МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

Высшая школа информационных технологий и автоматизированных систем

(наименование высшей школы / филиала / института / колледжа)

**КУРСОВОЙ ПРОЕКТ**

|  |  |  |
| --- | --- | --- |
| По дисциплине | | Глубокое обучение. Продвинутый уровень (модуль) |
|  | | |
|  | | |
| На тему | Разработка приложения по поиску аномалий спутниковой телеметрии с | |
| помощью машинного обучения | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О.  обучающихся | Наименование направления подготовки/ специальности | Курс | Группа | Ф.И.О. руководителя (-ей) должность / уч. степень / звание |
| Васильев Антон Дмитриевич | Информационные системы и технологии (Интеллектуальный анализ геопространственных данных) | 2 | 151266 | Васендина Ирина Сергеевна., доцент к.т.н |
| Сахник Арсений Алексеевич | Информационные системы и технологии (Интеллектуальный анализ геопространственных данных) | 2 | 151266 | Васендина Ирина Сергеевна., доцент к.т.н |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признать, что проект выполнен и защищен с отметкой |  |  |  |  |
|  |  | (отметка прописью) |  | (дата) |
| Руководитель |  |  |  | И.С. Васендина |
|  |  | (подпись руководителя) |  | (инициалы, фамилия) |

Архангельск 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

|  |
| --- |
| Высшая школа информационных технологий и автоматизированных систем |
| (наименование высшей школы / филиала / института / колледжа) |

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| по | Глубокое обучение. Продвинутый уровень (модуль) |
|  | (наименование дисциплины) |
| Студенту ВШИТиАС высшей школы 2 курса 151266 группы | |
| Васильеву Антону Дмитриевичу | |
| (фамилия, имя, отчество студента) | |
| 09.04.02 «Информационные системы и технологии» | |
| (код и наименование направления подготовки/специальности) | |
| Студенту ВШИТиАС высшей школы 2 курса 151266 группы | |
| Сахнику Арсению Алексеевичу | |
| (фамилия, имя, отчество студента) | |
| 09.04.02 «Информационные системы и технологии» | |
| (код и наименование направления подготовки/специальности) | |

|  |  |
| --- | --- |
| ТЕМА: | Разработка приложения по поиску аномалий спутниковой телеметрии с помощью машинного обучения |
| ИСХОДНЫЕ ДАННЫЕ:   1. Данные спутниковой телеметрии | |
| ЗАДАНИЕ:   1. Изучение предметной области; 2. Проектирование архитектуры приложения; 3. Реализация приложения 4. Сравнение результатов | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель работы |  | доцент |  |  |  | И. С. Васендина |
|  |  | (должность) |  | (подпись) |  | (инициалы, фамилия) |

Архангельск 2024

Лист для замечаний

## Содержание

[Введение 5](#_Toc164981577)

[1. Проблематика 7](#_Toc164981578)

[1.1 Постановка проблемы исследования 7](#_Toc164981579)

[1.2 Выделение объекта и предмета исследования 7](#_Toc164981580)

[1.3 Актуальность 8](#_Toc164981581)

[1.4 Цели и задачи исследования 9](#_Toc164981582)

[1.5 Анализ имеющихся методов, средств и технологий проектирования информационных систем 10](#_Toc164981583)

[1.6 Разработка методики научного исследования 13](#_Toc164981584)

[1.7 Анализ методов нахождения аномалий в телеметрии спутников 13](#_Toc164981585)

[1.8 Определение связи аномалий с космической погодой 35](#_Toc164981586)

[1.9 Выводы исследования 36](#_Toc164981587)

[2 Разработка программного средства 38](#_Toc164981588)

[2.1 Проектирование программного средства 38](#_Toc164981589)

[2.2 Разработка сервиса синхронизации космической погоды 38](#_Toc164981590)

[2.3 Разработка сервиса обработки данных и выделения аномалий космической погоды 39](#_Toc164981591)

[2.4 Реализация клиентского сервиса 40](#_Toc164981592)

[3 Структура магистерской диссертации 45](#_Toc164981593)

[3.1 Разработка структуры магистерской диссертации 45](#_Toc164981594)

[3.2 Написание отдельных глав диссертации 45](#_Toc164981595)

[Выводы 47](#_Toc164981596)

[Список использованных источников 49](#_Toc164981597)

[Приложение А (обязательное) Модуль для синхронизации космической погоды 51](#_Toc164981598)

[Приложение Б (обязательное) Модуль обработки данных 54](#_Toc164981599)

[Приложение В (обязательное) Модуль выделения аномалий 57](#_Toc164981600)

[Приложение Г (обязательное) Модуль выходных параметров 59](#_Toc164981601)

[Приложение Д (обязательное) Модуль визуализации результатов 61](#_Toc164981602)

ВВЕДЕНИЕ

Данные телеметрии являются одним из главных элементов в системе управления малых космических аппаратов, обеспечивая контроль состояния его отдельных узлов и параметров движения. Большой объем данных телеметрии, непрерывно поступающий с борта малого космического аппарата на приемные станции по всему миру, а также информация о космических аппаратах, оказывающей влияние на параметры систем спутников, требует разработки новых методик и алгоритмов обработки полученных данных с целью снижении количества сбоев в работе малых космических аппаратов, вызванных негативными факторами космической погоды. Помимо этого, операторы космических аппаратов заинтересованы также в визуализации данных и удобном представлении, и возможности использования различных способов поиска аномалий в данных спутниковой телеметрии.

Цель научно-исследовательской работы включает в себя изучение передовых способов хранения, обработки и передачи информации, способов построения архитектуры приложения и готовых и применяемых решений в области визуализации, используемые на предприятии. Помимо этого, целью практики является исследование предметной области, анализ методик и алгоритмов, направленных на выявление аномалий в данных, исследование связи между аномалиями и космической погодой, а также создание приложения, которое сможет автоматизировать этот процесс.

В ходе выполнения практики был проведен обширный анализ предметной области, включающий изучение передовых методов обработки данных телеметрии, а также анализ существующих решений и применяемых алгоритмов. Этот этап включал в себя ознакомление с литературными источниками, научными публикациями и практическим опытом в области космической телеметрии и ее обработки. Далее была разработана методология сбора и обработки данных, основываясь на полученных знаниях и лучших практиках в отрасли. Это включало в себя выбор подходящих алгоритмов для анализа данных, определение структуры и формата данных, а также планирование процесса визуализации и мониторинга. После этого перешли к разработке программного модуля для синхронизации данных между различными информационными системами. Этот модуль должен был обеспечивать эффективную передачу данных между разными базами данных, используемыми в рамках системы управления малыми космическими аппаратами. В ходе разработки учитывались требования к безопасности, надежности и эффективности передачи данных.

Параллельно с разработкой модуля проводилась настройка системы журналирования действий и визуализация нагрузки на сервер. Это включало в себя выбор подходящих инструментов для ведения журнала действий и настройку системы мониторинга с использованием инструмента Grafana. Завершающим этапом работы было тестирование разработанных модулей и интеграция их в общую систему управления космическими аппаратами. Проводилось как функциональное, так и нагрузочное тестирование, чтобы убедиться в правильной работе системы и ее способности обрабатывать большие объемы данных.

Результатом выполнения проекта стало успешное развертывание разработанных модулей и интеграция их в общую систему управления. Полученные навыки и опыт были применены для последующей разработки программного решения в рамках магистерской диссертации.

1. **Проблематика**
   1. Постановка проблемы исследования

Своевременное прогнозирование неисправностей на малых космических аппаратах играет ключевую роль в обеспечении их эффективной защиты в условиях агрессивной околоземной космической среды. Некоторые типы неисправностей, такие как электрические пробои и сбои в каналах связи, могут быть предсказаны путем непосредственного измерения различных параметров на борту космического аппарата. Этот подход позволяет операторам принимать меры предосторожности или активно вмешиваться для предотвращения серьезных инцидентов.

Другой подход к прогнозированию неисправностей на малых космических аппаратах основан на статистических методах [2, 3]. Совместный анализ сбоев на спутниках и изменений параметров окружающей среды позволяет выявить взаимосвязи между различными состояниями окружающей среды и возникновением нештатных ситуаций на космических аппаратах.

Интеграция этих двух подходов позволяет проводить комплексный анализ состояния группировок спутников с схожими конструктивными характеристиками. При учете геофизической обстановки [4, 5], параметров спутника и его положения в пространстве, становится возможным прогнозировать аномалии на борту конкретного космического аппарата [6]. Это дает операторам возможность предпринимать меры предосторожности заранее или оперативно реагировать на возникающие проблемы, минимизируя риск потенциальных повреждений или потерь.

* 1. Выделение объекта и предмета исследования

Объектом данного исследования является проведение анализа данных телеметрии с использованием специализированных методов и алгоритмов с целью выявления аномальных значений отдельных параметров. Кроме того, будет осуществлено исследование вероятных причин возникновения подобных аномалий.

Дополнительно, в рамках исследования будут изучены современные подходы к разработке приложений, обработке, хранению и передаче информации, а также визуализации различных типов данных. Это включает в себя изучение передовых методов обработки и анализа данных, использование новейших алгоритмов машинного обучения для выявления паттернов в данных телеметрии, а также применение современных технологий для эффективного хранения и передачи больших объемов информации.

Исследование аномальных значений в данных телеметрии и их возможных причин имеет важное практическое значение, поскольку это позволяет оперативно реагировать на возможные проблемы или неисправности на космических аппаратах, минимизируя риск негативных последствий. Кроме того, разработка и применение современных методов анализа данных способствует повышению эффективности управления космическими миссиями и обеспечивает более надежное функционирование космических аппаратов.

* 1. Актуальность

Тема обработки данных телеметрии в системах управления малых космических аппаратов актуальна в современной космической индустрии по нескольким ключевым причинам.

* объем данных - с развитием технологий сбора данных малых космических аппаратов объем получаемой телеметрии постоянно увеличивается. Это приводит к необходимости использования более эффективных методов обработки и анализа данных, чтобы обеспечить быстрый и точный контроль за состоянием аппаратов;
* негативные факторы космической погоды, такие как солнечные бури и радиационные пояса, могут вызывать сбои в работе космических аппаратов. Факторы, такие как солнечные бури и радиационные пояса, могут вызывать сбои и аномалии в работе аппаратов. Поэтому выявление и анализ аномалий в данных телеметрии, связанных с этими явлениями, становится критически важным для обеспечения надежности и безопасности космических миссий;
* визуализация и интерпретация данных. Операторы нуждаются в интуитивно понятных и информативных визуализациях данных для оперативного реагирования на изменения среды и ситуаций на борту аппаратов. Это требует разработки современных средств визуализации и анализа данных, которые учитывают специфику космических приложений;
* разработка новых методик и алгоритмов. Для оптимизации работы систем управления и улучшения диагностики состояния космических аппаратов необходимо постоянное совершенствование методов обработки и анализа данных. Это включает в себя исследование новых алгоритмов для выявления аномалий и разработку программных решений для автоматизации процессов анализа данных телеметрии.

Далее рассмотрим цели и основные задачи исследования.

* 1. Цели и задачи исследования

В цели и задачи исследования входят:

* изучение передовых методов хранения, обработки и передачи информации. Это включает в себя анализ современных технологий баз данных, методов компрессии данных, а также применение передовых протоколов передачи данных, для эффективного обмена информацией.;
* изучение архитектуры приложений и существующих решений в области визуализации данных телеметрии. Это позволит выявить особенности различных подходов к визуализации данных и выбрать наиболее подходящие решения для конкретных потребностей;
* исследование предметной области для анализа методик и алгоритмов выявления аномалий в данных телеметрии. Это включает в себя ознакомление с существующими методами машинного обучения, статистическими алгоритмами и экспертными системами, применяемыми для выявления аномалий;
* анализ связи между аномалиями в данных и космической погодой. Это позволит определить, какие конкретно факторы космической погоды могут оказывать влияние на работу космических аппаратов и какие аномалии в данных могут быть связаны с этими факторами;
* анализ имеющихся методов, средств и технологий проектирования информационных систем применительно к разработке. Это включает в себя анализ существующих архитектурных подходов, методологий разработки и инструментов, используемых для создания информационных систем.;
* создание приложения для автоматизации процесса выявления и анализа аномалий в данных телеметрии. Это включает в себя проектирование архитектуры приложения, выбор подходящих технологий разработки, реализацию функциональности для выявления и анализа аномалий, а также разработку пользовательского интерфейса для взаимодействия с приложением.

В итоге, глубокий анализ представленных аспектов позволяет сделать вывод о необходимости дальнейших исследований и разработок в области обработки данных телеметрии для управления малыми космическими аппаратами. Основываясь на полученных данных и анализе, можно сделать вывод о значимости развития современных методов и технологий для обеспечения надежности и эффективности космических миссий.

* 1. Анализ имеющихся методов, средств и технологий проектирования информационных систем

При выборе формата разрабатываемого программного модуля было принято решение о создании веб-приложения в связи с многочисленными преимуществами, которые оно предоставляет. Веб-приложения, будучи доступными через браузер, обеспечивают пользовательский доступ с любого устройства, что повышает их удобство использования и расширяет охват аудитории. Кроме того, веб-приложения обладают высокой масштабируемостью, что позволяет им легко адаптироваться к изменяющимся потребностям и объемам данных. Их гибкость настройки позволяет разработчикам создавать кастомные функциональные возможности и адаптировать приложение под конкретные требования заказчика.

Одним из ключевых преимуществ веб-приложений является централизованное управление данными. Они обеспечивают хранение информации на сервере, что позволяет обеспечить ее единообразный доступ для всех пользователей и облегчает ее обновление и синхронизацию.Интеграция с другими системами через API является еще одним важным аспектом, который обеспечивает веб-приложение. Это позволяет эффективно обмениваться данными между различными приложениями и сервисами, что расширяет функциональные возможности и повышает эффективность работы всей системы.

Наконец, веб-приложения обеспечивают обновления без необходимости ручной установки на устройствах пользователей. Это означает, что новые функции и исправления могут быть внедрены централизованно на сервере и автоматически доступны всем пользователям, что повышает удобство использования и обновляемость приложения.

При выборе архитектуры для реализации веб-приложения валидны два подхода - монолитная и микросервисная. Для оценки их плюсов и минусов следует рассмотреть оба варианта. Монолитная архитектура требует значительного количества связей между компонентами для реализации масштабируемого приложения. Микросервисная архитектура позволяет удобнее разрабатывать приложение, имея независимые компоненты приложения.

Одним из недостатков монолитной архитектуры является то, что любые изменения или изменения функций приведут к изменению всего приложения. Также увеличение серверных мощностей для монолитной архитектуры может быть осложнено или в принципе невозможным.

Из плюсов можно выделить простоту создания. Так как монолитная архитектура предлагает более легкий путь для создания приложений, поскольку все аспекты приложения собраны в одном монолите. Также такая архитектура обеспечивает более низкую потребность в ресурсах, благодаря локализации бизнес-логики в одной программной единице.

Из недостатков микросервисной архитектуры можно выделить то, что разработка приложения может занять больше времени и затрат, которые будет связаны со сложностью настройки взаимодействия между микросервисами, функции делятся на отдельные модули, которые должны быть взаимосвязаны с правильной инфраструктурой для их взаимодействия.

Преимущества микросервисной архитектуры заключаются в том, что приложение может быть разбито на много независимых программных модулей, которые связываются друг с другом с помощью модели запрос-ответ или передают данные через шину брокера сообщений. Такие приложения проще настраивать, запускать и масштабировать, а также легче поддерживать и обслуживать, так как предполагается, что модули будут небольших размеров. Помимо этого, с использованием микросервисной архитектуры появляется возможность избавиться от нужды в перезапуске всего проекта при изменении отдельных компонентов [7].

Из-за того, что в приложении предусмотрено дальнейшее расширение бизнес-логики, а также возможно увеличение используемого набора технологий была выбрана микросервисная архитектура. Данная архитектура даёт доступ к расширению и масштабированию бизнес-логики без изменения существующей логики, а также позволяет использовать программные модули других разработчиков.

При выборе решения для хранения временных рядов было выбрано решение InfluxDB, которое показывает наилучшие показатели по сравнению с аналогами [8] и обладает возможностью интеграции со средством визуализации Grafana.

Для технологических задач, которые не требуют высокой производительности и масштабируемости, было принято решение использовать СУБД PostgreSQL. Этот выбор обусловлен несколькими факторами. Во-первых, PostgreSQL известен своей надежностью и стабильностью, что делает его привлекательным для проектов, где необходимо обеспечить стабильную работу базы данных без сбоев и перебоев.

Во-вторых, PostgreSQL имеет обширную экосистему дополнений и расширений, а также обладает широкой поддержкой ORM-библиотек, что облегчает разработку приложений и интеграцию с другими технологиями. Это позволяет разработчикам быстрее создавать программные модули, не затрачивая много времени на настройку и интеграцию базы данных. Кроме того, PostgreSQL постоянно обновляется и развивается благодаря активному сообществу разработчиков, что гарантирует поддержку новых функций и технологий, а также исправление ошибок и уязвимостей. Это важно для обеспечения безопасности и актуальности базы данных в долгосрочной перспективе.

Таким образом, выбор PostgreSQL обусловлен его надежностью, гибкостью, и удобством использования, что делает его привлекательным вариантом для реализации программных модулей, не требующих высокой производительности, но при этом ценящих стабильность и удобство разработки.

Для визуализации данных было принято решение использовать инструмент Grafana. Этот выбор обусловлен несколькими преимуществами. Во-первых, Grafana предоставляет широкие возможности интеграции с различными системами управления базами данных, что позволяет эффективно работать с различными источниками данных, включая выбранную СУБД PostgreSQL. Это обеспечивает гибкость и масштабируемость в работе с информацией, что особенно важно для комплексного анализа телеметрии космических аппаратов.

Во-вторых, Grafana предоставляет мощные инструменты визуализации данных в реальном времени. Это позволяет операторам космических аппаратов мониторить и анализировать данные по мере их поступления, что особенно важно для оперативного реагирования на потенциальные проблемы и аномалии. Гибкие настройки графиков и дашбордов в Grafana позволяют создавать качественные и информативные визуализации, отражающие различные аспекты телеметрии спутников и параметров космической среды.

Для отображения промежуточных результатов и анализа данных в веб-приложении будут использоваться линейные графики и гистограммы. Эти типы графиков являются оптимальным выбором для отображения количественных показателей [9], таких как температура, давление, энергия и другие характеристики телеметрии. Они обеспечивают наглядное представление данных и позволяют операторам анализировать информацию с легкостью и точностью.

После тщательного анализа средств и технологий проектирования информационных систем было принято решение о разработке веб-приложения, которое будет состоять из отдельных микросервисов. Для управления данными была выбрана PostgreSQL, учитывая ее надежность, гибкость и хорошую интеграцию с другими технологиями. В качестве средства визуализации данных было выбрано Grafana, благодаря ее мощным возможностям и гибкости настройки. Это обеспечит эффективное мониторинг и анализ телеметрии космического аппарата.

* 1. Разработка методики научного исследования

Данная научно-исследовательская работа охватывает широкий спектр задач и целей, связанных с анализом данных телеметрии спутников. В первую очередь, сосредоточимся на изучении различных методов выявления аномалий в данных телеметрии. Это включает в себя как традиционные статистические методы, такие как анализ выбросов и контроль процесса, так и современные подходы машинного обучения, например, алгоритмы глубокого обучения и методы обнаружения аномалий.

Одной из основных целей нашего исследования является проверка гипотезы о возможном влиянии космической погоды на данные телеметрии спутников. Существует предположение, что различные феномены космической погоды, такие как солнечные вспышки, геомагнитные бури и другие, могут иметь влияние на работу космических аппаратов и проявляться в данных телеметрии. Для этого планируется провести обширный анализ временных рядов данных о космической погоде и телеметрии спутников, чтобы выявить возможные корреляции и зависимости между ними.

Кроме того, в рамках данного исследования есть намерение определить наиболее эффективную модель машинного обучения для выявления аномалий в данных телеметрии. Это включает в себя анализ различных алгоритмов машинного обучения, их сравнение и оценку производительности на основе набора данных о телеметрии спутника "СириусСат-1". Данное исследование стремится выявить наиболее подходящие и эффективные методы для обнаружения аномалий в данных телеметрии спутников.

Для анализа влияния космической погоды на телеметрию спутников будут использоваться обширные наборы данных, предоставленные ресурсом "Центр анализа космической погоды НИИЯФ МГУ". Эти данные содержат информацию о различных параметрах космической погоды, которые могут оказывать влияние на работу космических аппаратов. Анализ этих данных поможет нам лучше понять взаимосвязь между космической погодой и телеметрией спутников, а также выявить потенциальные факторы риска и разработать стратегии их управления.

* 1. Анализ методов нахождения аномалий в телеметрии спутников

При рассмотрении способов определения аномалий в данных рассматривались следующие модели:

* k-nearest neighbors (KNN);
* local outlier factor (LOF);
* isolation forest (IF);
* Gaussian mixture model (GMM);
* kernel density estimation (KDE);
* one-class support vector machine (OCSVM);
* DBSCAN;
* OPTICS;
* Gaussian anomaly detection (GAD);
* k-means anomaly detection (KMAD);
* hierarchical anomaly detection (HAD);
* autoencoder anomaly detection (AAD);
* PCA anomaly detection (PCAD);
* SVD anomaly detection (SVAD);
* convex hull anomaly detection (CHAD);
* minimum volume anomaly detection (MVAD);
* feature bagging;
* isolation forest ensemble (IFE);
* anomaly detection with multiple classifiers (ADMC);
* Feedforward Neural Networks;
* Generative Adversarial Networks (GANs);
* Recurrent Neural Networks (RNNs);
* Convolutional Neural Networks (CNNs).

Для ознакомления с работой каждого алгоритма рассмотрим показатель t\_amp, который демонстрирует температуру усилителя радиоэлектронного оборудования спутника.

KNN – это модель, которая основывается на принципе, что объекты с похожими признаками часто расположены близко друг к другу в пространстве признаков. Она хорошо подходит для рассматриваемого показателя, так как значения достаточно близко сгруппированы и показала хороший результат обнаружения аномалий. Результат выполнения модели показан на рисунке 1.

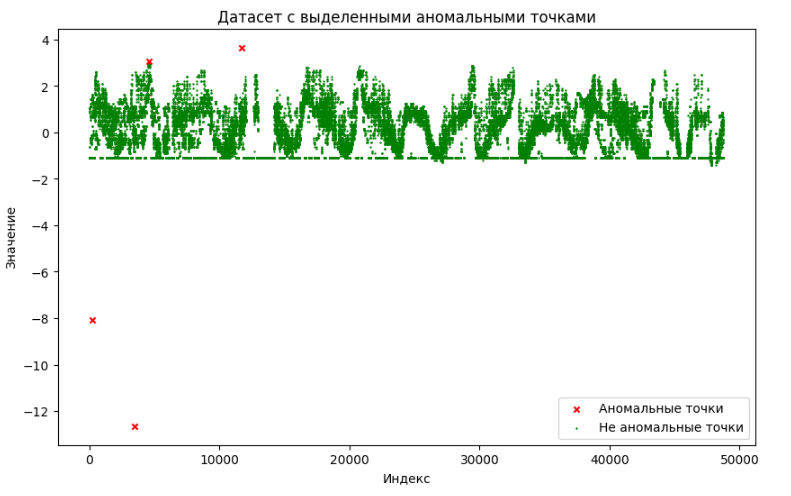


Рисунок 1 – Нахождение аномалий с помощью модели KNN с одним атрибутом

Однако вместе с временной меткой модель показывает уже плохой результат. Результат обучения модели с параметром t\_amp и временной меткой показан на рисунке 2.

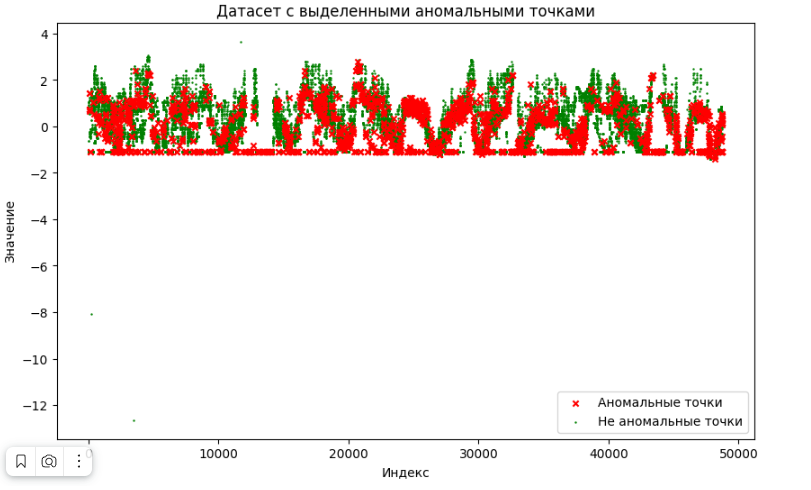


Рисунок 2 - Нахождение аномалий с помощью модели KNN с атрибутом и временной меткой

LOF – это модель, которая оценивает степень аномальности каждого объекта в его окрестности по сравнению с его соседями. Однако на рассматриваемых данных ей не удалось обнаружить аномалии как в паре в временной меткой, так и без нее.

Isolation Forest – это модель, которая строит лес деревьев решений, разделяя случайно выбранные признаки и разбивая объекты на более ранних уровнях деревьев, что позволяет быстро идентифицировать выбросы как объекты, требующие меньшего числа разбиений для изоляции. На рассматриваемых данных, увидеть которые можно на рисунке 3, можно увидеть, что эта модель определила определенное пороговое значение, значения выше которого, она определяет как аномалии.

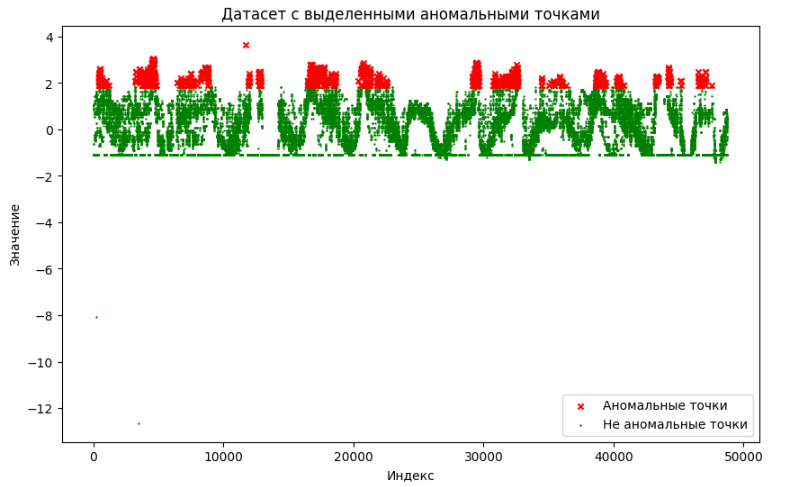


Рисунок 3 – Результат обучения модели Isolation Forest

Результаты модели в паре в ременной меткой показали более худший результат. Результат представлен на рисунке 4.

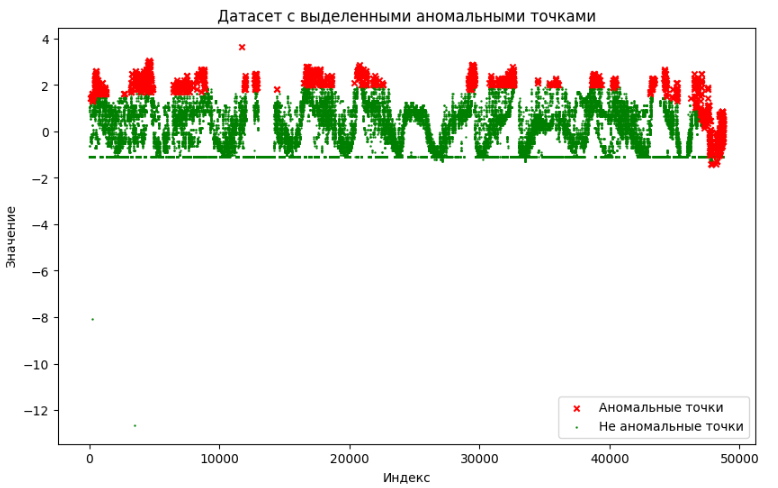


Рисунок 4 - Нахождение аномалий с помощью модели Isolation Forest с атрибутом и временной меткой

Gaussian mixture model (GMM) – это алгоритм, который моделирует распределение данных как смесь гауссовских распределений, идентифицируя аномалии как объекты с низкой вероятностью появления в смеси. Как можно увидеть на рисунке 5, модель выделила основное скопление точек и определила их как не аномальные. Добавление временной метки также не оказало положительного результата, как видно на рисунке 6.

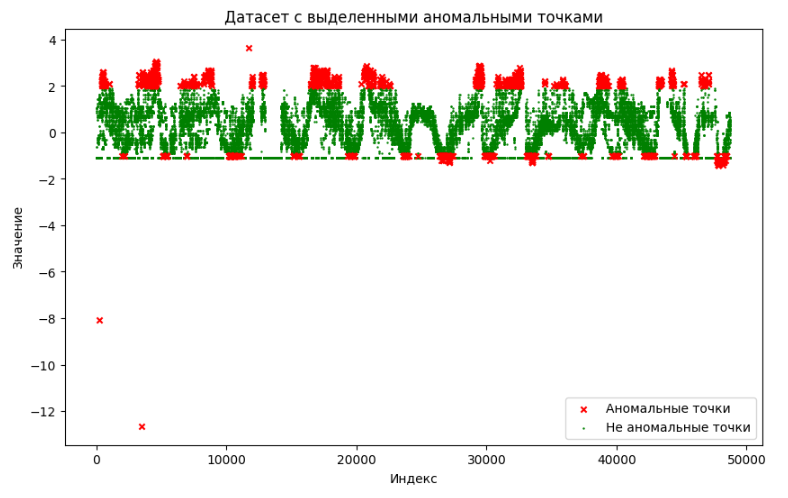


Рисунок 5 - Результат обучения модели GMM

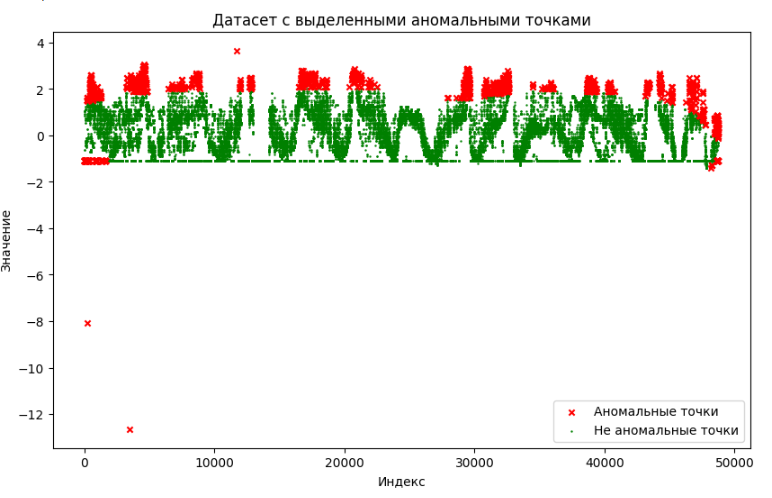


Рисунок 6 - Результат обучения модели GMM с атрибутом временной метки

Модель kernel density estimation (KDE) оценивает плотность данных, используя ядерную функцию, и обнаруживает аномалии как объекты с низкой плотностью в пространстве данных. Как видно на рисунке 7, результаты схожи с обучением модели GMM. Добавление же временного атрибута привела только а более худшим результатам, как это видно на рисунке 8.

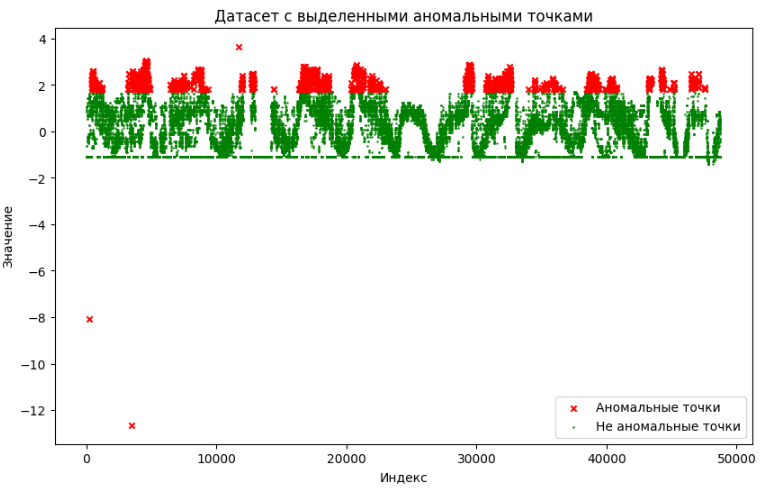


Рисунок 7 – Результат обучения модели KDE

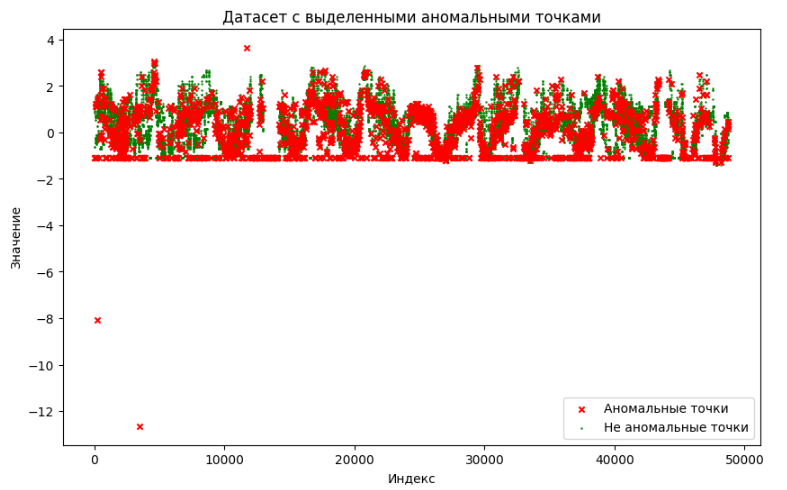


Рисунок 8 – Результат обучения модели KDE с временным атрибутом

Модель one-class support vector machine (OCSVM) строит гиперплоскость, разделяющую данные от нулевой плотности, и идентифицирует аномалии как объекты, находящиеся за пределами этой гиперплоскости. Можно сказать, что данная модель абсолютно не подходит для рассматриваемого параметра, как можно увидеть на рисунке 9.

Добавление временной метке также не делает эту модель оптимальной для рассматриваемого параметра, как видно на рисунке 10.

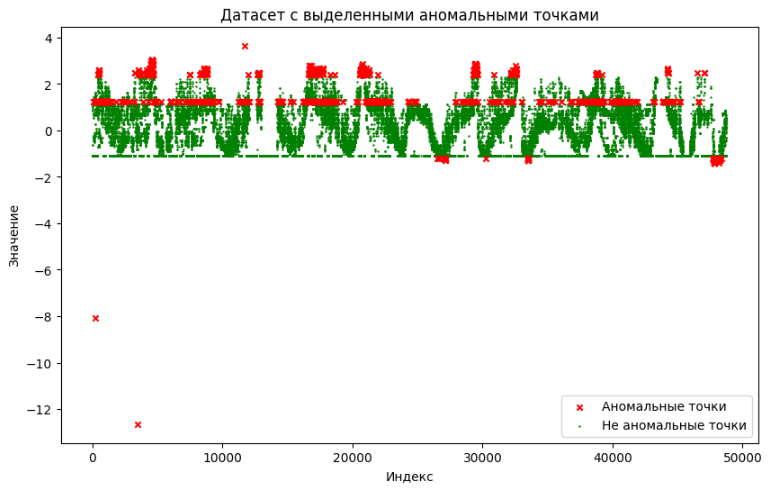


Рисунок 9 – Результат обучения модели OCSVM

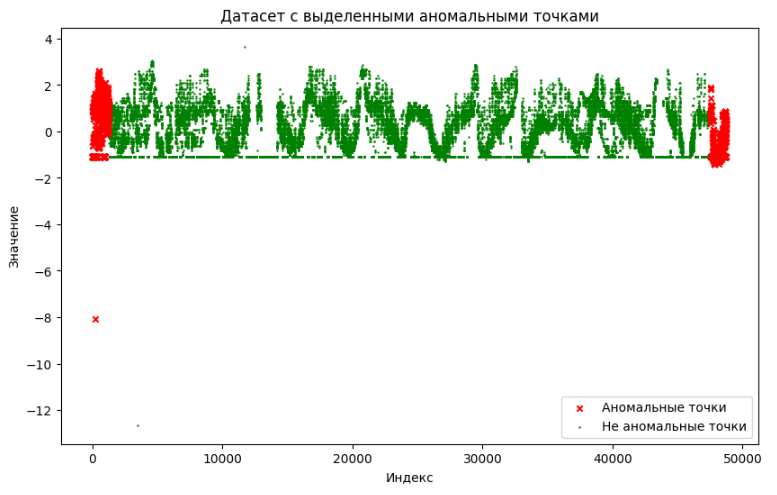


Рисунок 10 - Результат обучения модели OCSVM с временной меткой

Модель DBSCAN определяет кластеры объектов в пространстве данных, выявляя плотные области, и классифицирует объекты вне этих областей как выбросы. Результаты этой модели можно назвать положительными, так как и с временной меткой так и без нее были выбраны подходящие точки. Результаты представлены на рисунке 11 и 12, соответственно.

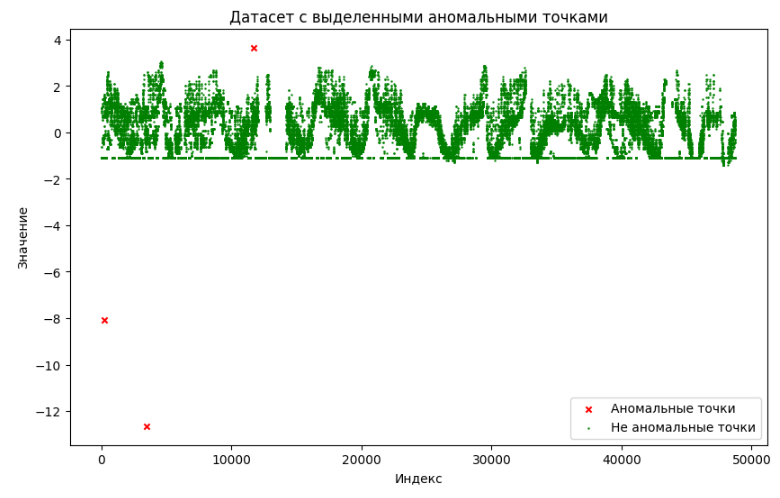


Рисунок 11 – Результат обучения модели DBSCAN

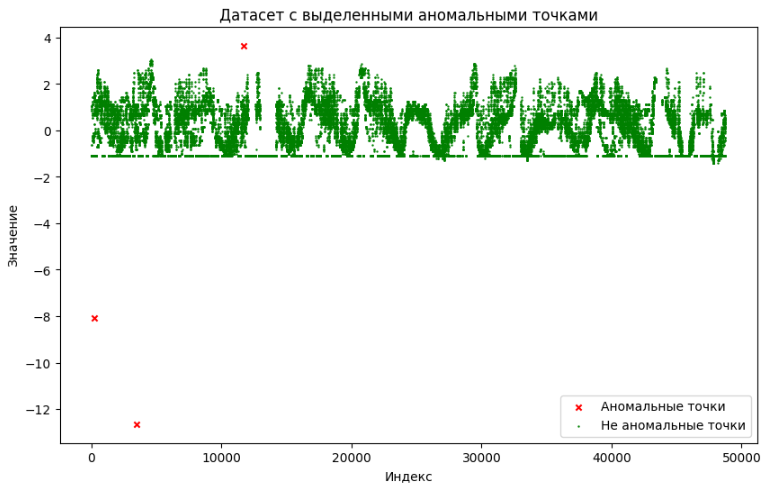


Рисунок 12 – Результат обучения модели DBSCAN

Модель OPTICSопределяет кластеры объектов, учитывая изменение плотности данных, и идентифицирует аномалии как объекты, находящиеся в областях низкой плотности. Как и модель DBSCAN она показала хорошие результаты, как видно на рисунке 12, но добавление временной метки сказалось исключительно негативно, как можно увидеть на рисунке 13.

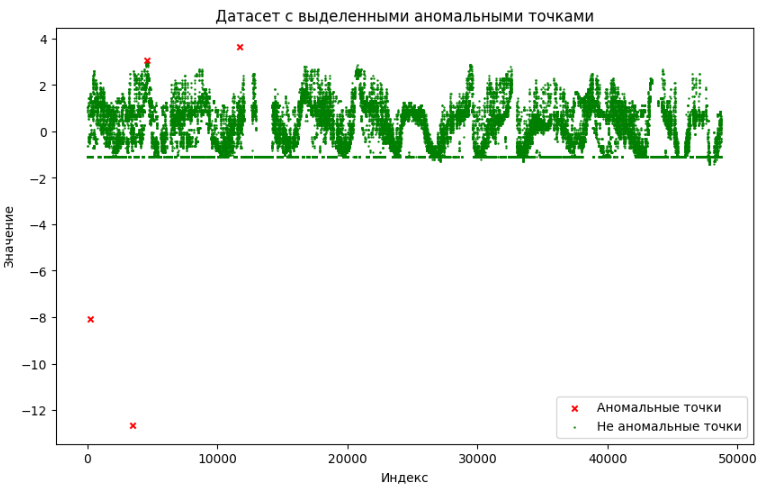


Рисунок 13 – Результат обучения модели OPTICS

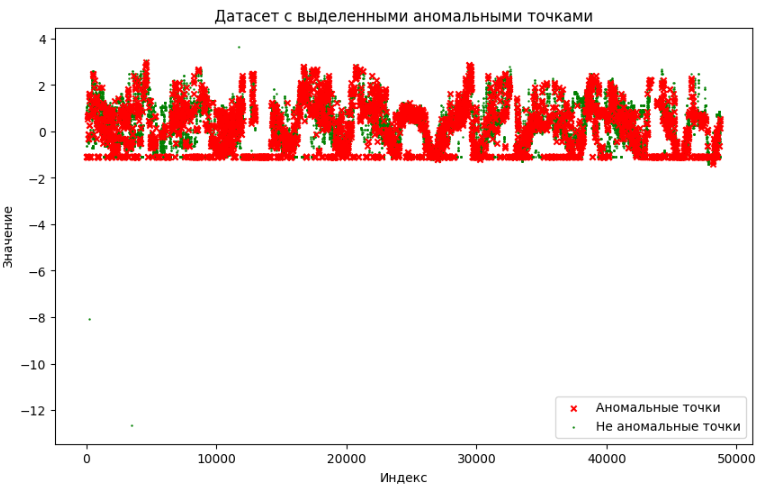


Рисунок 14 – Результат обучения модели OPTICS с атрибутом временной метки

Модель gaussian anomaly detection (GAD) моделирует нормальное распределение данных и выявляет аномалии как объекты, находящиеся в областях с низкой вероятностью появления в распределении. Она обучалась только с использованием значения и временной меткой, так как размер кортежа, подаваемого на обучения, минимально должен состоять из двух элементов. Эта модель показала плохой результат, как видно на рисунке 13.

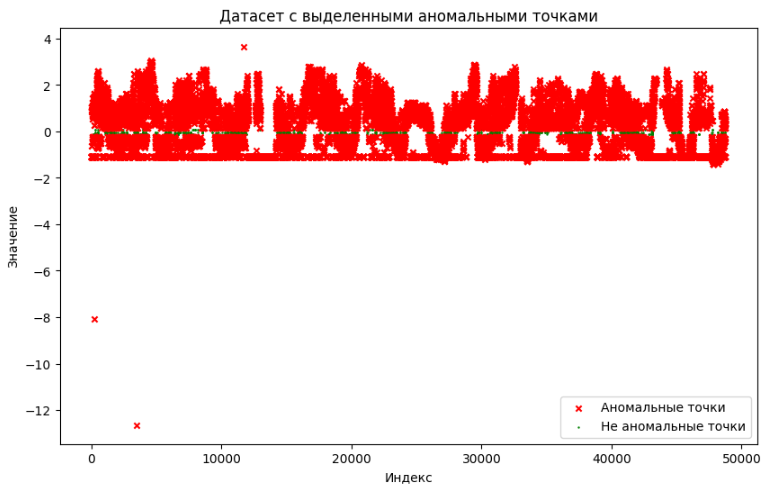


Рисунок 15 – Результат обучения модели GAD

Модель k-means anomaly detection (KMAD) назначает точки данных кластерам с использованием алгоритма k-means и определяет аномалии как точки, далекие от центроидов кластеров. Как можно увидеть на рисунке 14, результат схож на использование модели Isolation Forest. Добавление временной метки только ухудшило результат, как можно увидеть на рисунке 15.



Рисунок 16 – Результат обучения модели KMAD

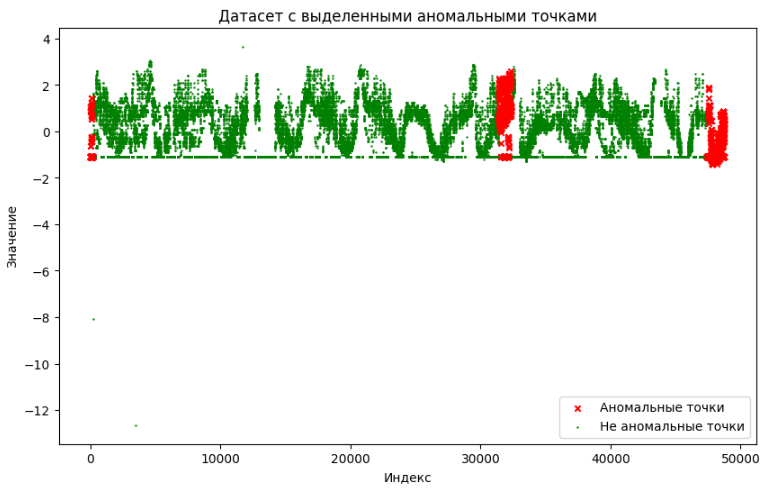


Рисунок 17 – Результат обучения модели KMAD

Модель hierarchical anomaly detection (HAD) HAD строит иерархическую структуру кластеров и идентифицирует аномалии на основе их расстояния от этой структуры. Результаты модели на рисунках 15 и 16, без временной метки и с временной меткой, соответственно, не выявили действительных аномалий в данных.

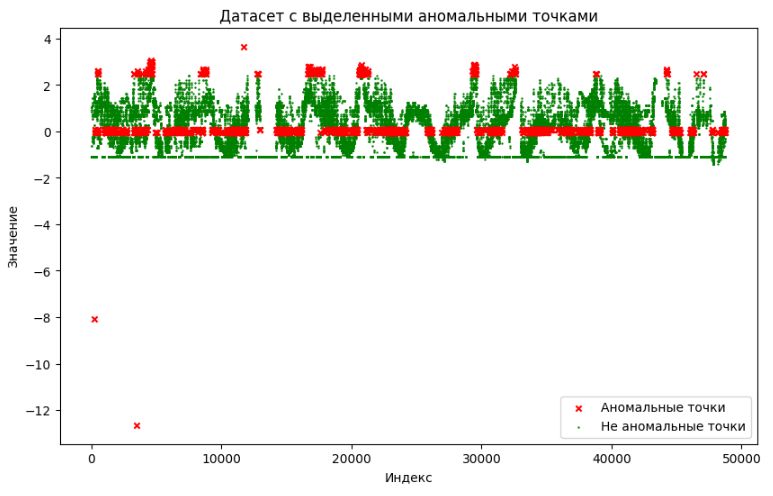


Рисунок 18 – Результат обучения модели HAD

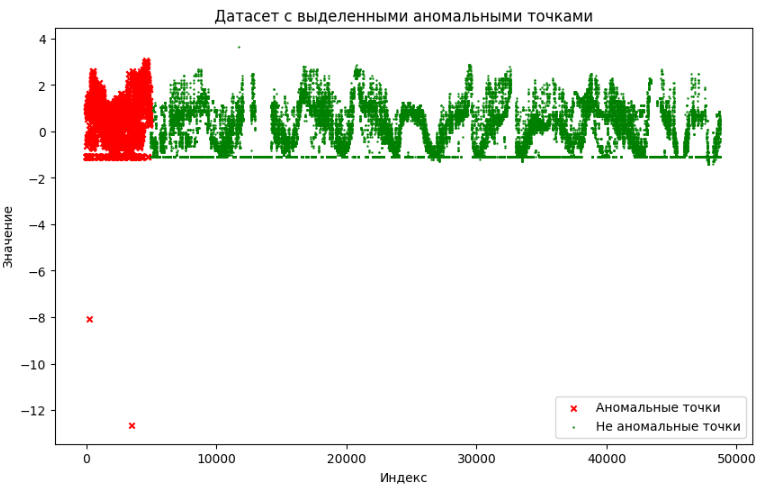


Рисунок 19 – Результат обучения модели HAD с временной меткой

Модель autoencoder anomaly detection (AAD) использует нейронные сети автокодировщики для восстановления данных и обнаружения аномалий на основе различий между входными и восстановленными данными. Она так же обучалась с использованием значения и временной метки, однако показала плохие результаты, которые можно увидеть на рисунке 16.

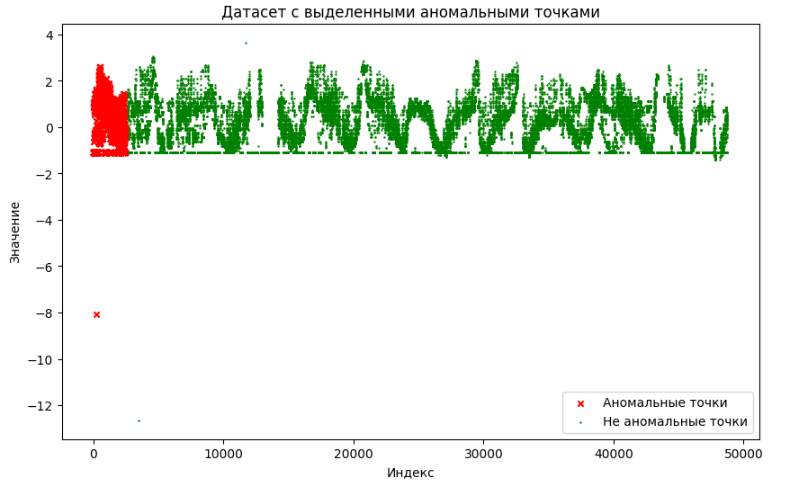


Рисунок 20 – Результат обучения модели AAD

PCA anomaly detection (PCAD) использует метод главных компонент для снижения размерности данных и обнаружения аномалий на основе остатков после восстановления данных. Как можно увидеть на рисунке 17 без указания временной метки эта модель не выделала аномалий, в то время как с указанием временной метки определила пороги, которые определяют являются ли данные аномальными, что представлено на рисунке 18.

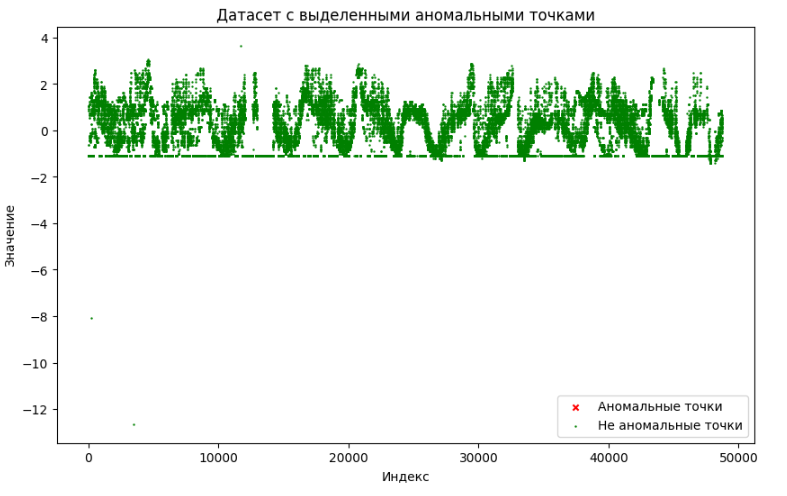


Рисунок 21 – Результат обучения модели PCAD

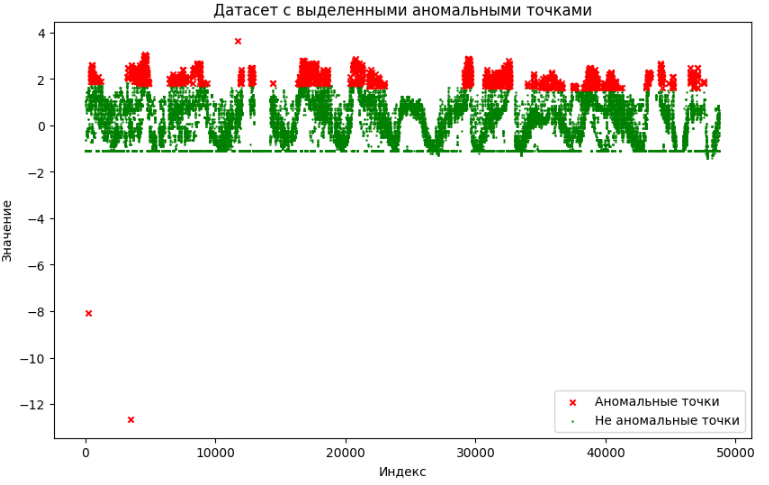


Рисунок 22 – Результат обучения модели PCAD с временной меткой

**S**VD anomaly detection (SVAD) SVAD использует сингулярное разложение для выделения аномалий в данных, основываясь на аномальности остатков после восстановления данных. Эта модель также определяет пороги, за пределами которых считает значения аномальными, как видно на рисунке 20. Добавление временной метки ухудшило результат, что можно увидеть на рисунке 21.

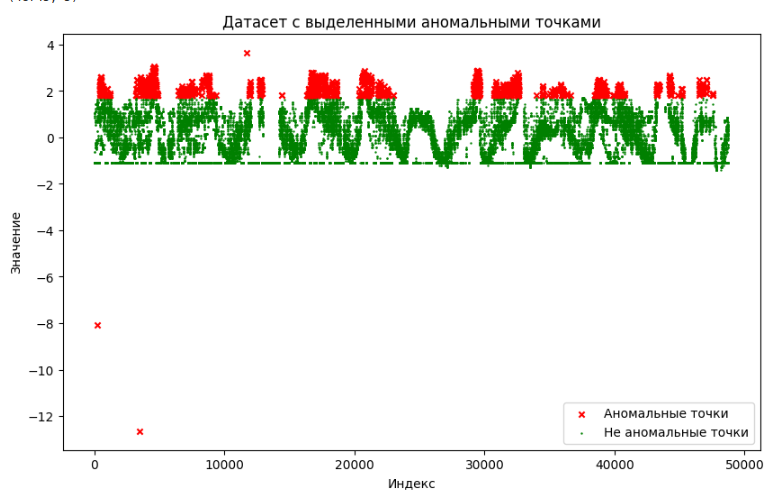


Рисунок 23 – Результат обучения модели SVAD

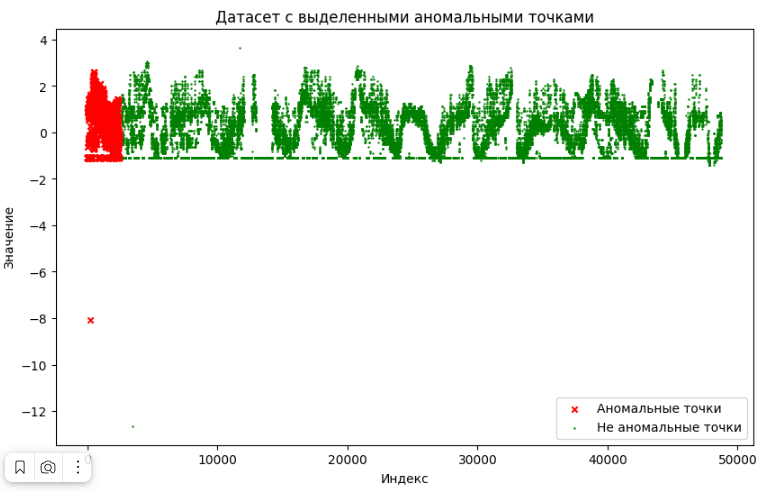


Рисунок 24 – Результат обучения модели SVAD с атрибутом временной меки

Модель convex hull anomaly detection (CHAD) CHAD идентифицирует аномалии на основе их расположения за пределами выпуклой оболочки, охватывающей нормальные точки данных. Эта модель так же обучалась с использованием атрибута временной метки, но не показала оптимальных результатов, как видно на рисунке 22.

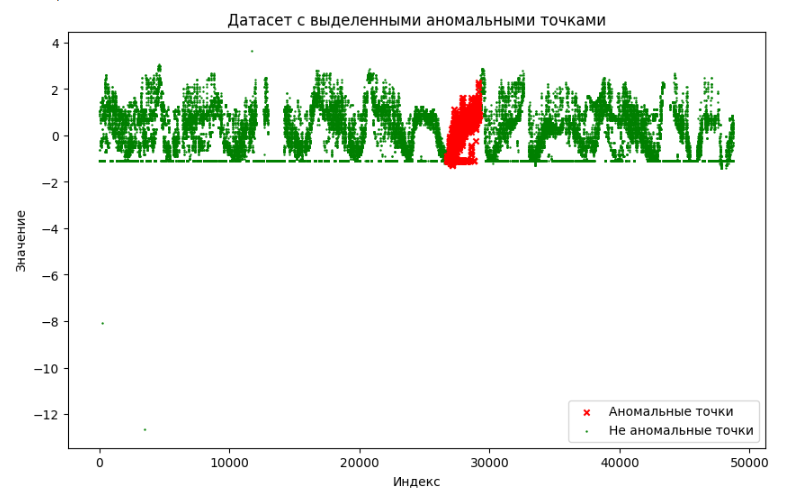


Рисунок 25 – Результат обучения модели CHAD

Модель minimum volume anomaly detection (MVAD) определяет аномалии как объекты, находящиеся в областях данных с наименьшим объемом, разделяя нормальные и аномальные области пространства данных. Эта модель так же определила пороги, а значения, которые за них выходят, как аномальные, что можно увидеть на рисунке 23

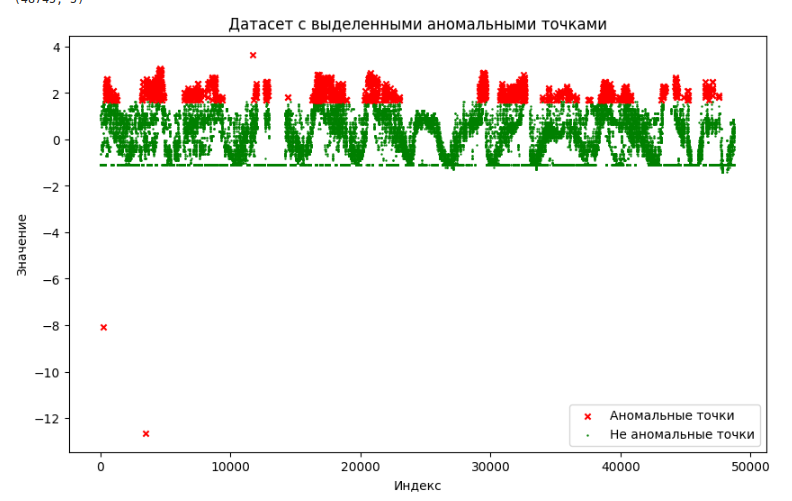


Рисунок 26 – Результат обучения модели MVAD

Модель feature bagging создает ансамбль моделей, используя случайные подмножества признаков для обнаружения аномалий. Использовались модели KNN, Isolation Forest и LSOF. Однако, как можно увидеть, ни результаты без временной метки, представленные на рисунке 24, ни результаты 25, без временной метки, нельзя назвать оптимальными

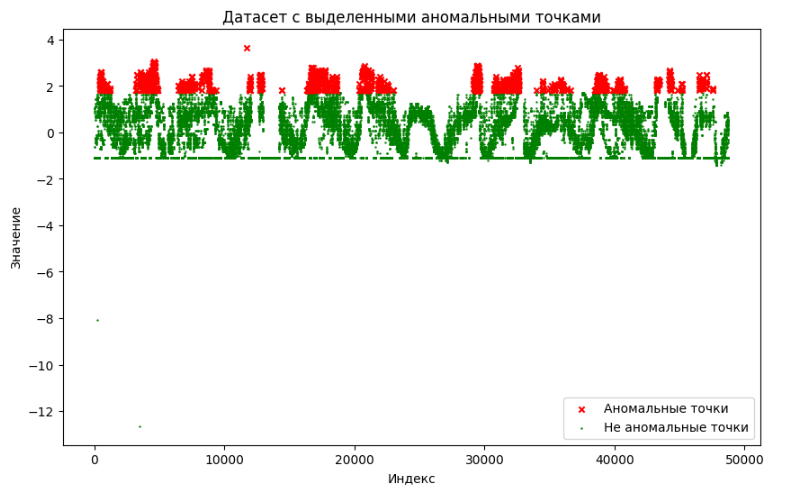


Рисунок 27 – Результат обучения модели future bagging

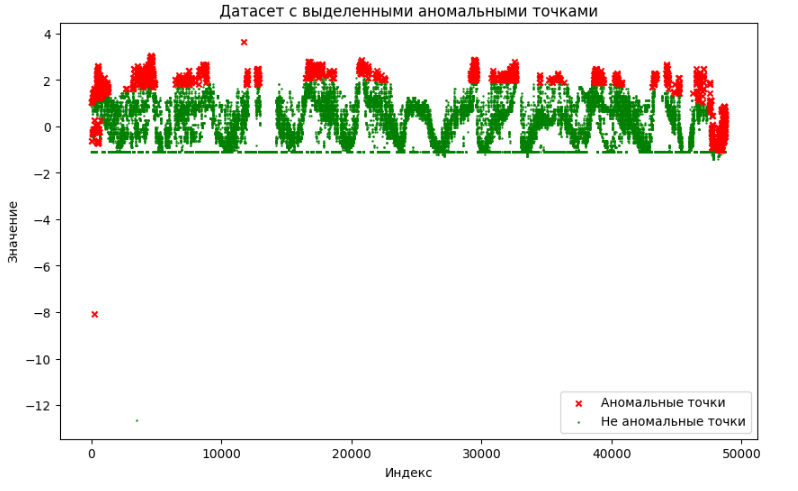


Рисунок 28 – Результат обучения модели future bagging с временной меткой

Модель isolation forest ensemble (IFE) строит лес из изолирующих деревьев решений для быстрого и эффективного обнаружения аномалий путем изоляции аномальных объектов. Так же использование пороговых значений для определения аномалий как с атрибутом временной метки, представленных на рисунке 26, так и без неё, представленных на рисунке 27, не показали оптимальных результатов.

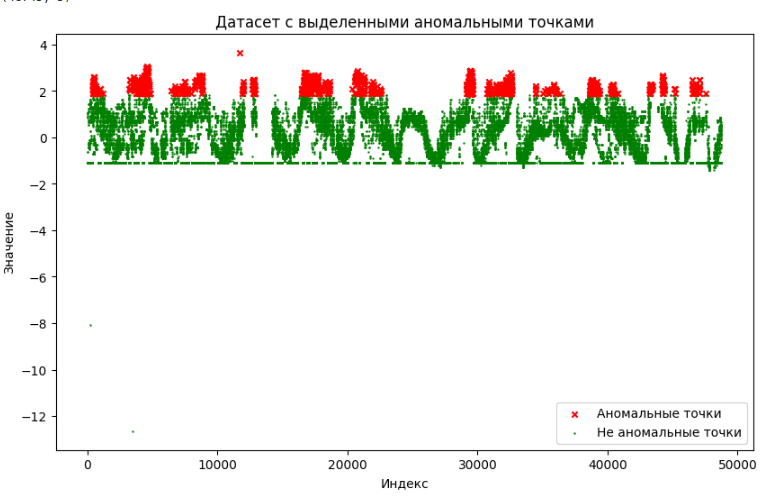


Рисунок 29 – Результат обучения модели IFE

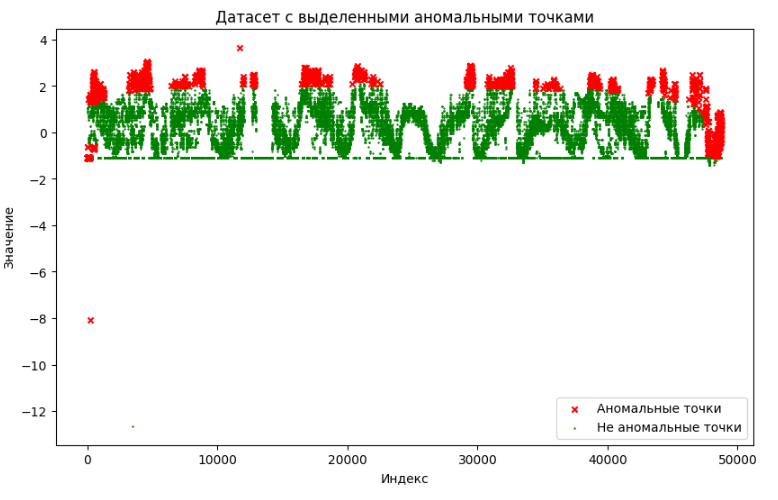


Рисунок 30 – Результат обучения модели IFE с использованием атрибута временной метки

Модель anomaly detection with multiple classifiers (ADMC) использует несколько классификаторов для обнаружения аномалий и объединяет их прогнозы для повышения общей производительности обнаружения аномалий. Использование этой модели показало хороший результат, за исключением того, что модель не помечала аномально низкие значения, что можно увидеть на рисунке 28. Использование временной метки только ухудшило результаты, что можно увидеть на рисунке 29.

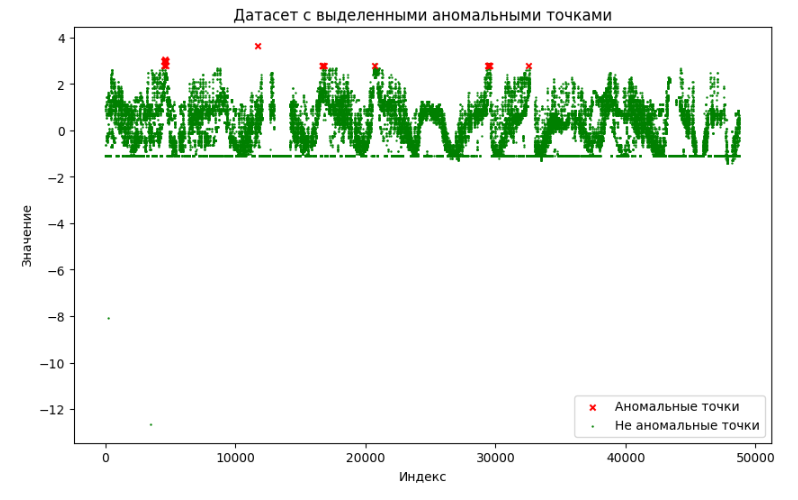


Рисунок 31 – Результат обучения модели ADMC

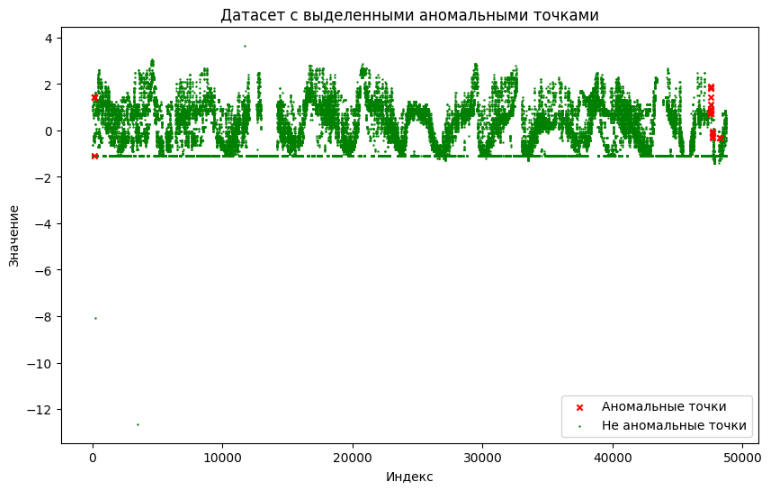


Рисунок 32 – Результат обучения модели ADMC

Модель autoencoder anomaly detection (AAD) использует нейронные сети автокодировщики для восстановления данных и обнаружения аномалий на основе различимежду входными и восстановленными данными. результаты схожи с работой модели LOF, результаты не являются оптимальными. Результаты модели представлены на рисунках 29 и 30, с временной меткой, и без неё.

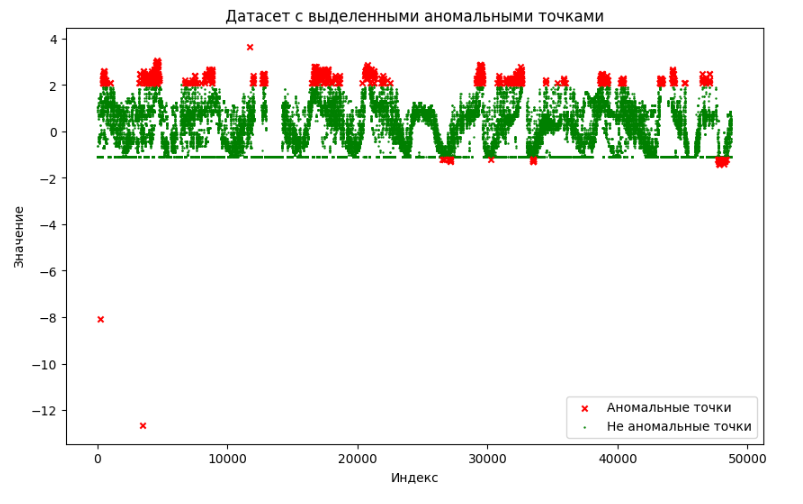


Рисунок 33 – Результат обучения модели AAD

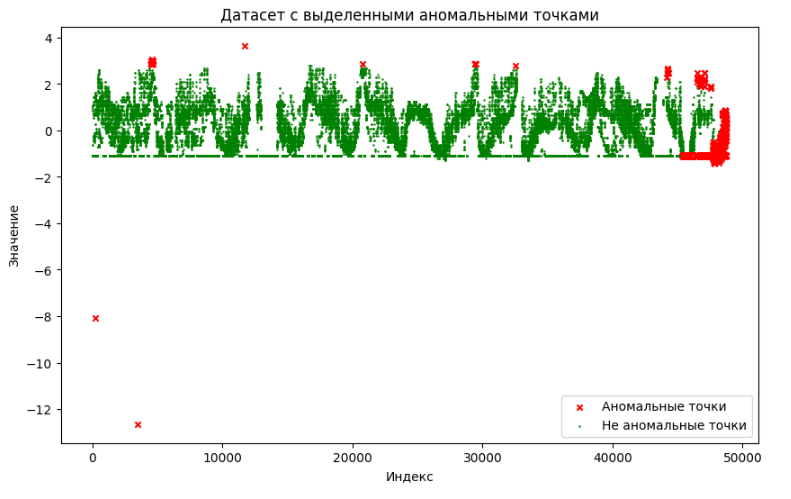


Рисунок 34 – Результат обучения модели AAD с временной меткой

Алгоритм Feedforward Neural Networks представляют собой модель нейронной сети, в которой данные передаются от входного слоя к выходному слою без обратной связи, используемой для решения задач классификации и регрессии. Использование этой модели не показало оптимального результата как без временной метки, так и с ней. Результаты обучения можно увидеть на рисунках 31 и 32.

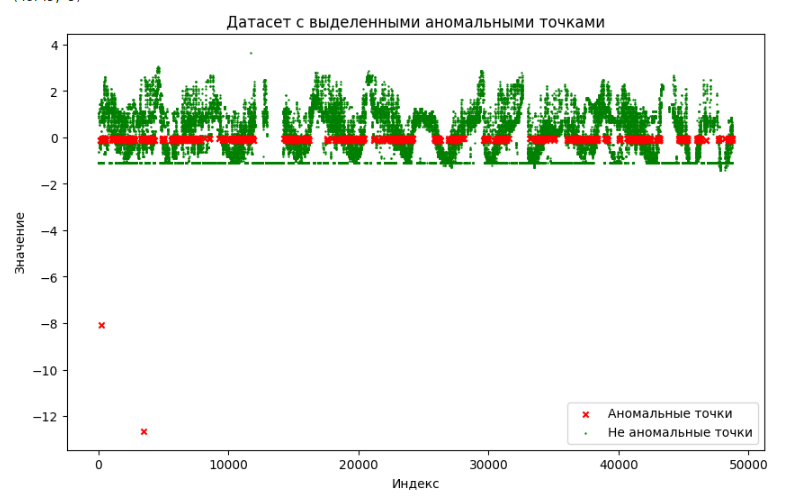


Рисунок 35 – Результат обучения модели Feedforward Neural Networks

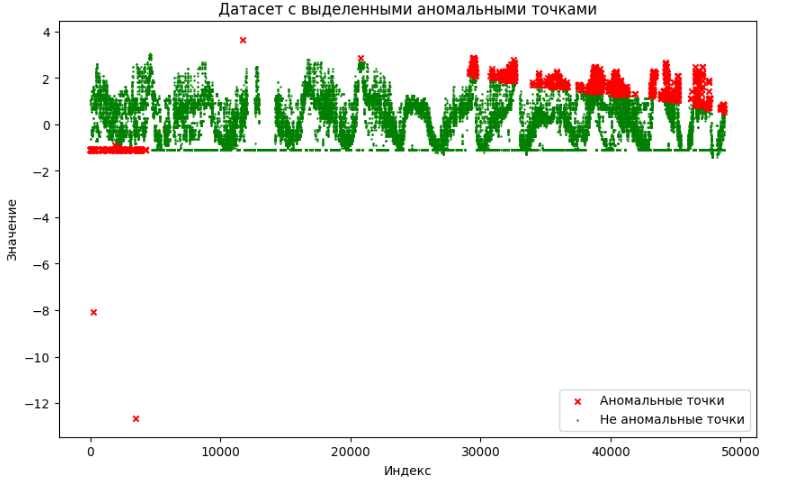


Рисунок 36 – Результат обучения модели Feedforward Neural Networks с использованием временной метки

Generative Adversarial Networks, GANs - это архитектура глубокого обучения, состоящая из генератора и дискриминатора, которые соревнуются друг с другом в процессе обучения для генерации реалистичных данных из случайного шума. Использование этой модели так же не является оптимальным. Результат обучения модели представлен на рисунке 32, с временной меткой, и на рисунке 33, без временной метки.

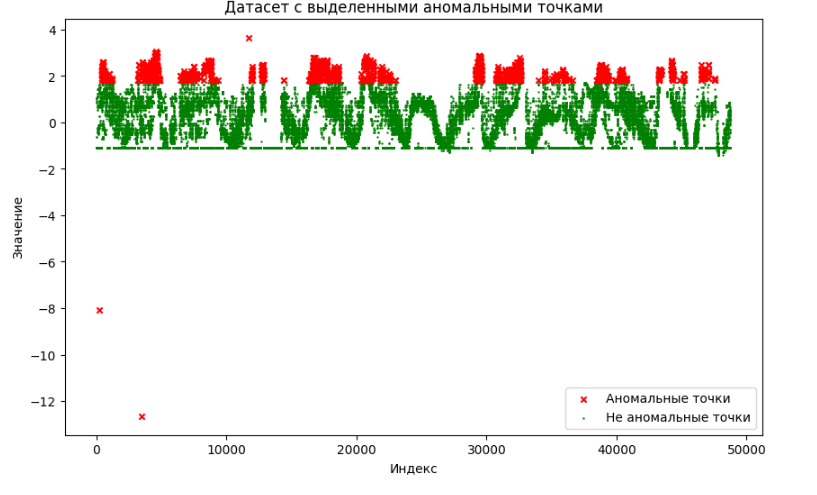


Рисунок 37 – Результат обучения GAN

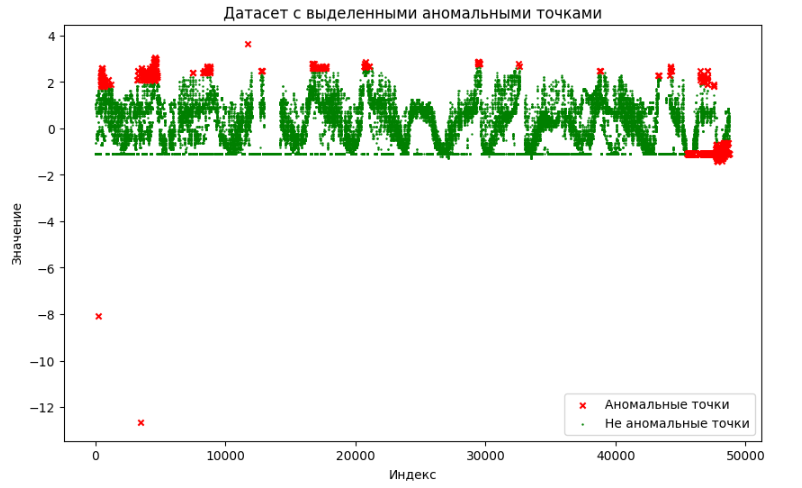


Рисунок 38 – Результат обучения GAN с временной меткой

Recurrent Neural Networks, RNNs это класс нейронных сетей, которые имеют обратные связи между нейронами, позволяющие им моделировать последовательные данные, такие как текст или временные ряды. Она обучалась с использованием временной метки и показала не оптимальный результат, который представлен на рисунке 34.

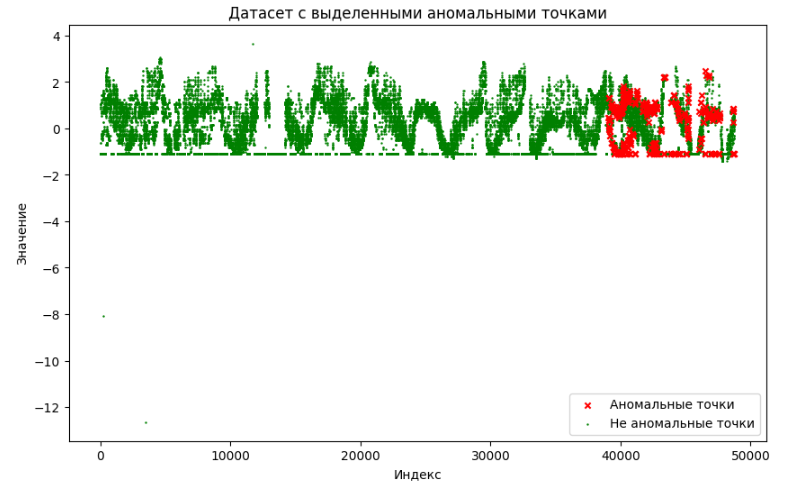


Рисунок 39 – Результат обучения GAN с временной меткой

Convolutional Neural Networks (CNN) - это тип нейронных сетей, специализированных для обработки структурированных данных с сеткой нейронов, имитирующих рецептивные поля нейронов в зрительной коре головного мозга, что делает их особенно эффективными для обработки изображений и видео. Она так же обучалась с временной меткой и показала плохой результат.

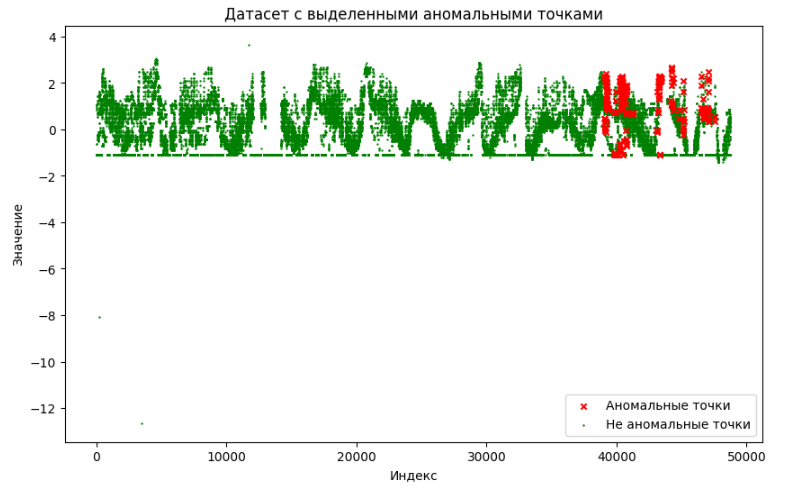


Рисунок 40 – Результат обучения CNN с временной меткой

По итогу рассмотрения моделей был сделан вывод, что наиболее подходящими являются алгоритмы, основанные на кластеризации, так как они лучше всего подходят для неразмеченных данных.

* 1. Определение связи аномалий с космической погодой

Для проверки гипотезы о влиянии космической погоды на телеметрию спутников был осуществлен тщательный отбор количественных показателей, характеризующих рассматриваемый спутник. Затем были выбраны модели и настройки гиперпараметров, которые обеспечили наиболее точные результаты по обнаружению аномалий в телеметрии. Для анализа связи с космической погодой была проведена проверка матрицы корреляции. Эта матрица позволяет исследовать взаимосвязь между каждым отдельным параметром космической погоды и телеметрией спутника. Например, на рисунке 41 представлена матрица корреляции для параметра "t\_amp", который отражает температуру батареи определенной части малого космического аппарата. Подобный анализ помогает определить, какие конкретные аспекты космической погоды могут влиять на работу спутника, а также выявить потенциальные аномалии или зависимости в данных телеметрии.

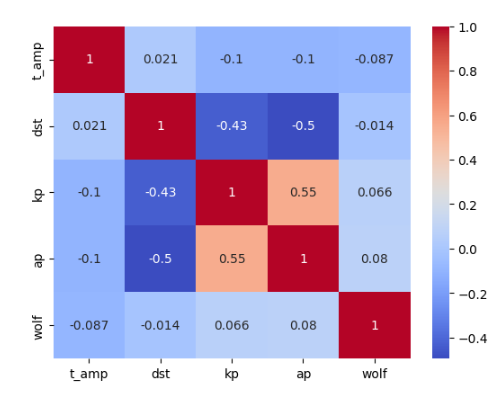


Рисунок 41 – Корреляционная матрица параметра «t\_amp» и космической погоды

Как следует из анализа, коэффициент корреляции между показателями телеметрии и параметрами космической погоды Kp, Ap, Dst, а также индексом интегральной солнечной активности оказался невысоким и составил всего 0.1. Это указывает на то, что между этими переменными практически отсутствует какая-либо значимая корреляция. В результате исследования всех показателей телеметрии космического аппарата "SiriusSat-1" не было обнаружено явных связей или зависимостей с рассмотренными параметрами космической погоды. Это означает, что изменения в космической погоде, представленные указанными параметрами, не оказывают существенного влияния на работу или состояние спутника "SiriusSat-1".

* 1. Выводы исследования

По завершении проведенного исследования пришли к выводу о том, что отсутствует универсальный алгоритм, способный однозначно решить задачу анализа телеметрии спутников. Вместо этого, для эффективного решения прикладных задач в данной области необходимо разработать инструмент, который позволит операторам малых космических аппаратов самостоятельно выбирать оптимальные модели анализа данных телеметрии спутников. Такой подход позволит учитывать специфические потребности и особенности каждого конкретного спутника, а также изменчивость условий космической среды, что в свою очередь способствует более точному и надежному контролю за состоянием и работой космических аппаратов.

**2 Разработка программного средства**

2.1 Проектирование программного средства

В ходе магистерской диссертации будет создано программное средство, состоящее из пяти сервисов, каждый из которых будет выполнять определенные функции:

* сервиса синхронизации космической погоды: отвечает за получение и обновление данных о космической погоде, таких как солнечные вспышки, магнитные бури и другие факторы, которые могут оказывать влияние на работу космических аппаратов;
* сервиса декодирования данных космической телеметрии: отвечает за расшифровку и обработку данных, полученных от космических аппаратов, чтобы сделать их доступными для анализа и дальнейшей обработки;
* сервиса обработки и выделения аномалий космических данных: занимается анализом данных телеметрии с целью выявления аномалий, которые могут свидетельствовать о неполадках или нештатных ситуациях на борту космических аппаратов;
* сервиса визуализации полученных результатов: отвечает за представление анализируемых данных в наглядной форме, такой как графики, диаграммы или трехмерные модели, для облегчения восприятия и анализа операторами космических аппаратов;
* сервиса для просмотра промежуточных результатов: предоставляет возможность операторам просматривать результаты промежуточного анализа данных, что помогает отслеживать процесс обработки и анализа данных в режиме реального времени.

На текущий момент были разработаны и реализованы следующие сервисы: сервис синхронизации космической погоды, сервис обработки и выделения аномалий космических данных и сервис для просмотра промежуточных результатов. Это позволяет продолжить работу над оставшимися сервисами и завершить разработку программного средства в целом.

2.2 Разработка сервиса синхронизации космической погоды

Для разработки сервиса синхронизации космической погоды был выбран язык программирования C# в сочетании с фреймворком ASP.NET Core. ASP.NET Core представляет собой мощный инструмент для создания веб-приложений на языке C#, обладающий высокой производительностью, масштабируемостью и безопасностью. Его кроссплатформенность и открытость делают его привлекательным выбором для разработчиков, и его модульная структура и интеграция с облачными сервисами обеспечивают удобство разработки и развертывания приложений на различных платформах.

Сервис синхронизации космической погоды осуществляет получение данных из ресурса «Центр анализа космической погоды НИИЯФ МГУ» с помощью HTTP запросов к их API. Этот сервис запускается периодически и выполняет запросы к API ресурса для получения актуальной информации о космической погоде. Полученные данные затем используются для дальнейшего анализа и интеграции с другими компонентами программного средства.

Программный модуль, отвечающий за основную логику работы сервиса, представлен в листинге, приложенном к документации. Этот модуль содержит код, который определяет способ получения данных из API ресурса и их обработку для дальнейшего использования в приложении.

2.3 Разработка сервиса обработки данных и выделения аномалий космической погоды

Сервис обработки данных и выделения аномалий космической погоды был реализован с использованием языка программирования Python и фреймворка FastAPI. Python — это мощный и универсальный язык программирования, который известен своей простотой в использовании, читаемостью и гибкостью. Он широко применяется во многих областях, включая веб-разработку, научные исследования, анализ данных, машинное обучение и многое другое благодаря обширной библиотеке и поддержке сообщества разработчиков. FastAPI — это современный и высокопроизводительный фреймворк для создания веб-приложений и API на Python. Он отличается автоматической валидацией запросов, автоматической документацией и поддержкой асинхронных операций. FastAPI обеспечивает легкость создания эффективных и надежных веб-сервисов с минимальным объемом кода, что делает его привлекательным выбором для разработки приложений, требующих высокой производительности и надежности.

Разработанный сервис обеспечивает эффективную обработку данных, выявление аномалий и предоставление результатов для дальнейшего использования в других компонентах программного обеспечения. Для обучения и визуализации промежуточных результатов используются следующие библиотеки:

* Pandas - библиотека для обработки и анализа данных в Python, предоставляющая высокопроизводительные структуры данных и инструменты для работы с временными рядами, таблицами и большими наборами данных.
* NumPy - основная библиотека для вычислительных операций с многомерными массивами и матрицами в Python, обеспечивающая эффективные методы для работы с числовыми данными и выполнения математических операций.
* PyTorch — это фреймворк глубокого обучения с открытым исходным кодом, который обеспечивает гибкость и скорость в разработке и обучении нейронных сетей, поддерживая автоматическое дифференцирование и динамические вычислительные графы.
* Scikit-learn (sklearn) - библиотека машинного обучения в Python, предоставляющая широкий спектр алгоритмов для классификации, регрессии, кластеризации и других задач, а также инструменты для предварительной обработки данных и оценки моделей.
* Matplotlib - основная библиотека визуализации данных в Python, позволяющая создавать различные типы графиков, диаграмм и визуализаций для анализа данных и презентации результатов.
* plt (часть Matplotlib) — это модуль Matplotlib, предоставляющий удобный интерфейс для построения графиков и диаграмм с помощью простых и интуитивных функций, облегчающих визуализацию данных.

Сервис состоит из трех основных компонентов – сервиса обработки данных, сервиса выделения аномалий и сервиса генерации графиков.

Сервис обработки данных занимается подготовкой данных к обработке, сервис выделения аномалий содержит различные модели для выделения аномалий и сервис генерации графиков для создания выходной информации. Листинг этих сервисов представлен в приложении Б, В и Г, соответственно.

2.4 Реализация клиентского сервиса

Приложение, которое представляет собой точку входа, разработано на языке C# и использует фреймворк ASP.NET Core. Это приложение выполняет роль в обработке запросов и раздаче статического контента, который создан с использованием фреймворка Vue.js версии 3. Vue.js 3 — это современный фреймворк на языке JavaScript, который предоставляет разработчикам мощные инструменты для создания динамических и интерактивных пользовательских интерфейсов веб-приложений. Версия 3 фреймворка отличается от предыдущих улучшенной производительностью, более удобным синтаксисом, расширенными возможностями композиции компонентов и более эффективным использованием виртуального DOM.

Таким образом, комбинация языка программирования C#, фреймворка ASP.NET Core для обработки запросов и Vue.js 3 для создания интерфейса делает это приложение привлекательным выбором для разработки современных и отзывчивых веб-приложений.Также были использованы две библиотеки Vue: Vue Router и Vuex. Vue Router — это официальная библиотека маршрутизации для Vue.js, которая позволяет организовать навигацию в одностраничных приложениях с помощью управления URL. Она позволяет определять маршруты, связывать их с компонентами Vue и обеспечивать переходы между ними без перезагрузки страницы. Vuex — это официальная библиотека управления состоянием для Vue.js, которая обеспечивает централизованное хранение данных и управление состоянием приложения. Он упрощает управление сложным состоянием, обеспечивает однонаправленный поток данных и обеспечивает механизмы для мутирования состояния с помощью мутаций и асинхронного изменения состояния с помощью действий.

С помощью данных средств было реализовано веб-приложение, предоставляющее доступ к созданию и настройке наборов данных спутниковой телеметрии, обучению на основе этих данных указанных ранее моделей для детектирования аномалий, а также представление результатов обучения с последующим сравнением результатов. На практике это свелось к реализации четырех компонентов:

- список проектов: компонент веб-приложения, который предоставляет пользователю обзор всех созданных в системе проектов и возможность по созданию новых проектов;

- страница проекта: компонент веб-приложения, который пользователь видит после выбора конкретного проекта из списка проектов. Этот компонент включает в себя обзор результатов обучения, доступ к датасетам и демонстрацию результатов поиска аномалий в последней обученной модели каждого типа;

- страница модели проекта: компонент веб-приложения, который пользователь видит после выбора конкретной модели из списка моделей в рамках выбранного проекта. Этот компонент включает в себя обзор всех результатов обучения модели и возможность сравнения результатов на разных датасетах;

- панель навигации: интерфейсный компонент веб-приложения, который обеспечивает пользователям возможность управления наборами данных и параметрами обучения моделей.

Далее на рисунках 42-44 показаны перечисленные выше компоненты. Листинг этого сервиса представлен в приложении Д.

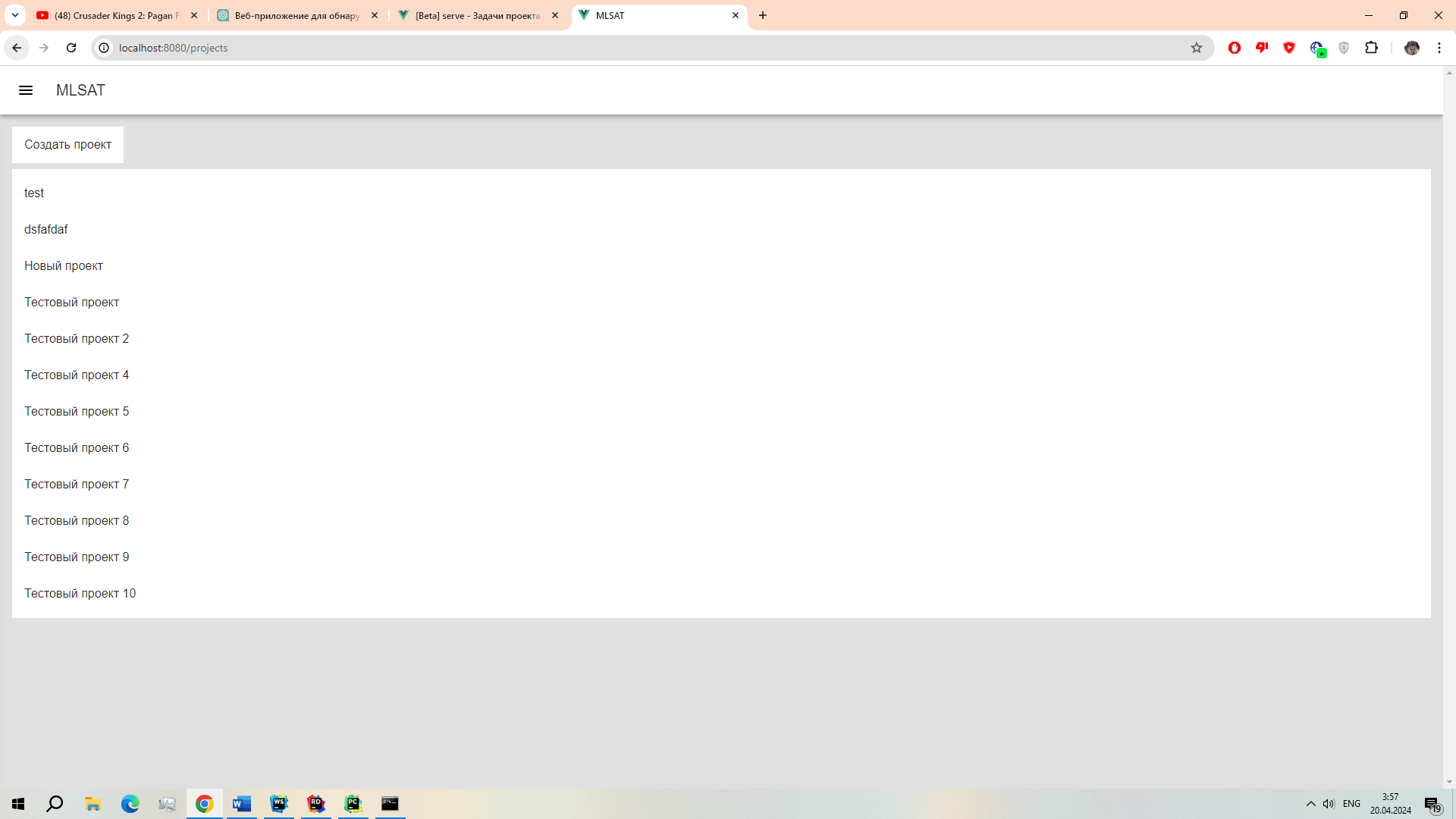


Рисунок 42 – Список проектов по анализу аномалий

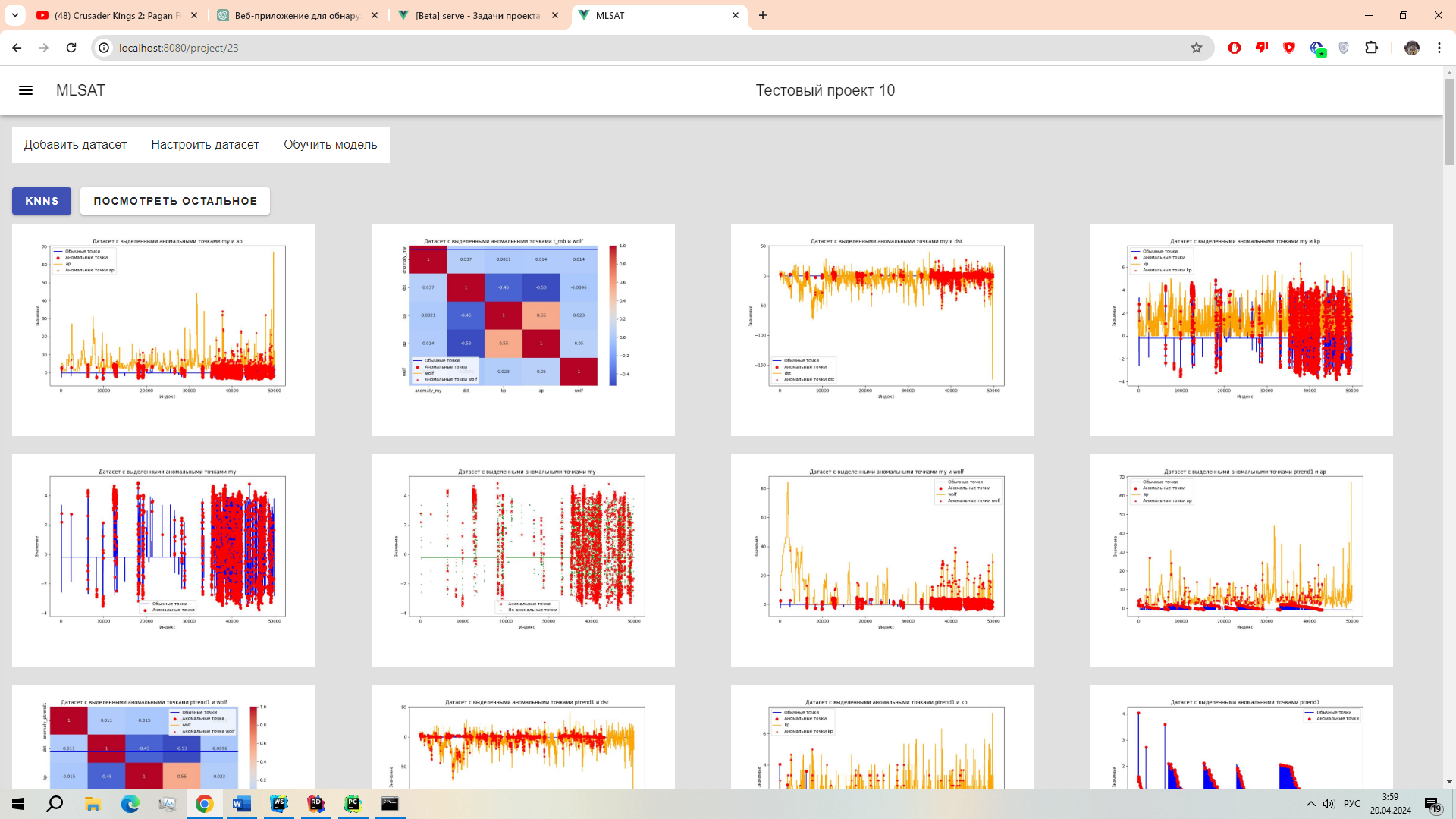


Рисунок 43 – Последние результаты поиска аномалий моделью knn



Рисунок 44 – Все результаты поиска аномалий моделью knn в проекте №10

Функция сравнения результатов обучения по конкретному параметру представляет собой ключевой инструмент анализа данных в системе. При активации этой функции пользователь имеет возможность сравнивать графики, отображающие значения одного и того же параметра в различных контекстах или временных точках.

После нажатия на соответствующую кнопку, которая располагается на развернутом графике с аномалиями, появляется список других графиков, относящихся к тому же параметру. Это позволяет пользователям изучать изменения параметра в различных условиях или сценариях, а также проводить сравнительный анализ для выявления трендов или аномалий.

Например, оператор спутника может использовать эту функцию для сравнения температурных показателей батареи в разные периоды времени или в разных условиях окружающей среды. Такой подход позволяет оперативно выявлять аномалии или нештатные ситуации, что является важным для обеспечения надежной работы космических аппаратов.

Представление результатов анализа в виде списка графиков с одним параметром упрощает визуальное восприятие данных и помогает пользователям более детально изучать характеристики параметров. Это существенно повышает эффективность принятия решений и обеспечивает оперативное реагирование на потенциальные проблемы или аномалии в работе космических аппаратов. Данная функция представлена на рисунке 45.

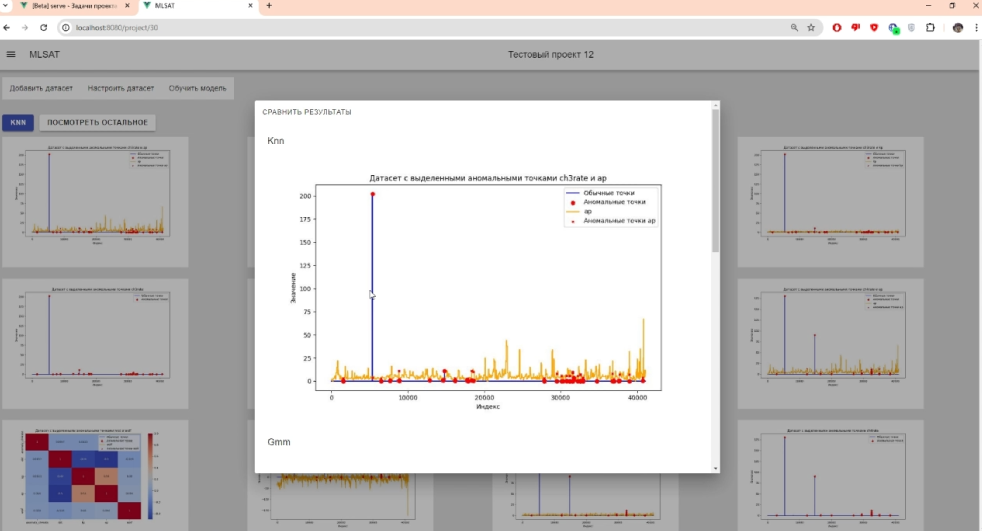


Рисунок 45 – Функция сравнения результата

**3 Структура магистерской диссертации**

3.1 Разработка структуры магистерской диссертации

Магистерская диссертация структурируется на три основные главы, каждая из которых фокусируется на ключевых аспектах исследования:

* проблематика: представляет введение в контекст исследования. Здесь будет детально рассмотрена постановка проблемы, выделены объект и предмет исследования. Важным аспектом этой главы является обоснование актуальности проведения исследования в контексте современных вызовов космической индустрии. Кроме того, будет сформулированы цели и задачи исследования, определяющие общий курс работы;
* изучение моделей выявления космических аномалий. Здесь предполагается провести обзор различных методов и моделей для выявления аномалий в данных космических аппаратов. Это включает в себя рассмотрение как классических, так и современных подходов к обнаружению аномалий в неразмеченных данных. Важным аспектом будет анализ существующих решений, выделение их преимуществ и недостатков, а также оценка их применимости для решения конкретных задач;
* реализация программного решения. Здесь будет представлена архитектура разрабатываемого программного решения, основные компоненты и функциональность приложения. Также будет рассмотрен процесс тестирования и оценки эффективности программного решения, включая его применение на практике;

В заключении раздела о структуре магистерской диссертации можно подчеркнуть важность выбора и организации основных глав и разделов их содержания. Каждая глава направлена на изучение определенных аспектов исследования, начиная с введения в проблематику и заканчивая практической реализацией программного решения.

3.2 Написание отдельных глав диссертации

В рамках научно-исследовательской работы было проведено исследование методов поиска аномалий в неразмеченных данных с учетом специфики космической телеметрии. Это включало анализ различных подходов и моделей для обнаружения аномалий, а также оценку их применимости к конкретным данным космических аппаратов. Особое внимание было уделено влиянию космической погоды на данные телеметрии, что позволило выявить потенциальные связи и факторы, влияющие на работу космических систем.

Исследовательская часть диссертации будет основана на этих аспектах и дополнена анализом результатов исследования. Важно также отметить, что в рамках работы будет реализовано существующее решение для вывода промежуточных результатов и его функциональность будет подробно описана и проанализирована.

Заключение данного раздела подчеркнет значимость проведенного анализа и исследования, а также обозначит перспективы дальнейших исследований и разработок в данной области.

Выводы

Наша работа представляет собой результат многогранного исследования, охватывающего различные аспекты анализа данных и разработки программного обеспечения в контексте космических приложений. Процесс проведения исследования позволил нам не только углубиться в понимание сложных технических аспектов, но и развить навыки работы с разнообразными методами и инструментами.

В ходе нашей работы мы приступили к изучению различных методов машинного обучения и анализа данных, применяемых для обнаружения аномалий. Этот этап оказался весьма разнообразным, включая как классические статистические методы, так и современные техники глубокого обучения. Мы провели обширный обзор литературы, изучив работы исследователей, посвященные этой проблематике, а также провели сравнительный анализ различных подходов. Такой глубокий анализ позволил нам лучше понять преимущества и ограничения каждого метода и определить их применимость в контексте нашей конкретной задачи.

Помимо этого, мы также провели исследование влияния космической погоды на работу космических систем. Этот аспект работы требовал от нас не только анализа данных, но и погружения в сложную тему космической физики и астрономии. Мы изучили различные параметры космической погоды и их влияние на работу космических аппаратов, включая их телеметрию. Результаты этого исследования имели важное значение для понимания возможных причин аномалий в данных и определения стратегий их обнаружения и предотвращения.

Однако наша работа не ограничивалась только теоретическим анализом. Мы также разработали и реализовали программное решение, включающее в себя микросервисную архитектуру и широкий спектр функциональных возможностей. Это включало в себя сервисы для синхронизации космической погоды, обработки данных и выявления аномалий, визуализации результатов и просмотра промежуточных данных. Разработка этого решения требовала от нас не только знаний в области программирования, но и понимания специфики космических систем и требований к их надежности и эффективности.

Таким образом, наше исследование не только позволило нам расширить наши знания и навыки в области анализа данных и разработки программного обеспечения, но и внесло важный вклад в научное понимание проблемы обнаружения аномалий в данных телеметрии космических аппаратов. Полученные результаты могут быть использованы как в дальнейших исследованиях и разработках в этой области, так и в практических приложениях, направленных на улучшение надежности и эффективности космических миссий и систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Беневольский С.В., Майорова В.И., Гришко Д.А., Ханеня Н.Н. Анализ телеметрии с космического аппарата" Юбилейный" //Машиностроение и компьютерные технологии. – 2011. – №. 13. – С. 59.

2. Романова Н.В., Пилипенко В.А., Ягова Н.В., Белов А.В. Статистическая связь частоты сбоев на геостационарных спутниках с потоками энергичных электронов и протонов // Космические исследования. – 2005. – Т. 43, № 3. – С. 186-193.

3. Pilipenko V., Yagova N., Romanova N., Allen J. Statistical relationships between satellite anomalies at geostationary orbit and high-energy particles // Advances in Space Research 37. – 2006. – С. 1192-1205.

4. Ефиторов А.О., Мягкова И.Н., Широкий В.Р., Доленко С.А. Прогнозирование Dst-индекса, основанное на методах машинного обучения //Космические исследования. – 2018. – Т. 56. – №. 6. – С. 420-428.

5. Camporeale E. The challenge of machine learning in space weather: Nowcasting and fore-casting //Space Weather. – 2019. – Т. 17. – №. 8. – С. 1166-1207.

6. Тасенко С.В., Чиженков В.А., Шатов П.В., Скороходов И.А. Перспективная система геофизического обеспечения управления космическими аппаратами в условиях высокой солнечной активности //Известия Южного федерального университета. Технические науки. – 2012. – Т. 128. – №. 3. – С. 234-241.

7 Морозов Р.В. Жучков Д. В., Межсистемное взаимодействие разнородных информационных систем [Электронный ресурс] / Электронные данные - [Москва] : Cyberleninka, 2012-2022 Режим доступа: <https://cyberleninka.ru/article/n/mezhsistemnoe-vzaimodeystvie-raznorodnyh-informatsionnyh-sistem/viewer>, свободный (06.12.2022). – Загл. с экрана.

8. Выбираем СУБД для хранения временных рядов [Электронный ресурс] / habr.com – Электрон дан. – [Москва] habr.com, 2002-2022, Режим доступа https://habr.com/ru/companies/oleg-bunin/articles/329062/, свободный (дата обращения – 03.05.2023). – Загл. с экрана.

9 Parkhomenko D.A.Data vizualization makes sense of big data [Electronic resource] / Big data and advanced analytics - Electronic data - [Moscow] : E-Library, 2000-2022 Mode of access: https://www.elibrary.ru/item.asp?id=46482839, free access (08.12.2022). – Title from screen.

10 Ершов К.С. Романова Т.Н., Анализ и классификация алгоритмов кластеризации [Электронный ресурс] / Электронные данные - [Москва] : Cyberleninka, 2012-2024 Режим доступа: <https://cyberleninka.ru/article/n/analiz-i-klassifikatsiya-algoritmov-klasterizatsii/viewer>, свободный (19.04.2024). – Загл. с экрана.

11 Клышинский Э.С. Метод кластеризации на основе анализа плотности точек [Электронный ресурс] / Электронные данные - [Москва] : Cyberleninka, 2012-2024 Режим доступа: <https://cyberleninka.ru/article/n/metod-klasterizatsii-na-osnove-analiza-plotnosti-tochek/viewer>, свободный (19.04.2024). – Загл. с экрана.

Приложение А  
(обязательное)  
Модуль для синхронизации космической погоды

using System.Globalization;  
using Mlsat.Models.Entities.SpaceWeather;  
using Mlsat.Services.SpaceWeatherServices.Dtos;  
using Newtonsoft.Json;  
  
namespace Mlsat.Services.SpaceWeatherServices;  
  
public class SpaceWeatherService  
{  
 private readonly HttpClient \_httpClient;  
 private const string **Uri** = "https://swx.sinp.msu.ru/db\_iface/api/v1/query";  
 private const string **Url** = "https://swx.sinp.msu.ru/gk0049/select\_bd\_ps.php?datetime={0}";  
 private static readonly object Plug = new();  
  
 public SpaceWeatherService()  
 {  
 \_httpClient = new HttpClient();  
 }  
  
 public async Task<(  
 IReadOnlyCollection<Ap> aps,  
 IReadOnlyCollection<Kp> kps,  
 IReadOnlyCollection<Dst> dsts,  
 IReadOnlyCollection<Wolf> wolfs)> GetSpaceWeather(DateTime startDate, DateTime endDate)  
 {  
 var aps = new List<Ap>();  
 var kps = new List<Kp>();  
 var dsts = new List<Dst>();  
 var wolfs = new List<Wolf>();  
  
 while (startDate < endDate)  
 {  
 var rangeStart = startDate;  
 var rangeEnd = startDate.AddDays(10);  
  
 var mainSpaceWeather = await GetMainSpaceWeather(rangeStart, rangeEnd);  
 var wolf = await GetWolf(rangeStart, rangeEnd);  
  
 aps.AddRange(mainSpaceWeather.Ap);  
 kps.AddRange(mainSpaceWeather.Kp);  
 dsts.AddRange(mainSpaceWeather.Dst);  
 wolfs.AddRange(wolf);  
  
 startDate = rangeEnd;  
 }  
  
 return (aps, kps, dsts, wolfs);  
 }  
  
 private async Task<MainSpaceWeatherDto> GetMainSpaceWeather(DateTime

startDate, DateTime endDate)  
 {  
 var requestDto = new MainSpaceWeatherRequestDto(startDate, endDate);  
 using var request = new HttpRequestMessage(HttpMethod.Post, **Uri**);  
 var stringContent = JsonConvert.SerializeObject(requestDto);  
 var content = new StringContent(stringContent);  
 request.Content = content;  
 using var response = await \_httpClient.SendAsync(request);  
 var responseText = await response.Content.ReadAsStringAsync();  
 var result = JsonConvert.DeserializeObject<MainSpaceWeatherResultDto>(responseText);  
  
 var x = result!.GetMainSpaceWeather();  
  
 return x;  
 }  
  
 public async Task<IEnumerable<Wolf>> GetWolf(DateTime startDate, DateTime endDate)  
 {  
 var spaceWeatherPoints = new LinkedList<Wolf>();  
  
 var dates = new List<DateTime>();  
  
 for (var dt = startDate; dt <= endDate; dt = dt.AddDays(1))  
 {  
 dates.Add(dt);  
 }  
  
 var tasks = new List<Task>();  
  
 foreach (var date in dates)  
 {  
 var taskDate = date;  
 tasks.Add(Task.Run(async () => await AddPoint(taskDate, spaceWeatherPoints)));  
 }  
  
 await Task.WhenAll(tasks);  
  
 return spaceWeatherPoints;  
 }  
  
 private async Task AddPoint(DateTime initDate, LinkedList<Wolf> points)  
 {  
 try  
 {  
 var date = initDate.ToString("yyyy/MM/dd hh:mm:ss", CultureInfo.InvariantCulture);  
 var dateQuery = date.Replace(" ", "%20");  
 var query = string.Format(**Url**, dateQuery);  
 using var request = new HttpRequestMessage(HttpMethod.Get, query);  
 using var response = await \_httpClient.SendAsync(request);  
 var responseText = await response.Content.ReadAsStringAsync();  
 var wolfResultDto = JsonConvert.DeserializeObject<WolfResultDto>(responseText);

for (var i = 0; i < wolfResultDto!.datetime.Length; i++)  
 {  
 var value = wolfResultDto.wolfnumber[i];  
  
 if (value == null) continue;  
  
 lock (Plug)  
 {  
 points.AddLast(new Wolf  
 {  
 Date = wolfResultDto!.datetime[i],  
 Value = value  
 });  
 }  
 }  
 }  
 catch  
 {  
 }  
 }  
}

Приложение Б  
(обязательное)  
Модуль обработки данных

import pandas as pd  
from typing import List, Dict, Optional  
import numpy as np  
from datetime import datetime  
from pydantic import BaseModel  
from sklearn.preprocessing import StandardScaler  
import os  
  
base\_directory = './python/space\_\_weathers'  
  
  
class SpaceWeatherPoint(BaseModel):  
 points: Dict[datetime, float]  
  
  
class DatasetSettings(BaseModel):  
 path: str  
 savePath: str  
 isNormalize: bool  
 dropna: bool  
 timeColumn: str = ''  
 columns: List[str] = [],  
 dst: Optional[Dict[datetime, float]] = None  
 kp: Optional[Dict[datetime, float]] = None  
 ap: Optional[Dict[datetime, float]] = None  
 wolf: Optional[Dict[datetime, float]] = None  
  
  
class DatasetService:  
 def create\_folder\_if\_not\_exist(self, path):  
 os.makedirs(path, exist\_ok=True)  
  
 def get\_columns(self, path):  
 df = pd.read\_csv(path)  
 return df.columns.tolist()  
  
 def get\_dates(self, path, time\_column):  
 df = pd.read\_csv(path)  
 self.convert\_values(df, time\_column)  
 return {  
 "startDate": df[time\_column].min(),  
 "endDate": df[time\_column].max()  
 }  
  
 def convert\_values(self, df, time\_column):  
 df[time\_column] = pd.to\_datetime(df[time\_column]).dt.tz\_localize(None)  
 df = df.sort\_values(by=time\_column)  
 df.set\_index(time\_column, inplace=True)  
  
 def add\_space\_weather(self, data, df, w, time\_column):  
 self.convert\_values(df, time\_column)  
 data[w] = None  
 for index, row in data.iterrows():  
 nearest\_row = df.iloc[(df[time\_column] - row[time\_column]).abs().argsort()[:1]]  
 nearest\_value = nearest\_row['Value'].values[0] if not

nearest\_row.empty else None

data.at[index, w] = nearest\_value  
  
 def get\_points(self, points: Dict[datetime, float], time\_column):  
 response = {  
 f'{time\_column}': [],  
 'Value': []  
 }  
  
 for key, value in points.items():  
 response[f'{time\_column}'].append(key)  
 response['Value'].append(value)  
  
 df = pd.DataFrame(response)  
  
 return df  
  
 def load\_wolf(self, df, time\_column, points: Dict[datetime, float]):  
 wolf = self.get\_points(points, time\_column)  
 self.add\_space\_weather(df, wolf, 'wolf', time\_column)  
  
 def load\_ap(self, df, time\_column, points: Dict[datetime, float]):  
 ap = self.get\_points(points, time\_column)  
 self.add\_space\_weather(df, ap, 'ap', time\_column)  
  
 def load\_dst(self, df, time\_column, points: Dict[datetime, float]):  
 dst = self.get\_points(points, time\_column)  
 self.add\_space\_weather(df, dst, 'dst', time\_column)  
  
 def load\_kp(self, df, time\_column, points: Dict[datetime, float]):  
 kp = self.get\_points(points, time\_column)  
 self.add\_space\_weather(df, kp, 'kp', time\_column)  
  
 def minimal\_processing(self, df, time\_column):  
 column\_types = df.dtypes  
 drop\_columns = []  
  
 for column, dtype in column\_types.items():  
 if column == time\_column:  
 continue  
 if dtype == 'object':  
 drop\_columns.append(column)  
 continue  
  
 df[column] = df[column].astype(float)  
  
 if len(drop\_columns) != 0:  
 df = df.drop(columns=[drop\_columns])  
  
 df.fillna(0, inplace=True)  
  
 def save\_csv(self, df, column, save\_path):  
 directory\_path = f'{save\_path}\\{column}'  
 self.create\_folder\_if\_not\_exist(directory\_path)  
 path = f'{directory\_path}\\{column}\_anomaly.csv'  
 df.to\_csv(path)  
  
  
 def create\_on\_basis(self, settings: DatasetSettings):  
 scaler = StandardScaler()  
 hasTimeColumn = len(settings.timeColumn) > 0

base\_data = pd.read\_csv(settings.path)  
  
 df = base\_data[settings.columns]  
  
 if hasTimeColumn:  
 df[settings.timeColumn] = base\_data[settings.timeColumn]  
  
 column\_types = df.dtypes  
  
 for column, dtype in column\_types.items():  
 if dtype == 'object' and column == settings.timeColumn:  
 df[column] = pd.to\_datetime(df[column])  
 df[column] = df[column].dt.strftime('%Y-%m-%d %H:%M:%S')  
 continue  
 elif dtype == 'object':  
 df = df.drop(columns=[column])  
 continue  
 elif dtype == 'int64':  
 df[column] = df[column].astype(float)  
 if settings.isNormalize:  
 df[column] = scaler.fit\_transform(df[[column]])  
  
 if settings.dropna:  
 df.dropna(axis=1, how='any', inplace=True)  
 else:  
 df.fillna(0, inplace=True)  
  
 if hasTimeColumn:  
 self.convert\_values(df, settings.timeColumn)  
  
 if hasTimeColumn and settings.wolf != None:  
 self.load\_wolf(df, settings.timeColumn, settings.wolf)  
  
 if hasTimeColumn and settings.ap != None:  
 self.load\_ap(df, settings.timeColumn, settings.ap)  
  
 if hasTimeColumn and settings.kp != None:  
 self.load\_kp(df, settings.timeColumn, settings.kp)  
  
 if hasTimeColumn and settings.dst != None:  
 self.load\_dst(df, settings.timeColumn, settings.dst)  
  
 df.to\_csv(settings.savePath)  
  
 return settings  
 def prepareData(self, data, column, time\_column, space\_wather\_columns):  
 columns = []  
 columns.append(column)  
 if len(time\_column) != 0:  
 columns.append(time\_column)  
  
 for item in space\_wather\_columns:  
 columns.append(item)  
  
 df = data[columns]  
 df['index'] = df.reset\_index().index + 1  
 df.rename(columns={time\_column: 'timestamp'}, inplace=True)  
 df['timestamp'] = pd.to\_datetime(df['timestamp'])  
 df['time\_epoch'] = (df['timestamp'].astype(np.int64) / 100000000000).astype(np.int64)  
 return df

Приложение В  
(обязательное)  
Модуль выделения аномалий

import pandas as pd

from pydantic import BaseModel

from typing import List

import numpy as np

from sklearn.neighbors import NearestNeighbors

from services.plot\_service.PlotService import PlotService

from services.dataset\_service.DatasetService import DatasetService

class KnnSettings(BaseModel):

datasetPath: str

savePath: str

timeColumn: str

nNeighbors: int = 5

algorithm: str

percentile: int = 95

spaceWeatherColumns: List[str] = []

columns: List[str] = []

class KnnService:

def \_\_init\_\_(self) -> None:

self.plot\_service = PlotService()

self.data\_service = DatasetService()

super().\_\_init\_\_()

def knn(self, df, column, settings: KnnSettings):

x = df[[column]]

# модель

knn\_model = NearestNeighbors(n\_neighbors=settings.nNeighbors, algorithm=settings.algorithm)

knn\_model.fit(x)

distances, \_ = knn\_model.kneighbors(x)

k\_distance = distances[:, -1]

threshold = np.percentile(k\_distance, settings.percentile)

# модель

anomalies\_indices = np.where(k\_distance > threshold)[0]

df[f'anomaly\_{column}'] = 0

df.loc[anomalies\_indices, f'anomaly\_{column}'] = 1

if len(settings.spaceWeatherColumns) != 0:

self.plot\_service.show\_corr\_matrix(df,

f'anomaly\_{column}',

settings.spaceWeatherColumns,

column,

settings.savePath)

self.plot\_service.print\_scutter\_only(df, anomalies\_indices, column, settings.savePath)

self.plot\_service.print\_plot(df, anomalies\_indices, column, settings.savePath)

for space\_weather in settings.spaceWeatherColumns:

self.plot\_service.print\_plot(df, anomalies\_indices, column, settings.savePath, space\_weather)

self.data\_service.save\_csv(df, column, settings.savePath)

def learn(self, settings: KnnSettings):

data = pd.read\_csv(settings.datasetPath)

self.data\_service.minimal\_processing(data, settings.timeColumn)

for column in settings.columns:

df = self.data\_service.prepareData(data, column, settings.timeColumn, settings.spaceWeatherColumns)

self.knn(df, column, settings)

Приложение Г  
(обязательное)  
Модуль выходных параметров

import os

import matplotlib

matplotlib.use('agg')

from matplotlib import pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

plt.ioff()

class PlotService:

def create\_folder\_if\_not\_exist(self, path):

os.makedirs(path, exist\_ok=True)

def print\_scutter\_only(self, df, anomaly\_indices, column, save\_path):

all\_indices = set(df.index)

non\_anomaly\_indices = list(all\_indices.difference(anomaly\_indices))

plt.figure(figsize=(10, 6))

plt.scatter(anomaly\_indices, df.iloc[anomaly\_indices][column], color='red', marker='x', zorder=5, s=20,

label='Аномальные точки')

plt.scatter(non\_anomaly\_indices, df.iloc[non\_anomaly\_indices][column], color='green', zorder=1, s=0.5,

label='Не аномальные точки')

plt.xlabel('Индекс')

plt.ylabel('Значение')

plt.title(f'Датасет с выделенными аномальными точками {column}')

plt.legend()

directory\_path = f'{save\_path}\\{column}'

path = f'{directory\_path}\\{column}\_scutter.jpg'

self.create\_folder\_if\_not\_exist(directory\_path)

plt.savefig(path)

def show\_corr\_matrix(self, df, anomaly, space\_weather\_columns, column, save\_path):

columns = [anomaly]

sns.reset\_defaults()

for c in space\_weather\_columns:

columns.append(c)

for\_corr\_matrix = df[[column for column in columns]]

correlation\_matrix = for\_corr\_matrix.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

directory\_path = f'{save\_path}\\{column}'

path = f'{directory\_path}\\{column}\_corr\_matrix.jpg'

self.create\_folder\_if\_not\_exist(directory\_path)

plt.savefig(path)

def print\_plot(self, df, anomaly\_indices, column, save\_path, space\_weather = None):

plt.figure(figsize=(10, 6))

plt.plot(df.index, df[column], color='blue', label='Обычные точки')

plt.scatter(anomaly\_indices, df.iloc[anomaly\_indices][column], color='red', zorder=3,

label='Аномальные точки')

if space\_weather != None:

plt.plot(df.index, df[space\_weather], color='orange', label=space\_weather)

plt.scatter(anomaly\_indices,

df.iloc[anomaly\_indices][space\_weather],

color='red',

marker='x',

zorder=2,

s=10,

label=f'Аномальные точки {space\_weather}')

plt.xlabel('Индекс')

plt.ylabel('Значение')

if space\_weather == None:

plt.title(f'Датасет с выделенными аномальными точками {column}')

else:

plt.title(f'Датасет с выделенными аномальными точками {column} и {space\_weather}')

plt.legend()

directory\_path = f'{save\_path}\\{column}'

if space\_weather != None:

path = f'{directory\_path}\\{column}\_{space\_weather}\_plot.jpg'

else:

path = f'{directory\_path}\\{column}\_plot.jpg'

self.create\_folder\_if\_not\_exist(directory\_path)

plt.savefig(path)

Приложение Д  
(обязательное)  
Модуль визуализации результатов

<template>  
 <v-container fluid>  
 <v-row class="mx-0 my-0" height="inherit">  
 <v-list height="inherit" class="d-flex px-0 py-0">  
 <template v-for="(item, index) in menuItems" :key="index">  
 <v-list-item class="text" @click="handleMenuItemClick(item)" :title="item.title"/>  
 </template>  
 </v-list>  
 </v-row>  
 <v-row v-if="models !== undefined">  
 <template v-for="(model, n) in *Object*.keys(models)" :key="n">  
 <v-col class="mt-2" cols="12">  
 <v-menu  
 :value="menuActive === n"  
 :close-on-content-click="false"  
 location="bottom"  
 :activator="`#category${n}`" >  
 <template v-slot:activator="{ on }">  
 <v-btn  
 :id="`category${n}`"  
 color="indigo"  
 @click="toggleMenu(model)" >  
 <strong>{{ getModelName(n) }}</strong>  
 </v-btn>  
 <v-btn  
 :id="`category${n}`"  
 color="second"  
 @click="openModelsView(getModelName(n))"  
 class="ml-3" >  
 <strong>Посмотреть остальное</strong>  
 </v-btn>  
 </template>  
 <v-card min-width="300">  
 <v-card-title>Параметры последней модели</v-card-title>  
 <v-list>  
 <template v-for="(value, key) in modelData" :key="key">  
 <v-list-item v-if="key !== 'id'">  
 {{ key }}: {{ value }}  
 </v-list-item>  
 </template>  
 </v-list>  
 </v-card>  
 </v-menu>  
 </v-col>  
 <template v-if="models[model]">  
 <v-row class="justify-start px-3">  
 <v-col  
 v-for="(path) in models[model][0].graphics"  
 :key="`${model}${path}`"  
 cols="12"  
 sm="6"  
 md="3" >  
 <v-sheet height="280" width="400">  
 <v-img  
 :src="`https://localhost:7268${path}`"  
 @click="openImageDialog(`https://localhost:7268${path}`)" />  
 </v-sheet>  
 </v-col>  
 </v-row>  
 </template>  
 </template>  
 </v-row>  
 <v-dialog v-model="addDataset" persistent width="auto">  
 <v-card width="600px">  
 <v-card-title>Добавить датасет</v-card-title>  
 <v-card-text>  
 <v-text-field label="Название датасета" v-model="text"/>  
 <v-text-field label="Колонка времени" v-model="timeColumn"/>  
 <v-file-input v-model="addedDataset"/>  
 </v-card-text>  
 <v-card-actions>  
 <v-btn @click="addDataset=false">  
 Отмена  
 </v-btn>  
 <v-btn @click="addNewDataset()">  
 Добавить  
 </v-btn>  
 </v-card-actions>  
 </v-card>  
 </v-dialog>  
 <v-dialog v-model="customDataset" persistent width="auto">  
 <v-card width="600px">  
 <v-card-title>Настроить датасет</v-card-title>  
 <v-card-text>  
 <v-text-field label="Название нового датасета" v-model="text"/>  
 <v-select v-model="selectedDataset"  
 label="Выберите датасет"  
 :items="projectDatasets"  
 @update:modelValue="acceptBaseDataset(selectedDataset)"  
 :item-props="itemProps"/>  
 <v-select label="Выберите колонку времени" v-model="timeColumn" :items="availableColumns"/>  
 <v-row>  
 <v-col v-for="(value, key) in checkboxes" :key="key" cols="3">  
 <v-checkbox v-model="checkboxes[key]" :label="key"/>  
 </v-col>  
 </v-row>  
 <v-select  
 v-model="selectedColumns"  
 :items="availableColumns"  
 label="Выберите колонки датасета"  
 multiple  
 chips/>  
 </v-card-text>  
 <v-card-actions>  
 <v-btn @click="customDataset=false">  
 Отмена  
 </v-btn>  
 <v-btn @click="addNewCustomDataset()">  
 Добавить  
 </v-btn>  
 </v-card-actions>  
 </v-card>  
 </v-dialog>  
 <v-dialog v-model="newModel" persistent width="auto">  
 <v-card width="600px">  
 <v-card-title>Укажите данные и гиперпараметры модели</v-card-title>  
 <v-card-text>  
 <v-select v-model="selectedModel"  
 label="Выберите модель"  
 :items="availableModels"/>  
 <v-select v-model="selectedDataset"  
 label="Выберите датасет"  
 :items="projectDatasets"  
 @update:modelValue="acceptBaseDataset(selectedDataset)"  
 :item-props="itemProps"/>  
 <v-select label="Выберите колонку времени" v-model="timeColumn" :items="availableColumns"/>  
 <v-select  
 v-model="selectedColumns"  
 :items="availableColumns"  
 label="Выберите колонки датасета"  
 multiple  
 chips/>  
 </v-card-text>  
 <div class="d-flex justify-center">  
 <v-progress-circular indeterminate v-if="buttonDisabled"/>  
 </div>  
 <v-card-actions>  
 <v-btn @click="newModel=false" :disabled = "buttonDisabled">  
 Отмена  
 </v-btn>  
 <v-btn @click="addNewModel(selectedModel)" :disabled = "buttonDisabled">  
 Добавить  
 </v-btn>  
 </v-card-actions>  
 </v-card>  
 </v-dialog>  
 <v-dialog v-model="imageDialog" max-width="1000" max-height="800">  
 <v-card>  
 <v-card-actions>  
 <v-btn>Сравнить результаты</v-btn>  
 </v-card-actions>  
 <v-img :src="imageUrl" contain/>  
 </v-card>  
 </v-dialog>  
 </v-container>  
</template>  
<script>  
import ProjectsRequest from "@/services/ProjectsRequest";  
import *axios* from "axios";  
import DataSourcesRequest from "@/services/DataSourcesRequest";  
import {*mapActions*} from "vuex";  
import ModelRequest from "@/services/ModelRequest";  
export default {  
 name: 'ProjectView',  
 data: () => ({  
 models: [],  
 projectDatasets: [],  
 selectedModel: null,  
 availableModels: ['knn', 'isolation-forest', 'lof', 'auto-encoder', 'gan', 'siamese'],  
 selectedDataset: null,  
 selectedAlgorithm: null,  
 text: null,  
 drawer: null,  
 createProject: false,  
 newModel: false,  
 addDataset: false,  
 customDataset: false,  
 menuActive: -1,  
 menuItems: [{title: "Добавить датасет"}, {title: "Настроить датасет"}, {title: "Обучить модель"}],  
 availableColumns: [],  
 selectedColumns: null,  
 addedDataset: null,  
 timeColumn: '',  
 checkboxes: {  
 isDstLoaded: false,  
 isNormalize: false,  
 isLoadDst: false,  
 isLoadKp: false,  
 isLoadAp: false,  
 isLoadWolf: false,  
 isNaDropped: false  
 },  
 modelData: {},  
 buttonDisabled: false,  
 imageDialog: false,  
 imageUrl: '',  
 }),  
 created() {  
 this.initialize()  
 },  
 computed: {  
 getProjectId() {  
 const {selectedProjectID} = this.$route.params;  
 return selectedProjectID;  
 }  
 },  
 methods: {  
 ...*mapActions*(['updateAppBarTitle']),  
 changeAppBarTitle(newTitle) {  
 this.updateAppBarTitle(newTitle);  
 },  
 openModelsView(typeModel) {  
 let selectedProjectID = parseInt(this.getProjectId);  
 let type = typeModel  
 this.$router.push({name: 'modelView', params: {selectedProjectID, type}}).catch(() => {  
 });  
 },  
 openImageDialog(imageUrl) {  
 this.imageUrl = imageUrl;  
 this.imageDialog = true;  
 },  
 itemProps(item) {  
 const pathComponents = item.path.split('\\');  
 const fileName = pathComponents[pathComponents.length - 1];  
 return {  
 title: fileName,  
 };  
 },  
 getModelName(index) {  
 const modelKeys = *Object*.keys(this.models);  
 if (modelKeys.length > index && index >= 0) {  
 return modelKeys[index];  
 } else {  
 return ''  
 }  
 },  
 async acceptBaseDataset(item) {  
 if (item.id) {  
 this.availableColumns = await this.getDatasetForID(item.id)  
 }  
 },  
 handleMenuItemClick(item) {  
 if (item.title === 'Добавить датасет') {  
 this.addDataset = true  
 } else if (item.title === "Обучить модель") {  
 this.newModel = true  
 } else {  
 this.customDataset = true  
 }  
 },  
 async getDatasetForID(id) {  
 const dataset = new DataSourcesRequest()  
 let datasetData  
 await dataset.getDataSource(id).then(response => {  
 datasetData = response.data.columns.map(column => column);  
 }).catch(x => *console*.log(x))  
 return datasetData  
 },  
 async getDatasetProjectForID() {  
 const project = new ProjectsRequest()  
 let datasets  
 let titleProject  
 await project.getProjectId(this.getProjectId).then(response => {  
 datasets = response.data.dataSources.map(dataset => dataset)  
 titleProject = response.data.title  
 }).catch(x => *console*.log(x))  
 this.changeAppBarTitle(titleProject)  
 return datasets  
 },  
 async addNewDataset() {  
 const formData = new FormData()  
 formData.append("File", this.addedDataset)  
 await axios.post(`https://localhost:7268/api/DataSources?projectId=${this.getProjectId}&timeColumn=${this.timeColumn}&dataSourceTitle=${this.text}`, formData).finally(() => {  
 this.addDataset = false  
 this.text = ''  
 })  
 },  
 async addNewCustomDataset() {  
 const dataSourcesRequest = new DataSourcesRequest()  
 let body = {  
 "dataSourceId": this.selectedDataset.id,  
 "projectId": this.getProjectId,  
 "title": this.text,  
 "isDstLoaded": this.checkboxes.isDstLoaded,  
 "isNormalize": this.checkboxes.isNormalize,  
 "isLoadDst": this.checkboxes.isLoadDst,  
 "isLoadKp": this.checkboxes.isLoadKp,  
 "isLoadAp": this.checkboxes.isLoadAp,  
 "isLoadWolf": this.checkboxes.isLoadWolf,  
 "isNaDropped": this.checkboxes.isNaDropped,  
 "timeColumn": this.timeColumn,  
 "columns": this.selectedColumns  
 }  
 await dataSourcesRequest.postCustomDataFrame(body).catch(x => {  
 *console*.log(x)  
 }).finally(() => {  
 this.text = ''  
 this.customDataset = false  
 })  
 },  
 async addNewModel(modelType) {  
 let modelRequest = null;  
 let body = {}  
 this.buttonDisabled = true  
 switch (modelType) {  
 case 'knn':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "nNeighbors": 12,  
 "algorithm": "auto",  
 "percentile": 95,  
 "columns": this.selectedColumns,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"]  
 }  
 await modelRequest.postKNN(body).catch(error => *console*.error(error));  
 break;  
 case 'isolation-forest':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "columns": this.selectedColumns,  
 "contamination": 0.25,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"],  
 }  
 await modelRequest.postIF(body).catch(error => *console*.error(error));  
 break;  
 case 'lof':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "nNeighbors": 12,  
 "contamination": 0.25,  
 "columns": this.selectedColumns,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"]  
 }  
 await modelRequest.posLof(body).catch(error => *console*.error(error));  
 break;  
 case 'auto-encoder':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "columns": this.selectedColumns,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"]  
 }  
 await modelRequest.postAutoEncoder(body).catch(error => *console*.error(error));  
 break;  
 case 'gan':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "columns": this.selectedColumns,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"]  
 }  
 await modelRequest.postGAN(body).catch(error => *console*.error(error));  
 break;  
 case 'siamese':  
 modelRequest = new ModelRequest();  
 body = {  
 "projectId": parseInt(this.getProjectId),  
 "dataSourceId": this.selectedDataset.id,  
 "columns": this.selectedColumns,  
 "spaceWeatherColumns": [ "dst", "kp", "ap", "wolf"]  
 }  
 await modelRequest.postSiamese(body).catch(error => *console*.error(error));  
 break;  
 default:  
 *console*.error('Неизвестный тип модели');  
 }  
 this.newModel = false  
 this.buttonDisabled = false  
 await this.initialize()  
 },  
 async toggleMenu(id) {  
 this.menuActive = (this.menuActive === id) ? -1 : id;  
 this.modelData = await this.getModel(id)  
 return this.modelData  
 },  
 async getModel(id) {  
 const model = new ModelRequest()  
 let data  
 await model.getModelsId(id).then(response => {  
 data = response.data  
 }).catch(x => *console*.log(x)).finally(() => {  
 })  
 return data  
 },  
 async getModelsForID() {  
 const project = new ModelRequest()  
 let models  
 await project.getModelsId(this.getProjectId).then(response => {  
 models = response.data;  
 }).catch(x => *console*.log(x))  
 return models  
 },  
 async initialize() {  
 let cal;  
 cal = await this.getModelsForID();  
 this.models = cal;  
 for (let key in this.models) {  
 if (!this.models[key] || !this.models[key].length) {  
 this.models[key] = [{}];  
 } else {  
 let maxIdModel = this.models[key].reduce((max, model) => (model.id > max.id ? model : max), this.models[key][0]);  
 this.models[key] = [maxIdModel];  
 }  
 }  
 *console*.log(this.models)  
 cal = await this.getDatasetProjectForID();  
 this.projectDatasets = cal  
 }  
 ,  
 }  
}  
</script>