004348	-0.001269	0.145165	-0.034807	-9.828311
1	947976312	4996	4996	0.001431	0.005915	-0.001310	0.116024	0.020439	-9.769987
2	947983469	4998	4998	0.016173	0.007350	0.000573	0.092866	0.060234	-9.827328
3	947996546	4994	4994	-0.001588	0.000100	-0.003505	0.065213	0.016342	-9.804899
4	947995456	4995	4995	0.008024	0.012801	0.000042	0.082880	0.064342	-9.800008

Рисунок 4. Обработанные данные с датчиков.

Таким образом, на этом этапе выполняется детальная подготовка и обработка исходных данных, которая включает в себя извлечение, фильтрацию, преобразование и сохранение данных в нужном формате, готовом для дальнейшего анализа.

2.2 Анализ данных

На втором этапе работы проводится анализ данных, который включает в себя вычисление частоты дискретизации и построение спектрограмм.

Поскольку частота дискретизации для акселерометра и гироскопа не была указана в исходных данных, необходимо вычислить её самостоятельно. Для этого в ходе обработки данных для каждого класса и файла извлекаются временные метки, которые затем сортируются по времени. Для каждой пары соседних временных меток вычисляется разница во времени, и на основе этих различий оценивается среднее время между измерениями. Это значение затем используется для расчёта частоты дискретизации, которая выражается как обратная величина среднего времени между соседними измерениями. После выполнения этих вычислений получается значение частоты дискретизации, которое составляет ровно 195 Гц. Это значение используется при анализе спектрограмм и определяет, с какой частотой поступают данные с датчиков.

Следующим этапом является построение 3D-спектрограмм для каждого класса данных. Для этого считываются данные из CSV файлов, содержащих измерения акселерометра и гироскопа. Далее для данных, соответствующим оси Z акселерометра, рассчитывается новое значение путем добавления смещения 9.81 для более корректного отображения графика.

Процесс извлечения необходимой информации для построения спектрограммы начинается с деления данных на перекрывающиеся сегменты (наложение составляет 50%) фиксированной длины (в данном случае 256 точек). Далее все строки данных в сегменте сортируются по времени и проверяется условие, что этот сегмент не лежит на стыке данных двух разных полетов — конца первого и начала второго. Если все в порядке, то сегмент обрабатывается с использованием окна Блэкмана-Наталла [18, 19, 20], чтобы минимизировать эффекты утечек спектра, и применяется быстрое преобразование Фурье (БПФ) [21] для получения частотных составляющих. Результат преобразования отображает амплитуду сигнала на различных частотах для каждого сегмента во времени.

Для всех сегментов данных, полученных с использованием БПФ, сохраняются амплитудные значения для каждой из анализируемых частот, которые затем используются для построения 3D-графиков. Эти графики отображают изменение амплитуды сигнала в зависимости от времени и частоты. Оси на графиках отображают номер окна (время) по оси X, частоту по оси Y и амплитуду по оси Z. Такие спектрограммы позволяют анализировать, как изменяются частотные компоненты сигналов с течением времени для каждого канала датчиков. Примеры спектрограмм для каждой из осей акселерометра и гироскопа для классов 0 и 4 приведены на рисунках 5 и 6 соответственно. По ним можно заметить заметное увеличение амплитуды для всех трех осей акселерометра и появление дефектных частот также для всех трех осей гироскопа в диапазоне примерно от 10 Гц. В дальнейшем используем эту информацию для извлечения признаков из частотной области.

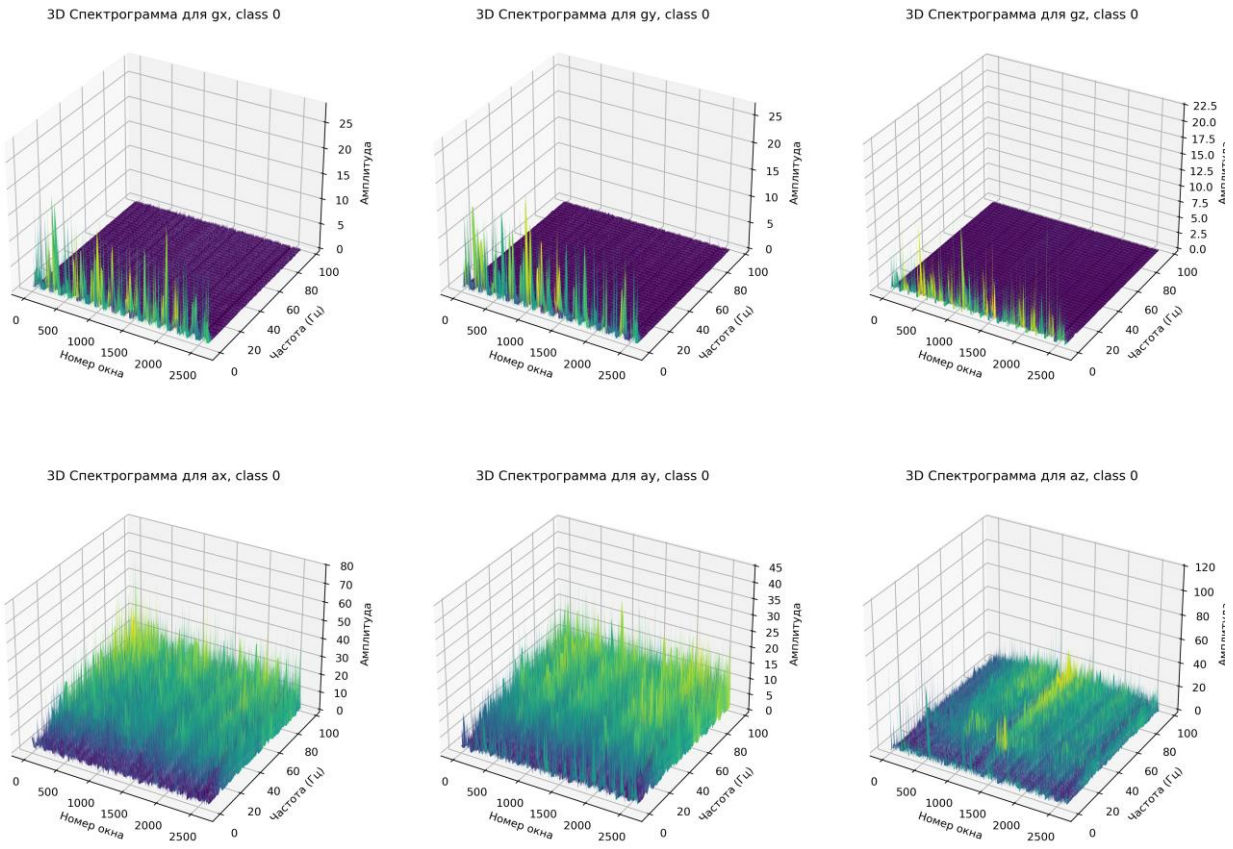

Рисунок 5. 3D-спектрограмма для класса «0».

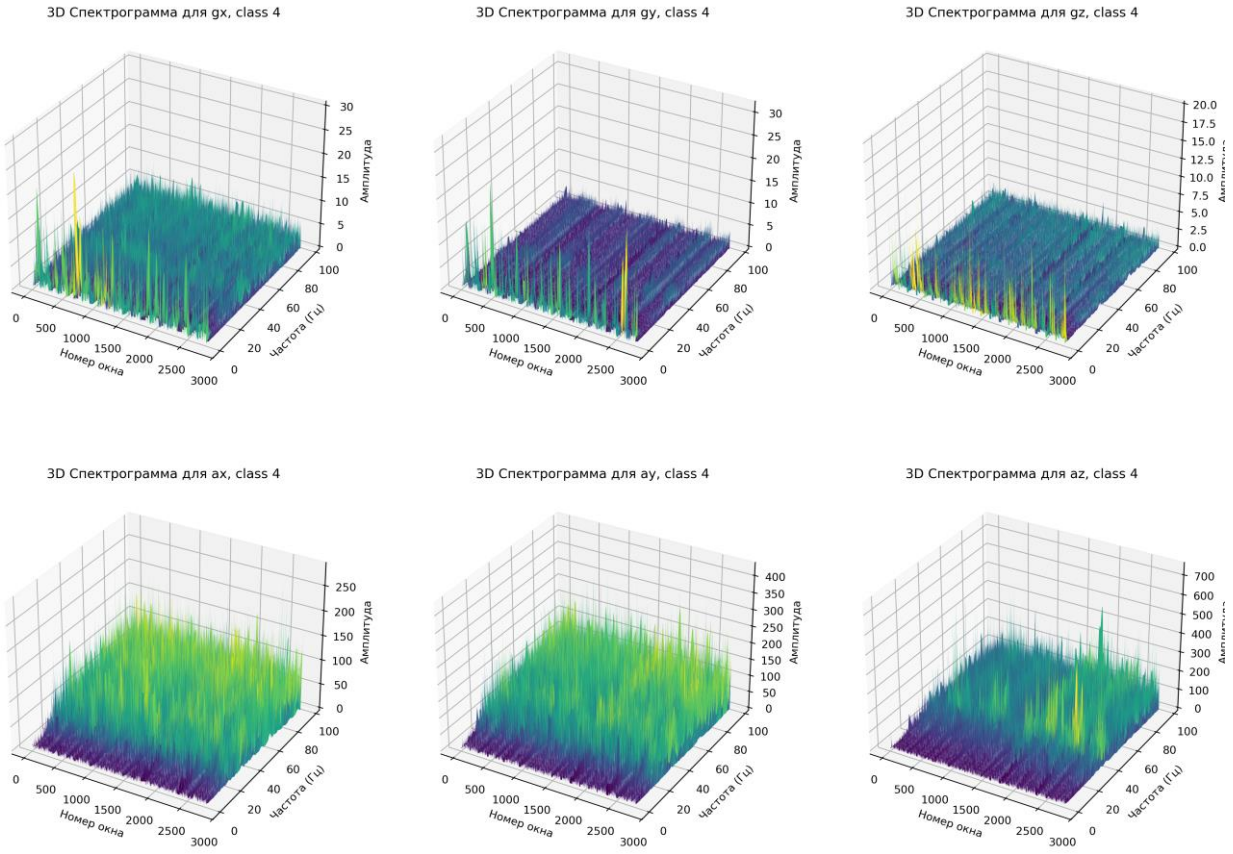

Рисунок 6. 3D-спектрограмма для класса «4».

2.3 Извлечение признаков

Третьим шагом является извлечение признаков. Данный процесс представляет собой преобразование сырых данных в числовые характеристики, которые могут быть использованы для дальнейшего анализа и машинного обучения. Извлечение признаков – это ключевая часть работы с данными, так как оно позволяет преобразовать сложные многомерные временные ряды в более простые и информативные данные, которые могут быть интерпретированы моделью. На данном этапе используется комбинация методов, как во временной, так и в частотной области, чтобы получить полное представление о характере сигналов акселерометров и гироскопов.

Для начала создается список всех признаков, которые будут извлекаться из данных для каждого канала. Признаки делятся на две основные категории:

- Признаки во временной области (ВО) — это статистические и физические характеристики, которые извлекаются непосредственно из значений сигналов, таких как амплитуды, процентили и другие параметры.
- Признаки в частотной области (ЧО) — это характеристики, которые вычисляются после преобразования сигналов в частотную область с помощью быстрого преобразования Фурье (БПФ). Признаки частотной области позволяют выявить особенности сигнала, которые не видны в временной области, например, информацию о частотах, энергию спектра и другие аспекты сигнала.

Выбранные признаки для временной области включают медиану, среднее значение, стандартное отклонение (STD), максимальное и минимальное значение, процентильные значения (25-й, 75-й, 90-й), куртозис, асимметрию и энергию [22, 23]. Также рассчитываются такие характеристики, как коэффициент вариации (отношение стандартного отклонения к среднему) и количество пиков или выбросов (значения, превышающие определенный порог).

Большинство признаков для частотной области аналогичны, но вместо анализа временных данных мы рассматриваем амплитуды на различных

частотах, полученные после применения БПФ. Здесь исследуются такие характеристики, как энергия на частотах больше 10 Гц, а также различные статистические параметры для спектра сигнала, такие как медиана, стандартное отклонение и т. д.

После того как определены признаки, которые необходимо извлечь, начинается процесс работы с данными. Для каждого класса данных (всего существует 5 классов) загружаются CSV файлы, содержащие данные для каждого сегмента. Важно отметить, что перед обработкой данные сортируются по времени и проверяется условие, что сегмент не лежит на стыке данных двух разных полетов — конца первого и начала второго, чтобы устранить возможные ошибки.

Кроме того, каждый файл представляет собой большое количество данных, которые нужно разбить на более мелкие сегменты. Для этого используется метод оконного преобразования. Данные делятся на сегменты фиксированного размера, при этом сегменты перекрываются друг с другом (с наложением 50%), что позволяет получить более детальную информацию о динамике сигналов.

Для обработки сигналов используется оконная функция Блэкмана-Наталла. Это окно применяется для каждого сегмента данных, чтобы минимизировать эффекты утечек спектра при вычислении БПФ. Окно Блэкмана-Наталла позволяет плавно снижать амплитуду сигналов на краях сегмента, что делает спектральный анализ более точным.

После того как окно применяется к данным, они подвергаются преобразованию Фурье. БПФ преобразует временные данные в частотные, разделяя сигнал на различные частотные компоненты. Применяя БПФ, получаем амплитуды для каждого частотного компонента, которые затем будут использованы для извлечения признаков частотной области.

Рассмотрим признаки, которые извлекаем из осей акселерометра и гироскопа.

Медиана, среднее, стандартное отклонение, максимальное и минимальное значение амплитуды, которые дают общее представление о распределении

амплитуд на различных частотах. Медиана и среднее значение показывают центральную тенденцию амплитуд, в то время как стандартное отклонение показывает, насколько сильно амплитуды колеблются вокруг среднего значения.

Процентильные значения (25-й, 75-й и 90-й), которые описывают, как распределяются амплитуды на различных частотах. Процентильные значения помогают понять, какой процент амплитуд находится ниже или выше определенного значения.

Куртозис — это статистическая мера, которая характеризует «пик» распределения данных, а также «хвосты» распределения. Проще говоря, куртозис позволяет определить, насколько распределение отклоняется от нормального в плане пиковости и тяжести хвостов.

Высокий куртозис указывает на то, что распределение данных имеет более острый пик и более тяжелые хвосты по сравнению с нормальным распределением. Это означает, что данные имеют больше выбросов (экстремальных значений), чем в нормальном распределении.

Низкий куртозис означает, что распределение более плоское, с более тонкими хвостами. В таком случае данные имеют меньше экстремальных значений и более равномерно распределены.

Асимметрия — это статистическая мера, которая описывает симметричность распределения данных относительно среднего значения. Она позволяет понять, насколько распределение сдвинуто в одну из сторон, то есть имеет ли оно "длинный хвост" влево или вправо.

Положительная асимметрия (или "правая асимметрия") означает, что распределение имеет длинный хвост справа, т.е. значения часто оказываются выше среднего, но есть редкие экстремальные выбросы в правую сторону. Это указывает на то, что большинство значений сосредоточено в нижней части диапазона, а высокие значения встречаются реже, но значительно выше среднего.

Отрицательная асимметрия (или "левая асимметрия") означает, что распределение имеет длинный хвост слева, т.е. данные имеют редкие экстремальные значения, которые сильно ниже среднего. В этом случае

большинство значений находится в верхней части диапазона, а низкие значения встречаются реже, но значительно ниже среднего.

Энергия вычисляется как интеграл от амплитуды в квадрате. Это позволяет оценить, сколько энергии содержится в каждой частотной компоненте сигнала.

Коэффициент вариации (КВ) — это статистический показатель, который используется для оценки относительной изменчивости данных. Он показывает, насколько велико стандартное отклонение в сравнении со средней величиной. В контексте анализа сигналов КВ может быть полезен для того, чтобы понять степень нестабильности или колебания амплитуды в сигнале. Чем выше коэффициент вариации, тем более изменчивым и нестабильным является сигнал.

Количество пиков (выбросов) — это количество участков данных, где амплитуда значительно превышает нормальные значения для данного сигнала. Наличие таких пиков может быть связано с экстренными событиями или системными аномалиями.

Также извлекаются дополнительные признаки, связанные с частотами выше 10 Гц. Для этого выбираются только те частоты, которые превышают этот порог, и для них рассчитываются статистические характеристики: средняя амплитуда, стандартное отклонение и энергия.

После обработки данных в частотной области извлекаются признаки для временной области. Эти признаки аналогичны признакам для частотной области и включают медиану, среднее, стандартное отклонение, максимальные и минимальные значения, процентильные значения (25-й, 75-й и 90-й), куртозис, ассиметрию, коэффициент вариации и количество пиков.

Для оценки плотности мощности сигнала используется метод `Welch`. Этот метод позволяет получить информацию о мощности сигнала в разных частотных диапазонах и выявить, какие частоты вносят наибольший вклад в общий сигнал. Из данных, полученных после работы функции welch, извлекаются два признака: общая и средняя мощность.

После того как все признаки для каждого сегмента и каждого класса были извлечены, они добавляются в таблицу признаков. Для каждого сегмента создается строка с извлеченными признаками и целевым классом, и эта строка добавляется в итоговый DataFrame, который в конечном итоге сохраняется в CSV файл.

В финальном CSV файле каждая строка соответствует одному сегменту данных, а столбцы содержат все извлеченные признаки. Суммарно их 186 — по 31 признаку для каждой из осей акселерометра и гироскопа. Файл с извлеченными признаками будет использоваться в дальнейшем для обучения моделей.

Таким образом, извлечение признаков – это процесс преобразования исходных временных рядов в набор числовых характеристик, которые могут быть использованы для классификации, прогнозирования или других видов анализа. В этом шаге учитываются как временные, так и частотные аспекты сигналов, что позволяет более глубоко понять поведение системы и выявить скрытые закономерности, которые могут быть полезны для дальнейшего моделирования и предсказания.

2.4 Построение, тестирование и сравнение моделей

Построение, тестирование и сравнение моделей машинного обучения — это ключевые этапы в процессе анализа данных, направленные на создание эффективных предсказательных моделей и их оценку. Эти действия включают несколько ключевых шагов, от загрузки и подготовки данных до выбора, обучения и тестирования различных моделей с последующей их оценкой.

На первом этапе необходимо загрузить данные. Эти данные содержат уже извлеченные признаки, которые будут использованы для обучения. В данном случае, данные загружаются из CSV-файла, и затем создаются переменные для признаков и целевой переменной (класса). Признаки хранятся в матрице X, а целевая переменная (или метки классов) в векторе y.

Следующий шаг — это разделение данных на обучающую и тестовую выборки. Обычно данные делятся на две части: одна используется для обучения модели (обучающая выборка), а другая — для оценки ее эффективности (тестовая выборка). В данном случае, данные делятся с использованием функции `train_test_split`, при этом 20 процентов данных резервируются для тестирования. Важно установить параметр `random_state`, чтобы результаты были воспроизводимыми.

После разделения данных на обучающую и тестовую выборки необходимо провести предварительную обработку. Она включает в себя несколько этапов преобразования признаков, для улучшения качества обучения моделей. Во-первых, применяется метод преобразования Yeo-Johnson, который позволяет привести данные к более нормальному распределению. Это важно для некоторых моделей машинного обучения, которые чувствительны к распределению данных. Далее данные подвергаются стандартизации. Этот процесс необходим для моделей, чувствительных к масштабу признаков, например, для моделей, использующих расстояния между точками (например, K ближайших соседей). Стандартизация гарантирует, что все признаки будут иметь одинаковый масштаб, что помогает моделям обучаться более эффективно. Например, если есть признаки с разными единицами измерения, то без стандартизации признаки с большими значениями могут "доминировать" в модели. В следующем шаге проводится нормализация данных с помощью MinMaxScaler, которая ограничивает значения признаков диапазоном от 0 до 1 что также может быть полезно для алгоритмов, чувствительных к масштабу данных, например, нейросетей.

Если алгоритм чувствителен к масштабу признаков и предполагает нормальность данных, можно использовать и стандартизацию, и приведение данных к нормальному распределению одновременно. Например, для линейной регрессии данные часто стандартизируют, чтобы привести их к единому

масштабу, а затем проводят приведение к нормальному распределению, чтобы улучшить работу модели, если остатки отклоняются от нормальности.

Стоит отметить, что сначала происходит анализ и вычисление параметров преобразования (например, параметры для стандартизации, такие как среднее и стандартное отклонение, параметры для нормализации, такие как минимальные и максимальные значения, или для приведения к нормальному распределению) только на тренировочных данных. Затем, когда преобразование завершено, эти параметры фиксируются и применяются к тестовым данным.

Это нужно для того, чтобы избежать утечки информации (data leakage), когда информация из тестовой выборки используется в процессе обучения модели. Если преобразование сначала будет применено на тестовых данных, модель может "увидеть" информацию, которая должна быть скрыта, что приведет к завышенной точности и некорректной оценке производительности. Поэтому все преобразования должны быть выполнены на тренировочных данных, а затем использоваться для обработки тестовых данных, чтобы сохранить чистоту эксперимента и корректность результатов.

После того как данные подготовлены, необходимо выбрать подходящие модели машинного обучения, которые будут обучаться на тренировочных данных и затем оцениваться на тестовых данных. В данном случае используется широкий спектр моделей, каждая из которых имеет свои особенности и преимущества. Рассмотрим каждый из выбранных типов моделей.

K-ближайших соседей (KNN) — это алгоритм, который классифицирует объекты на основе их близости к соседям в пространстве признаков. Он не строит явную модель, а просто использует данные для поиска наиболее схожих примеров, что делает его простым и эффективным в некоторых задачах, особенно при наличии четких классов.

Наивный байесовский классификатор (Naive Bayes) — это статистический метод, который предполагает независимость признаков между собой и использует теорему Байеса для предсказания вероятности принадлежности объекта к классу.

Метод опорных векторов (SVM) — это мощный метод классификации, который находит гиперплоскость, разделяющую данные с максимальным зазором. SVM особенно хорош в задачах с высокоразмерными данными, например, в задачах классификации с большим количеством признаков.

Логистическая регрессия — это статистическая модель, которая используется для предсказания вероятности принадлежности объекта к одному из двух классов. Логистическая регрессия работает хорошо, если данные линейно разделимы. Она может быть расширена для многоклассовых задач.

Дерево решений — алгоритм, который принимает решение путем построения дерева, где каждый узел представляет собой проверку одного признака, а ветви — возможные результаты. Этот метод часто используется для классификации и регрессии и может быть легко интерпретируемым.

Многослойный перцептрон (MLP) — это искусственная нейронная сеть, которая состоит из нескольких слоев нейронов. MLP хорошо работает с данными, содержащими сложные нелинейные зависимости.

AdaBoost — это метод, который улучшает слабые классификаторы путем комбинирования их предсказаний. Он назначает больший вес ошибочным примерам, что помогает моделям сосредоточиться на сложных, трудно классифицируемых объектах.

Gradient Boosting — это метод ансамблирования, который строит серию слабых моделей (обычно деревьев решений), каждая из которых пытается исправить ошибки предыдущей.

Histogram-based Gradient Boosting (HGB) — это оптимизированная версия Gradient Boosting, которая работает быстрее и более эффективно с большими наборами данных. Алгоритм использует гистограммы для улучшения быстродействия при построении моделей.

XGBoost — это одна из самых популярных и эффективных реализаций градиентного бустинга, которая включает множество оптимизаций для скорости и точности. Этот метод широко используется в соревнованиях по машинному обучению, благодаря своей мощности и высокой производительности.

CatBoost — это еще одна эффективная реализация градиентного бустинга, но с особенностями, которые делают его более удобным для работы с категориальными данными. CatBoost также включает в себя различные улучшения, позволяющие достичь отличных результатов без необходимости в настройке гиперпараметров.

LightGBM — это еще одна популярная и мощная библиотека для градиентного бустинга, которая специализируется на обработке больших данных. LightGBM использует более быстрые алгоритмы для построения деревьев и оптимизации вычислений.

Stacking Classifier — это метод, который использует несколько моделей в качестве базовых классификаторов, а затем комбинирует их выводы с помощью мета-модели (обычно это логистическая регрессия). Stacking Classifier позволяет эффективно использовать сильные стороны разных моделей, повышая точность предсказаний.

Voting Classifier — это метод, при котором несколько классификаторов делают свои предсказания, а итоговое решение принимается на основе голосования. В случае soft voting, предсказания вероятности каждого классификатора комбинируются, а в случае hard voting — модель голосует за класс, который был предсказан большинством классификаторов. Voting Classifier увеличивает общую точность, поскольку позволяет использовать «коллективный разум» нескольких моделей.

Классические методы обычно быстрее обучаются и интерпретируются, но могут не справляться с более сложными зависимостями в данных. Ансамблевые методы, напротив, часто могут значительно улучшить точность за счет сочетания различных моделей и их сильных сторон, но при этом они требуют больше вычислительных ресурсов и времени для обучения, поэтому стоит заранее определиться, какой из параметров предпочтительнее в решении конкретной задачи.

Стоит отметить, что для подбора оптимальных гиперпараметров моделей был использован метод Grid Search с кросс-валидацией. Этот подход позволяет

исследовать пространство гиперпараметров, определяя наилучшие значения, которые минимизируют ошибку на тестовых данных. В процессе выполнения данного этапа была настроена сетка возможных значений для различных параметров, таких как регуляризация, выбор ядра, параметры, связанные с метриками расстояния и другие, соответствующими используемой модели. Кросс-валидация обеспечила надежную оценку производительности модели на различных поднаборах данных, что поспособствовало получению стабильных и точных результатов. Метод Grid Search позволил эффективно настроить модели и выбрать оптимальные параметры, обеспечивающие наилучшую точность классификации.

В следующем этапе происходит обучение моделей. Каждая модель из списка обучается на тренировочных данных с помощью метода `fit`, где в качестве входных данных подаются матрица признаков `X_train` и вектор целевых переменных `y_train`. После этого модель используется для предсказания меток классов на тестовых данных с помощью метода `predict`. Полученные предсказания затем сравниваются с реальными метками классов из тестовой выборки `y_test`.

Stacking Classifier:

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	528
1.0	0.98	0.99	0.98	561
2.0	0.96	0.96	0.96	538
3.0	0.94	0.94	0.94	582
4.0	0.96	0.96	0.96	550
accuracy			0.97	2759
macro avg	0.97	0.97	0.97	2759
weighted avg	0.97	0.97	0.97	2759

Accuracy: 96.59 %

Рисунок 7. Результат тестирования обученной модели Stacking Classifier.

Для оценки качества модели используются несколько показателей. Во-первых, выводится отчет о классификации (на рисунке 7 приведен пример для Stacking Classifier), который включает такие метрики, как Precision, Recall, F1-score, для каждого класса, которые позволяют более подробно оценить, как хорошо модель работает для разных категорий.

Также отдельно выводится общая точность модели (Accuracy), которая представляет собой долю правильных предсказаний среди всех сделанных, и является одной из самых популярных метрик для классификации. Эти показатели дают полное представление о производительности каждой модели.

Кроме того, после обучения и тестирования всех моделей важно сравнить их результаты. Это позволит выбрать наиболее эффективную модель для дальнейшего использования. В данном случае, ключевой метрикой для оценки является Accuracy, которая показывает, сколько правильных предсказаний модель сделала по сравнению с общим числом предсказаний. Для визуализации результатов используется горизонтальная столбчатая диаграмма, которая представлена на рисунке 8.

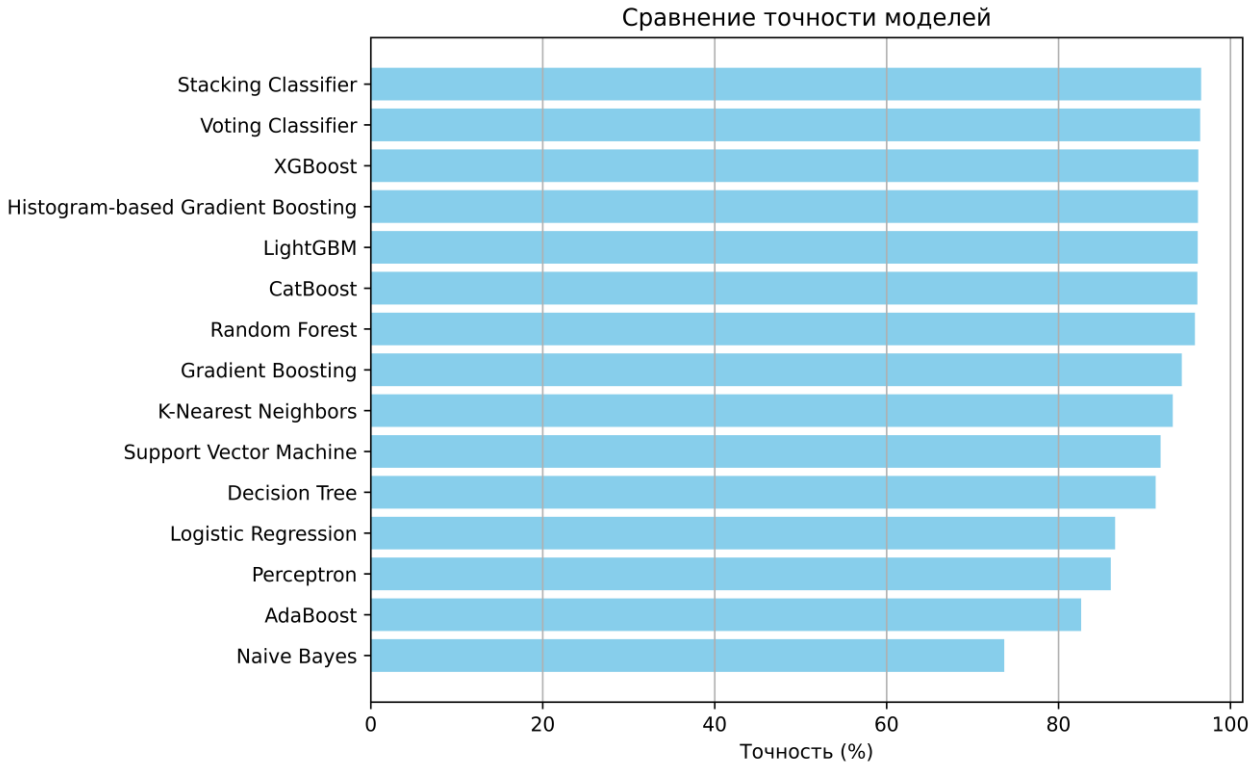

Рисунок 8. Диаграмма точности моделей

Такая диаграмма позволяет наглядно сравнить точности различных моделей. На оси X откладываются значения точности, а на оси Y — названия моделей. Этот график дает наглядное представление о том, какая модель наиболее точна, что помогает сделать выбор модели, которая будет использоваться для решения задачи.

Для оценки и сравнения моделей классификации важно учитывать несколько метрик, которые дают полное представление о их производительности.

Accuracy измеряет, какой процент всех предсказаний был правильным. Однако эта метрика может быть недостаточной. В таком случае важными становятся другие метрики, представленные ниже.

Precision измеряет долю правильно классифицированных положительных примеров среди всех примеров, которые модель классифицировала как положительные. Это особенно важно, когда важно минимизировать количество ложноположительных предсказаний.

Recall показывает, какую долю всех реальных положительных примеров модель правильно предсказала. Она необходима, когда важно не пропустить важные положительные примеры, даже если это приведет к большему числу ложных срабатываний.

F1-score является гармоническим средним между точностью и полнотой, что позволяет получить сбалансированную оценку, особенно в случае, когда точность и полнота сильно различаются. Она используется, когда важно учитывать как ошибочные отрицательные, так и положительные предсказания.

После вычисления этих метрик для каждой модели, результаты сохраняются в словарь, где для каждой модели хранится Accuracy, Precision, Recall и F1-score. Эти данные затем преобразуются в таблицу (DataFrame), что позволяет их удобно анализировать. Далее таблица сортируется по точности, чтобы можно было легко определить, какая модель работает наилучшим образом с точки зрения общего процента правильных предсказаний. Результат оценки моделей по четырем метрикам приведен на рисунке 9.

	Accuracy	Precision	Recall	F1 Score
Stacking Classifier	96.593	96.595	96.593	96.593
Voting Classifier	96.484	96.486	96.484	96.484
XGBoost	96.267	96.270	96.267	96.267
Histogram-based Gradient Boosting	96.231	96.229	96.231	96.230
LightGBM	96.194	96.199	96.194	96.197
CatBoost	96.158	96.155	96.158	96.154
Random Forest	95.868	95.857	95.868	95.860
Gradient Boosting	94.346	94.354	94.346	94.347
K-Nearest Neighbors	93.295	93.292	93.295	93.291
Support Vector Machine	91.881	91.874	91.881	91.874
Decision Tree	91.301	91.281	91.301	91.280
Logistic Regression	86.589	86.625	86.589	86.592
Perceptron	86.082	87.877	86.082	86.311
AdaBoost	82.639	87.389	82.639	80.943
Naive Bayes	73.686	76.717	73.686	73.304

Рисунок 9. Таблица результатов оценки моделей по четырем метрикам.

Такой подход дает возможность не только оценить, какая модель лучше по точности, но и выявить модели, которые демонстрируют лучшие результаты по другим важным метрикам, что поможет сделать более взвешенный выбор для конкретной задачи.

Наиболее точные предсказания осуществляет модель Stacking Classifier, структура которой представлена на рисунке 10.

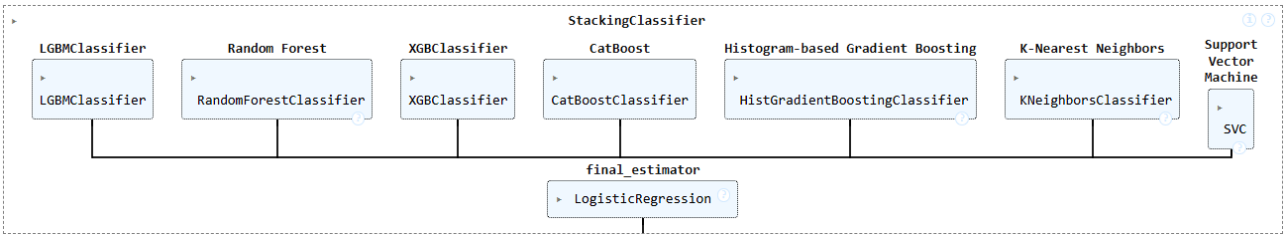

Рисунок 10. Структура модели Stacking Classifier.

Чтобы более наглядно показать, как модель работает при предсказаниях, часто используют матрицу ошибок. Это инструмент, который помогает детально

оценить качество классификации, давая информацию о том, сколько объектов модель правильно или неправильно классифицировала. Она показывает, как модель предсказывает каждый класс относительно истинных меток, что особенно полезно при работе с несбалансированными данными, где одни классы могут быть представлены гораздо реже, чем другие.

Матрица ошибок позволяет не только оценить общую точность работы модели, но и выявить, в каких случаях она ошибается. Например, если модель часто путает два определенных класса, это может быть сигналом, что данные требуют доработки или что модель нуждается в улучшении. Также с помощью матрицы ошибок можно увидеть, насколько стабильно модель работает для разных классов, и в случае необходимости изменить алгоритм или добавить веса для редких классов. Матрица ошибок для классификатора Stacking Classifier приведена на рисунке 11.

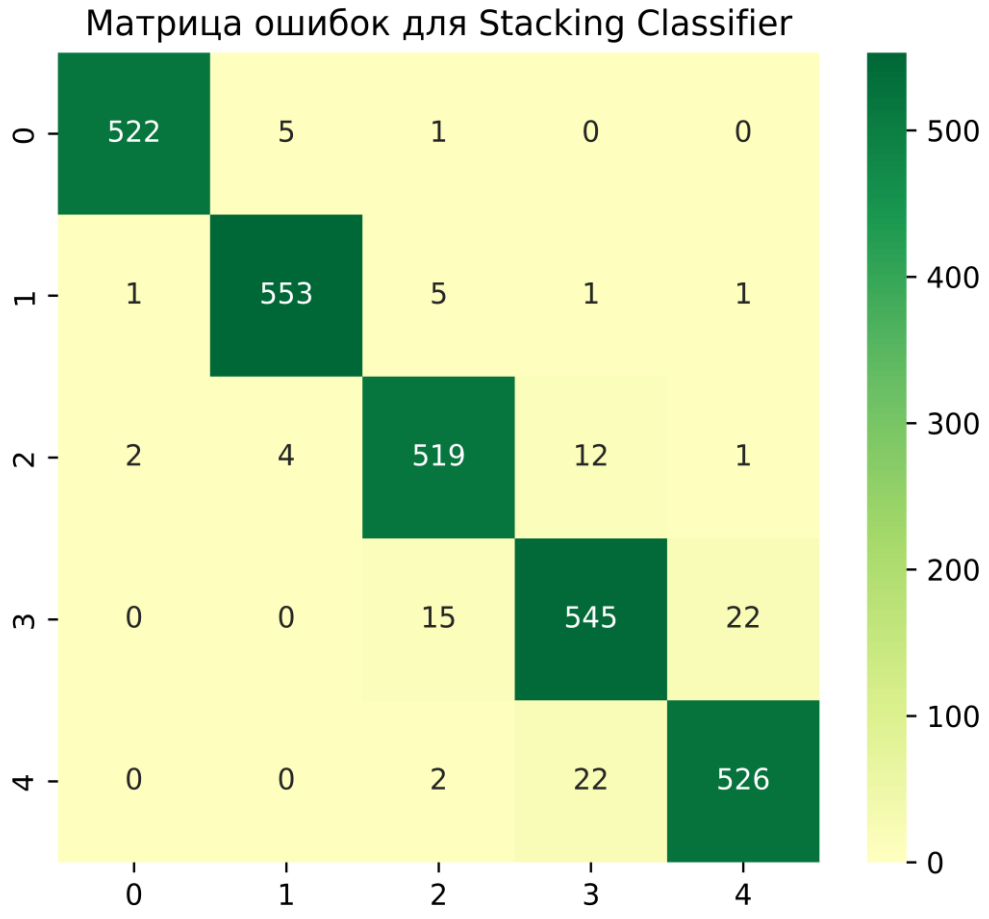

Рисунок 11. Матрица ошибок для классификатора Stacking Classifier.

Для более ясного понимания того, как модель выполняет предсказания, полезно использовать ROC-кривую. Она помогает оценить эффективность классификационной модели, показывая, как изменяются показатели точности и ошибки при разных порогах классификации. ROC-кривая отображает зависимость между чувствительностью (True Positive Rate, TPR), то есть долей правильно классифицированных положительных объектов, и ложноположительным результатом (False Positive Rate, FPR), который показывает долю объектов, ошибочно отнесенных к положительному классу.

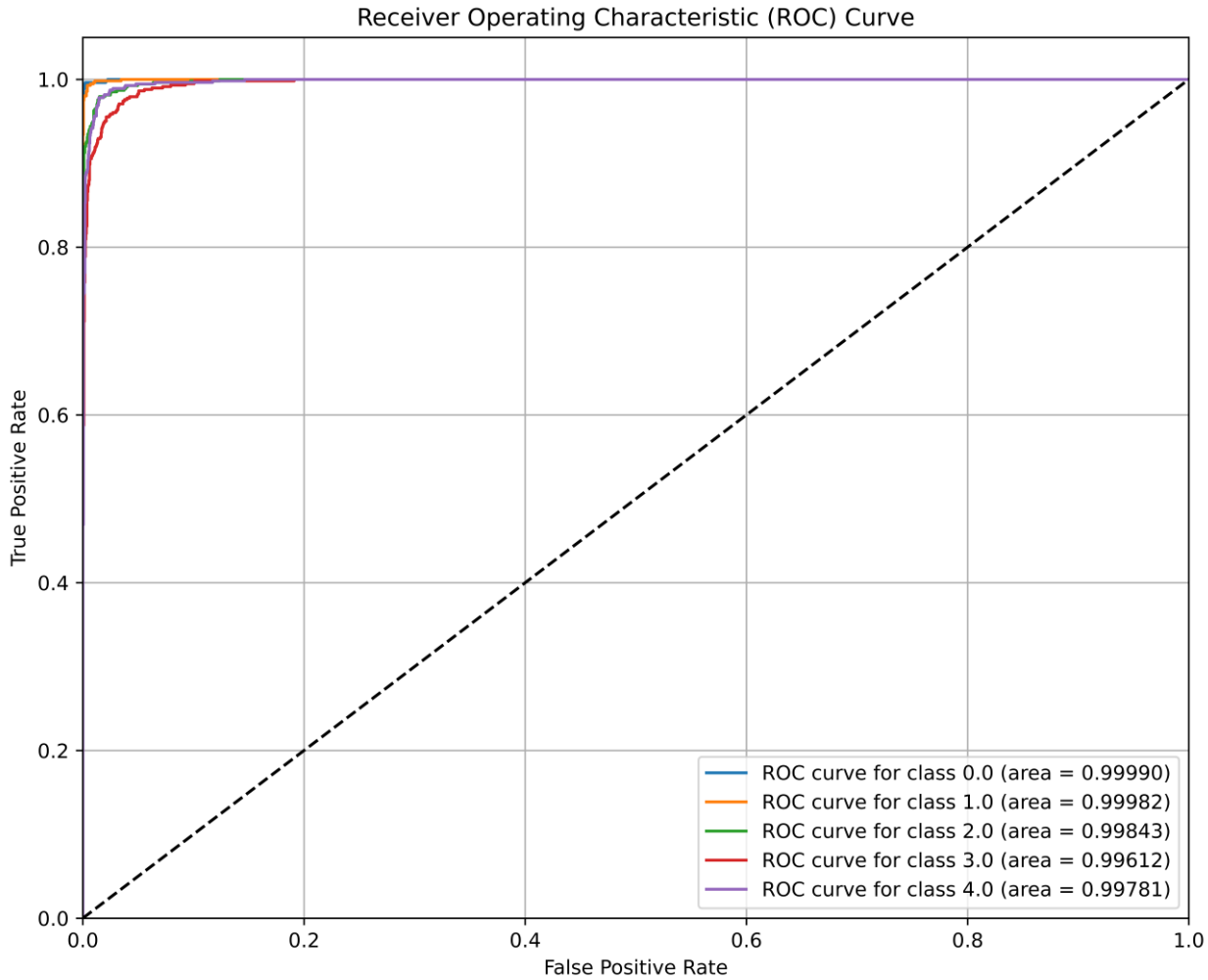

Рисунок 12. ROC-кривая для классификатора Stacking Classifier.

Для многоклассовых задач ROC-кривая строится для каждого класса отдельно. Данный подход позволяет оценить, как хорошо модель различает каждый класс. Одним из важных показателей при интерпретации ROC-кривой

является площадь под кривой (AUC — Area Under Curve). Чем ближе значение AUC к 1, тем лучше модель справляется с разделением классов. Значение AUC около 0.5 говорит о том, что модель работает не лучше случайного выбора, а если AUC меньше 0.5, то модель ошибается чаще, чем случайный классификатор. ROC-кривая для классификатора Stacking Classifier приведена на рисунке 12.

Построение ROC-кривой позволяет не только оценить производительность модели в целом, но и выбрать оптимальный порог классификации, при котором модель будет наилучшим образом балансировать между количеством правильных предсказаний и количеством ложных срабатываний. Это делает ROC-кривую полезным инструментом для тонкой настройки модели и её дальнейшего улучшения.

Таким образом, в ходе построения и оценки моделей машинного обучения важно учитывать различные аспекты их производительности. Это помогает выбрать наиболее подходящую модель для конкретной задачи и улучшить её работу. На каждом этапе анализа, начиная с подготовки данных и разделения их на обучающую и тестовую выборки, до выбора и обучения моделей, необходимо тщательно следить за качеством работы алгоритмов. После обучения моделей важно провести их оценку с помощью различных метрик точности. Это позволяет не только понять, насколько хорошо модели работают в целом, но и выявить их слабые места. Кроме того, для более полной и наглядной оценки модели можно использовать различные визуальные инструменты, такие как матрицы ошибок и ROC-кривые. Важно помнить, что для более точной оценки следует использовать ансамблевые методы, которые объединяют несколько моделей и могут повысить общую точность.

2.5 Создание, обучение и тестирование нейронной сети

Нейронные сети играют ключевую роль в современных методах машинного обучения, поскольку позволяют эффективно решать сложные задачи. Эти модели способны выявлять скрытые закономерности в данных, что делает их мощным инструментом для анализа больших объемов информации. Одним из

самых популярных и удобных инструментов для разработки нейронных сетей является библиотека Keras, которая предоставляет простой и гибкий интерфейс для построения, обучения и оценки моделей. Благодаря своей интеграции с TensorFlow, Keras позволяет быстро реализовывать сложные нейронные сети с минимальными усилиями, что делает его идеальным выбором как для исследователей, так и для разработчиков.

Для решения задачи классификации с использованием нейронной сети была проведена последовательная обработка данных и построение модели. Сначала данные были загружены из файла "features.csv", а затем разделены на признаки и целевую переменную. Для подготовки целевой переменной к многоклассовой классификации была использована функция to_categorical, преобразующая её в формат one-hot encoding.

После этого данные были разделены на обучающую и тестовую выборки с помощью функции train_test_split из библиотеки scikit-learn. В результате 80% данных были использованы для обучения модели, а оставшиеся 20% — для её тестирования. Для улучшения сходимости модели и ускорения обучения признаки были масштабированы с помощью MinMaxScaler.

Следующим шагом было построение модели нейронной сети с использованием библиотеки Keras [24, 25]. Для этого была выбрана последовательная архитектура модели, состоящая из нескольких скрытых слоев с функцией активации ReLU. Каждый из этих слоев был дополнен слоем Dropout(0.2), что позволило уменьшить вероятность переобучения модели, случайным образом исключая часть нейронов на каждом шаге обучения. Выходной слой был имел функцию активации Softmax, так как задача представляла собой многоклассовую классификацию. Архитектура созданной нейронной сети представлена на рисунках 13 и 14.

Модель была скомпилирована с использованием оптимизатора Adam, который является эффективным для обучения нейронных сетей, и функцией потерь categorical_crossentropy, подходящей для многоклассовых задач.

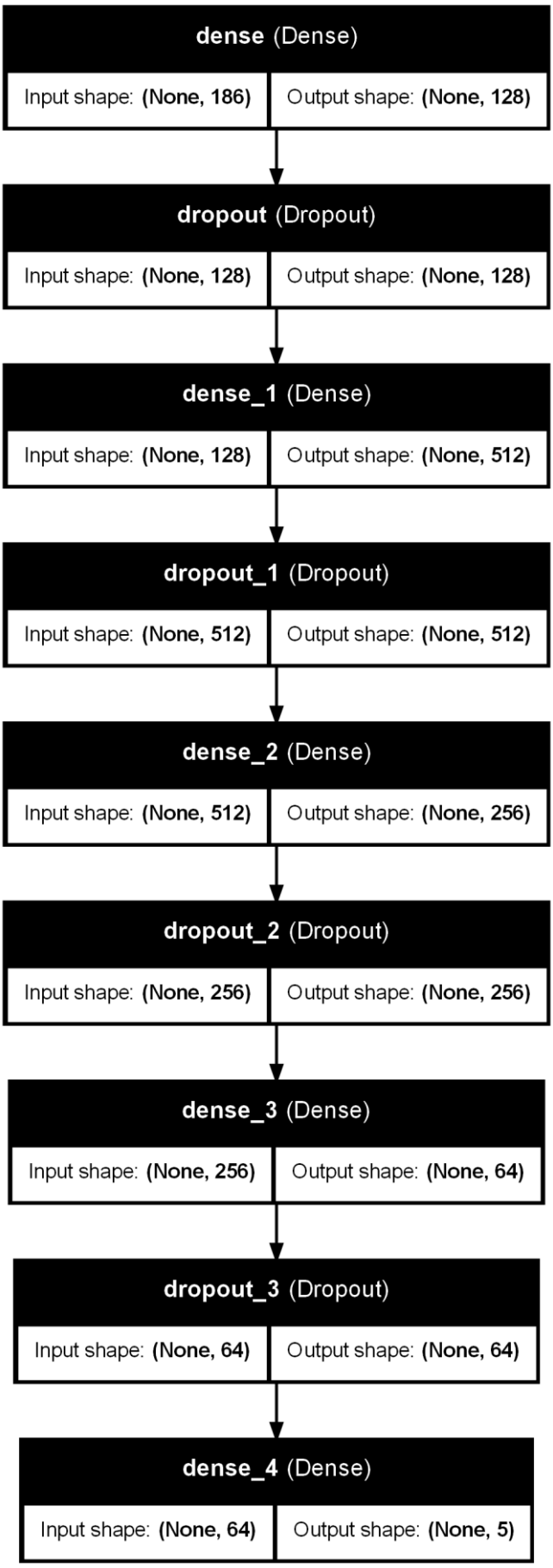

Рисунок 13. Архитектура нейронной сети.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	23,936
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 512)	66,048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131,328
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 64)	16,448
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 5)	325

Total params: 714,257 (2.72 MB)
Trainable params: 238,085 (930.02 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 476,172 (1.82 MB)

Рисунок 14. Архитектура и количество параметров нейронной сети.

Для предотвращения переобучения и ускорения процесса обучения была использована техника ранней остановки (EarlyStopping). Она позволяет автоматически завершить обучение, если ошибка на валидационной выборке не улучшалась в течение 200 эпох.

После компиляции модели было выполнено её обучение, которое продолжалось 1000 эпох, однако благодаря использованию ранней остановки процесс был завершен раньше. Далее модель была протестирована на тестовой выборке, и предсказания были преобразованы в метки классов с помощью функции np.argmax, которая выбирает класс с наибольшей вероятностью. На следующем шаге была использована функция classification_report из библиотеки sklearn.metrics, которая выдала подробный отчет о точности модели для каждого из классов. Для оценки эффективности модели (как для каждого класса, так и для всей модели в целом) были использованы несколько метрик точности: Accuracy, Precision, Recall и F1-score. Результат представлен на рисунке 15.

	precision	recall	f1-score	support
0	0.96	0.96	0.96	528
1	0.95	0.95	0.95	561
2	0.92	0.91	0.92	538
3	0.87	0.90	0.89	582
4	0.92	0.90	0.91	550
accuracy			0.93	2759
macro avg	0.93	0.93	0.93	2759
weighted avg	0.93	0.93	0.93	2759

Accuracy: 92.534 %
Precision: 92.572 %
F1-Score: 92.545 %
Recall: 92.534 %

Рисунок 15. Результат тестирования обученной нейронной сети.

Для более глубокого анализа поведения модели в процессе обучения были построены два графика, представленные на рисунке 16, которые показывают изменение точности и функции потерь на обучающей и валидационной выборках. Эти графики полезны для диагностики процесса обучения, выявления проблем с переобучением и недообучением, а также для оценки того, насколько хорошо модель адаптируется к данным.

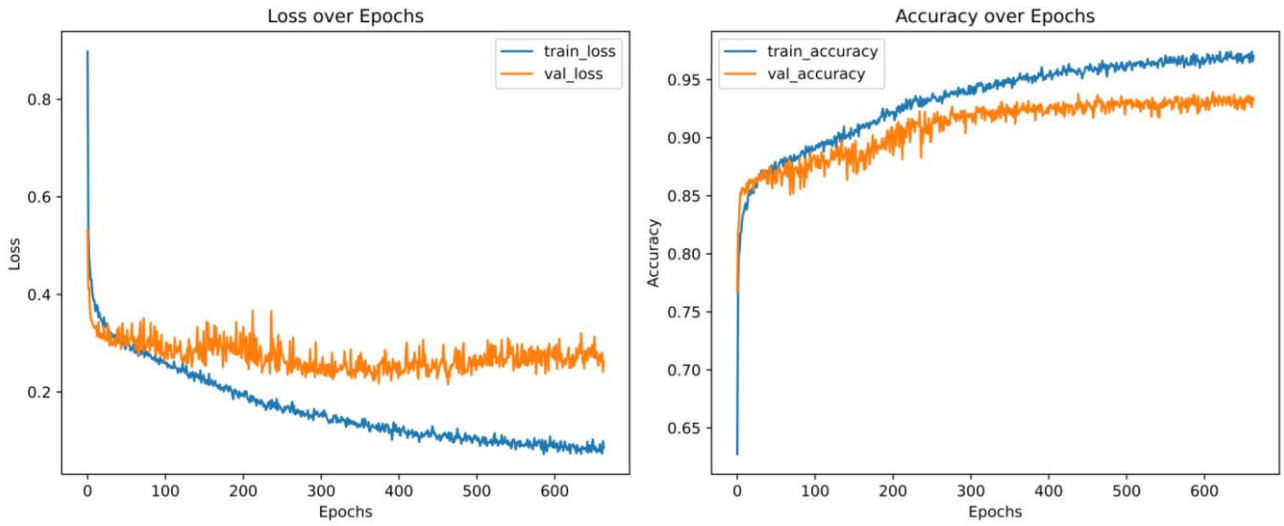

Рисунок 16. Графики точности и функции потерь модели в зависимости от количества эпох обучения.

График точности отображает, как изменялась точность модели на обучающей и валидационной выборках в процессе обучения. На оси X откладываются эпохи обучения, на оси Y — значения точности. График показывает, как модель становится более точной с каждым шагом обучения, а также позволяет увидеть возможное снижение точности на валидационной выборке, что может свидетельствовать о переобучении.

График потерь (Loss**)** представляет собой график изменения функции потерь как на обучающей, так и на валидационной выборках. На оси X также откладываются эпохи, а на оси Y — значения функции потерь. Функция потерь измеряет, насколько предсказания модели отличаются от реальных значений, и является ключевым показателем эффективности алгоритма обучения. Если функция потерь не снижается или начинает расти, это может означать, что модель не обучается должным образом или происходит переобучение.

Далее была построена матрица ошибок (confusion matrix), представленная на рисунке 17, которая показала, насколько корректно модель классифицирует объекты различных классов. Это особенно полезно в случае многоклассовой классификации, где важно увидеть, какие классы чаще всего путает модель.

Несмотря на тщательную настройку модели нейронной сети и использование различных техник для повышения её производительности, не удалось добиться значительных улучшений по сравнению с другими моделями. Во время экспериментов с нейронной сетью, несмотря на применение различных архитектур, оптимизаторов и методов предотвращения переобучения, результат модели не показал существенного превосходства над более простыми методами машинного обучения.

Возможно, выбранная архитектура нейронной сети (с несколькими слоями и нейронами) была избыточной для задачи. В некоторых случаях для задач классификации даже простая однослойная нейронная сеть может показывать сопоставимые результаты с более сложными моделями, если правильно настроены параметры.

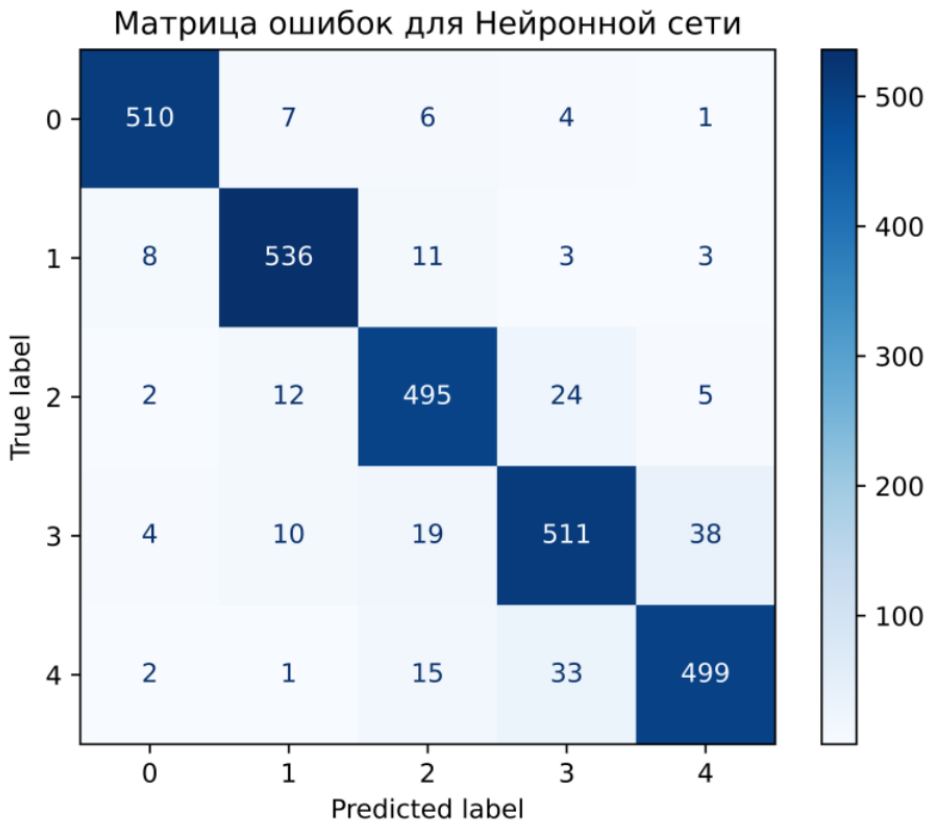

Рисунок 17. Матрица ошибок для нейронной сети.

Модели, основанные на решающих деревьях и ансамблях, такие как случайный лес или градиентный бустинг, продемонстрировали более стабильные и предсказуемые результаты. Эти методы, как правило, гораздо быстрее сходятся и менее склонны к переобучению.

Это подчеркивает важность выбора подходящего алгоритма и настройки модели в зависимости от специфики данных и задачи. Возможно, для улучшения результатов нейронной сети потребовалась бы дополнительная работа над выбором оптимальной архитектуры и подбором гиперпараметров.

2.6 Разработка приложения

Приложения — это программные решения, которые выполняют определенные задачи или предоставляют пользователю определённые услуги. В современном мире приложения охватывают широкий спектр функционала и могут быть использованы для самых разных целей, от развлечений и общения до сложных аналитических задач и автоматизации бизнес-процессов. Они могут

работать на различных платформах: мобильных устройствах, персональных компьютерах, серверных системах и даже в облаке.

В этом проекте создается веб-приложение с использованием фреймворка Flask [26, 27], а также Bootstrap [28, 29], которые позволяет пользователю загружать файл лога полетов в формате JSONL, а затем обрабатывать его, извлекая признаки из данных и выполняя предсказания с использованием заранее обученной модели машинного обучения. Приложение состоит из нескольких компонентов, каждый из которых выполняет определенную роль.

Веб-приложение состоит из нескольких HTML-шаблонов [30] и одного основного Python-файла. Шаблоны — это файлы с разметкой, которые отвечают за отображение страниц пользователю. В проекте есть три таких файла: error.html, home_page.html, и result.html.

Страница home_page.html, которая показана на рисунке 18, отображается при переходе на главную страницу. На ней есть форма, в которой пользователь может загрузить файл в формате JSONL. Этот файл содержит данные, которые будут обработаны в приложении. Форма отправляет файл на сервер с помощью метода POST. Если файл загружен корректно, то пользователь перенаправляется на страницу обработки файла.

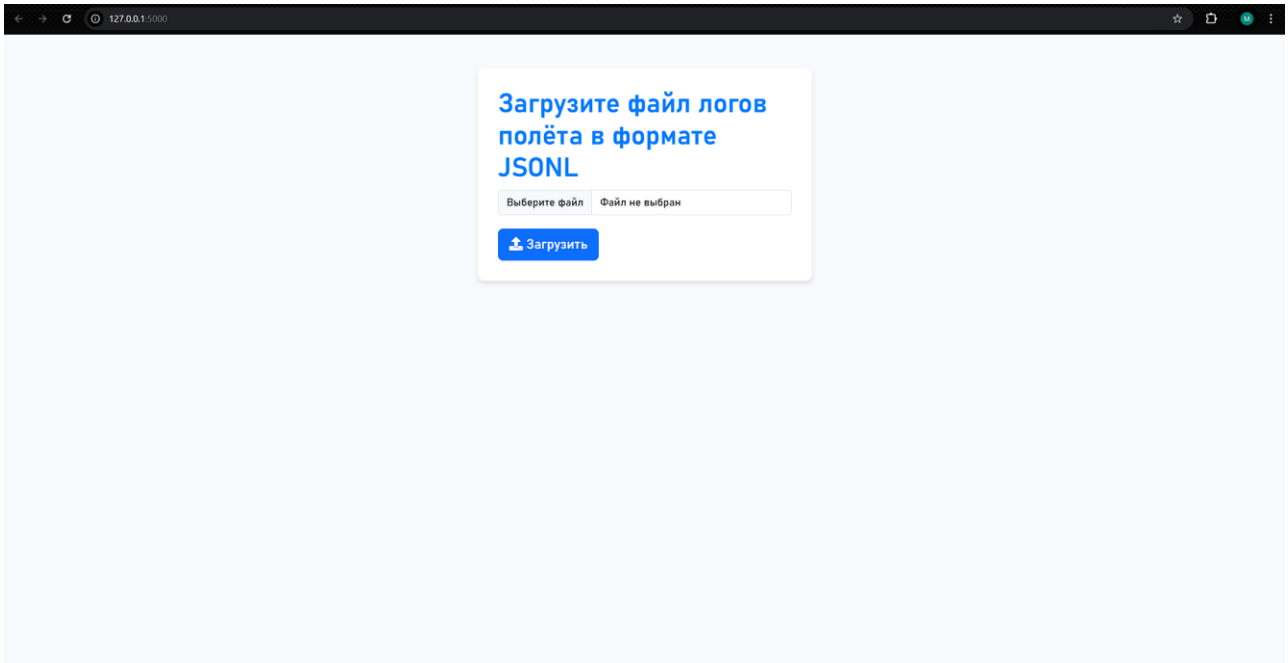

Рисунок 18. Интерфейс главной страницы.

Если же пользователь загружает файл неподдерживаемого формата, то вместо обработки данных показывается страница ошибки, показанная на рисунке 19.

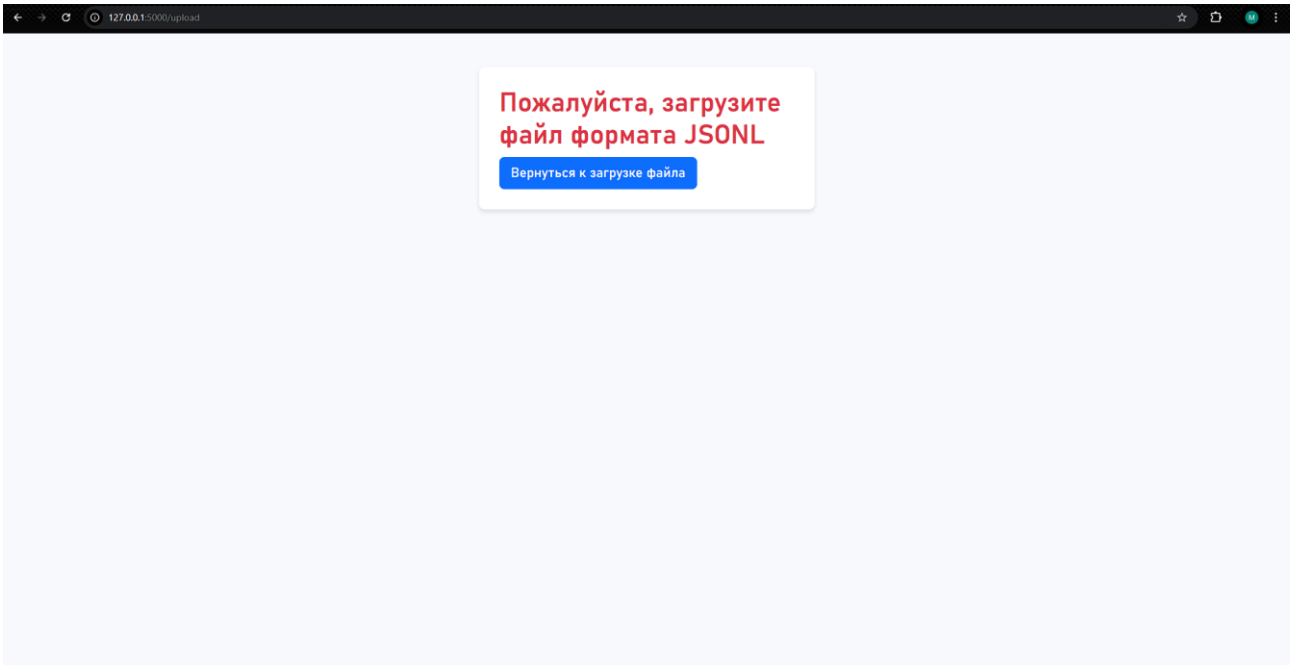

Рисунок 19. Интерфейс страницы ошибки.

Если файл прошел проверку, он сохраняется на сервере в специально выделенной папке uploads, и запускается процесс обработки данных.

Процесс обработки данных включает в себя несколько важных шагов. Во-первых, приложение читает файл JSONL, где каждая строка представляет собой отдельный объект JSON. Эти объекты затем преобразуются в строки, из которых формируется структура данных, представляемая DataFrame с помощью библиотеки pandas.

Далее происходит извлечение данных и организация их в соответствующие столбцы. Затем некоторые ненужные столбцы удаляются для упрощения дальнейшей обработки.

После этого начинается этап извлечения признаков из временной и частотной областей.

Когда признаки извлечены, наступает этап нормализации данных. Для этого используется несколько методов масштабирования, таких как Power

Transformer, Standard Scaler и MinMax Scaler, обученных заранее, каждый из которых преобразует данные в определенный диапазон или распределение, что улучшает производительность модели машинного обучения.

После того как данные нормализованы, они подаются на вход модели машинного обучения. Модель (XGBoost), обученная заранее и сохраненная в файле trained_model.pkl, выполняет предсказания на основе переданных данных.

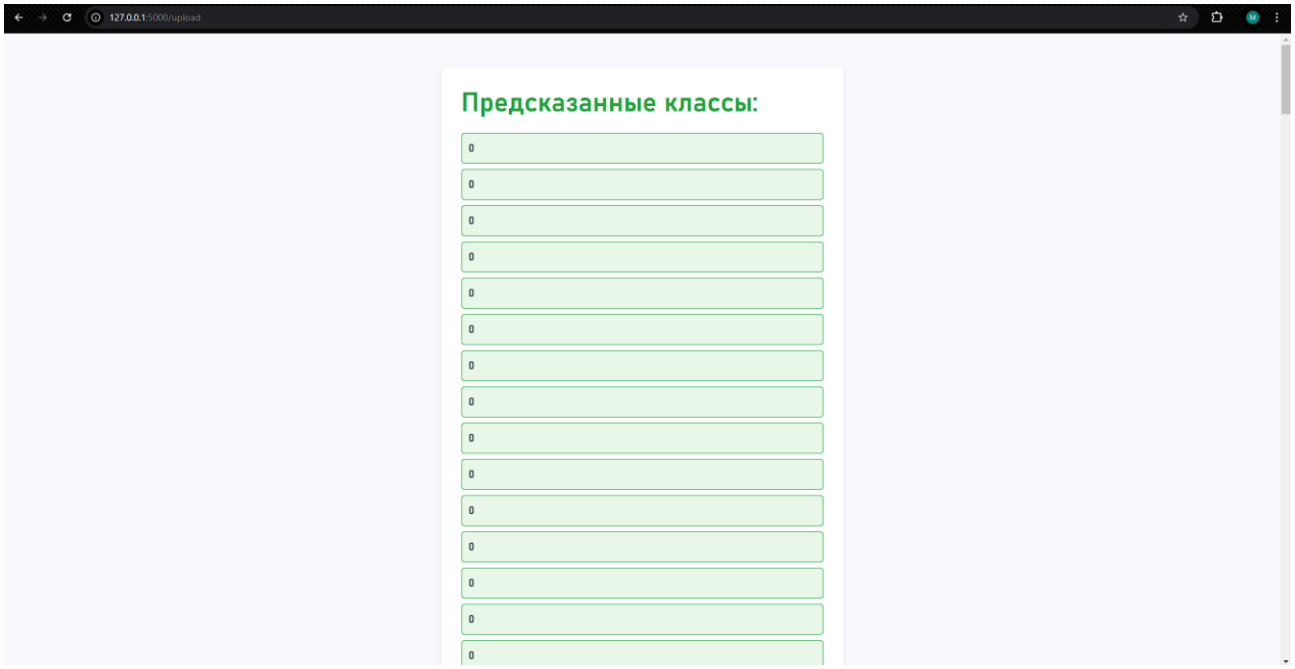

Рисунок 20. Интерфейс страницы с результатами (начало).

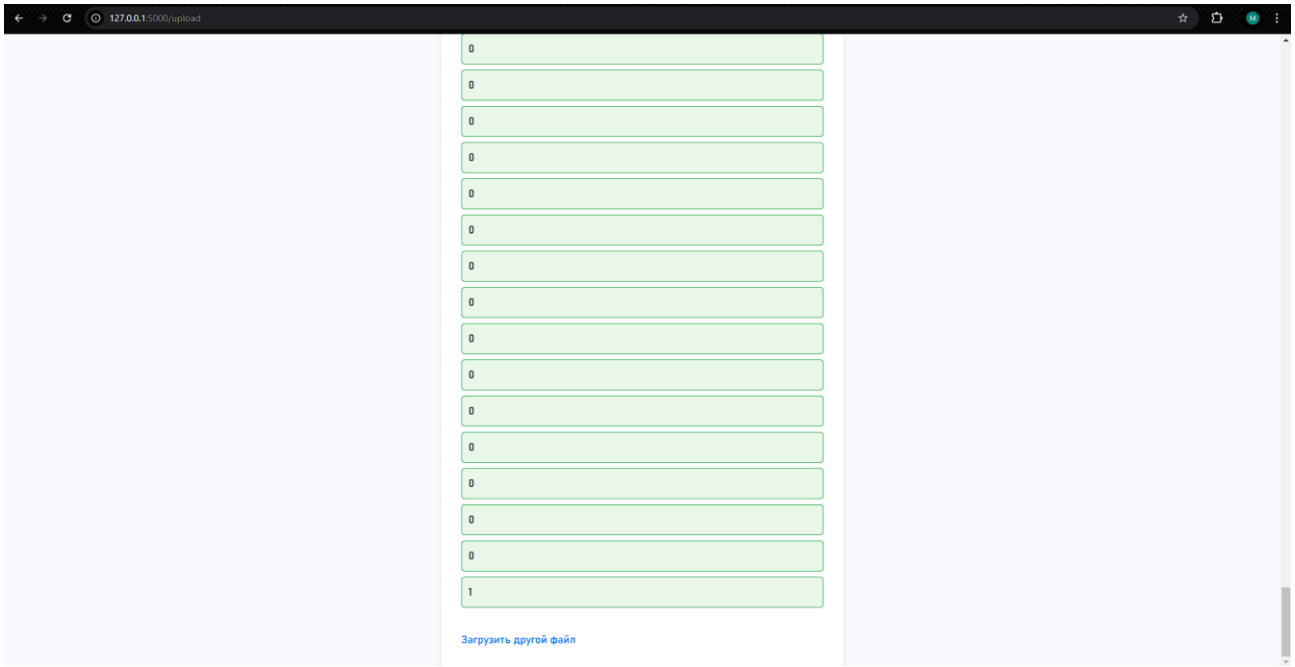

Рисунок 21. Интерфейс страницы с результатами (конец).

Когда предсказания получены, они передаются в шаблон result.html, где отображаются пользователю, как показано на рисунках 20 и 21. На странице с результатами выводится список предсказанных классов, а также предоставляется ссылка для загрузки другого файла.

В случае, если при обработке файла возникла ошибка, приложение сообщает пользователю об этом, как показано на рисунке 22, например, если загруженный файл в формате JSONL не имеет структуры данных, необходимой для корректного извлечения данных.

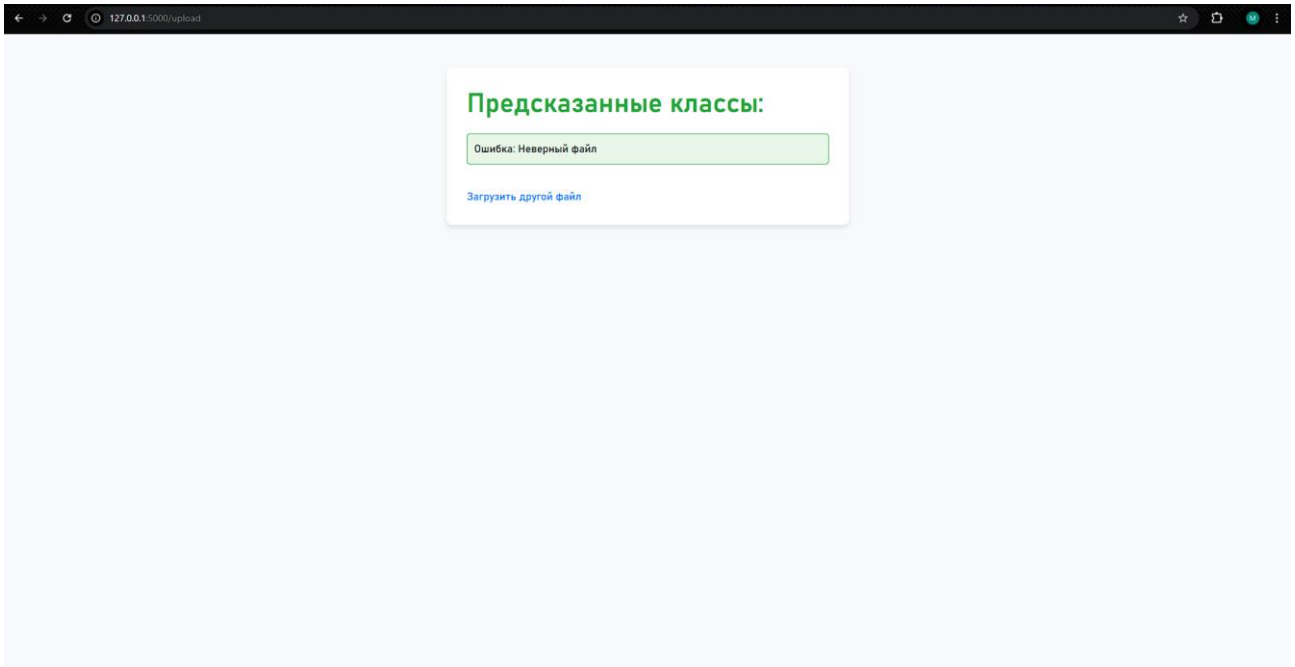

Рисунок 22. Интерфейс страницы с результатами при некорректном загруженном фале формата JSONL.

Приложение запускается с помощью команды python app.py. После запуска сервер будет доступен по адресу http://127.0.0.1:5000.

Для работы с моделью в проекте используется библиотека joblib [31], которая позволяет загружать заранее обученные модели и преобразователи. Важно отметить, что обученные модели и преобразователи должны быть сохранены в формате .pkl для их использования в приложении.

Таким образом, приложение позволяет пользователю легко загружать данные и получать результаты анализа и предсказаний, не требуя от него

углубленных знаний о работе с моделями машинного обучения. Все ключевые операции выполняются автоматически на сервере, а пользователю достаточно просто загрузить файл и получить результат.

2.7 Создание удаленного репозитория

Для данного исследования был создан удаленный репозиторий на платформе GitHub, который служит для хранения и управления кодом, а также для совместной работы над проектом. Репозиторий был создан с целью упрощения доступа к материалам исследования и для последующего использования в других проектах.

В репозиторий были загружены все необходимые файлы, относящиеся к исследованию: исследовательский notebook, датасет извлеченных признаков, датасет извлеченных из JSONL данных акселерометра и гироскопа, приложение, пояснительная записка и презентация.

Ссылка на репозиторий: https://github.com/amigoris/BMSTU_Diplom

Заключение

В рамках данной работы была разработана система для детектирования аварийных ситуаций на базе мультироторных беспилотных летательных аппаратов (БПЛА), основанная на анализе данных с инерциальных датчиков, таких как акселерометры и гироскопы. Одной из главных задач работы было создание эффективного алгоритма, который мог бы своевременно обнаружить неисправности и предупредить оператора о возможных аварийных ситуациях, тем самым повышая уровень безопасности и надежности эксплуатации БПЛА. Это особенно важно, поскольку аварийные ситуации могут иметь критические последствия, и их своевременное выявление может предотвратить повреждения аппарата и потерю оборудования.

Современные БПЛА, в частности мультироторные системы, становятся все более востребованными в таких областях, как сельское хозяйство, мониторинг окружающей среды, логистика, инспекция инфраструктуры и поисково-спасательные операции. Однако для эффективного применения этих технологий крайне важно обеспечить их безопасность и надежность в реальных условиях эксплуатации. Поэтому использование данных с инерциальных датчиков, которые позволяют отслеживать состояние аппарата в процессе полета, является ключевым шагом к решению этой проблемы.

Вибрация, являющаяся результатом работы механизмов, и другие параметры, получаемые с датчиков, могут служить индикаторами возможных неисправностей. Понимание закономерностей этих данных помогает разрабатывать алгоритмы, способные прогнозировать возможные аварийные ситуации до того, как они приведут к критическим последствиям.

Для реализации задачи был использован подход машинного обучения, который позволяет анализировать большие объемы данных, получаемых с датчиков, и выявлять аномалии, указывающие на неисправности. В процессе работы рассматривались и применялись различные алгоритмы классификации в том числе и нейронная сеть. В ходе тестирования этих моделей были получены

результаты, которые показали, что использование ансамблевых методов, таких как случайный лес, градиентный бустинг и XGBoost, позволяет достичь наилучших показателей точности и устойчивости к ошибкам. Эти методы продемонстрировали высокую способность к обобщению на новых данных, что крайне важно для обеспечения надежности системы в реальных условиях эксплуатации.

Одним из ключевых аспектов работы стало создание программного обеспечения, которое включает в себя несколько этапов: импорт и первичную обработку данных, анализ данных, извлечение признаков и построение моделей. Признаки были использованы для обучения моделей машинного обучения, которые затем были протестированы на данных, имитирующих реальные условия эксплуатации БПЛА.

Результаты тестирования и анализа показали, что предложенная система успешно решает задачу своевременного обнаружения неисправностей на базе БПЛА, демонстрируя высокую точность и низкую вероятность ложных срабатываний.

Таким образом, реализация системы детектирования аварийных ситуаций на базе БПЛА с использованием машинного обучения и данных с инерциальных датчиков является важным шагом в направлении повышения безопасности и надежности эксплуатации беспилотных летательных аппаратов. Результаты работы подчеркивают высокий потенциал применения данной системы в реальных условиях, а также открывают новые возможности для разработки более сложных и универсальных решений, которые могут быть интегрированы в различные области, такие как экологический мониторинг, поисково-спасательные операции и инспекция инфраструктуры. Работа в данной области имеет значительный научный и практический вклад в развитие технологий БПЛА и их безопасного использования в разнообразных сферах.

Библиографический список

1. Dynamic Consultees: Flexible Planning for Heterogeneous Teams of Robotic Agents [Электронный ресурс] : – Режим доступа: https://secplab.ppgia.pucpr.br/files/papers/2023icra.pdf. (дата обращения: 02.12.2024).
2. Support Vector Machines [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/svm.html. (дата обращения: 03.12.2024).
3. k-Nearest Neighbors [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/neighbors.html. (дата обращения: 03.12.2024).
4. MLPClassifier [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#mlp-classifier. (дата обращения: 03.12.2024).
5. Logistic Regression [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. (дата обращения: 03.12.2024).
6. Naive Bayes Classification [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/naive_bayes.html. (дата обращения: 03.12.2024).
7. Decision Trees [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/tree.html. (дата обращения: 03.12.2024).
8. Random Forests [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#random-forests. (дата обращения: 03.12.2024).
9. Машинное обучение для начинающих. Алгоритм случайного леса (Random Forest) [Электронный ресурс] : – Режим доступа: https://proglib.io/p/mashinnoe-obuchenie-dlya-nachinayushchih-algoritm-sluchaynogo-lesa-random-forest-2021-08-12. (дата обращения: 03.12.2024).

10. CatBoost: Gradient Boosting with Categorical Features Support [Электронный ресурс] : – Режим доступа: https://catboost.ai/. (дата обращения: 03.12.2024).
11. AdaBoost [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#adaboost. (дата обращения: 03.12.2024).
12. Gradient Boosting Machines [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting. (дата обращения: 03.12.2024).
13. XGBoost Documentation [Электронный ресурс] : – Режим доступа: https://xgboost.readthedocs.io/. (дата обращения: 03.12.2024).
14. LightGBM Documentation [Электронный ресурс] : – Режим доступа: https://lightgbm.readthedocs.io/. (дата обращения: 03.12.2024).
15. HistGradientBoosting [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#histgradientboosting. (дата обращения: 03.12.2024).
16. Stacking Classifier [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#stacking-classifier. (дата обращения: 03.12.2024).
17. Voting Classifier [Электронный ресурс] : – Режим доступа: https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier. (дата обращения: 03.12.2024).
18. Understanding Windowing Functions for Signal Processing [Электронный ресурс] : – Режим доступа: https://www.mathworks.com/help/signal/ug/understanding-windowing-functions-for-signal-processing.html. (дата обращения: 04.12.2024).
19. Функции оконного сглаживания [Электронный ресурс] : – Режим доступа: https://ru.dsplib.org/content/windows/windows.html. (дата обращения: 04.12.2024).

20. Digital Signal Processing: Blackman-Nuttall Window [Электронный ресурс] : – Режим доступа: https://www.ti.com/lit/an/slyt235/slyt235.pdf. (дата обращения: 04.12.2024).
21. Fast Fourier Transform and Its Applications [Электронный ресурс] : – Режим доступа: https://www.mathworks.com/help/matlab/fast-fourier-transform.html. (дата обращения: 04.12.2024).
22. Statistical Functions (mean, median, std, kurtosis, skew, etc.) [Электронный ресурс] : – Режим доступа: https://docs.scipy.org/doc/scipy/reference/stats.html. (дата обращения: 06.12.2024).
23. Python Statistics: Mean, Median, Mode, Variance, and Standard Deviation [Электронный ресурс] : – Режим доступа: https://realpython.com/python-statistics/. (дата обращения: 06.12.2024).
24. TensorFlow Guide: Keras [Электронный ресурс] : – Режим доступа: https://www.tensorflow.org/guide/keras. (дата обращения: 08.12.2024).
25. Keras Documentation [Электронный ресурс] : – Режим доступа: https://keras.io/. (дата обращения: 08.12.2024).
26. Flask Documentation [Электронный ресурс] : – Режим доступа: https://flask.palletsprojects.com/en/2.0.x/. (дата обращения: 10.12.2024).
27. Jinja2 Documentation [Электронный ресурс] : – Режим доступа: https://jinja.palletsprojects.com/en/3.1.x/. (дата обращения: 10.12.2024).
28. Bootstrap Documentation [Электронный ресурс] : – Режим доступа: https://getbootstrap.com/docs/5.3/. (дата обращения: 11.12.2024).
29. CSS Documentation [Электронный ресурс] : – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/CSS. (дата обращения: 11.12.2024).
30. HTML5 Documentation [Электронный ресурс] : – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/HTML. (дата обращения: 11.12.2024).

31. Joblib Documentation [Электронный ресурс] : – Режим доступа: https://joblib.readthedocs.io/en/latest. (дата обращения: 12.12.2024).