Евразийский национальный университет имени Л.Н. Гумилёва

УДК 004.896 На правах рукописи

ГРНТИ 50.43

**МЫРЗАТАЙ АЛИ АЛТЫНБАЙҰЛЫ**

**Разработка алгоритма прогнозирования и оптимизации принятия решений в системе ЛВС корпоративной сети**

6D070200 – Автоматизация и управление

Диссертация на соискание степени

доктора философии (PhD)

Научные консультанты

доктор PhD,

Рзаева Л.Г.,

зарубежный консультант

профессор

Бандини С.

(Миланский университет Бикокка)

Республика Казахстан

Астана, 2024

СОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
|  | НОРМАТИВНЫЕ ССЫЛКИ ……………………..………………..... | 4 |
|  | ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ……………………….……….... | 5 |
|  | ВВЕДЕНИЕ…….………………….…………………..………………….. | 9 |
| 1 | АНАЛИЗ И ЛИТЕРАТУРНЫЙ ОБЗОР СОВРЕМЕННОГО СОСТОЯНИЯ СИСТЕМ ПРОГНОЗИРОВАНИЯ СБОЕВ ЛВС...... | 14 |
| 1.1 | Проблемы сбоев ЛВС ………………………………............................... | 14 |
| 1.2 | Прогнозирование сбоев в локальных сетях (LAN) и их влияние на корпоративные сети..........................……………………............................ | 16 |
| 1.3 | Модели, построенные на базе алгоритмов машинного обучения….… | 17 |
| 1.4 | Анализ проблем при разработке предиктивной модели на базе МО…. | 24 |
| 1.4.1 | Несбалансированные данные…………………………………………… | 24 |
| 1.4.2 | Обзор проблемы при выборе алгоритмов машинного обучения…….. | 25 |
| 1.4.3 | Обзор проблемы доступности данных…………………………………. | 26 |
| 1.4.4 | Обзор проблемы симуляции работы модели……………………………. | 27 |
| 1.5 | Обзор применяемых метрик оценки прогностических моделей……..... | 28 |
| 1.6 | Сравнительный анализ работ по прогнозированию отказов……..……. | 29 |
|  | Выводы по разделу 1. …………………………….……………….……… | 31 |
| 2 | ОПИСАНИЕ РАБОТЫ ОБЪЕКТА УПРАВЛЕНИЯ И СБОР ВХОДНЫХ ДАННЫХ………………………………………….……... | 33 |
| 2.1 | Описание работы ЛВС корпоративной сети ………………………..….. | 33 |
| 2.2 | Процесс предварительной подготовки данных.……………………..… | 38 |
| 2.3 | Автоматизация процесса сбора и преобразования исходных данных…. | 40 |
|  | Выводы по разделу 2.……………………………….…………………… | 44 |
| 3 | ОБЩАЯ МЕТОДОЛОГИЯ СОЗДАНИЯ МОДЕЛЕЙ ПРОГНОЗИРОВАНИЯ СБОЕВ ОБОРУДОВАНИЯ ЛВС ИЗ ДИСКРЕТНОГО НАБОРА ДАННЫХ СОБЫТИЙ………………… | 46 |
| 3.1 | Определение модели прогнозирования.………….………….…………. | 46 |
| 3.2 | Масштабируемость алгоритмов прогнозирования………….………… | 49 |
| 3.2.1 | Вычислительная среда…………………………………………………… | 49 |
| 3.3 | Предварительное моделирование в ПО и корреляционный анализ данных …………………………………………………………………….. | 50 |
| 3.4 | Разработка рабочей модели на основе алгоритмов машинного обучения ……………………………………………….…………………. | 58 |
| 3.5 | Математическая модель алгоритмов предсказывания сбоев на основе методов машинного обучения………………………………………….. | 61 |
| 3.6 | Комбинированный метод DES+LR/RF/DT/SVM……………………… | 70 |
|  | Выводы по разделу 3. ……………………………………………………. | 72 |
| 4 | ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ ПО ПРИМЕНЕНИЮ DES-ML АЛГОРИТМОВ ПРОГНОЗИРОВАНИЯ НА ДАННЫЕ ИЗ ПРЕДПРИЯТИЯ …….. | 74 |
| 4.1 | Построение и описание обучаемого алгоритма на прикладном уровне. DES-ML.…………………………………..………………………………... | 74 |
| 4.1.1 | SVM………………………………………………………………………. | 74 |
| 4.1.2 | Описание алгоритма на основе метода деревьев решений и случайных деревьев (DT и RF)………………………………………………………. | 78 |
| 4.2 | Сравнение результатов в ходе обучения моделей на основе МО…..… | 80 |
| 4.3 | Сравнение результатов гибридной модели DES-ML(RF/LR/DT/SVM).. | 93 |
| 4.4 | Оптимизация принятий решений в ЛВС………………………………… | 98 |
| 4.5 | Оптимизация плановых профилактических и экстренных ремонтных работ ……………………………………………………………….……… | 100 |
|  | ЗАКЛЮЧЕНИЕ ……………………………………………..…….…….. | 105 |
|  | СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ. ……….……..… | 108 |
|  | ПРИЛОЖЕНИЕ А – Код процесса преобразования и дальнейшей работы алгоритмов: ………………. | 118 |

**НОРМАТИВНЫЕ ССЫЛКИ**

В настоящей диссертации используются ссылки на следующие нормативные документы и стандарты:

П ЕНУ 19-21. Положение о диссертационном совете Евразийского национального университета им. Л.Н. Гумилёва.

СТ РК ISO/IEC 27002-2015. Информационная технология.

Методы и средства обеспечения безопасности.

Свод правил по средствам управления защитой информации.

Постановление Правительства Республики Казахстан от 20 декабря 2016 года, №832. Об утверждении единых требований в области информационно-коммуникационных технологий и обеспечения информационной безопасности.

ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе.

ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

**ОПРЕДЕЛЕНИЯ**

В настоящей диссертации используются следующие термины, имеющие соответствующие определения, обозначения и сокращения:

**Алгоритм** – последовательность четко определенных инструкций или шагов для выполнения задачи или решения проблемы.

**Базы знаний и данных** – это организованные коллекции информации и фактов, систематически структурированные для удобного хранения, поиска и извлечения данных и знаний.

**Весовой вектор** – набор числовых значений, присваиваемых переменным или элементам в математической, статистической или вычислительной модели, отражающих их относительную важность или вклад в общий результат.

**Весовой коэффициент** – числовое значение, присваиваемое переменной или элементу в математической или статистической модели, которое определяет степень влияния этой переменной или элемента на итоговый результат или выходные данные модели, то есть *xi*​ на зависимую переменную *y.*

**Гиперпараметр** – параметр, используемый для управления процессом обучения алгоритма машинного обучения, который, в отличие от внутренних параметров модели, устанавливается до начала обучения и не изменяется в процессе обучения.

**Гистограмма** – графическое представление распределения данных, где данные группируются в дискретные интервалы (бины), а высота каждого столбца на графике отражает частоту или количество данных в соответствующем интервале.

**Генератор случайных чисел** – алгоритм или устройство, предназначенное для создания последовательности чисел, которые не обладают каким-либо узнаваемым шаблоном или порядком, создавая эффект случайности.

**Детерминированные модели** – математические или компьютерные модели, в которых одинаковые начальные условия всегда приводят к одному и тому же результату, без случайных вариаций.

**Датасет** – собрание данных, обычно структурированных в форме таблицы, где строки представляют отдельные записи (наблюдения), а столбцы – различные атрибуты или переменные этих записей.

**Dataframe/датафрейм** – табличная структура данных, часто используемая в языках программирования для анализа данных, таких как Python (в библиотеке pandas), которая позволяет хранить и манипулировать данными с метками строк и столбцов.

**Звездная топология** – конфигурация сети, в которой все узлы (компьютеры, принтеры и т.д.) подключены к центральному устройству, такому как коммутатор или концентратор, образуя звездообразную структуру.

**Загрузка данных** – процесс передачи данных из одного источника в систему или приложение для дальнейшей обработки, анализа или хранения.

**Кросс-валидация** – метод оценки производительности статистической модели, при котором данные разбиваются на несколько частей, и модель обучается и тестируется на различных комбинациях этих частей.

**Коммутатор** – сетевое устройство, используемое для соединения и переключения пакетов данных между различными устройствами в компьютерной сети, такими как компьютеры, серверы и другие коммутаторы.

**Компилятор** – программа, которая переводит исходный код, написанный на одном языке программирования, в машинный код или другой формат, который может быть выполнен компьютером.

**Корреляция данных** – Статистическая мера, показывающая степень взаимосвязи между двумя переменными.

**Корреляционная матрица** – Таблица, показывающая корреляционные коэффициенты между различными переменными.

**Качество обслуживания (QoS)** – мера производительности сети, которая отражает её способность обеспечивать эффективную передачу данных с учётом таких параметров, как задержка, пропускная способность и надёжность.

**Логистическая регрессия** – статистическая модель, используемая для анализа данных, где зависимая переменная является категориальной, чаще всего бинарной, и предсказывает вероятность наступления события.

**Линейная регрессия** – статистический метод, используемый для моделирования взаимосвязи между зависимой переменной и одной или несколькими независимыми переменными, предполагая линейную зависимость между ними.

**Логит** – функция, используемая в логистической регрессии, преобразующая линейную комбинацию входных переменных в вероятности, ограниченные интервалом от 0 до 1.

**Математическая модель** – система математических выражений, уравнений, зависимостей и ограничений, отражающие особенностей процессов, протекающих в моделируемом объекте моделирования, позволяющая на основе заданного алгоритма прогнозировать поведение объекта, определить значения выходных параметров моделируемого объекта при изменении значений его входных, режимных параметров

**Математическое моделирование** – Процесс разработки и использования математических моделей для имитации и анализа реальных систем или процессов.

**Машинное обучение** – Область искусственного интеллекта, фокусирующаяся на разработке алгоритмов, которые позволяют компьютерам учиться и делать предсказания или принимать решения на основе данных.

**Метод случайного леса (Random forest/RF)** – Ансамблевый метод машинного обучения, использующий множество деревьев решений для повышения точности и устойчивости предсказаний.

**Метод деревьев решений (Decision Tree/DT)** – Алгоритм машинного обучения, использующий древовидную структуру для принятия решений на основе правил, выведенных из данных.

**Метод Опорных Векторов (SVM/SVM)** – Алгоритм машинного обучения, использующий гиперплоскости для классификации или регрессии в многомерном пространстве.

**MAC-адрес** – Уникальный идентификатор, присвоенный сетевому интерфейсу для коммуникации в сетевых технологиях.

**Нормализация** – Процесс приведения данных к общему масштабу без искажения различий в диапазонах значений.

**Оптимизация** – Процесс нахождения наилучшего решения или максимального/минимального значения функции в заданных условиях.

**Протокол передачи данных** – Набор правил и стандартов, определяющих способ передачи данных между устройствами в сети.

**Процессор** – Центральный исполнительный компонент компьютера/устройства, отвечающий за выполнение программных инструкций.

**Предварительная подготовка данных** – Процесс очистки, трансформации и организации данных перед их анализом или использованием в машинном обучении.

**Статистические модели** – Математические конструкции, используемые для анализа и интерпретации статистических данных.

**Структурная идентификация** – Процесс определения структуры модели, наиболее подходящей для анализа данных.

**Telnet** – Протокол сетевого доступа, позволяющий устанавливать текстовое соединение с удаленным компьютером.

**Trunk port** – Сетевой порт, используемый для передачи трафика между несколькими VLAN в сетях.

**URL-адрес** – Унифицированный указатель ресурсов, используемый для идентификации и доступа к ресурсам в интернете.

**F-мера/F1 score** – Статистическая мера, используемая для оценки точности бинарной классификации, учитывающая баланс между точностью и полнотой.

**Информационная система** – организованная система для сбора, хранения, обработки и распространения информации, обычно в контексте организации или предприятия.

**Искусственный интеллект** – область компьютерных наук, занимающаяся созданием систем, способных выполнять задачи, требующие человеческого интеллекта, такие как: восприятие, обучение, рассуждение и решение проблем.

**Локальная Вычислительная Сеть** – это сеть, ограниченная небольшими географическими областью, такими как: офисное здание или кампус, предназначенная для связи между компьютерами и другими сетевыми.

**Оперативное Запоминающее Устройство** – Вид компьютерной памяти, используемый для временного хранения данных, с которыми активно работает процессор.

**Постоянное запоминающее устройство** – энергонезависимая память, используется для хранения массива неизменяемых данных.

|  |  |
| --- | --- |
| DES | – Метод двойного экспоненциального сглаживания, используемый в анализе временных рядов для прогнозирования будущих значений |
| ИС ППР | – Компьютеризированная система, предназначенная для поддержки процессов принятия решений в организации, обеспечивая сбор, обработку и анализ данных |
| ИС | – Информационная система |
| ИИ | – Искусственный интеллект |
| ЛВС | – Локальная Вычислительная Сеть |
| ОЗУ | – Оперативное Запоминающее Устройство |
| ПЗУ | – Постоянное запоминающее устройство |
| HTTP | – Протокол передачи гипертекста, основной протокол для передачи данных во Всемирной паутине |
| HTTPS | – Расширение протокола HTTP, обеспечивающее защищенную передачу данных |

**ВВЕДЕНИЕ**

**Актуальность темы.** В связи с глобальной информатизацией и развитием информационных технологий, все больше стран модернизируют и производят цифровизацию отрасли и механизмы деятельности для достижения быстрого роста как экономики, так и социума. На данный момент в распоряжении крупных компании находятся огромные массивы данных. Анализируя эти данные, можно найти пути решения некоторых задач. Однако экспертам-аналитикам процедуры анализа сотен и тысяч показателей могут показаться избыточно длительными и трудоёмкими, если вообще осуществимыми. Однако алгоритмы на базе машинного обучения могут справится с поставленными задачами анализа быстро и эффективно.

Развитие информационных технологий приводит к расширению локальных сетей предприятий и увеличению сложности их эффективного управления и мониторинга. В этих сетях каждую минуту создаётся масса разнообразных данных, что делает управление сетевой инфраструктурой ЛВС сложной задачей. С учётом того, что сетевая инфраструктура включает в себя множество различных устройств (таких как коммутаторы, серверы, маршрутизаторы, компьютеры) и на каждом из них работает множество приложений, генерирующих большие объёмы данных, алгоритмы машинного обучения могут эффективно прогнозировать потенциальные отказы. Точный прогноз отказов оборудования повысит надёжность, безопасность и экономическую эффективность эксплуатации локальных вычислительных сетей. Машинное обучение и искусственный интеллект уже находят широкое применение в промышленности, позволяя выявлять риски и детали, которые трудно обнаружить с помощью традиционных аналитических методов.

Одним из областей применения машинного обучения – анализ данных для последующего прогнозирования. Например, с каждым днём появляются все больше IT-компаний, оказывающих широчайший спектр услуг. Успешная работа таких компаний всегда зависит от эффективного построения внутренней инфраструктуры систем и ЛВС. Однако, чем больше ЛВС, тем больше вероятность что одна из аппаратных средств ЛВС выйдет из строя. Крупные IT-компаний используют системы мониторинга сетей, чтобы в режиме online контролировать всю транспортную среду ЛВС. Контроль осуществляется с помощью интегрирования в ЛВС множества датчиков и сенсоров, которые собирают огромное количество данных, и группируя, выводятся на экраны в том или ином виде в ситуационных центрах, диспетчерских, ЦОД-ах (центры обработки данных). Однако все эти данные обрабатываются оператором, а не машиной.

В связи с этим разработка алгоритма прогнозирования и оптимизации принятия решений в системе ЛВС корпоративной сети на основе машинного обучения является актуальной научной задачей.

**Целью исследования является** построение алгоритма, позволяющее обрабатывать входящий поток данных от мониторинговой системы ЛВС и выдавать результаты в виде прогноза возможного инцидента в структуре ЛВС и, тем самым, позволяющий ускорить и повысить эффективность принятия решений оператором.

**Задачи исследования.**

1. Выполнить обзор и провести сравнительный анализ по теме исследования.
2. Проведение корреляционного анализа входных параметров системы ЛВС;
3. Построение математической модели алгоритма прогнозирования инцидентов в ЛВС корпоративной сети.
4. Разработка моделей на базе алгоритмов ML по предложенному алгоритму прогнозирования.
5. Оптимизация принятия решений по внеплановым ремонтным работам в системе ЛВС через разработанный алгоритм.
6. Апробация сформированных моделей и алгоритмов для обработки данных в системе мониторинга ЛВС. Оценка эффективности предложенной концепции внедрения системы прогнозов, основанной на методах машинного обучения.

**Объектом исследования** **является** сегмент ЛВС, состоящий из коммутаторов L2 и L3 уровня.

**Предметом исследования** являются методы и алгоритмы прогнозирования отказов оборудования ЛВС корпоративной сети.

**Методы исследования**. В данной работе для достижения поставленных целей применялись различные методы исследования, включая математическое моделирование, методы оптимизации, техники машинного обучения, методы деревьев решений и случайных лесов, алгоритмы двойного экспоненциального сглаживания (DES), а также интеграция этих подходов. Дополнительно использовались программные инструменты и языковые ресурсы, соответствующие современным компьютерным технологиям.

**Основные научные результаты и положения, выносимые на защиту:**

1. Концепция построения системы предварительной подготовки данных, и концепция построения системы прогнозирования, основанной на моделях, использующих методы машинного обучения.
2. Общая методология построения модели прогнозирования.
3. Аналитическая работа по определению эффективных прогностических моделей на основе комбинированной модели с использованием алгоритмов МО и DES.
4. Процесс оптимизации принятия решений внеплановых ремонтных работ ЛВС.

**Научная новизна результатов исследования.**

1. **Комплексный анализ методов**: Исследование представляет собой одно из немногих, которое проводит комплексный анализ применения различных методов машинного обучения – деревьев решений, случайного леса, метода опорных векторов логистической регрессии и алгоритмы двойного экспоненциального сглаживания (DES) – для прогнозирования сбоев в локальных сетях (ЛВС).
2. **Фокус на интерпретируемости**: Уделено внимание на интерпретируемость моделей, что представляет собой важный шаг в понимании, какие сетевые параметры наиболее критичны для прогнозирования сбоев.
3. **Анализ нелинейных зависимостей**: Введение в анализ сложных нелинейных зависимостей между сетевыми параметрами и сбоями через использование ядер в SVM и сложных структур в деревьях решений.
4. **Отбор признаков**: Исследование также затрагивает вопрос отбора признаков, который является ключевым для повышения эффективности прогнозных моделей.
5. **Практическая ориентация**: Фокус на применении в индустриальных масштабах и реальных сетевых средах делает исследование особенно актуальным для практического применения.
6. **Сравнительный анализ**: Финальное сравнение эффективности различных методов в конкретном контексте может служить основой для будущих исследований и практических рекомендаций.
7. **Процесс оптимизации принятия решений внеплановых ремонтных работ ЛВС:** Было уделено отдельное внимание к условным вероятностям и их влиянию на процесс принятия решений.

**Связь темы диссертационной работы с другими исследовательскими работами.** Диссертация выполнена в соответствии с планами научно-исследовательских работ кафедры Системного анализа и управления ЕНУ им. Л.Н. Гумилева.

**Практическая значимость диссертационного исследования.**

Практическая значимость данного исследования многоаспектна и может быть применена в различных областях, включая, но не ограничиваясь следующими:

1. **Обеспечение надёжности сети (QoS)**: Применение эффективных методов прогнозирования сбоев в сетевых коммутаторах и других элементах ЛВС способствует повышению надёжности и устойчивости всей сетевой инфраструктуры.
2. **Оптимизация принятий решений в ЛВС**: Практическая значимость заключается в разработке эффективной системы прогнозирования поломок сетевого оборудования, которая позволяет своевременно идентифицировать потенциальные неисправности и оптимизировать процесс ремонтных работ. Это обеспечивает повышение надёжности и экономической эффективности эксплуатации оборудования в локальных вычислительных сетях, снижая риски и убытки, связанные с непредвиденными отказами и простоями.
3. **Улучшение качества сервиса**: Уменьшение времени простоя и повышение надёжности сети приводят к улучшению качества предоставляемых сетевых сервисов и, соответственно, удовлетворённости пользователей.
4. **Принятие управленческих решений**: Информация о важности различных сетевых параметров и их влиянии на вероятность сбоев может быть полезной для принятия управленческих решений на уровне планирования и развития сетевой инфраструктуры.
5. **Адаптивность и масштабируемость**: Исследование учитывает вопросы масштабируемости, что особенно важно для крупных и растущих сетевых систем.
6. **Работа с неполными данными**: Методы обработки пропущенных данных, описанные в исследовании, особенно актуальны в реальных сетевых условиях, где данные могут быть неполными или неточными.
7. **База для дальнейших исследований**: Проведённый анализ и выводы могут служить отправной точкой для дальнейших исследований в этой области, включая разработку новых методов прогнозирования сбоев и их применения в различных сетевых конфигурациях.

Таким образом, исследование представляет собой ценный вклад в область сетевых технологий и машинного обучения, обладая широким спектром практического применения и потенциалом для дальнейшего развития и инноваций.

**Научные публикации по результатам исследования.** Основные результаты и положения диссертационной работы опубликованы в 7 научных трудах, в т.ч. в журналах рекомендованных Комитетом по обеспечению качества в сфере образования и науки Министерства образования и науки Республики Казахстан – 4 научные статьи:

1. The effect of the amount of data array on the results of forecasting network equipment failures // Известия НАН РК. Серия физико-математическая. – 2021. – №6. – С. 28-36.
2. Автоматизация сбора и подготовки данных из системы мониторинга ЛВС для дальнейшего применения в системах прогнозирования базирующихся на методах машинного обучения // Вестник КазАТК. – 2022. – Т. 121, №2. – С. 294-301.
3. The implementation and the use of the LAN monitoring systems for systematisation of the input data of the incident forecasting systems // Известия НАН РК. Серия физико-математическая. – 2022. – №2. – С. 54-63.
4. Использование методов интеллектуального анализа данных для прогнозирования поломок устройств сети передачи данных // Труды университета – 2022. – №3(88). – С. 304-313.

В журналах, входящие в базу Scopus опубликованы – 4 статьи:

1. Enhancing LAN Failure Predictions with Decision Trees and SVMs: Methodology and Implementation // Electronics. – 2023. – Т. 12, №18. – Р. 3950.
2. Predicting LAN Switch Failures: An Integrated Approach with DES and Machine Learning Techniques (RF/LR/DT/SVM) //Results in Engineering. – 2024. – С. 102356.
3. Security Information Event Management Data Acquisition and Analysis Methods with Machine Learning Principles // Results in Engineering. – 2024. – Р. 102254.
4. Abitova G. et al. Neural network modeling of agglomeration firing process for polymetallic ores // International Journal of Electrical & Computer Engineering (2088-8708). – 2022. – Т. 12, №4.

**Структура диссертации.** Диссертация состоит из введения, четырех разделов, заключения, списка использованных источников – 144, 45 рисунков, 4 таблиц и одного приложения, объем диссертации – 122 страницы.

**1 АНАЛИЗ И ЛИТЕРАТУРНЫЙ ОБЗОР СОВРЕМЕННОГО СОСТОЯНИЯ СИСТЕМ ПРОГНОЗИРОВАНИЯ СБОЕВ ЛВС**

* 1. **Проблемы сбоев ЛВС**

Локальные вычислительные сети (ЛВС) служат основой связи в корпоративных условиях. Как следует из названия, ЛВС – это сети, ограниченные относительно небольшой площадью, обычно одним зданием или группой соседних зданий, таких как офисы, школы или дома. ЛВС позволяют множеству устройств в этой определенной области обмениваться данными и делиться ресурсами, такими как файлы или принтеры, содействуя созданию взаимосвязанности и эффективности [1-4].

В корпоративных условиях ЛВС играют жизненно важную роль в обеспечении повседневных бизнес-операций. Они обеспечивают передачу данных между сотрудниками, содействуют сотрудничеству и обеспечивают совместный доступ к ресурсам, таким как серверы, принтеры и интернет-подключение. Эти сети также позволяют централизованное управление данными, что имеет решающее значение для безопасности, обслуживания и эффективного ведения бизнеса [2; 3, р. 57-69].

Управление сетью, с другой стороны, это процесс администрирования и управления работой компьютерной сети. Он включает в себя широкий спектр обязанностей, включая, но не ограничиваясь, поддержанием сетевого оборудования (например, коммутаторы, маршрутизаторы), контролем программного обеспечения, мониторингом работы сети, устранением проблем сети и планированием будущего роста сети. Основная цель управления сетью – обеспечение плавной работы сети с минимальным простоем, так как перебои могут привести к значительным потерям в производительности и, следовательно, финансовым затратам [1; 2].

Выбор между развёртыванием устройств ЛВС одного поставщика и нескольких поставщиков в корпоративных сетях часто сводится к компромиссу между простотой и гибкостью.

Окружения с одним поставщиком, где все сетевые устройства поставляются одним производителем, предлагают несколько преимуществ. Основное преимущество – это простота и лёгкость управления. В среде с одним поставщиком устройства разработаны так, чтобы работать вместе без швов, что часто приводит к меньшему количеству проблем с совместимостью, упрощает устранение неполадок и позволяет централизованное управление. Кроме того, производительность во всей сети более предсказуема, так как все устройства работают в рамках похожих параметров. Этот сценарий потенциально может привести к более эффективному управлению сетью и снижению общей стоимости владения за счёт уменьшения затрат на обслуживание.

Однако у окружений с одним поставщиком также есть свои недостатки. Они часто ограничивают гибкость, так как корпорации привязаны к продуктовой линейке одного производителя. Если у поставщика нет устройства с определенными функциями, корпорации приходится идти на компромисс или искать обходные пути. Кроме того, привязанность к одному поставщику может потенциально привести к более высоким затратам на закупки, особенно если цены поставщика неконкурентоспособны. Также риски зависимости от поставщика могут привести к долгосрочной зависимости от определенного поставщика, что может быть проблематично, если бизнес или продуктовая линейка поставщика изменятся в будущем.

С другой стороны, среды с несколькими поставщиками предлагают повышенную гибкость. Корпорации могут выбирать из более широкого спектра устройств, каждое из которых имеет свои уникальные функции и цены, позволяя им адаптировать свою сетевую инфраструктуру к своим конкретным потребностям. Среды с несколькими поставщиками также обеспечивают избыточность, так что, если один поставщик сталкивается с проблемами (например, банкротство или нарушения в цепочке поставок), альтернативы уже присутствуют в сети.

Однако управление средами с несколькими поставщиками может быть сложным. Устройства каждого поставщика могут иметь разные операционные процедуры, что приводит к более сложным задачам управления и устранения неполадок. Проблемы совместимости между устройствами разных производителей также могут возникнуть, потенциально вызывая сбои в сети. Кроме того, оптимизация производительности может быть более сложной из-за различных характеристик производительности устройств от разных производителей.

В итоге, хотя среды с одним поставщиком предлагают простоту и лёгкость управления, они могут не обладать гибкостью и диапазоном функций, предоставляемых средами с несколькими поставщиками. Наоборот, хотя среды с несколькими поставщиками предлагают повышенную гибкость и выбор, они также могут привести к увеличению сложности в управлении сетью. Выбор между двумя будет зависеть от конкретных требований и возможностей каждой корпорации [2].

Управление ЛВС в корпоративных условиях является сложной задачей из-за динамичного характера сетевого трафика и разнообразного диапазона устройств, которые должны без проблем обмениваться данными. Администраторы ЛВС должны постоянно контролировать работу сети, выявлять потенциальные проблемы, прежде чем они обострятся, обеспечивать безопасность данных и планировать рост сети. Задача усложняется необходимостью свести к минимуму простои сети при управлении обновлениями или изменениями сети, так как любое прерывание может нарушить бизнес-операции [3, р. 57-69].

Кроме того, значительной проблемой в управлении ЛВС является прогнозирование и устранение отказов коммутаторов. Сетевые коммутаторы, которые облегчают общение между устройствами в ЛВС, могут периодически выходить из строя по причинам, таким как аппаратные проблемы, ошибки программного обеспечения или перегрузка. Такие отказы могут серьёзно влиять на работу сети, приводя к сбоям в общении и возможной потере данных. Существующие стратегии управления сетью часто не могут прогнозировать эти сбои, склоняясь к реакции на проблемы, а не к их предотвращению. Это область, где методы машинного обучения могут потенциально обеспечить значительный прогресс [5]. Соответственно, выбор направления проблемы обуславливается практической необходимостью и отсутствием оптимального решения в широком смысле.

**1.2 Прогнозирование сбоев в локальных сетях (LAN) и их влияние на корпоративные сети**

Надёжность и устойчивость корпоративных систем локальных сетей (LAN) имеют решающее значение для поддержания бесперебойного ведения бизнеса. Среди множества проблем, которые могут нарушить работу сети, особое значение имеют сбои коммутаторов, которые могут иметь серьёзные последствия. Коммутаторы сети, критически важные компоненты, направляющие и распределяющие данные по сети, могут сталкиваться со сбоями из-за различных факторов, таких как неисправности оборудования, ошибки программного обеспечения или перегрузка. Когда коммутатор выходит из строя, это может вызвать узкие места в сети, потерю данных или даже полную остановку сети, что приводит к значительным операционным нарушениям и потенциальным финансовым потерям [6].

Учитывая эти возможные последствия, было проведено немало исследований по прогнозированию сбоев коммутаторов в локальных сетях. Одно из ранних исследований в этой области, проведённое G. Weiss и другими работами [7-10], подчеркнуло значимость мониторинга сетевых журналов и трафика для прогнозирования возможных сбоев коммутаторов. Они предложили систему, основанную на правилах, которая использует исторические сетевые данные для прогнозирования сбоев. Однако их подход сталкивался с ограничениями в отношении адаптивности и масштабируемости, особенно в сложных и динамичных корпоративных сетевых средах.

С развитием машинного обучения и искусственного интеллекта исследование в области прогнозирования сбоев LAN переключилось на эти техники. Например, Zhang K. и др. [5, р. 29-39] применили сочетание глубокого обучения и анализа временных рядов для прогнозирования сетевых сбоев. Их метод продемонстрировал значительные улучшения в точности прогнозирования по сравнению с предыдущими методами, но требовал значительных вычислительных ресурсов, что затруднило его реализацию в приложениях в реальном времени.

Несмотря на эти достижения, многие современные методы прогнозирования сбоев в локальных сетях имеют ограничения. Большинство методов требуют обширных вычислительных ресурсов или не могут адаптироваться к динамическим изменениям в сетевом трафике. Кроме того, некоторые методы основаны на всесторонних исторических данных, которые могут быть недоступны во всех корпоративных настройках [8, р. 75-104; 9, р. 166-169; 10, р. 58307-58322; 11].

Таким образом, существует явная потребность в эффективных, адаптивных и масштабируемых методах прогнозирования сбоев в локальных сетях, которые могут быть использованы в корпоративных сетях. Разработка такого алгоритма, способного точно прогнозировать сбои коммутаторов, может потенциально революционизировать стратегии управления сетью, приведя к более надёжным и устойчивым корпоративным системам локальных сетей.

Анализ существующих работ показал, что наиболее эффективным является применение алгоритмов машинного обучения для построения моделей предиктивной аналитики сбоев ЛВС. Таким образом, выбор направления решения проблемы данного диссертационного исследования обосновывается качественным анализом трудов исследователей в этой области.

**1.3 Модели, построенные на базе алгоритмов машинного обучения**

Методы машинного обучения охватывают широкий спектр подходов, целью которых является создание модели, обучаемой на основе конкретных данных. Это обучение позволяет модели анализировать и генерировать выводы, соответствующие обнаруженным в данных закономерностям. В обзоре, опубликованном в работе [12], была предложена таксономия алгоритмов классификации в рамках машинного обучения, подчёркивая концепцию индуктивного машинного обучения. Этот метод включает изучение набора правил из конкретных примеров (например, примеров в обучающем наборе) или, более широко говоря, создание классификатора, способного обобщать из новых примеров. Этот подход к обучению, обычно основанный на статистических методах, происходит автоматически. Модели, созданные в рамках этого процесса, оцениваются на основе того, насколько результаты, которые они генерируют, соответствуют ожидаемому поведению. Уникальной особенностью этих моделей является то, что они могут быть запитаны образцами, не наблюдаемыми во время обучения, что требует способности обобщать и адаптироваться к сценариям, не охваченным обучающими данными.

Цели применения машинного обучения очень разнообразны, но с момента его зарождения они тесно связаны с проблемами прогнозирования [13-15]. Идея использования вычислительных способностей этих алгоритмов для выявления скрытых взаимосвязей или закономерностей в данных привлекательна во многих областях.

Понятие прогнозирования не имеет универсального или формального определения, но в большинстве контекстов его можно понимать, как объявление или догадку о будущем событии, основанную на текущих и прошлых знаниях или опыте. Основной аспект будущего – его неопределённость; таким образом, любое решение, предложенное для проблемы прогнозирования, никогда не будет совершенно точным, а будет представлять собой приближение наиболее вероятного исхода. Кроме того, краткосрочные прогнозы, как правило, более достижимы, чем долгосрочные.

Для создания прогнозных моделей необходимо наличие исторических данных для анализа и обучения модели. Характер этих данных будет различаться в зависимости от конкретной области применения, и, следовательно, прогнозируемая цель также будет разнообразной. В медицине прогнозы могут помогать специалистам в диагностике [16] или в оценке рисков, связанных с определенными лечениями [17]. Прогностические возможности машинного обучения были использованы в экономической сфере для оценки доходов от рекламных инвестиций или для моделирования возникающих экономических динамик [18]. Различные геологические исследования использовали прогнозирование в попытках предвидеть землетрясения [19, 20]. В областях, более тесно связанных с информационными технологиями, машинное обучение использовалось для прогнозирования сбоев в телекоммуникационных инфраструктурах [21], выявления дефектов в программном обеспечении [22], прогнозирования сетевых и памятных сбоев в суперкомпьютерах [23] или обнаружения мошенничества в веб-платежах [24]. Ещё одной обычной проблемой, как правило, связанной с последней областью, является прогнозирование сбоев.

Прогнозирование сбоев сосредоточено на разработке моделей, которые могут предвидеть неисправность или поломку в компонентах программного или аппаратного обеспечения. На основе всестороннего изучения существующей литературы, Salfner и др. [25] разъясняют три ключевых понятия в этой области:

1. сбой: это относится к событию, которое происходит, когда предоставленная услуга отклоняется от правильной или ожидаемой услуги;
2. ошибка: она составляет часть общего состояния системы, которая может вызвать её последующий сбой в обслуживании;
3. дефект: они считаются или предполагаются как истоки ошибки, представляя собой фундаментальную причину самой ошибки.

Эти определения устанавливают основополагающее понимание понятий, связанных с прогнозированием сбоев, тем самым предлагая рамки для оценки, диагностики и предотвращения сбоев в различных системах и компонентах.

Понимание вышеуказанных трёх понятий необходимо для осознания границ проблем прогнозирования сбоев. Эти модели направлены на выявление отклонений от правильного функционирования системы. Поэтому в некоторых случаях они опираются на техники обнаружения аномалий [26, 27]. Основная цель этих моделей – узнать, когда произойдут эти сбои, а не установить причину их возникновения (определить, в чем заключается неисправность) или предотвратить их. Эти два аспекта, вместе с прогнозированием сбоев, составляют автоматизированную систему управления неисправностями объекта управления, которая может быть описана следующим образом:

1. прогнозирование сбоев является основным аспектом этих систем, сосредоточенным на обнаружении предстоящих сбоев;
2. автоматизированный отклик на сбои стремится решить вопрос о правильном отклике на сбой, направленный на его устранение или смягчение эффектов;
3. диагностика или анализ корневых причин направлен на выявление того, что вызвало сбой, используя исторические журналы и записи сделанных прогнозов.

В совокупности эти компоненты предлагают всеобъёмлющую рамку для проактивного управления неисправностями, облегчая раннее обнаружение, стратегический отклик и тщательное расследование сбоев. Этот всеобъёмлющий подход служит для усиления надёжности, и эффективности различных систем и приложений.

Хотя эти три проблемы тесно связаны, они существенно различаются, и каждая из них требует подробного изучения. В этой работе, основной фокус сосредотачивается на первой проблеме. Существует множество исследований по прогнозированию сбоев в различных областях, включая прогнозирование сбоев жёстких дисков [28], сбоев суперкомпьютеров [29], сбоев аппаратных компонентов [30], сбоев телекоммуникационных систем [31] и сбоев распределённых систем [32].

В контексте распределённых систем прогнозирование обычно основывается на данных, собранных путём мониторинга сети системы. Эти мониторинговые данные обычно включают отчёты об ошибках, события мониторинга или даже события обнаружения сбоев. Прогнозисты используют эту информацию для выявления шаблонов активности, которые предвещают сбои [33-35].

Независимо от области, каждая модель прогнозирования требует набора данных для обучения. Этот набор данных должен включать типичную активность системы, но также должен охватывать случаи, где произошли сбои. В зависимости от уровня активности системы, с которой работает этот набор, данные могут охватывать период от нескольких дней до нескольких лет. Сырые данные обычно не могут обучить модель напрямую, поэтому всегда требуется этап предварительной обработки и подготовки набора данных [36].

В функциональной системе количество сбоев, которые происходят, не может быть чрезмерно велико. Сбои обычно являются редкими событиями, и собранный набор данных будет преимущественно содержать случаи, где сбои не произошли, а не те, где они произошли. Этот аспект подчёркивает значимость осмотрительного выбора и подготовки данных для обучения и анализа, важного шага в разработке точных и надёжных моделей прогнозирования.

Способ решения проблемы ограниченной информации об ошибках в наборах данных заключается в синтетической генерации новых данных. Эти искусственно созданные образцы обогащают набор данных более конкретными сведениями об ошибках, помогая в обучении модели. Существует несколько методов создания синтетических данных, начиная от простого объединения существующих образцов до сложных симуляций, имитирующих поведение системы или процесс создания данных [37-39].

Симуляторы, сами по себе, демонстрируют разные уровни сложности. Более простые ограничены созданием данных, которые строго соответствуют оригинальному формату. В отличие от этого, некоторые стремятся отразить внутренние связи между различными элементами, в то время как самые сложные пытаются имитировать систему, ответственную за генерацию данных, напрямую. Область симуляторов, хотя и обширная в исследованиях, видит синтетическую генерацию данных только как один из своих незначительных сегментов. Дополнительным преимуществом симуляторов перед более простыми методами синтетического создания данных является их обычная способность генерировать данные в реальном времени, верно воспроизводя выходные данные изучаемой системы.

Эта возможность работы в реальном времени особенно актуальна в контексте прогнозирования сбоев, особенно в подполе прогнозирования в реальном времени. Использование симулятора, генерирующего данные в реальном времени, предоставляет ценный инструмент для оценки отзывчивости обученных моделей, предлагая более динамичное и непосредственное понимание системы.

Прогноз событий в реальном времени не является необходимостью для всех прогнозирующих задач. Некоторые системы предиктивного обслуживания, например, требуют выполнения только раз в день или неделю [40]. Прогнозирование в реальном времени особенно актуально при работе с системами, которые производят значительный поток информации и позволяют предпринимать немедленные действия во время выполнения. В этом контексте модели интерпретируют входные данные как непрерывный поток образцов, а не отдельные образцы, и выход состоит из постоянного потока прогнозов. Этот метод функционирования требует, чтобы модели соответствовали определенным критериям производительности, главным из которых является способность быстро генерировать выход, соответствующий одному входному образцу, чтобы быть готовым к следующему входному образцу.

Проекты, связанные с прогнозированием в области машинного обучения, обладают уникальной особенностью: они всегда должны учитывать временное измерение данных. Прогнозы основаны на понимании будущего, которое формируется настоящим и прошлым. Хотя временной аспект может быть потерян в фазе обработки данных, и входные данные моделей могут не иметь временной метки, время все равно будет фактором в обработке. Традиционно, времязависимые данные обрабатывались с использованием временных рядов, но они не подходят для всех проблем [41]. С середины 90-х годов алгоритмы машинного обучения все чаще используются для решения таких вопросов. Основываясь на поиске в Google Scholar по запросу «прогнозирование с помощью машинного обучения» в разных временных интервалах, за последнее десятилетие было опубликовано примерно полмиллиона публикаций, в то время как в 90-х годах их было всего 20 000. Эта тенденция отражена на рисунке 1.1. Данные алгоритмы разрабатывают модели через тренировочный процесс, который использует набор данных, часто извлекаемых из данных, собранных со временем, и преобразованных в статический набор образцов, не обязательно упорядоченных хронологически, с помощью методов агрегации или снимков. Временное представление данных в каждом проекте варьируется в зависимости от целей и области применения, и этот процесс обычно называется оконным методом данных.

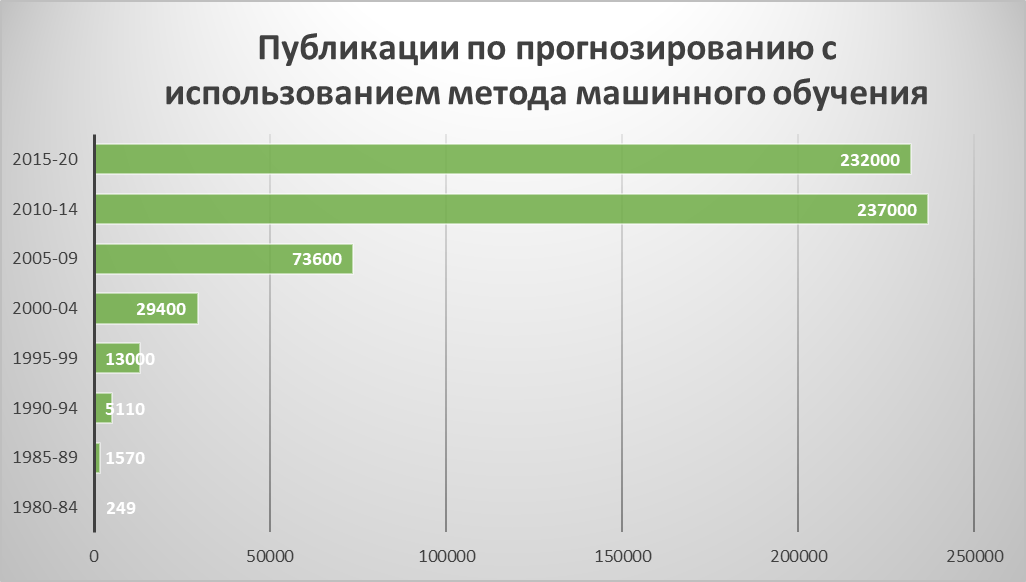
****

Рисунок 1.1 – Рост числа публикаций по прогнозированию с использованием методик машинного обучения между 1980 и 2020 годами

Методология оконных данных относится к процессу захвата состояния системы путём наблюдения за её состоянием в течение определенного периода времени, называемого временным окном [42]. Временное окно характеризуется своей продолжительностью. Эта продолжительность обычно измеряется во времени (например, 5 минут, 1 час или 1 раз в день). В некоторых случаях, когда данные представлены в виде дискретных событий, длина или вместимость окна может измеряться в количестве образцов [43] (например, последние сто произошедших событий в системе мониторинга или последние двадцать образцов, полученных датчиком).

Данные, собранные во временном интервале, объединяются и представляются в виде численных переменных. Тип агрегации, который применяется, зависит от проблемы. В некоторых случаях это может быть подсчёт событий, скользящая средняя или другие специфические для области типы агрегации. В задачах прогнозирования окно, с помощью которого формируется вход моделей, известно, как «окно наблюдения». В этих ситуациях необходимо определить окно, которое определяет время действия прогноза или вывода. Это второе окно называется окном прогноза или временем прогноза [25, р. 1-41].

В онлайн-прогнозировании сбоев текущий момент обозначается как *t*. Сбои прогнозируются с некоторым запасом времени *tl*, который должен быть больше минимального времени предупреждения *tw*. Прогноз считается действительным в течение определенного периода времени, называемого периодом прогноза, *tp*. Для осуществления прогноза используются данные вплоть до временного горизонта *td*, который называется размером окна данных (рисунок 1.2) [25, р. 1-41].

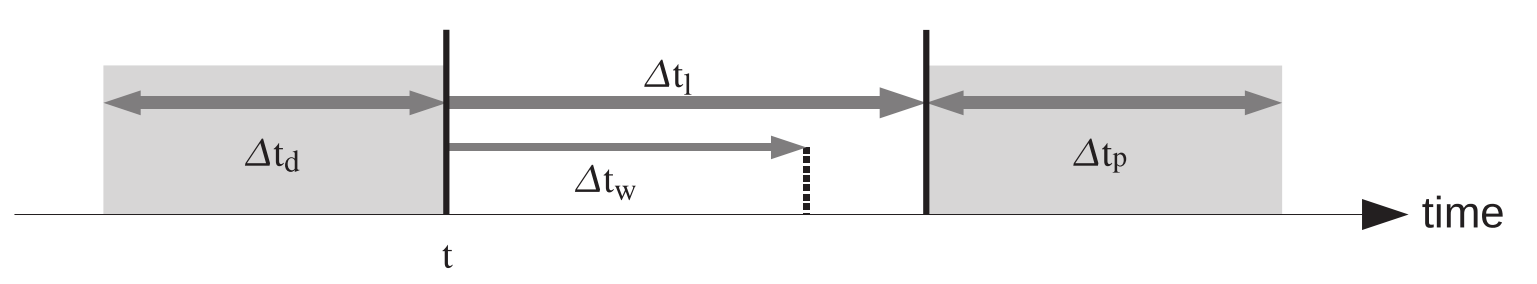


Рисунок 1.2 – Схема онлайн-прогноза

Окно наблюдения играет ключевую роль в способности модели изучать систему для определения выходных данных. Размер (длина) окна указывает на количество информации, которое оно может содержать, и напрямую связано с характером данных, который может варьироваться от секунд [44] до часов [45] и даже дней [46]. Выявление оптимального размера для каждого проекта является нерешённым вопросом в научной литературе, хотя было предложено различные методы. В некоторых случаях авторы сами выбирают размер окна без обсуждения его влияния [47]. Другие используют поиск по сетке для изучения влияния длины окна на производительность модели [48, 49]. Также применяются более сложные методы, такие как использование генетического алгоритма для определения оптимального размера окна, как показано в исследованиях Вайса и др. в нескольких статьях [33, р. 83-89; 50, 51].

Что касается размера окна прогнозирования, он меняется в зависимости от изучаемой проблемы и может варьироваться от минут до дней или больше. Прогнозы, связанные с отказами жёстких дисков или компонентов аппаратного обеспечения, могут иметь окна в несколько часов [45, р. 1-11; 52], в то время как прогнозы, связанные с обнаружением вторжений в облачных системах, могут потребовать окно в минутах [53]. Иногда окно может быть определено количеством использований, а не временным измерением, как в прогнозах, связанных с физическими компонентами [54]. В [55] был предложен альтернативный подход, известный как прогнозирование на основе событий (в отличие от традиционного оконного прогнозирования). Этот метод устраняет окно прогнозирования, подтверждая прогнозы в один момент времени, который отделяется от окна наблюдения временем, называемым время набега.

Другим методом прогнозирования во временном промежутке является метод двойного экспоненциального сглаживания (Double Exponential Smoothing (DES)), широко применяемый в анализе временных рядов. Данный метод разработан для более точного прогнозирования данных, содержащих как уровни (или базовые значения), так и тренды. Этот метод является расширением простого экспоненциального сглаживания и особенно эффективен в случаях, когда данные демонстрируют трендовую составляющую. Это может включать прогнозирование продаж, погоды, финансовых индикаторов и, в вашем случае, отказов оборудования в системах ЛВС.

Примером использования этого метода можно указать результаты исследования [56] где группа исследователей смогли комбинировать метод DES с алгоритмом МО, использующим метод опорных векторов. В этом исследовании особое внимание уделяется моделям, учитывающим риски в оптических сетях, и исследуется, как спрогнозировать риск отказа оборудования. Результаты экспериментов показали среднюю точность прогнозирования состояния отказа оптического оборудования 95%.

Другие исследования также изучали эффективность использования метода DES в различных контекстах прогнозирования. Например, в ислледовании [57] оценивается эффективность экспоненциального сглаживания для прогнозирования данных временных рядов с тенденциями и несезонными характеристиками, обнаружив, что методы экспоненциального сглаживания эффективны для таких данных. В другом исследовании [58] применяется двойное экспоненциальное сглаживание для прогнозирования факторов окружающей среды, таких как осадки и температура. В работе [59] исследователи представили гибридную модель, основанную на двойном экспоненциальном сглаживании, для прогнозирования скорости ветра, подчёркивающий преимущества этой модели по сравнению с традиционными моделями МО. Также метод DES применялся в исследовании [60] для прогнозирования продаж лекарств.

Если сравнить эти два метода, метод оконных данных и метод двойного экспоненциального сглаживания, можно выделить преимущества и недостатки каждого из методов (таблица 1.1).

Таблица 1.1 – Сравнение методов оконных данных и двойного экспоненциального сглаживания

|  |  |  |
| --- | --- | --- |
| Сравнение | Метод оконных данных | Метод двойного экспоненциального сглаживания (DES) |
| Преимущества | Сосредоточенность на последних тенденциях: Сосредоточив внимание на последних данных, использование метода оконных данных может сделать модель более чувствительной к недавним изменениям и аномалиям. | Осведомлённость о тенденциях: DES эффективно фиксирует как уровень, так и тенденцию данных, что может иметь решающее значение для прогнозирования сбоев, которые возникают постепенно |
| Гибкость: размер окна можно регулировать, чтобы сбалансировать актуальность и объём рассматриваемых данных. | Эффект сглаживания: сглаживает краткосрочные колебания, что может быть полезно для уменьшения шума в прогнозе. |

Продолжение таблицы 1.1

|  |  |  |
| --- | --- | --- |
|  | Уменьшенная сложность: анализ меньшего набора данных может снизить сложность вычислений и повысить скорость обработки. | Использует более полный набор данных для создания более обоснованных прогнозов. |
| Недостатки | *Потеря исторического контекста.* Сосредоточение внимания на меньшем окне данных может привести к потере важных исторических тенденций и закономерностей. | *Задержка реакции: DES* может не реагировать быстро на внезапные изменения или аномалии, поскольку по своей сути он сглаживает недавние наблюдения. Но этот недостаток можно нивелировать, используя другие алгоритмы МО параллельно с DES. |
| *Чувствительность параметра:* Выбор размера окна может существенно повлиять на производительность модели. | *Допущение линейности:* DES предполагает линейный тренд, который не всегда подходит для сложных, нелинейных закономер ностей в данных об отказах |

Подводя итог сравнения двух методов, хочется заметить, что, если характер сбоев в коммутаторах локальной сети демонстрирует чёткую тенденцию с течением времени, DES может оказаться более подходящим. Однако, если сбои более резкие или недавние данные более указывают на будущие сбои, оконное управление может быть лучше.

Что касается вычислительных ресурсов то для крупномасштабных сетей вычислительная эффективность оконной обработки данных может быть более практичной. Однако это требует дальнейших исследований, так-как тот же метод DES совместно с другими алгоритмами МО могут дать не менее эффективный результат.

В данном исследовании, после анализа множества работ по схожей тематике, был выбран метод DES как более приемлемый метод для решения поставленных задач. Более подробно об использовании метода DES совместно с другими алгоритмами МО будет рассмотрено в следующих главах.

**1.4 Анализ проблем при разработке предиктивной модели на базе МО**

1.4.1 Несбалансированные данные

В прогнозирующем моделировании, независимо от конкретной области или цели, одна из постоянных проблем – это недостаток образцов данных, представляющих события, которые необходимо прогнозировать. Часто эти события редки или необычны, что приводит к доминированию в наборе данных отрицательных образцов, где желаемый результат не происходит. Положительные образцы, где присутствует интересующий исход, обычно превосходят отрицательные. Это приводит к ситуации, которую обычно называют «несбалансированным набором данных» или просто «несбалансированной проблемой». Этот дисбаланс между положительными и отрицательными образцами оказывает влияние на всю процедуру создания модели машинного обучения. Некоторые широко используемые прогнозные алгоритмы, такие как Наивный байесовский алгоритм или метод опорных векторов, изначально предполагают сбалансированный набор данных во время обучения. Следовательно, при их применении к несбалансированному набору данных они могут столкнуться с трудностью в точной оценке значимости паттернов в положительных образцах или ошибочно идентифицировать их как шум. Поэтому для правильной оценки производительности модели необходимо либо сбалансировать классы в данных, либо использовать алгоритмы, специально разработанные для решения этой проблемы. Основным недостатком второго варианта является ограниченный выбор альтернатив и ограниченная поддержка в литературе, что может затруднить обоснование. Распространённый метод решения проблемы несбалансированных данных – это перевыборка данных, которая включает добавление или удаление образцов для достижения равновесия классов. Если добавляются положительные образцы, это называется перепроизводством; если удаляются отрицательные образцы, это называется недопроизводством. Существуют также гибридные методы обработки данных, сочетающие оба этих подхода [61-73].

1.4.2 Обзор проблемы при выборе алгоритмов машинного обучения

При выборе алгоритма для создания модели прогнозирования варианты разнообразны и многочисленны. В зависимости от характера прогноза, будь то оценка значения переменной, как в прогнозировании на фондовом рынке [74], [75], или прогнозирование сбоя или событий в виде бинарной классификации, можно использовать различные алгоритмы. Бинарные прогнозы, например, упрощаются до двух исходов: событие произойдёт (положительный) или не произойдёт (отрицательный). Классические алгоритмы машинного обучения подходят к этому по-разному, например, математическое моделирование с Naive Bayes, корреляция признаков с целевой переменной с помощью логистической регрессии или различение возможных значений целевой переменной с помощью Supported Vector Machine (SVM) и его вариации. Алгоритмы на основе деревьев, такие как дерево решений, случайный лес или градиентно-усиленные деревья, часто используются из-за их эффективности в прогнозировании на основе паттернов [76-78]. В разделе 2 и 3 описаны подробные сведения об этих алгоритмах, а также связанные концепции обучения и оценки. Некоторые исследователи даже модифицируют эти общие алгоритмы, чтобы лучше соответствовать уникальным требованиям своей задачи [79-81].

Стоит отметить, что алгоритмы реагируют по-разному на предварительную обработку обучающих данных, важный аспект повышения эффективности модели [82, 83]. Предварительная обработка, включая уменьшение шума и удаление несущественной информации, обычно включает в себя уменьшение размерности или выбор признаков. Несколько работ подчёркивают значение этого этапа и его положительное влияние на модели [84, 85].

Выбор алгоритма для создания модели прогнозирования является открытой проблемой, и хотя некоторые результаты предлагают модифицированные алгоритмы, ни один из них не имеет значительного признания. Поэтому по-прежнему необходимо сравнивать наиболее распространённые алгоритмы для каждого конкретного сценария. Другие проблемы с широко принятыми решениями, такие как использование методов выборки для работы с несбалансированными наборами данных или выбором признаков, были успешно протестированы в отдельности. Хотя многие из этих новых техник предлагают интересные новшества, отсутствие примеров их использования в различных сценариях ограничивает их возможное использование. Однако они редко присутствуют вместе в одной работе, поэтому сложно понять, как они влияют друг на друга.

1.4.3 Обзор проблемы доступности данных

В этой области одной из признанных проблем является проблема доступности данных. Большая часть исследовательских результатов основана на данных полученными частным способом, и контексты, в которых они применяются, часто очень специфичны [86, 87]. Эта ситуация затрудняет возможность сравнения достижений в этой области.

Однако есть пример, который отклоняется от этого общего правила: набор данных Blue Gene. Суперкомпьютер IBM Blue Gene/L в Национальной лаборатории Лоуренса Ливермора (LLNL) состоит из 128K процессоров. Его показатели надёжности, доступности и обслуживаемости (RAS) собираются в журнальный файл через систему мониторинга, и этот журнальный файл, содержащий события RAS, доступен публично. Эта доступность способствовала многочисленным исследовательским проектам, основанным на этом наборе данных, в частности тем, которые направлены на прогнозирование сбоев [88, 89]. Наличие этого набора данных в общественном достоянии предоставляет возможность сравнивать различные научные исследования. Хотя использование специфических атрибутов или точных деталей данных может улучшить результаты в конкретном сценарии, это также может значительно ограничить применимость предлагаемых решений в разных контекстах.

Проблема, озвученная ранее в этом разделе о доступности данных, особенно в проблеме прогнозирования сбоев компьютерных сетей, затрудняет повторное использование опубликованных результатов. Единственная доступная документация – это результаты, которые публикуются в статьях, которые обычно сводятся к окончательным результатам без включения промежуточного результата. Ещё один факт, который обычно не включается в публикации, – это производительность выполнения проекта. Однако обычно можно найти времена обучения или прогнозирования, но, как подчёркивается в данной работе, не весь процесс. Что касается методов моделирования, область знаний очень широкая и может быстро уходить от проблем прогнозирования, так как её применения разнообразны. Существуют решения для моделирования наборов данных в формате журнала или дискретных событий в целом. Однако следует подчеркнуть, что моделирование событий сбоев на основе взаимодействия событий – мало изученное поле. Некоторые модели моделирования, такие как основанная на агентах, включают взаимодействие и обмен информацией между различными участниками.

1.4.4 Обзор проблемы симуляции работы модели

В предыдущих разделах было подчёркнуто необходимость достаточного объёма данных для разработки эффективных моделей, в частности, решая проблему дисбаланса данных, преимущественно в задачах прогнозирования. Распространённым средством решения этой проблемы является создание синтетических данных. Существует несколько способов производства синтетических данных, например, объединение существующих данных для создания новых экземпляров, таких как техника SMOTE. SMOTE [90] – частый алгоритм в ситуациях дисбаланса данных – используется как для создания, так и для удаления образцов, то есть для перебора и недооценки. Его концепция заключается в создании синтетических экземпляров путём интерполяции текущих образцов в пространстве признаков, и он был эффективно использован для повышения эффективности прогнозных алгоритмов [91], включая прогнозирование результатов компиляции программного обеспечения.

Более сложные варианты могут включать в себя моделирование изучаемой системы. Инструменты симуляции имеют различные применения и могут использоваться для генерации синтетических данных или проверки функциональности модели или артефакта. Обзор в этой главе будет сосредоточен на симуляциях, используемых для генерации данных, в частности данных для обучения и оценки моделей машинного обучения.

Симуляции могут пытаться эмулировать работу всей системы или сосредоточиться на инъекции данных. В [92] проводится сравнение алгоритмов машинного обучения для прогнозирования сбоев программного обеспечения на основе вставленных синтетических данных. Инструмент [93] контролирует количество синтетических данных, внедряя сбои в регулярную работу системы. Другие варианты имитируют поведение системы с акцентом на дискретную симуляцию событий [94, 95], которая воспроизводит условия, генерирующие события. Несмотря на свою довольно долгую историю, основным недостатком этого метода является необходимость точного знания системы, что затруднительно в сложных средах, таких как распределённые системы. Здесь выделяется агентное моделирование как заметная альтернатива [97].

Агентное моделирование, известное под такими аббревиатурами, как ABM, ABS, MABS и ABMS, основано на концепции программных агентов 90-х годов [98, 99]. Оно строит сложные системы, способные к продвинутому принятию решений, через агентов, обрабатывающих простые задачи. Различные области, такие как биология, потребительские рынки, высокопроизводительные вычислительные системы, такие как Blue Gene, применяли эту симуляцию [100-102].

Совмещение агентного моделирования с машинным обучением – относительно неизведанная область до недавнего времени [103]. Авторы предлагают использовать агентов, которые учатся на данных, полученных из реального мира, для моделирования поведения агентов в симуляции данных [103]. Недавние работы, такие как [104, 105], используют агентные симуляции для наборов тренировок нейронных сетей и симуляций различных задач. Также предлагаются алгоритмы, позволяющие агентам проводить усиленное обучение [106].

Применения симуляции данных в машинном обучении не ограничиваются улучшением работы модели; они также могут тестировать системы, реализующие эти модели [104]. Некоторые учёные выступают за использование синтетических данных для проверки модели [107], обеспечивая определенные условия, в отличие от фактических данных, которые не позволяют этого контроля. В исследованиях также рассматривается использование алгоритмов машинного обучения для объяснения сложных поведений, возникающих из агентных симуляций [108], и применение симуляций в сочетании с машинным обучением для оценки рисков [109].

**1.5 Обзор применяемых метрик оценки прогностических моделей**

Методы оценки, используемые для анализа прогностических моделей, являются многообразными. Выбранные метрики могут различаться в зависимости от конкретной задачи, преследуемых целей или определенных интересующих аспектов. При работе с несбалансированными наборами данных выбор этих метрик должен осуществляться с большей осторожностью, так как различие в количестве образцов между классами может повлиять на репрезентативность метрики [67].

Например, в классификационных моделях метрика точности, рассчитанная как отношение правильных ответов классификатора к общему количеству ответов, может быть вводящей в заблуждение, если один класс значительно превосходит другой. Если модель хорошо справляется с большинством классов, но плохо с меньшинством (который обычно представляет интерес), метрика точности может все равно оставаться высокой. Поэтому выбранная метрика оценки должна учитывать этот дисбаланс.

Несмотря на описанный выше пример, метрика точности обычно используется из-за её лёгкой интерпретируемости [69, р. 935-946]. Другая простая метрика, которая используется, – это точность, которая рассчитывается как отношение правильно предсказанных положительных образцов ко всем положительным образцам. Эта метрика пропорциональна количеству правильных положительных прогнозов, поэтому она полезна, когда количество ложноположительных результатов не важно. Однако высокая точность не обязательно фиксирует редкие или неожиданные события [70, р. 233-239]. Полнота же измеряет отношение правильных положительных предсказаний ко всем фактическим положительным случаям. Вместе точность и полнота дают более полное представление о возможностях модели, хотя их объединение в кортеж (*упорядоченный набор фиксированной длины из двух значений*) может усложнить сравнения. F-мера, рассчитанная как гармоническое среднее точности и полноты, решает эту проблему и часто считается более информативной относительно эффективности классификатора. Согласно [69, р. 935-946], F-мера обычно используется и является более информативной о эффективности классификатора в правильном прогнозировании случаев, которые важны для пользователя. Другие метрики, такие как кривая рабочей характеристики приёмника (ROC), работают по-другому, будучи вычисленными как кривая на основе корректировок порога принятия решения модели. Тем не менее, эта метрика может представлять слишком оптимистичный взгляд в несбалансированных наборах данных [71, р. 359-362]. Выбор любой из вышеуказанных метрик также требует осознания того, что задачи прогнозирования часто сложны, в результате чего диапазоны метрик могут быть ниже, чем в других областях машинного обучения. Например, при прогнозировании отказов телекоммуникационного оборудования точность может быть всего 3%, а полнота 63% [72, р. 1-9], в то время как обнаружение отказов жёстких дисков может достигать 30% полноты с отношением ложных срабатываний 0,3% [28, р. 202-208], и прогнозирование отказов в центрах обработки данных может привести к F-мере 48,21% [73, р. 6-11].

**1.6 Сравнительный анализ работ по прогнозированию отказов**

Модели машинного обучения применимы в различных областях, в том числе и к прогностическим задачам, включая прогнозирование отказов. Обширный объём литературы подтверждает эффективность использования машинного обучения в этом контексте. Тем не менее, в отличие от многих проектов машинного обучения, детальная процедура создания прогностической модели из первичного сбора данных часто не описывается с аналогичной подробностью в этой конкретной области. В ходе обзора литературы не была найдена полностью всеобъемлющая работа, которая бы полностью описывала весь процесс. Некоторые публикации выделяют отдельные аспекты этого процесса, такие как сравнение алгоритмов, решение проблемы дисбаланса данных или выбор метрик оценки. Удивительно, что некоторые исследования даже не раскрывают подготовку данных, этап, известный своим существенным влиянием на окончательные результаты моделирования [110]. Таблица 1.2 была создана для визуализации сравнения выборки работ по прогнозированию отказов. В этом сравнении рассматриваются аспекты создания системы прогнозирования, охваченные в этих работах, выявляя, и выделяя подробно описанных аспектах, или наоборот, не описанных, либо недостаточно объяснённых. Задачей этого обзора не является намёк на то, что статьи в этом сравнении неполны или недействительны. Каждая из них вносит ценный вклад в знание в области и была опубликована в авторитетных научных журналах, подвергнутых должным методам валидации. Основной целью было лишь подчеркнуть, что публикации в этой области обычно концентрируются на определенных гранях процесса.

Таблица 1.2 – Сравнение рассмотренных аспектов в опубликованных работах по прогнозированию отказов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | DR | AB | MD | U | Тематика |
| 1 | 2 | 3 | 4 | 5 | 6 |
| + | - | - | + | - | Отказы телекоммуникационного оборудования |
| ∼ | + | - | + | - | Отказы компьютерных систем |
| ∼ | - | ∼ | - | - | Выявление и распознавание шаблонов ошибок, приводящих к сбоям |
| + | + | + | + | - | Прогнозирование сбоев в BlueGene/L |
| - | - | ∼ | - | - | Прогнозирование сбоев в онлайн-сервисах во время их реального жизненного цикла |
|  | | | | | |
| - | + | + | ∼ | ∼ | Прогнозирование редких событий сбоев в производстве |
| ∼ | - | - | + | + | Прогнозирование сбоев в распределённых бизнес-процессах |
| + | + | + | - | + | Обнаружение мошенничества в системах веб-платежей |
| + | ∼ | - | + | - | Сравнение прогнозирования на основе событий с прогнозированием с использованием окна наблюдения |
| - | + | - | + | - | Прогнозирование в режиме онлайн отказов в облачных дата-центрах |
| ∼ | ∼ | - | + | - | Универсальный алгоритм прогнозирования отказов в режиме онлайн |
| ∼ | + | + | + | ∼ | Прогнозирование отказов коммутаторов |
| Примечания:  1. + – есть и подробно описано.  2. ∼ – есть, но не решено или не описано.  3. - – нет.  4. P – Предварительная обработка.  5. DR – Уменьшение размерности.  6. AB – Сравнение алгоритмов.  7. MD – Обсуждение метрик.  8. U – Несбалансированность  9. Составлено по источникам [24, р. 219-221; 31, р. 891-895; 32, р. 1391-1398; 35, р. 220-234; 40, р. 2024-2027; 48, р. 1-8; 60, р. 106-116; 88, р. 15-21; 111-114] | | | | | |

Например, работа [31, р. 891-895] предлагает использовать генетический алгоритм для определения области поиска событийных образцов, которые приводят к сбоям в телекоммуникационных системах. В статье предлагается использование анализа образцов, поэтому его нельзя применить для решения проблемы дисбаланса в данном проекте. Также в статье содержится интересное обсуждение о том, какие метрики полезны для оценки моделей прогнозирования. Статья [48, р. 1-8] предлагает прогнозирующую программу сбоев в компьютерных системах на основе SVM с учётом образцов событий. Происходит поиск по сетке, чтобы найти лучшее окно наблюдения, но интервалы и границы этой сетки не обоснованы и приняты на основании мнения авторов. В работе [111, р. 504-510] авторы предложили использовать скрытые модели Маркова (HMM) для проблем прогнозирования сбоев. Это предложение аргументирует жизнеспособность HMM в таких задачах, но не сравнивает его с другими вариантами. Оно также упоминает предварительную обработку, которую нужно выполнить над обучающими данными, но не детализирует ее. Авторы статьи [88, р. 15-21] прогнозируют сбои в BlueGen/L. Исследователи предлагают агрегирующую функцию для создания входных данных модели, но функция специфична для проблемы и обоснована знанием предметной области. В статье приведено отличное сравнение алгоритмов между RIPPER, SVM, KNN и их настраиваемым KNN, с хорошим обсуждением о том, какую метрику они использовали для оценки моделей. Далее, работа [32, р. 1391-1398] предлагают использовать синтетические данные для обучения прогнозирующей модели на основе собственного алгоритма классификации. Хотя она сравнивается с другими алгоритмами, сравнение не является строгим, поскольку они использовали точность для оценки модели, и некоторые из сравниваемых алгоритмов не детализированы (например, нейронные сети) или имеют разные цели (например, кластеризация). Работа [40, р. 2024-2027] использует важность признаков для уменьшения размерности признаков, и авторы дают аргументацию и детальное сравнение алгоритмов. Однако проблема дисбаланса упоминается, но не решается. Исследовательская работ [35, р. 220-234] – это предложение по прогнозированию сбоев в распределённых бизнес-процессах, в котором авторы используют контролируемую инъекцию сбоев для работы с дисбалансом данных. Они также обсуждают правильную метрику для оценки моделей среди точности и полноты или MCC [115]. [24, р. 219-221] – это работа по обнаружению мошенничества, применённого к веб-платёжным системам, где используются выбор признаков и недопроизводство для улучшения моделей. [60, р. 1003-1019] – отличный пример исследования влияния окна наблюдения на прогнозы, но оно изучается только с одним алгоритмом. В [113, р. 1-28] предварительная обработка и уменьшение размерности упоминаются, но не детализируются. Odersky M., Spoon L., Venners B. предлагают функцию агрегации признаков для уменьшения количества входных переменных и работы с дисбалансом. Однако эта агрегация и многие принятые решения основаны на знании предметной области и не обоснованы, что затрудняет их повторное использование [116-128].

Наиболее распространённой метрикой в этих статьях является F-мера, которая аргументируется как баланс между точностью и полнотой. Только немногие из них включают сравнение алгоритмов. Проблема дисбаланса обычно упоминается, но не всегда решается. Выбор признаков и предварительная обработка обычно осуществляются на основе знаний в предметной области [129-144].

**Выводы по разделу 1**

В разделе 1 обсуждается ранее предложенные научные исследования по прогнозированию с использованием машинного обучения и симуляции в области предиктивной аналитики. Большая часть рассмотренной литературы состоит из научных статей, опубликованных в рецензируемых журналах. Кроме того, были процитированы книги и технические отчёты по выбранному направлению, что является характерным для некоторых упомянутых обзоров. Поиск осуществлялся с использованием специализированных поисковых систем, таких как IEE Explore и Google Scholar, а также путём анализа ссылок в изученных статьях и работах, ссылающихся на них.

В данном разделе подробно описан выбор направления исследования включающая обоснования направления исследования. Приведён обзор существующих методов решения поставленной задачи и проведена их сравнительная оценка. Таким образом, в следующем разделе будет подробно описана работа исследуемого объекта управления, процесс сбора данных, анализ и выбор входных параметров для построения качественной модели предиктивной аналитики сбоев ЛВС, а также, будет описан процесс подготовки датасета к дальнейшему использованию в предиктивной модели.

**2 ОПИСАНИЕ РАБОТЫ ОБЪЕКТА УПРАВЛЕНИЯ И СБОР ВХОДНЫХ ДАННЫХ**

**2.1 Описание работы ЛВС корпоративной сети**

Исследуемый набор данных извлечен из инфраструктуры провайдерской сети состоящих из коммутаторов и маршрутизаторов. В основном, активные оборудования ЛВС на 99% состояли из оборудования поставщика Cisco различных моделей, таких как WS C2960-48P, WS-C3750-48PS, WS-C3560-48PS-S, Catalyst 6509-E и другие более современные модели. Все сетевые оборудования, которых было более 110, были размещены в стандартизированных кросс и серверных комнатах. Соединения между коммутаторами осуществлялись через ВОЛС (волоконно-оптическую линию связи), используя звездную топологию, где коммутаторы Catalyst 6509-E служили ядрами сети, как показана на примере рисунка 2.1. По крайней мере половина (50%) коммутаторов была обеспечена источниками бесперебойного питания (ИБП), но только около 30% коммутаторов были оснащены системами климатического охлаждения и вентиляции. Кроме того, сеть передачи данных имела многоуровневую сертифицированную защиту нового поколения от внешних угроз, которая устраняла до 95% причин отказов, возникших вследствие потенциальных интернет-атак или действии зловредных программ. Кроме того, компания интегрировала систему мониторинга активного оборудования PRTG Network Monitor, которая отслеживала состояние сетевых коммутаторов и других устройств в реальном времени по протоколу SNMP и сложным внутренним системам датчиков. Но несмотря на эти усилия, время от времени происходили случайные отказы отдельных устройств/коммутаторов по неопределенным причинам. Этот конкретный случай был выбран для расследования, и впоследствии было предложено решение в виде системы прогнозирования отказов с использованием метода машинного обучения.

Процесс вышеупомянутого мониторинга провайдерской сети приведен на рисунке 2.2. Сбор данных осуществлялся с помощью программного обеспечения мониторинга PRTG Network Monitor, которое регистрирует все сообщения Trap, переданные по протоколу SNMP. Эти сообщения содержат подробности о предупреждениях, сигналах или сбоях, происходящих в сети. Хотя большинство из них относятся к сетевому уровню, некоторые касаются уровня приложения (обычно связано с услугами или приложениями, которые перестают функционировать из-за потери доступа к сети). Для поддержания конфиденциальности все поля данных, включая IP-адреса, имена машин, URL-адреса и т.д., были анонимизированы путем замены на обобщенные значения или удалены вовсе.

Наблюдаемая сеть представляет собой лишь срез из восьми L2 коммутаторов единого вендора, входящий в состав огромной развёрнутой сети. Набор данных включает все сообщения TRAP, сгенерированные этими устройствами с 12 декабря 2019 года по 10 декабря 2020 года, всего 363 дня. Сообщения собираются в итоговые файлы журнала в формате CSV, разделенном запятыми. Объединив журналы за двенадцать месяцев, всех восьми коммутаторов, результирующий журнал содержал 2923 записи, в среднем 8 сообщения в день от каждого коммутатора. Хотя это можно отнести к небольшому потоку, это соответствует ожиданиям от набора данных о сбоях, учитывая, что сбои должны быть случайным явлением.

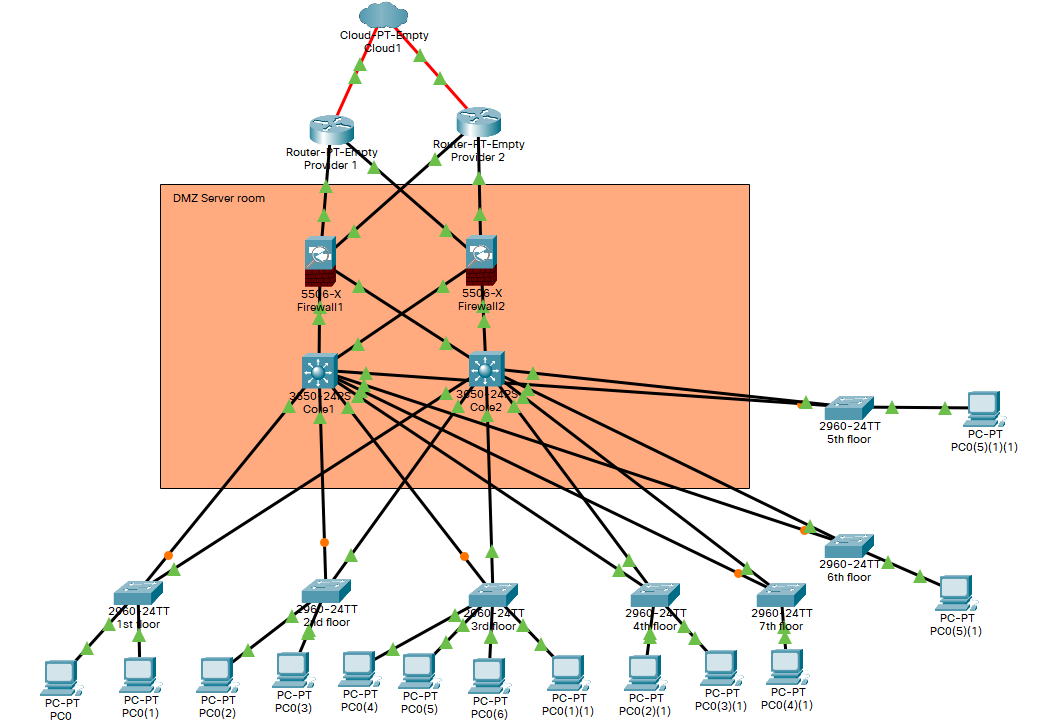


Рисунок 2.1 – Пример рассматриваемой ЛВС в виде структурной схемы

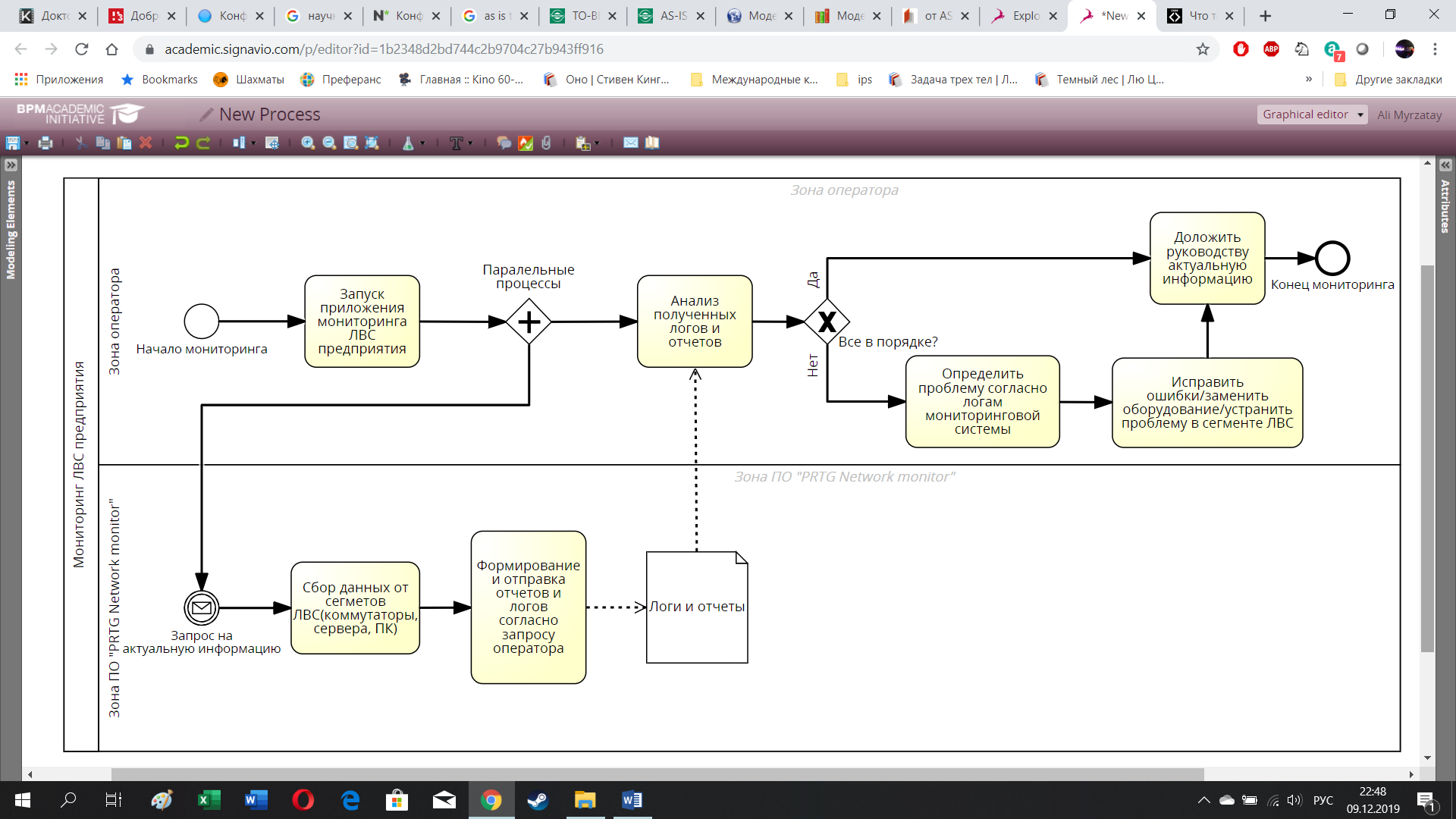


Рисунок 2.2 – модель процесса мониторинга ЛВС оператором

Каждый журнал соответствует единообразному формату, включая четырнадцать переменных, которые были выделены вручную из системы как имеющие значение для целей данного исследования:

1. *Дата:* дата, когда сообщение о статусе оборудования была записана. Формат записи была следующей: «ДД.ММ.ГГГГ», например, «10.12.2020». В дальнейшем, этот столбец был убран, так как не нёс никакой информативной нагрузки.
2. *Время бесперебойной работы:* дни, сколько работало оборудование без сбоев. Формат записи: числовой, например – «95,00», что означало что коммутатор работал 95 дней без перерыва и сбоя.
3. *Downtime или «Время простоя»:*это показатель периода времени, когда определенный коммутатор или устройство в сети недоступно или не функционирует должным образом. Это может произойти по различным причинам, таким как аппаратные сбои, программные ошибки, проблемы с подключением и так далее. Формат записи: процентный, например, если показатель 0%, это значит, что оборудование работало 24 часа без перебоев. Если показатель 0,1%, то это значит, что оборудование был простое до от 01 до 0,2% времени 24 часов, что приравнивается приблизительно к 2 минутам.
4. *CPU Load*:загруженность центрального процессора коммутатора. Формат записи: числовой, например «8,12», что означало загруженность ЦП на 81,2% от полной 100% мощности
5. *CPU 1*:Коммутаторы серии Catalyst 6500/6000 оснащены двумя центральными процессорами (ЦП). Один из них служит процессором для управления сетью, известным как процессор сетевого управления (NMP) или процессор коммутации (SP), в то время как второй ЦП, известный как MSFC или процессор маршрутизации (RP), отвечает за маршрутизацию на 3 уровне. В данном контексте рассматривается загрузка процессора коммутации (SP), который выполняет ряд ключевых функций:
   * участвует в процессе обнаружения и обновления MAC-адресов, процесс также известен как настройка маршрутизации;
   * отвечает за запуск протоколов и процессов, необходимых для управления сетью, включая протокол связующего дерева (STP), протокол обнаружения Cisco (CDP), транкинговый протокол VLAN (VTP), динамический транкинговый протокол (DTP) и протокол агрегации портов (PAgP);
   * обрабатывает сетевой управляющий трафик, направленный к ЦП коммутатора, включая трафик от протоколов Telnet, HTTP и SNMP.

Формат записи – числовой, например – «8,76».

1. *Доступная память процессора 1*: числовое значение доступности памяти процессора 1 на момент измерения. Формат записи: числовой, единица измерения – Мбайт. Например, «52,00» означает что доступно 52 Мбайт памяти для процессора.
2. *Процентное соотношение доступной памяти процессора 1***:** процентное соотношение доступности памяти процессора коммутатора от 100%. Формат записи – процентный, например, «37%». Это означало, что ещё 37% от 100% памяти процессора коммутатора доступна для выполнения задач. При очистке элементов данных был убран символ «%» из-за того, что модель прогнозирования не могла считывать этот символ.
3. *Доступная память 2:*В этом разделе представлена информация о памяти второго процессора, который не является стандартным компонентом всех коммутаторов. Процессор маршрутизации (RP) выполняет ряд ключевых функций:
   * ответственен за формирование и обновление таблиц маршрутизации третьего уровня и таблиц ARP (протокол разрешения адресов);
   * создаёт таблицы базы данных пересылки (FIB) для CEF и таблицы смежности, после чего загружает их на плату поддержки политик (PFC);
   * обрабатывает сетевой управляющий трафик, направляемый к RP, включая данные от протоколов Telnet, HTTP и SNMP.

Формат записи – числовой, единица измерения – Мбайт, например, «1,73» означает что ещё 1,73 Мбайт памяти доступно для выделения на обработку каких либо процессов.

1. *Процентное соотношение доступной памяти процессора 2:* процентное соотношение доступности памяти процессора коммутатора от 100%. Формат записи – процентный, например, «43%». Это означало что ещё 37% от 100% памяти процессора коммутатора доступна для выполнения задач. При очистке элементов данных был убран символ «%» из-за того, что модель прогнозирования не могла считывать этот символ.
2. *Доступная память (в общем):* показатель доступности памяти в процентном соотношении. Формат записи: процентный, например «100%». Это означало что все 100% памяти/мощности ОЗУ доступно для использования. При очистке элементов данных нами был убран символ «%» из-за того, что модель прогнозирования не могла считывать этот символ. Далее, для упрощения данных число 100 было сокращено до 1.
3. *Индекс времени отклика:*это метрика, используемая для измерения времени отклика сетевого устройства или приложения. Это может быть важным показателем качества обслуживания (QoS) в сети, особенно в сценариях, где задержка является критическим фактором, таких как голосовые или видеоконференции.

Время отклика является временем между отправкой запроса и получением ответа. Оно включает в себя задержку передачи по сети, задержку обработки на устройстве и другие факторы. RTI может быть использован для мониторинга и анализа производительности сети, и, если значение RTI становится слишком высоким, это может указывать на проблемы в сети, которые требуют расследования.

Формат измерения – числовой, в миллисекундах (мс). Эта метрика отражает время, необходимое для передачи пакета от отправителя к получателю и обратно, и может включать различные виды задержек, такие как задержка передачи, задержка обработки и др. Например: «70», что означает 70 миллисекунд:

* + оптимальный RTI: Если у вас есть RTI равный 20 мс, это может считаться отличным временем отклика для большинства приложений.
  + средний RTI: RTI в 100 мс может быть приемлемым для некоторых типов трафика, но уже может вызывать заметные задержки в реальном времени для голосовых и видео приложений.
  + высокий RTI: Если RTI достигает 300 мс или более, это, скорее всего, будет указывать на серьезные проблемы с задержкой в сети, которые могут негативно сказаться на производительности сетевых приложений.

1. *CPU Load Index:*этот показатель представляет собой информацию, которая отражает нагрузку на центральный процессор (CPU) в данное время. Это количественная оценка того, насколько интенсивно используется процессорное время, и она может быть важным индикатором общей производительности и состояния устройства. В коммутаторах Cisco, CPU Load Index обычно измеряется в процентах от 1% до 100%, где:
   * 1-30%: Обычно считается нормальной нагрузкой, указывающей на здоровую работу устройства.
   * 31-70%: Умеренная нагрузка, которая может потребовать дополнительного наблюдения. Это может быть связано с временным увеличением трафика или задачей, которая использует больше ресурсов.
   * 71-100%: Высокая или критическая нагрузка, которая может указывать на проблему. Это может быть связано с неправильной конфигурацией, атакой, дефектом оборудования или другими проблемами, которые могут негативно сказаться на общей работе сети.
2. *Traffic Index:* является показателем, используемым для количественной оценки объёма трафика, проходящего через коммутатор в определенный промежуток времени. Анализ Пропускной Способности: Определение скорости передачи данных через определенный порт или интерфейс коммутатора. Например, Traffic Index может показать, что определенный порт используется на 70% от его максимальной пропускной способности. В данном случае записывались данные с TRUNK портов (канальных, соединяющих коммутатор с другими сетевыми коммутаторами). Формат записи: числовой. Например, показатель «9,32» означает что канальный порт работал в среднем со скоростью 9,32 Мбайт/сек.
3. *Alarms:*являются записями системы оповещений, которая предназначена для уведомления администраторов сети о различных событиях, проблемах или аномальных состояниях, которые могут потребовать внимания или вмешательства. Формат записи – числовой. Например, «1» значит, что была 1 тревога от системы. «3» – значит, что было 3 тревоги от системы.
4. *Температура (Temperature):* Большинство коммутаторов Cisco оснащены встроенными датчиками температуры, которые непрерывно измеряют температуру в различных точках устройства. В данном случае, формат измерения был числовой, в градусах C0, например, запись «42,0» означал что на момент измерения, температура оборудования была в области 42 C0. При очистке элементов данных был убран символ «C0» из-за того, что модель прогнозирования не могла считывать этот символ.

Была проведена сравнительная работа с другими вендорами на наличие аналогичных сенсоров. Выявлено что большинство современных коммутаторов L2 и L3 уровней имеют датчики необходимые для мониторинга состояния ЛВС. Так, например, в вендоре Juniper мониторинг состояния оборудования возможно осуществить через встроенный шаблон Template Net Juniper SNMPv2 (предварительно на свиче надо настроить параметр community=read-only). В этом случае, встроенный шаблон начал работать и может собрать большой массив данных: как нужных для анализа, так и не нужных – около 600 различных элементов данных. В качестве оптимизации сбора показателей, можно создать уникальный шаблон Template Juniper EX Custom, и скопировать в него 10 элементов данных из встроенного шаблона. Например: температура; загрузка процессора; сигналы тревоги (alarms); серийный номер; модель; используемая и свободная память; CPU Usage %. Далее идет процесс назначения имен процессов за которым следует вести мониторинг, и другие тонкие настройки, после чего можно добиться реузльтатов в виде записей сенсоров.

Сетевые устройства вендора Huawei, например S5700-28C-HI, также можно вести мониторинг состояния через SNMPv2 протокол. В данном случае, эти устройства лишь были добавлены нами в уже существующую мониторинговую систему PRTG Network Monitor, затем выбрали из списка сенсоров необходимые, например: ping, время бесперебойной работы, траффик на портах и т.п. Однако, чтобы иметь возможность контролировать дополнительные сенсоры, нам потребовалось импортировать MIB – Huawei Enterprise MIB. После проделанных действий появилась возможность вести мониторинг коммутаторов вендора Huawei по дополнительным параметрам: загрузка ЦП, используемая и свободная память.

При обсуждении топологии ЛВС исследуемого субъекта, было решено пренебречь этим условием, так как главная задача состояла в прогнозировании поломок отдельных устройств сегмента ЛВС, так как любая поломка сетевого оборудования являлась критичной и приводило к застою в работе организации.

**2.2 Процесс предварительной подготовки данных**

Одним из критических аспектов данного исследования является разработка моделей, способных предсказывать наступление определенных событий. Для достижения целей, в процессе проверки выбранных моделей применялся метод k-блочных кросс-валидации (K-Folds) (k-блочная перекрёстная проверка). Предложенные модели предварительно имитировались в ПО RapidMiner Studio version 9.8.

Перед непосредственно моделированием системы прогнозов, был этап предварительной подготовки загружаемых данных и предварительного моделирования данных – процесс, где загружаемые данные анализировались и на него применялись шаблоны существующих базовых типов моделей машинного обучения для прогноза параметра «Breaking» (Поломка). Согласно полученным результатам предварительного моделирования, описанного в разделе 3.3, наиболее перспективными моделями определились как kNN, Decision Tree, Random Forest, SVM. В дальнейшем, все предложенные модели кроме kNN показали хорошие результаты. Модель kNN был ориентирован на показатели других, соседствующих коммутаторов, что в итоге смазывало результат прогнозов, так как требовалось прогноз сбоя тестируемого коммутатора на основе его собственных показателях, а не на показателях других, соседствующих коммутаторах.

Следующей проблемой, которую необходимо было решить – проблема объединения в единый дата-фрейм (массива данных) разрозненных записей. Если описывать более подробно, то существует проблема в разрозненных массивах данных, которые выдают лицензированные, или любые другие системы мониторинга. Для систем прогнозирования требуются файлы, где разрозненные и не идентифицированные данные объединены в одну единую форму, чтобы корректно обработать запросы и выдать корректные прогнозы на выходе системы.

В ходе решения данной задачи была автоматизирована процедура преобразования исходных данных, получаемых от системы мониторинга, в формат, подходящий для использования в предсказательных моделях, основанных на машинном обучении. Пример выгрузки данных из системы мониторинга демонстрируется на рисунке 2.3. В процессе разработки приложения для трансформации и предварительной обработки данных обнаружился ряд проблем:

1. Основной трудностью при работе с данными от программного обеспечения PRTG Network Monitor было то, что все данные представлены в формате PDF, а метаданные – в формате CSV, что не подходило для их дальнейшего анализа в рамках модели машинного обучения.
2. Кроме того, увеличение количества сенсоров, данные которых необходимо экспортировать как метаданные, приводило к созданию большого числа отдельных файлов CSV. Это создавало значительные трудности, так как работа с множественными базами данных в контексте машинного обучения становится сложной и неэффективной.
3. Ключевой аспект проблемы заключался в том, что ручная подготовка данных специалистом занимала бы значительно больше времени, что делало автоматизацию этого процесса критически важной для последующего использования в системах предсказательного анализа.
4. Финальный продукт должен был быть экономически выгодным. Например, разработка собственного скрипта или программы компанией могла бы исключить необходимость в приобретении дорогостоящего лицензионного программного обеспечения сторонних производителей.

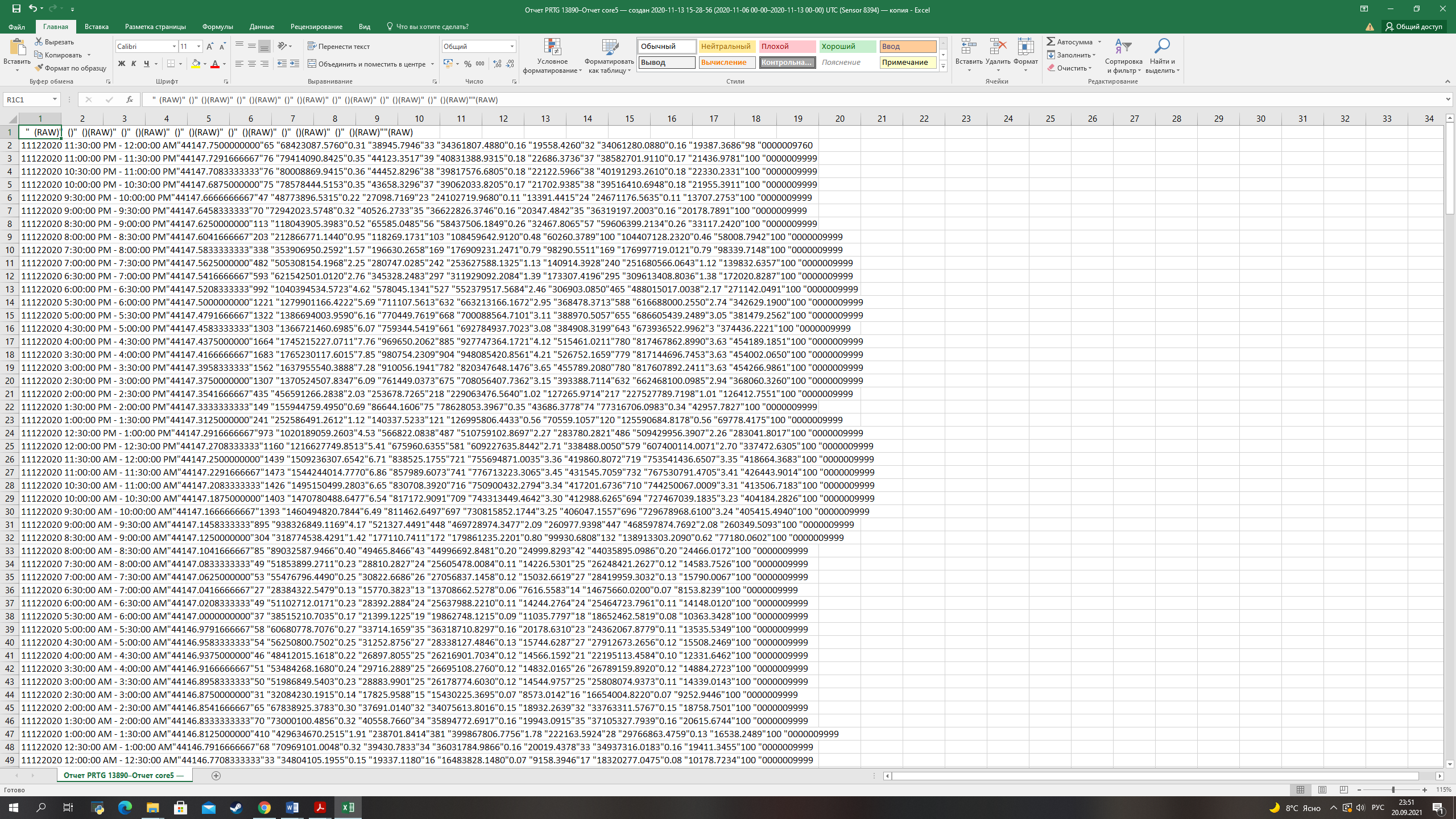


Рисунок 2.3 – Пример выгрузки данных из программы мониторинга ЛВС

В качестве примера, данные за год работы были собраны с датчиков нескольких коммутаторов с использованием программного обеспечения мониторинга системы. В исходном виде эти данные не могут быть легко обработаны другими системами, поэтому они были структурированы с помощью приложения-скрипта, который будет описан далее.

**2.3 Автоматизация процесса сбора и преобразования исходных данных**

Интегрированная среда разработки (IDE) представляет собой комплексное программное решение, предназначенное для облегчения процесса создания программного обеспечения разработчиками. Она объединяет в себе ряд инструментов, включая текстовый редактор, компилятор или интерпретатор, инструменты для автоматизации процесса сборки и отладки. Кроме того, IDE часто включает в себя средства для интеграции с системами контроля версий и другие дополнительные утилиты. Некоторые IDE специализируются на одном языке программирования, в то время как большинство современных IDE поддерживают работу с множеством языков. Примеры таких сред включают PhpStorm, CLion, Microsoft Visual Studio, PyCharm, Eclipse и Komodo. Во всех этих примерах оптимизирован процесс работы с данными. Для данных целей был выбран PyCharm и Jupyter Notebook за их комплексный набор инструментов, способствующих эффективной разработке на Python. Важно отметить, что разработка программного обеспечения с пользовательскими интерфейсами на объектно-ориентированном языке Python осуществляется более гладко и эффективно.

Для решения задач, изложенных в предыдущей главе, авторы использовали встроенную библиотеку Python – Pandas, а также интерпретатор Jupyter Notebook:

– в рамках этой работы был разработан скрипт для извлечения и объединения нескольких CSV-файлов в один общий dataframe (таблицу), как показано на рисунке 2.4.

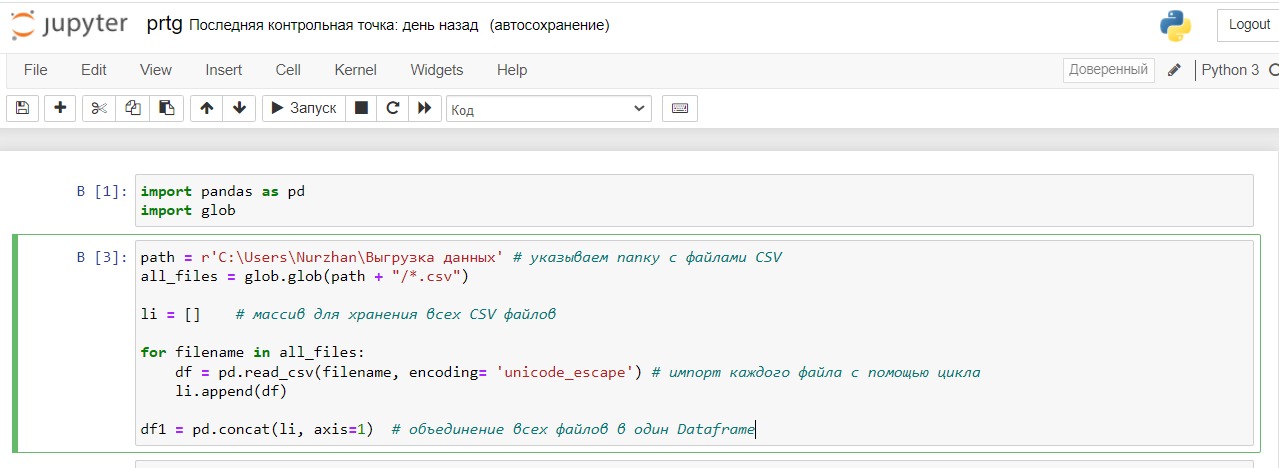


Рисунок 2.4 – Код объединения нескольких CSV файлов

После объединения данных из разных CSV-файлов в один dataframe, было обнаружено наличие повторяющихся столбцов, не несущих уникальной информации, как видно на рисунке 2.5.

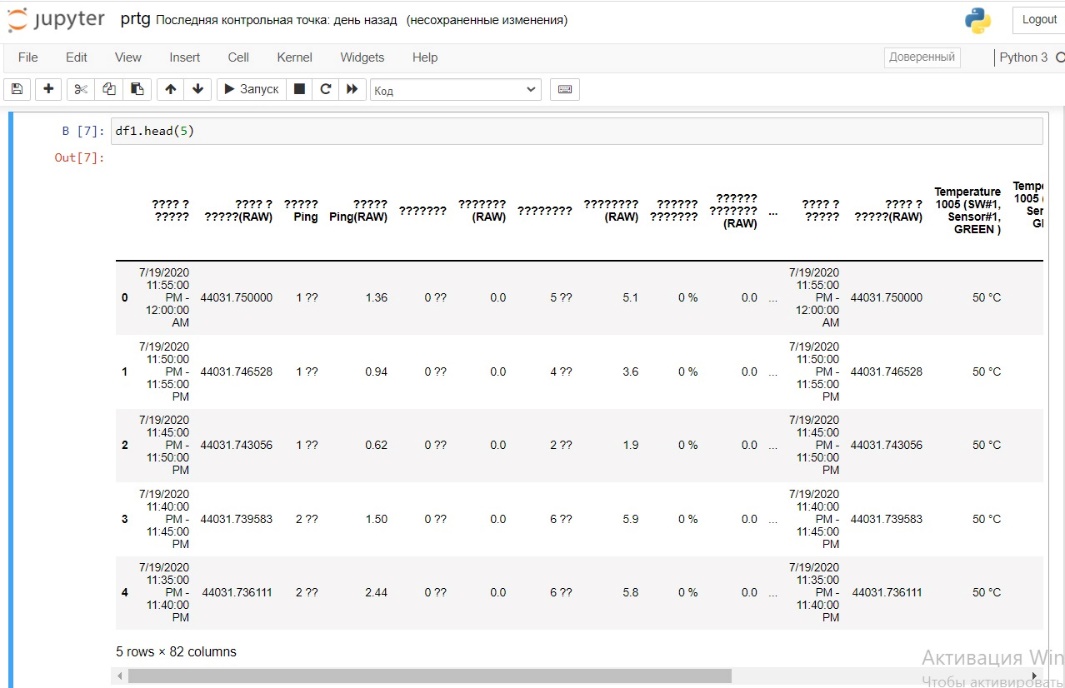


Рисунок 2.5 – Объединение файлов в один dataframe файл

Следующий этап включал удаление этих повторяющихся столбцов и столбцов с сырыми данными, помеченными как RAW, для упрощения анализа. В результате из 82 столбцов было отобрано 20 уникальных, как показано на рисунке 2.6.

После очистки dataframe от дублирующихся и сырых данных последовала лингвистическая корректировка названий столбцов. Этот этап завершает подготовку данных для их последующего использования в моделях машинного обучения, как демонстрируется на рисунке 2.7.

Срез кода (2.1), представленный ниже предназначен для слияния данных из всех CSV-файлов в указанной директории в один общий датафрейм с использованием библиотеки pandas. Путь к папке с файлами задан в переменной path, и в дальнейшем используется библиотека glob для получения списка всех CSV-файлов в данной папке. Основная функция этой библиотеки – 'glob.glob()', которая возвращает список файлов, соответствующих заданному шаблону. Шаблоны указываются с использованием обычных символов шелла, таких как \*, ?, []. Например, \*.csv будет соответствовать всем файлам с расширением .csv в текущем каталоге.

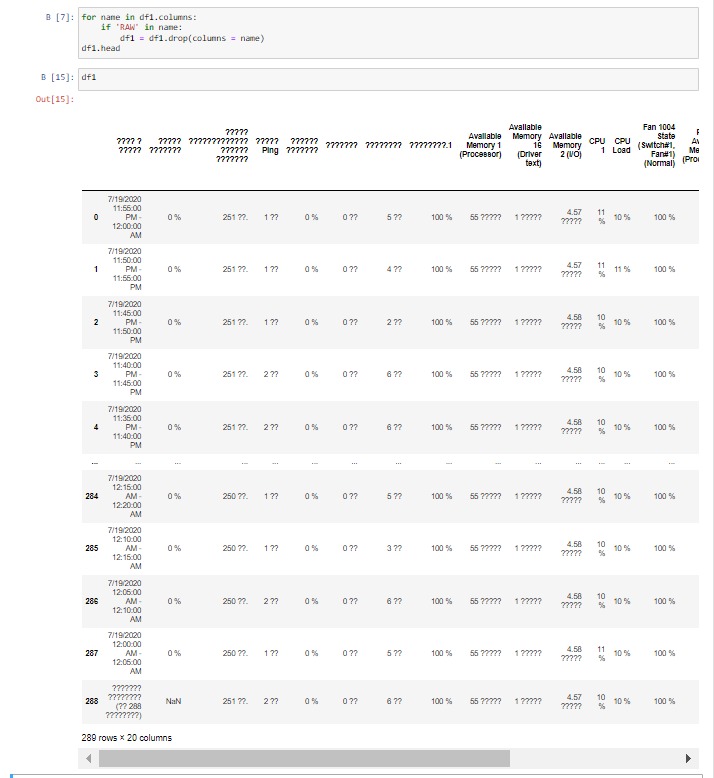


Рисунок 2.6 – Предитоговый вид dataframe-а

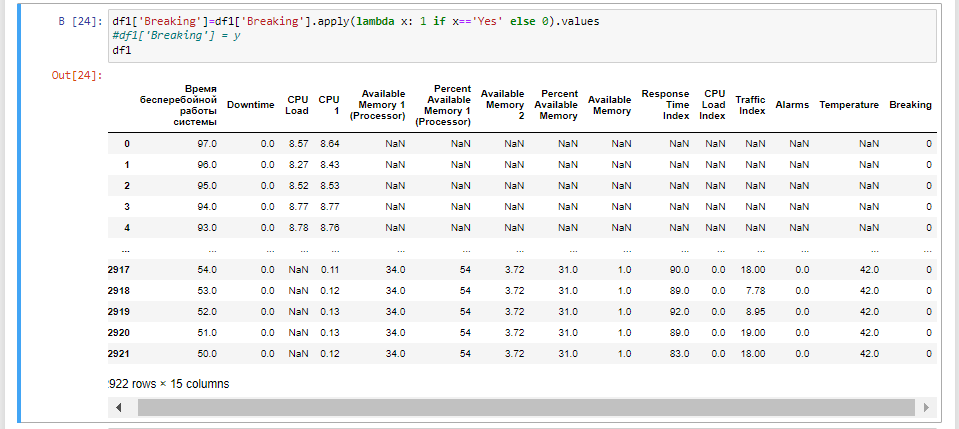


Рисунок 2.7 – Итоговая версия dataframe файла

Затем, каждый файл считывается поочерёдно, и датафреймы, полученные из каждого файла, добавляются в список. Процесс объединения происходит по оси 1, то есть файлы сливаются горизонтально. Однако, если вдруг, в других случаях структура файлов одинакова и необходимо объединить их вертикально, следует изменить значение параметра оси на 0 (axis=0). При чтении CSV-файлов используется кодировка 'unicode\_escape'. Если в файлах используется кириллица, возможно, потребуется изменить кодировку на 'utf-8' или другую подходящую. В конце все эти датафреймы объединяются в один с помощью функции pd.concat.

Присутствующий элемент 'groupby(level=0)' – это группировка транспонированного DataFrame по уровню индекса 0. В контексте этого кода, это служит удалению дубликатов в названиях столбцов. Далее следует указание что если название столбца содержит подстроку «RAW», то данный столбец удаляется из DataFrame. В итоге, все столбцы, содержащие «RAW» в названии, будут удалены из df1. Весь код процесса преобразования и объединения файлов в единый датафрейм представлен в (Приложении А).

(2.1)

Математическая модель этого процесса будет выглядеть следующим образом:

*Обозначения:*

1. *Mi* – *i*-тая матрица данных, представляющая *i*-ный .csv файл. Каждая матрица имеет размер 365×14 (365 строк и 14 столбцов).
2. *S –* множество всех i-тых матриц Mi​, S={M1​,M2​,…,M8​}.
3. *Cj​ –* j-тый столбец в матрице Mi​. Cj​∈Mi​, j=1,2,…,14.
4. *C*uniq​ – множество уникальных столбцов после удаления дубликатов.
5. *U* – итоговая матрица, составленная из столбцов *C*uniq​.

*Операции:*

1. Объединение матриц:

(2.2)

где *n=8* – количество CSV-файлов.

1. Удаление дубликатов в столбцах:

(2.3)

где – *j-*тый столбец *i-*ной матрицы *Mi*.

1. Формирование итоговой матрицы выглядит следующим образом:

(2.4)

где – *j-*тый уникальный столбец из *C*uniq​.

1. Удаление столбцов с определенным маркером (например, 'RAW'):

(2.5)

где – операция удаления столбцов с маркером 'RAW'.

В этом разделе подробно описывается контекст всех экспериментов и проверок, сосредоточенных вокруг набора данных, извлеченного из реальной производственной сети, предоставленной провайдерской компанией. Природа данных, происходящих из реальной инфраструктуры, подразумевает, что сырые данные будут нечистыми. Это требует предварительной обработки и фильтрации, поскольку не каждая информация может быть использована в анализе данных. Это может быть связано с неточностями в записанном количестве записей или другими ошибками измерений.

В следующих разделах будут подробно представлены набор данных, отчет об аналитическом исследовании, описывающий его основные характеристики, и обязательную очистку и трансформацию, выполненную для получения формата, подходящего для экспериментов.

**Выводы по разделу 2:**

В данном разделе был представлен объект исследования – ЛВС сеть организации, состоящей из коммутаторов L2 и L3 уровней, по топологии звезда. Большинство компонентов ЛВС были от одного вендора, что исключает мультивендорность ЛВС. При этом, наличие компонентов ЛВС отличающегося от основного вендора не является препятствием для сбора данных с датчиков устройств ЛВС. При детальном разборе объекта исследования было выявлено несколько проблем: проблема сырых данных, получаемых от системы мониторинга; проблема выбора наиболее актуальных моделей машинного обучения для прогнозирования сбоев; проблема улучшения качества получаемых результатов. Все проблемы нашли свои технические и алгоритмические решения. Были определены ключевые модели для дальнейшего моделирования в интегрированной среде разработки. В следующих главах будут описаны процессы предварительного моделирования, а также процесс создания трёх моделей прогнозов отказов элементов ЛВС, и описаны ключевые параметры этих моделей.

**3 ОБЩАЯ МЕТОДОЛОГИЯ СОЗДАНИЯ МОДЕЛЕЙ ПРОГНОЗИРОВАНИЯ СБОЕВ ОБОРУДОВАНИЯ ЛВС ИЗ ДИСКРЕТНОГО НАБОРА ДАННЫХ СОБЫТИЙ**

**3.1 Определение модели прогнозирования**

Модель прогнозирования не подпадает под стандартизированное понятие, поскольку различные методы могут быть отнесены к таковым. В рассматриваемом конкретном контексте, модель прогнозирования может рассматриваться как модель классификации, которая принимает числовой вход, отражающий историческое поведение системы, и выдаёт двоичный вывод, символизируя возможное наличие или отсутствие определенного будущего события. Различные алгоритмы машинного обучения (МО), такие как логистическая регрессия, случайный лес, машина опорных векторов и др., могут формулировать эти модели, если они устанавливают двоичную модель классификации. В разделе 2.2 было проведено предварительное моделирование входных данных с последующей проверкой на построенной модели с перекрёстной проверкой. По результатам построенной модели с перекрёстной проверкой, была выявлена высокая эффективность методов SVM, деревьев решений и случайных деревьев. В дальнейшем, основной упор был на это моделирование и попытались проверить вышесказанные методы с реальными данными. Подробности этого процесса будут описаны в следующем разделе.

Таким образом, эти модели прогнозирования будут относиться к программному обеспечению, выполняющему функцию *f*, как описано:

(3.1)

Основная идея заключается во взаимосвязи встречаемых событий в отслеживаемой системе. Определяя определенные обыденные модели, может быть выполнимо предвидеть предстоящие события. Использование МО становится существенным, поскольку эти модели сложные и не жёсткие, возможно показывая временные изменения. Даже в умеренно масштабной системе, как в этом рассматриваемом, множество различных событий делает невозможным для человека-наблюдателя определение или даже предположение о моделях и влияниях. Важно осознать, что данная диссертационная работа не пытается раскрыть причинно-следственную связь между событиями. Искомые модели определяют события, которые обычно предшествуют другим, а не события, которые вызывают другие. Хотя это может показаться тонкой разницей, истинное понимание причины события требует знаний в определенной области. В противном случае можно сделать только предположения. Кроме того, как упоминалось ранее, работа выполняется без консультации с экспертом в данной области.

Модели прогнозирования, которые используются в данной работе, являются моделями бинарной классификации, и это имеет ряд преимуществ, таких как возможность использования стандартных методов оценки для моделей классификации. В случае бинарной классификации большинство этих методов основаны на матрице ошибок, которая состоит из четырёх значений и позволяет оценить, сколько раз модель выдаёт правильный или неправильный прогноз. Матрица ошибок регистрирует четыре состояния (таблица 3.1):

Таблица 3.1 – Матрица ошибок

|  |  |  |  |
| --- | --- | --- | --- |
| Матрица ошибок | | Реальные значения | |
| *Положительные* | *Отрицательные* |
| Показатели прогностической модели | *Положительные* | *Истинно-положительные* | *Ложно-положительные* |
| *Отрицательные* | *Ложно-отрицательные* | *Истинно-отрицательные* |
| Примечания:   1. *Истинно-положительные (True Positive [TP]):* Количество положительных примеров, которые модель классифицирует как положительные. 2. *Истинно-отрицательные (True Negative [TN]):* Количество отрицательных примеров, которые модель классифицирует как отрицательные. 3. *Ложно-положительные (False Positive [FP]):* Количество отрицательных примеров, которые модель классифицирует как положительные. 4. *Ложно-отрицательные (False Negative [FN]):* Количество положительных примеров, которые модель классифицирует как отрицательные. | | | |

На основе этих определений общее количество правильно классифицированных образцов равно TP+TN, а количество неправильно классифицированных образцов равно FP+FN. Общее количество образцов равно N=TP+TN+FP+FN. Как говорилось ранее, данные, используемые в этой диссертационной работе, являются сильно несбалансированными, что предполагает, что значение TP будет гораздо больше других значений. Кроме того, образцы TP являются менее важными, поскольку они могут быть интерпретированы как «ничего не происходит, и модель прогнозирования не генерирует тревогу». В связи с этим, использовалась метрика, в которой это значение имеет низкий вес или вообще не учитывается. Некоторые из наиболее распространённых метрик, полученных из этой матрицы, включают:

* общая точность предсказанной модели по всем классам (Accuracy): Это отношение правильно классифицированных образцов к общему числу образцов.

. (3.2)

Однако, как говорилось выше, *TN* доминирует, и эта метрика даёт хорошие результаты даже для плохих моделей прогнозирования.

* точность модели при определении класса «Положительно» (Positive) (Precision): Точность относится к доле положительных образцов, которые были правильно классифицированы, и представлена формулой (3.3):

. (3.3)

С точки зрения прогнозирования, это рассматривается как мера того, как часто прогноз модели о наступлении события соответствует фактическому наступлению этого события.

* полнота (Recall): это метрика, используемая для оценки способности классификационной модели правильно идентифицировать все релевантные экземпляры. В контексте бинарной классификации полнота вычисляется как отношение истинно положительных результатов (TP) к сумме истинно положительных и ложно отрицательных результатов (FN), то есть формула (3.4) для расчёта полноты выглядит так:

. (3.4)

* False Positive Rate (FPR):

. (3.5)

уровень ложноположительных срабатываний – это метрика, которая измеряет долю неверно классифицированных отрицательных экземпляров среди всех истинно отрицательных. В других словах, это доля отрицательных экземпляров, которые неверно были классифицированы как положительные.

* F-мера (F-score или F1 score): Этот показатель объединяет две ключевые метрики оценки модели: точность (precision) и полноту (recall). Он представляет собой гармоническое среднее этих двух метрик, предоставляя комплексную оценку эффективности модели классификации, особенно в случаях, когда классы представлены в данных неравномерно. Расчёт F1 score осуществляется по следующей формуле (3.6):

. (3.6)

F1 score принимает значения в диапазоне от 0 до 1. Значение 1 указывает на идеальную точность и полноту, в то время как значение 0 указывает на полное отсутствие точности и полноты.

В рамках данной диссертации, из соображения простоты и надёжности выбрали метрику «*accuracy*» так как эта метрика является наиболее распространенным для поставленной задачи.

**3.2 Масштабируемость алгоритмов прогнозирования**

Хотя сценарий, из которого извлекаются данные для этой работы, является реальной компанией, не все сети имеют одинаковые размеры или возможности использования ПО. Решения должны быть применимы к более обширным сетям, большему объёму данных или большему разнообразию событий. Простейший способ обеспечить это требование – заставить решения работать в распределённой среде. В рамках данной диссертации в качестве библиотеки был выбран Scikit-learn из-за его легко использовать для быстрого прототипирования. Вот несколько причин который обуславливал выбор данной библиотеки:

*Простота Интерфейса:* Scikit-learn имеет интуитивно понятный и унифицированный интерфейс. Это облегчает работу с различными алгоритмами машинного обучения, даже если вы впервые с ними работаете.

*Наличие Готовых Решений:* Библиотека включает в себя широкий спектр готовых алгоритмов и функций для предварительной обработки данных, обучения моделей и оценки их качества. Это сокращает время разработки.

*Поддержка Сообщества:* Обширное сообщество и наличие многочисленных примеров и руководств в интернете делают процесс обучения и применения Scikit-learn менее затратным по времени.

*Фокус на Экспериментирование:* Возможность быстрого изменения моделей, параметров и алгоритмов обработки данных позволяет проводить эксперименты на начальных этапах проекта, чтобы определить наилучшие подходы.

Соответственно, эти особенности делают Scikit-learn отличным инструментом для быстрого прототипирования, то есть для создания начальных, экспериментальных версий моделей машинного обучения, которые затем можно усовершенствовать и оптимизировать.

3.2.1 Вычислительная среда

Необходимо вкратце описать конфигурации, в которых проводились все эксперименты. Хотя характеристики машин не влияют на результаты, они влияют на продолжительность вычислений. Есть две различные конфигурации: первая – это мощный рабочий ПК для работы над различными программами и проектами, а вторая – личный ноутбук. Рабочий ПК – исключительная машина с достаточно мощной производительностью позволяющая запускать распределённые алгоритмы и исследовать их горизонтальную масштабируемость до определенного уровня. Местная конфигурация, личный ноутбук, менее мощна, но достаточна для многих экспериментов, требующих ограниченных вычислений, и обеспечивает более простой доступ к результатам по сравнению с распределённой средой.

* *среда A: Рабочий ПК:*

1. Модель процессора: Ryzen 5 5800X, 8 ядер и 16 потоков, рабочая частота – 3,8 ГГц.
2. Операционная система: Windows 10.
3. ОЗУ: 32 ГБ DDR4 с частотами 3200 МГц.
4. SSD: файловая система с 4 ТБ места для хранения, содержащая исходный набор данных и его различные трансформации.

* *среда B: Ноутбук:*

1. Модель: Legion Y530.
2. Операционная система: Windows 10.
3. Процессор: Intel Core i5-8300h 2.3 ГГц с 4 ядрами.
4. ОЗУ: 16 ГБ DDR4 2667 МГц.
5. HDD: файловая система с 500 Gb места для хранения, содержащая исходный набор данных и его различные трансформации.

Этот ноутбук использовался для большинства задач, включая сканирование данных, сбор результатов и создание диаграмм. Здесь также проводились тесты, включая обучение моделей. Таким образом, вычислительная мощность рабочего ПК примерно в два-два с половиной раза превышает мощность ноутбука.

**3.3 Предварительное моделирование в ПО и корреляционный анализ данных**

В самом начале исследовательской деятельности был произведён анализ входных данных собранных, описанным методом в главе 2, через программное обеспечение RapidMiner Studio. RapidMiner Studio – это программный продукт для работы с данными, который предоставляет инструменты для анализа, машинного обучения, предварительной обработки данных, визуализации и статистического моделирования. Он широко используется в различных отраслях для аналитических и исследовательских задач. Один из основных плюсов RapidMiner Studio заключается в том, что в нем имеются средства для создания и валидации различных статистических и машинно-обучающих моделей. Это позволяет быстро находить закономерности в данных и делать прогнозы. Также, несомненный ключевой аспект заключается в его графическом интерфейсе, который позволяет пользователям создавать и управлять рабочими процессами обработки данных без необходимости написания кода. Это делает его доступным для специалистов без глубоких знаний программирования.

Самым первым шагом было визуализация входных данных, где загружая готовый датафрейм, можно выделить параметр который надо спрогнозировать (рисунок 3.1). В данном случае, прогнозируемым критерием являлась сама «поломка» коммутатора (Breaking).

Далее была использована функция быстрого прототипирования: Как и многие другие инструменты для работы с данными, RapidMiner создает и тестирует различные прототипы использующих все доступные в библиотеке алгоритмы МО на основе загруженного датафрейма. В данном случае, было протестировано 9 видов алгоритмов в режиме «быстрого прототипирования»:

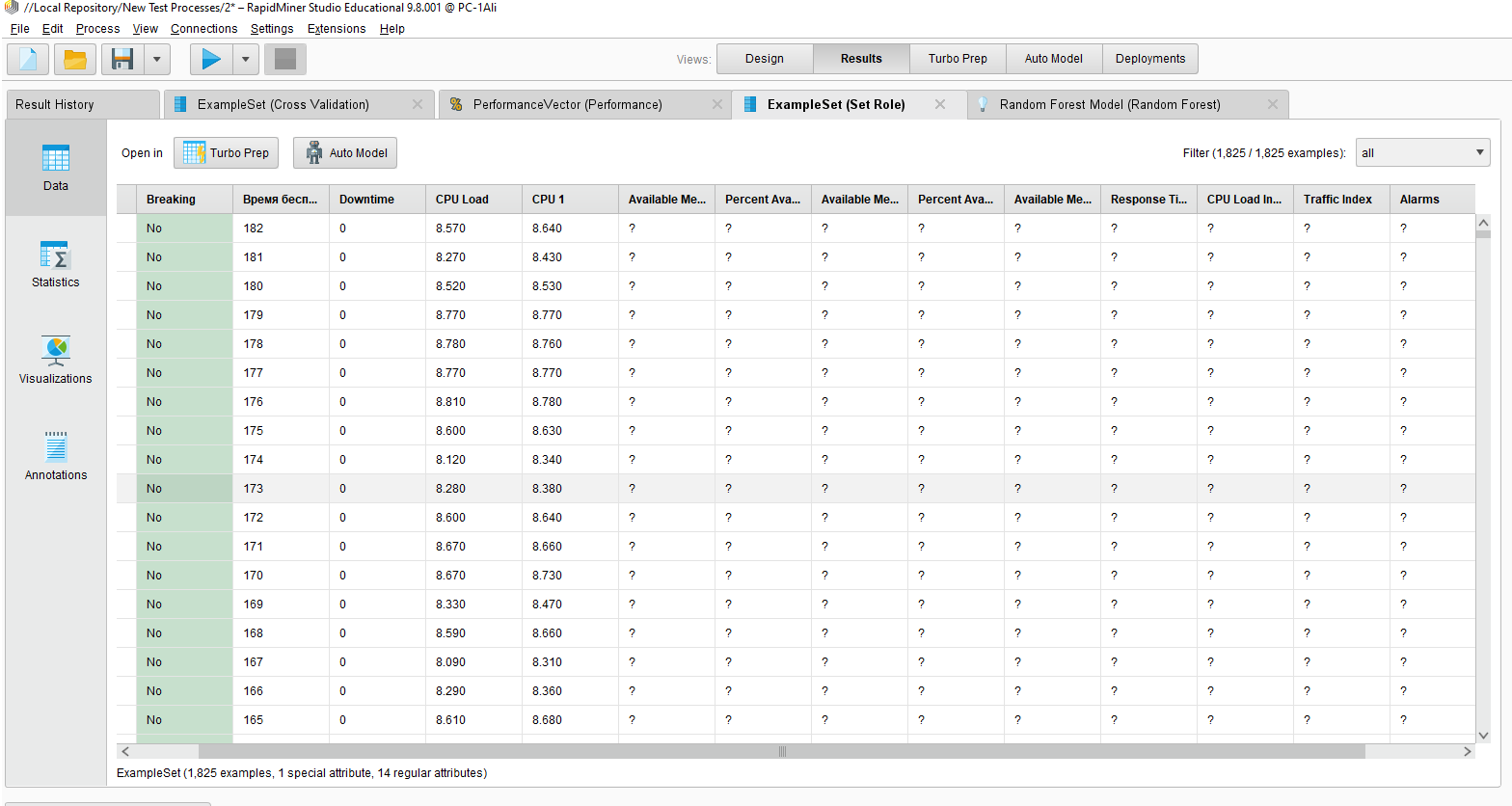


Рисунок 3.1 – Визуализация данных, загруженных в ПО

1. Логистическая регрессия (LR): статистическая модель, используемая для прогнозирования вероятности наступления события путём подгонки данных к логистической кривой.
2. Случайный лес (RF): ансамблевый метод машинного обучения, использующий множество деревьев решений для выполнения задач классификации и регрессии.
3. Градиентный бустинг деревьев (GBT): техника машинного обучения для задач регрессии и классификации, которая строит модель в форме ансамбля слабых предсказательных моделей, обычно деревьев решений.
4. SVM (Метод опорных векторов): модель машинного обучения, используемая для классификации и регрессии, которая ищет гиперплоскость в многомерном пространстве, разделяющую различные классы.
5. kNN (метод k-ближайших соседей): алгоритм машинного обучения, используемый для классификации и регрессии, который предсказывает значение объекта на основе значений его k ближайших соседей.
6. Линейная регрессия (LinR): статистический метод, используемый для моделирования взаимосвязи между зависимой переменной и одной или несколькими независимыми переменными.
7. Нейронная сеть (NN): компьютерная модель, вдохновлённая структурой и функциями биологических нейронных сетей, используемая для решения разнообразных задач машинного обучения.

По итогу прототипирования был получен результат наибольшей применимости алгоритма по отношению к входным данным. Самым положительным предварительным результатом были алгоритмы деревьев решений, случайных деревьев, SVM, логистической регрессии и kNN. Они показывали более 60% точности предварительного предсказывания поломок и меньшее время обучения по сравнению с другими видами алгоритмов.

Алгоритмы, о которых идёт речь, ограничены определенным набором гиперпараметров, требующих настройки, обычно необходимой для эмпирической коррекции. Поскольку они все принадлежат одной и той же среде, можно провести различные оценки относительно времени обучения или использования памяти. Также имеется возможность гибко использовать единый формат набора данных для всех алгоритмов. По итогам прототипирования, как показано на рисунке 3.2, было решено протестировать более основательно, в том же ПО rapidminer studio алгоритм случайных деревьев и алгоритм деревьев решений с перекрёстной проверкой.

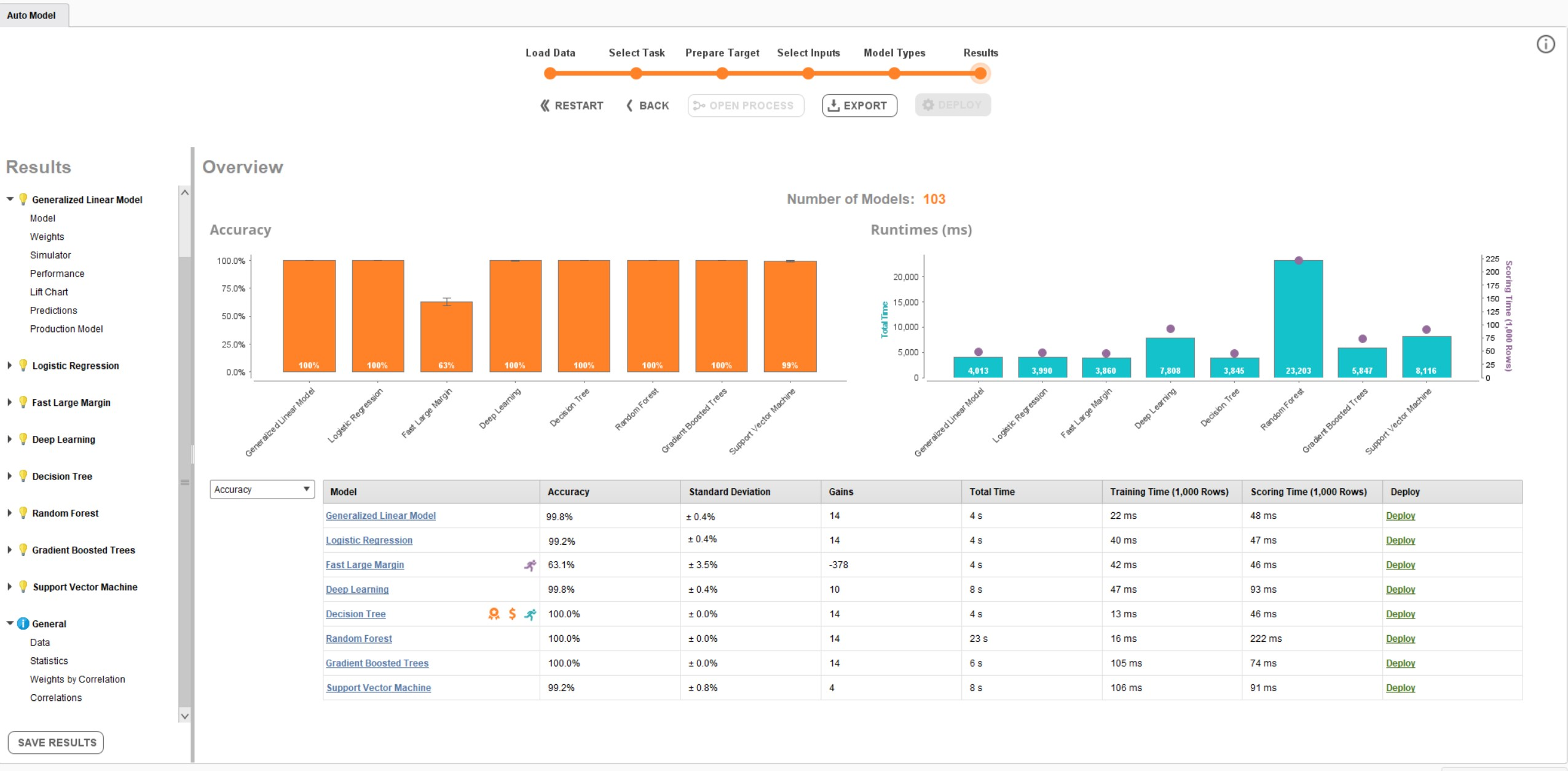


Рисунок 3.2 – Результаты предварительного моделирования

Следующим шагом после предварительного моделирования, результаты которого показаны на рисунке 3.2 стало построение соответствующей модели предсказания поломок, показанная на рисунке 3.3. Первоначально была построена модель, в которую был загружен входной набор данных, состоящий примерно из 1800 записей. Впоследствии были выбраны атрибуты, назначены роли и проведена перекрёстная проверка. Соотношение тестовых блоков к обучающим составляло 40 на 60; т. е. модель была обучена на 60% данных и делала прогнозы по оставшимся 40%. Для обеспечения надёжности оценки модели было использовано 5-кратная перекрёстная проверка, хотя можно было увеличить и до 15-20, однако некоторые события имеют уменьшенное количество положительных образцов, и это может привести к тому, что некоторые фолды (блоки) будут иметь мало или вообще не иметь положительных образцов, делая невозможным расчёт некоторых метрик. Схема работы модели приведён на рисунке 3.3. и 3.4

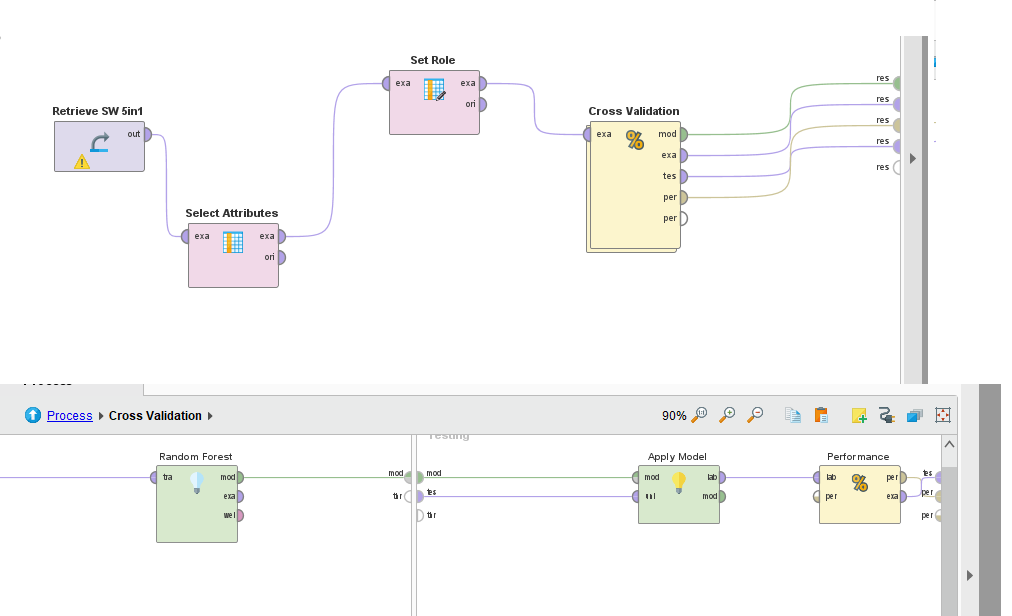


Рисунок 3.3 – Схема модели прогнозирования сбоев, построенная в ПО Rapidminer

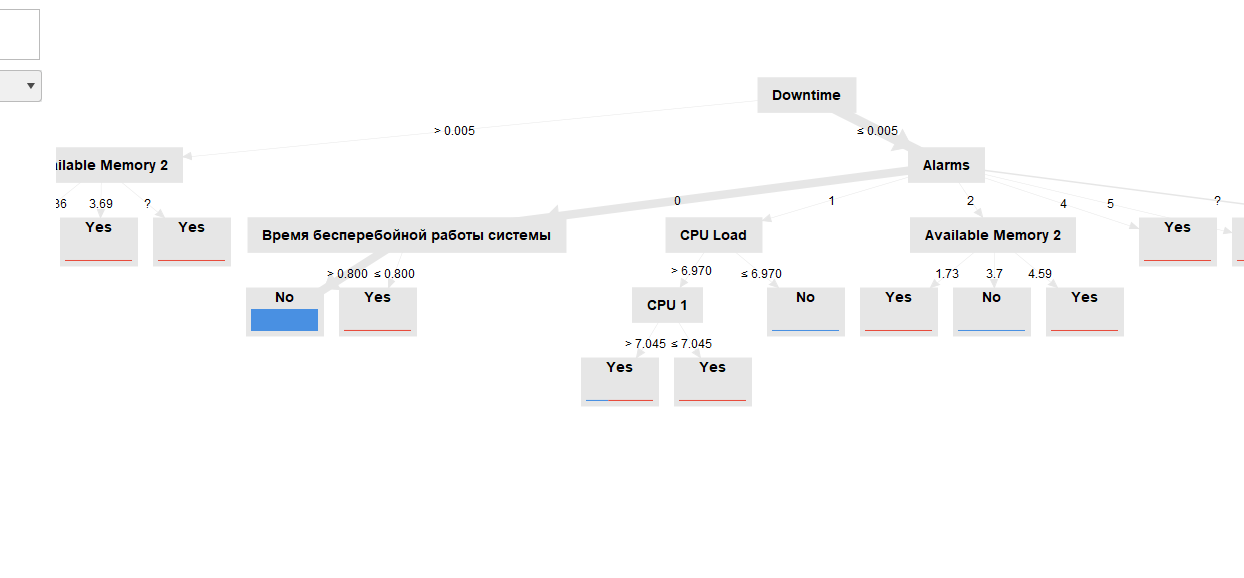


Рисунок 3.4 – Один из множества построенных деревьев метода

Результатом этой проверки модели стали показатели *accuracy* в 99,84%, однако показатели полноты предсказаний были на отметке в 72,73%. Такой высокий процент точности прогнозов был показателем что выбранная стратегия была верной, но требовала некоторых доработок, так как построенная система спрогнозировала 3 ложно отрицательных сигнала (рисунок 3.5), что является недопустимым в данном сценарии.

В связи с вышенаписанным, были произведены следующие изменения в модели для улучшения показателей на выходе: добавлены данные из ещё трёх коммутаторов, увеличив общее количество входных данных до 2900 записей, была увеличена кратность перекрёстных проверок с 5 до 20 но оставив неизменным соотношение тренируемых и прогнозируемых блоков 60/40.

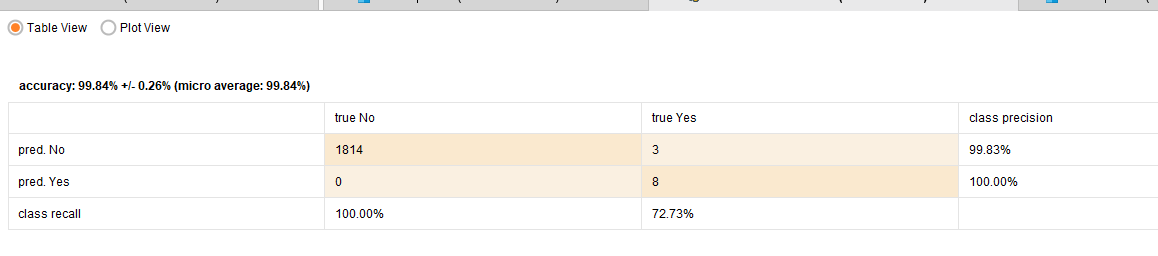


Рисунок 3.5 – Показатели модели

Результатом произведённых изменений был качественный рост конечных прогнозов, как показано на рисунках 3.6 и 3.8. Показатель критерия точности (accuracy) вырос до 99,97%, а критерий полноты предсказаний поломок до 96%, как показано на рисунке 3.6. Тем не менее, время обучения и предсказывания событий увеличилось кратно, с 4 секунд до около 12 секунд в среднем, как показано на рисунке 3.7. В дальнейшем, при построении алгоритмов и тестировании в личном ноутбуке и на рабочем ПК, в условиях отличных от имитации в ПО, временные показатели обучения и прогнозирования будут существенно отличаться. Подробнее про это будет написано далее в следующих главах.

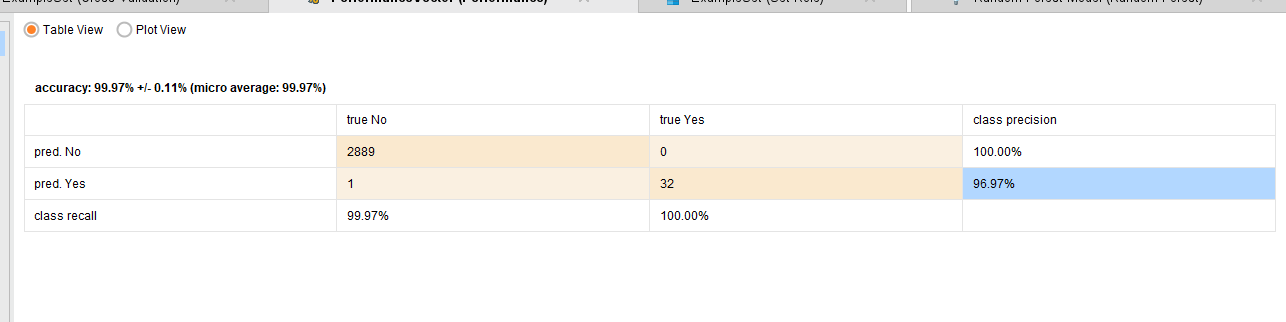


Рисунок 3.6 – Показатели улучшенной модели

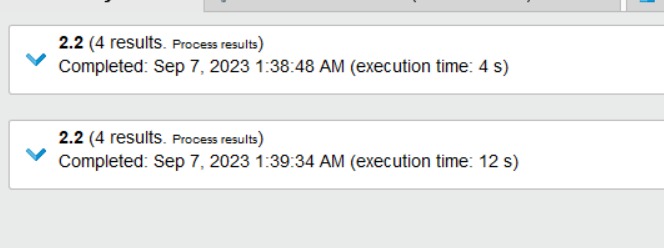


Рисунок 3.7 – Время, затраченное на модели 1 и 2

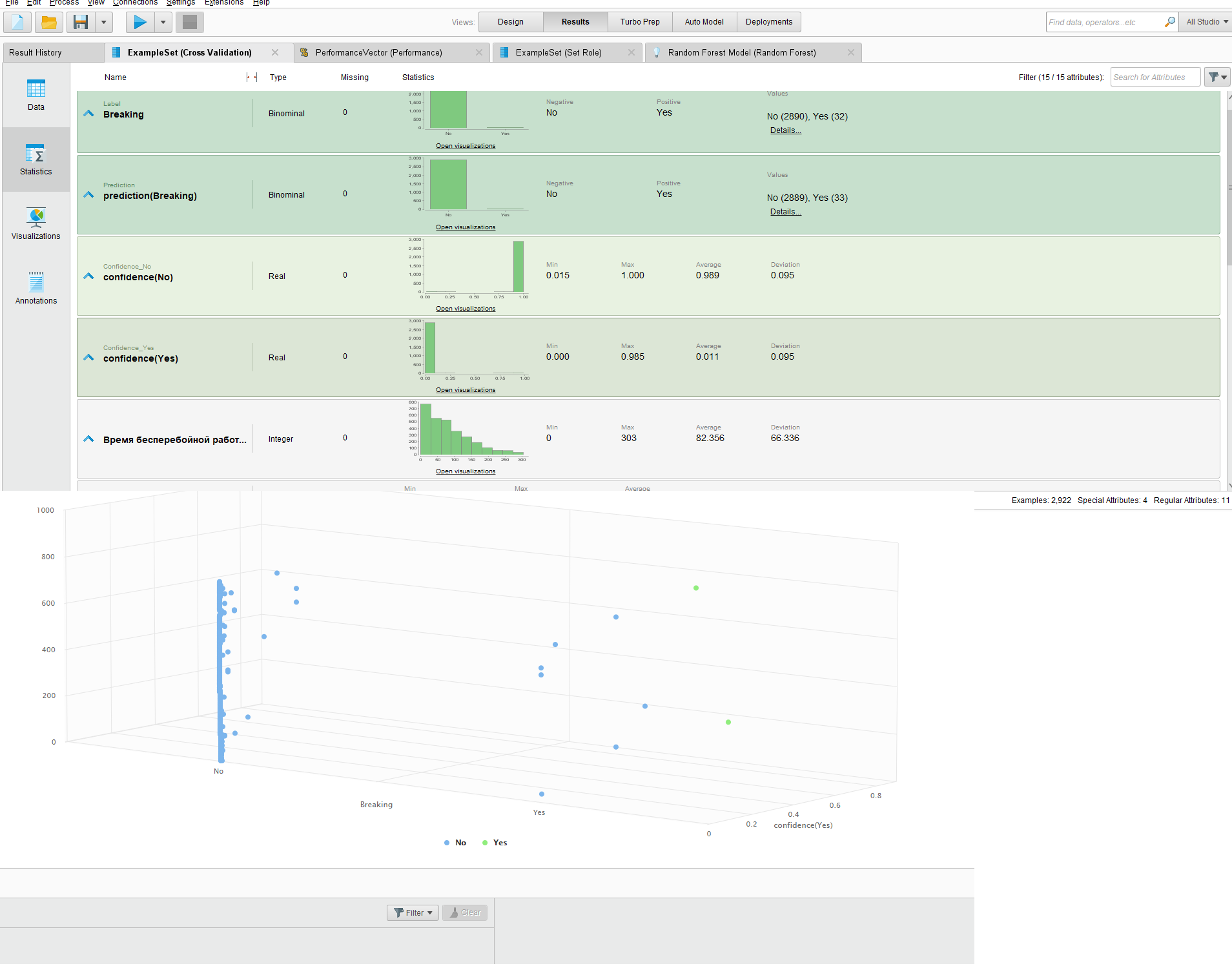


Рисунок 3.8 – Визуализация результатов модели в виде статистики и графов

Как итог, хочется отметить, что были проанализированы ещё 3 вида алгоритмов, таких как деревья решений, kNN, и SVM. Их моделирование показали следующие результаты:

1. Алгоритм на базе деревьев решений показал схожие результаты с методом случайных деревьев, за тем исключением что на обучение уходило в среднем на 10% меньше времени.
2. Модель kNN была трудно применима из-за множества признаков на входе(14 видов), и требовалось проделать огромную работу по масштабированию признаков. Как итог, время, которое затрачивалась на обучение было больше, более 27 секунд в среднем, а результаты менее точными, на уровне 78%.
3. Модель на основе алгоритма SVM была более успешна, чем модель на основе kNN, но обучалась быстрее, чем модель на основе деревьев решений и случайных деревьев. Но при этом, данная модель не требовала масштабирования признаков, и могла считывать те же данные что и модели на основе алгоритмов случайных деревьев и деревьев решений. Подводя итог, можно сказать, что модель kNN была ориентирована на показатели других, соседствующих коммутаторов, что в итоге смазывало результат прогнозов, так как требовался прогноз сбоя тестируемого коммутатора на основе его собственных показателей, а не на основе показателей других, соседствующих коммутаторов.

Стоит заметить, что выбор оптимального алгоритма зависит от конкретных бизнес-требований и контекста применения. В данном случае анализ показывает важность настройки параметров и предварительной обработки данных для достижения желаемых результатов. В случае с ЛВС-отказами, высокая полнота может быть критическим показателем, чтобы не упустить важные отказы. Это подчеркивает важность оптимизации не только точности, но и других метрик.

В рамках предподготовки данных для дальнейшего применения в моделях прогнозирования, *корреляционный анализ* играет ключевую роль в оценке и интерпретации взаимосвязей между различными переменными. Этот метод статистического анализа позволяет оценить, как одна переменная изменяется в ответ на изменение другой, и выразить эту зависимость численно.

Прежде всего, необходимо определить тип корреляции, который будет использоваться: линейный (Пирсон (3.7)), ранговый (Спирмен (3.8), Кендалл (3.9)) или другие. Выбор зависит от характера данных и конкретных задач исследования. Корреляционный анализ может быть проведён как для всех пар переменных, так и для конкретных пар, которые предположительно могут быть взаимосвязаны.

Также важно учитывать мультиколлинеарность: когда две или более независимых переменных тесно связаны между собой, это может вызвать проблемы мультиколлинеарности в регрессионных моделях. В таких ситуациях одну из коррелирующих переменных обычно исключают из модели. Кроме того, необходимо провести предварительную обработку данных: перед выполнением корреляционного анализа данные могут быть нормализованы или стандартизированы, особенно если переменные измеряются в разных единицах.

Как говорилось выше, есть несколько типов корреляции. Их формулы (3.7), (3.8), (3.9) приведены ниже:

(3.7)

где среднее значение переменных *x* и *y*;

*n –* количество наблюдений.

(3.8)

где *di*​ – разница между рангами каждой пары наблюдений.

(3.9)

где *C* – количество согласованных пар;

*D* – количество несогласованных пар;

*T* и *U* – количество связей только в *X* или *Y*, соответственно.

В данной системе прогнозного моделирования было решено использовать коэффициент корреляции Пирсона в качестве основного метода корреляционного анализа. Это решение основано на характере этих данных и конкретных требованиях данного исследования. Метод Пирсона особенно хорошо подходит для наборов данных, в которых связь между переменными предполагается линейной. Учитывая, что данное исследование включает переменные в сетевой среде, где распространены линейные отношения, корреляция Пирсона обеспечивает эффективное средство количественной оценки силы и направления этих отношений. Пример применения, когда n=5 параметра x1- temperature, приведён на (3.10):

где = 37.53, 0.011, соответственно,

(3.10)

Аналогичным образом рассчитывается корреляция для всех параметров, и получается корреляционная карта, показанная на рисунке 3.9.

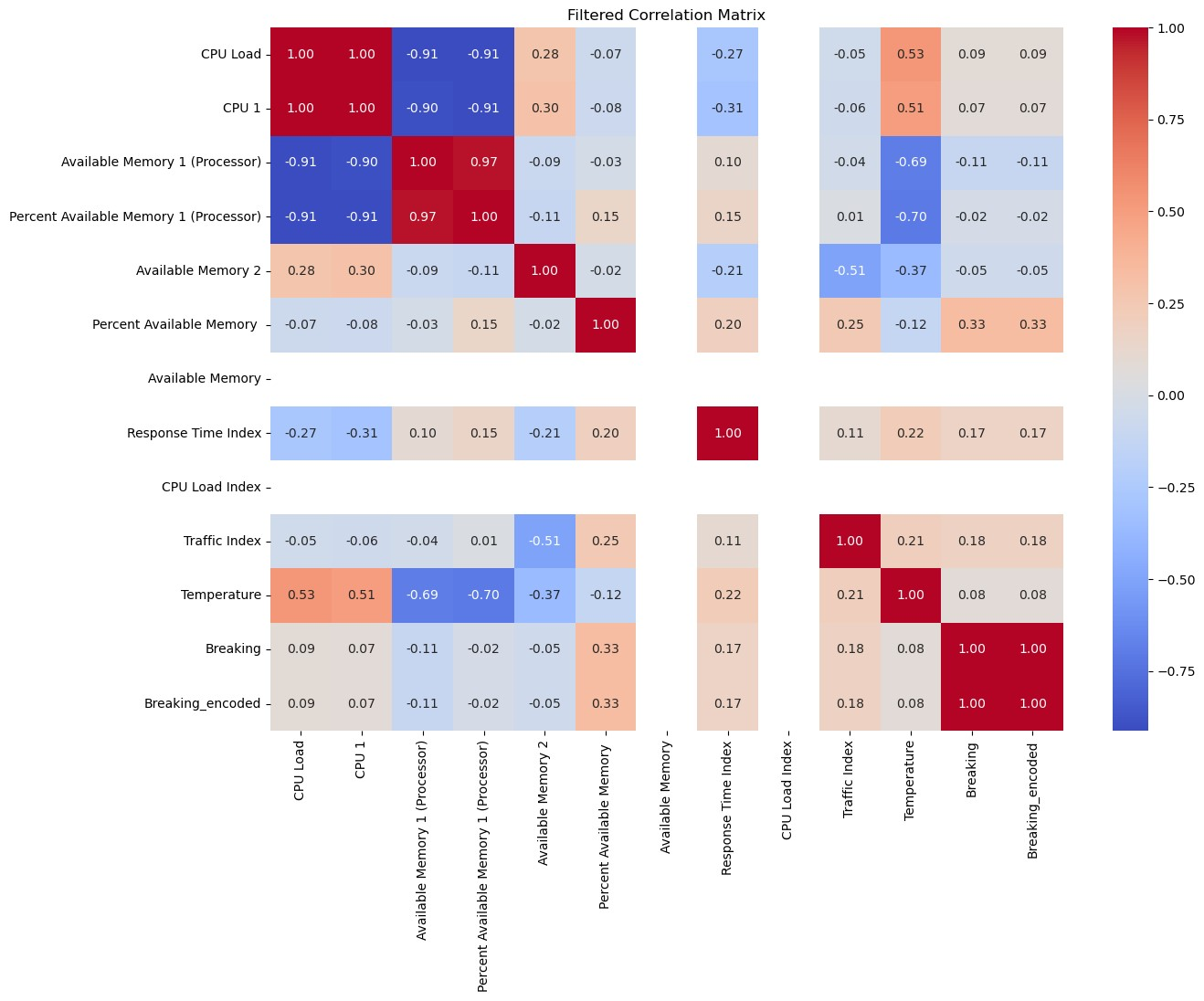


Рисунок 3.9 – Матрица корреляции данных

**3.4 Разработка рабочей модели на основе алгоритмов машинного обучения**

Универсальная методология была разработана в данной главе для получения моделей прогнозирования событий по сценариям, описанных в главе 2. Эта методология разработана обобщённым образом и способна адаптироваться к различным ЛВС, в том числе и с мультивендорным оснащением. Примечательной особенностью является то, что это не просто теоретическая последовательность шагов, но её можно с определенной долей успеха применить на практике. В случае этого исследования, алгоритмы были протестированы на реальном наборе данных, полученном из производственной среды компании. Конечно, утверждать, что разработанные алгоритмы и их результаты представленные здесь не являются идеальными, и некоторые из шагов могут показаться не очень эффективными, но необходимо подчеркнуть, что в условиях исследуемого объекта все они были необходимыми. Другое отличие в работе с реальными данными, а не с теоретическим процессом, заключается в том, что предложения должны быть осуществимыми, и необходимо принимать решения относительно компромиссов между результатами и трудностью их получения. Например, ранее упомянутая техника валидации моделей в ПО Rapidminer представляет собой кросс-валидацию с K-блоками. В этом типе валидации чем больше разбиений сделано, тем более надёжной является оценка модели. Однако важно учитывать, с какими данными вы работаете; в моделированном сценарии есть события с всего около 10-15 положительными образцами, это означает, что, хотя теоретически использование 20 разбиений лучше, чем использование всего 5, как это было протестировано, для этих редких событий это будет означать, что некоторые из сделанных разбиений возможно не будут иметь положительных образцов, и, следовательно, оценка не будет чётко определена.

Методология состоит из пяти последовательных шагов. Каждый из этих шагов включает в себя решение одной из обычных проблем в процессе машинного обучения (например, выбор алгоритма или уменьшение переменных), учитывая характеристики проблемы. Изначально все события пытаются смоделировать одинаково, но каждый шаг методологии включает в себя решение о конфигурации модели, которое влияет на последующие шаги и заканчивается индивидуализированным моделированием для каждого типа события. Принятые на каждом из шагов решения могут быть автоматизированы на основе заданной конфигурации, что позволяет собрать все шаги в один процесс. Тот факт, что методология обобщённо адаптируется к набору данных и может быть автоматизирована, отвечает одной из поставленных в главе 1 задач: это должен быть осуществимый процесс с максимальной эффективностью для человека принимающий решения, то есть, система помощи принятия решений.

Последовательность шагов методологии показана на рисунке 3.10, а краткое описание каждого из них приведено ниже:

Шаг 1. Предварительная подготовка и оценка данных: Установить конкретные метрики и методы валидации для оценки алгоритмов с равными параметрами. Процесс предварительного моделирования.

Шаг 2. Корреляционный анализ: На этом этапе оптимизируются параметры, сокращаются количество параметров согласно корреляционному анализу.

Шаг 3. Разработка предиктивной модели на базе DES+LR/RF/DT/SVM: Процесс составления математической модели выбранных алгоритмов с количеством параметров. Для каждого датафрейма ищутся наилучшие алгоритм и комбинация гиперпараметров.

Шаг 4. Процесс оптимизации принятия решений внеплановых ремонтных работ ЛВС. На этом этапе определяется процесс или его часть, которая будет подвержена оптимизации через применение системы прогнозирования.

Шаг 5. Практическая реализация и выборка: На языках прикладного программирования реализуется исполнение алгоритма. Для оценки истинной производительности модели в несбалансированном наборе применяются техники, которые изменяют обучающий набор, чтобы учитывать количество положительных и отрицательных образцов.

Предыдущие разделы рассказывали подробно про шаги 1 и 2. Следующие разделы этой главы подробно расскажут о том, как работает каждый из дальнейших шагов, и о результатах, которые получились, по итогам тестирования на данных с производства.

Рисунок 3.10 – Общая методология

**3.5 Математическая модель алгоритмов предсказывания сбоев на основе методов машинного обучения**

Датасет в общем виде представлен в таблице 3.2.

Таблица 3.2 – Датасет в общем виде

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | Y |
| CPU Load | CPU 1 | Available Memory 1 (Processor) | Available Memory 2 | Percent Available Memory | Response Time Index | Traffic Index | Alarms | Temperature | Breaking |
| 9,31 | 9,38 | 37,00 | 3,73 | 31,00 | 37,00 | 21,00 | 1,00 | 42,00 | 1 |
| 9,03 | 9,10 | 37,00 | 3,73 | 31,00 | 89,00 | 18,00 | 0,00 | 41,00 | 0 |
| 9,32 | 9,41 | 37,00 | 3,73 | 31,00 | 51,00 | 18,00 | 1,00 | 41,00 | 0 |
| 9,25 | 9,37 | 37,00 | 3,73 | 31,00 | 89,00 | 20,00 | 1,00 | 56,00 | 1 |
| 9,01 | 9,03 | 37,00 | 3,73 | 31,00 | 51,00 | 18,00 | 0,00 | 41,00 | 0 |

Датасет в общем виде выглядит следующим образом:

(3.11)

Loss function (с помощью этой функции происходит обучение)

(3.12)

где это значение *y,* которое рассчитывает модель предсказывания поломок.

*Модель линейной и логистической регрессии для данного случая:*

(3.13)

Допустим, что x0=1, тогда вектор признаков *x=(1, x1, x2,…, x9)*

Тогда, в общем виде, модель линейной регрессии будет выглядеть следующим образом:

(3.14)

В данном случае, выборка состоящая из более чем 2500 строк определяется как *l=2500* пар *(xi, yi)* где

(3.15)

Для данного случая, для обучения *l=*80% от *n l=2000* строк запишем, что требуется получить в виде скалярного производного: X =y. В развёрнутом виде это выглядит следующим образом:

(3.16)

Далее, для нахождения оптимальных весовых коэффициентов вектора , логичнее всего воспользоваться функцией потерь (Loss function).

(3.17)

Запишем её в векторном виде:

(3.18)

Эта векторная форма является удобной для анализа и вычислений, поскольку она позволяет работать с большими наборами данных эффективно, используя операции линейной алгебры. Далее берётся производная этой формулы и упрощается, сгруппировав члены до следующего вида:

(3.19)

Это и есть градиент функции потерь, который используется для оптимизации весов ω в методах, таких как линейная регрессия.

Так как матрицы *X* – линейно независимые столбцы, то можно приравнять производную к нулю и получим:

(3.20)

где – это обратная матрица . Однако это решение является очень сложным, так как данный датасет состоит из множества параметров и больших объёмов данных.

Поскольку решается задача классификации, надо преобразовать эти функции: , то есть *y* – это вектор из 0 и 1.

(3.21)

Далее постараемся вывести функцию потерь из теоремы статистики: для этого введём вероятности в модель. X – случайная величина вектора признаков; y – случайная величина целевых перемен. Соответственно, функция распределения вероятностей примет вид:

(3.22)

Вероятность того, что случится поломка при заданных значениях, где данная модель – , функция распределения примет вид: . Правдоподобие (likelihood) для этой модели в общем виде будет представлено как произведение вероятностей для каждого наблюдения из выборки: . Теорема из статистики гарантирует что если найти параметры модели, которые максимизирует правдоподобие, то они будут хорошими:

(3.23)

Далее берётся логарифмический вид и разложив в сумму логарифмов от производных, получается Loss Function:

(3.24)

Исходя из этого, можно увидеть, что функция потерь – это логарифмическая вероятность правильного класса.

Задача максимального правдоподобия и задача минимизации функции потерь – это одно и то же определение.

Определение модели логистической регрессии для текущих параметров – это преобразованная формула линейной регрессии с применением сигмоидальной функции:

(3.25)

В данном случае, *n=*9*.* Как известно, . А это значит, что формула (3.25) преобразуется в:

(3.26)

Формула (3.27) является логарифмом функции правдоподобия (log-likelihood) для бинарной классификационной задачи в контексте модели логистической регрессии.

(3.27)

Если подставить в формулу (3.24) функцию правдоподобия для бинарной классификационной задачи в контексте модели логистической регрессии (3.27), то получается функция потери LogLoss LogRegress.

(3.28)

Если ввести в эту формулу логита и использовав формулу функции *softmax(zi)=* , то в итоге получается

(3.29)

Переход к функции softmax предполагает, что в дальнейших исследованиях будет открыта возможность на расширение рассматриваемой модели для прогнозирования различных типов сбоев или статусов, а не только двоичного исхода.

*Математическая модель для случая с алгоритмом случайных деревьев решений:*

Для построения модели случайного леса (Random Forest) на основе набора данных, показанных на таблице 3.2 и задачи классификации с использованием энтропии в качестве критерия, можно определить следующие математические и методологические элементы.

Основные определения и параметры: датасет в общем виде имеет тот же вид как показано в 3.11, а функция потерь в контексте случайного леса L обычно не минимизируется явно, но для оценки модели может использоваться средняя энтропия по всем деревьям:

(3.30)

где Т – это количество деревьев в лесу, Т=100;

*H* – энтропия для узла с заданным набором данных, который используется для задачи классификации, и он определяется как:

(3.31)

гдеь это вероятности положительного и отрицательного классов в данном узле.

*Формирование Дерева:*

Каждое дерево *Ti*​ строится на основе бутстрапированной выборки из исходных данных. При каждом разделении узла:

1. Выбирается подмножество признаков.
2. Из этого подмножества выбирается признак, который максимизирует уменьшение энтропии (или эквивалентно, увеличивает прирост информации).

*Предсказание:*

Для нового наблюдения *x*, каждое дерево *Ti*​ в лесу делает предсказание *yi*​. Глобальное предсказание делается путём голосования между всеми деревьями:

(3.32)

Таким образом, модель случайного леса f(x) является ансамблем деревьев решений:

(3.33)

*Математическая модель для случая с алгоритмом деревьев решений:*

Дляразбиения узла дерева в случае использования энтропии в качестве функции потерь применяется следующая формула (3.34):

(3.34)

*Entropy(S)* часто обозначает энтропию узла *S*, который может быть как родительским, так и дочерним узлом, в зависимости от контекста. *Entropy(S)* определяется по формуле (3.31). При этом *N* – это общее число образцов в родительском узле, а *Ni* – число образцов в дочернем узле *i*.

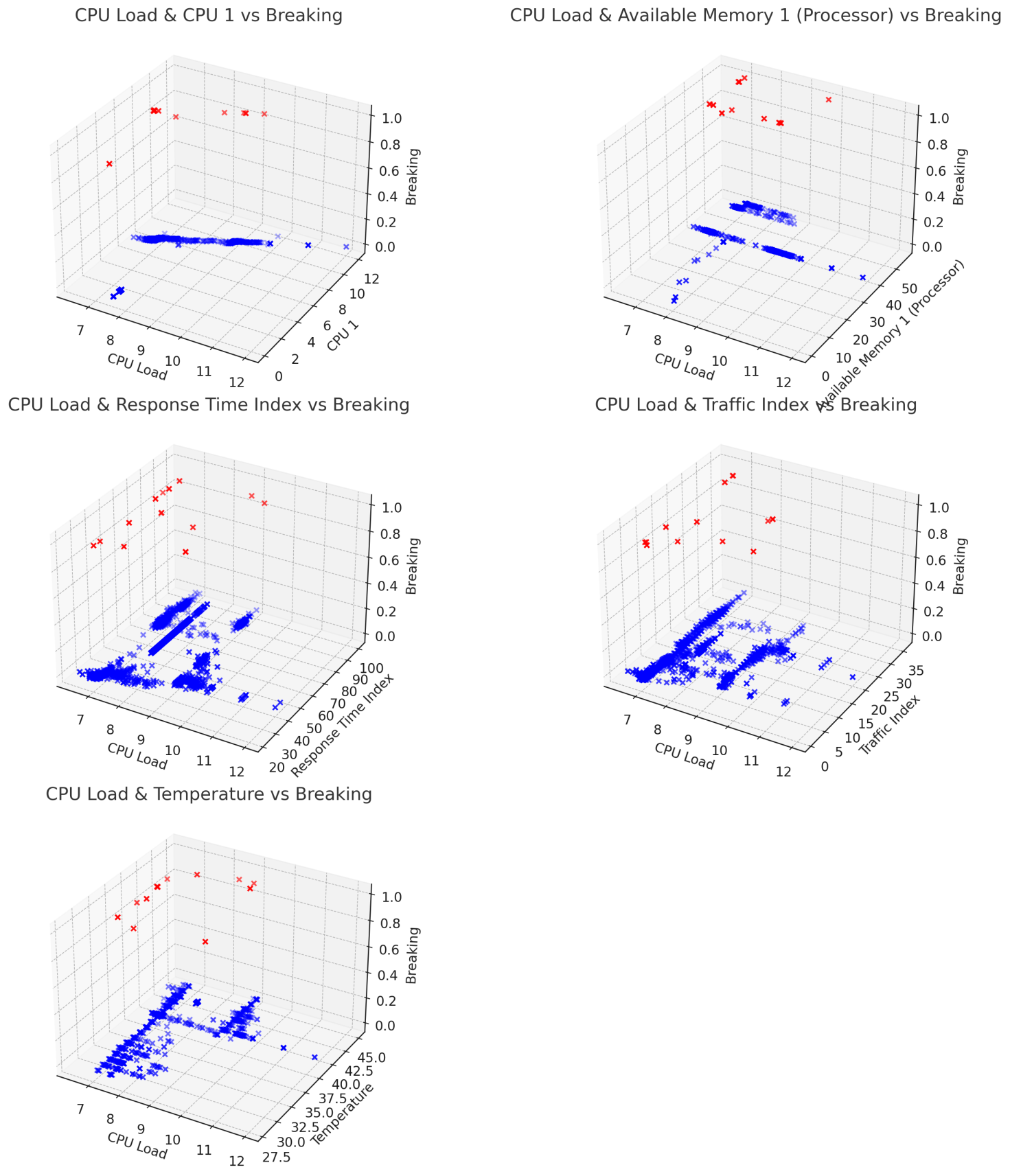
Обученное дерево решений *f*(*x*) представляет собой рекурсивную функцию, которая на каждом уровне *d*, 0≤*d*<*D*, где *D* – максимальная глубина дерева, в данном случае это глубина в 3 или 4 веток (лепестка).

(3.25)

где *xi* – это признак, на основе которого происходит разбиение;

*t* – пороговое значение для *xi*, и – рекурсивные функции для левого и правого дочерних узлов.

*Математическая модель для случая с алгоритмом использующий метод опорных векторов:*



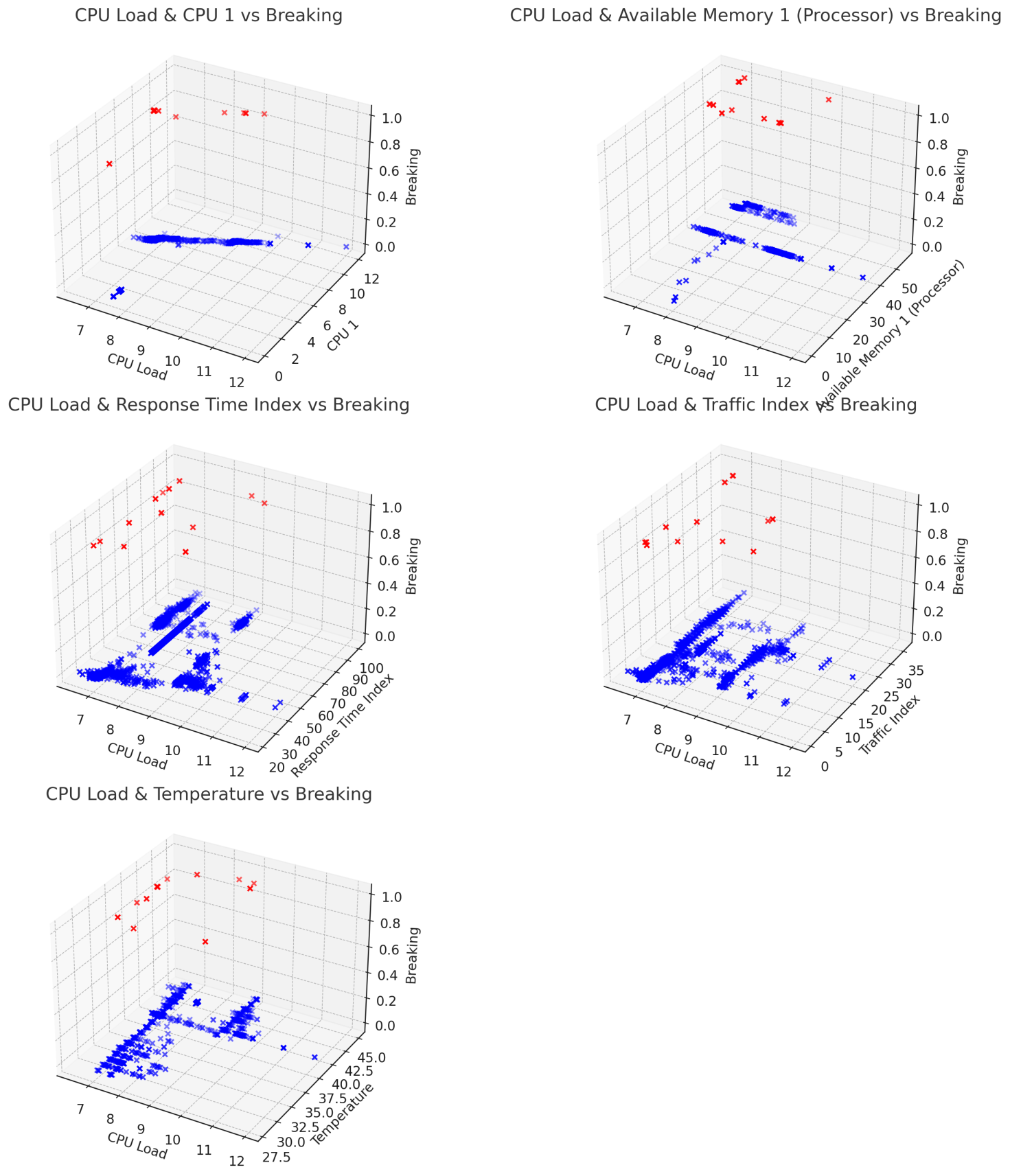
а б

Рисунок 3.11 – Комбинация функций модели SVM и графики целевых переменных, лист 1

Датасет в виде таблицы 3.2 и в общем виде как показано на рисунке 3.11 остается неизменным. Алгоритм машины опорных векторов (SVM), надежный инструмент двоичной классификации, чтобы различать рабочие состояния и состояния отказа коммутаторов локальной сети. SVM работает путем определения критических точек данных, известных как векторы поддержки, из данного набора обучающих данных. Эти векторы играют важную роль в определении границы принятия решений, которая эффективно разделяет набор данных на две отдельные категории: случаи сбоев оборудования и нормальные эксплуатационные данные.

Уникальная сила SVM заключается в его способности определять наиболее оптимальную границу решения, известную как гиперплоскость. Эта гиперплоскость характеризуется максимальным запасом между двумя классами, что обеспечивает максимально четкое и отчетливое разделение. Целью SVM является обнаружение этой гиперплоскости, тем самым максимизируя точность классификации. (рисунок 3.11). Математическая формулировка этой решающей функции в SVM определяется следующим образом:

(3.26)



в

Рисунок 3.11, лист 2

где вектор «*w»* и скаляр «*b»* представляют собой неопределённые коэффициенты гиперплоскости. Входной параметр «*x»* представляет элементы данных, называемые вектором признаков. Предположим, что в наборе данных имеется N векторов признаков, выраженных как x1, x2,…, xN. Чтобы получить w и b, SVM сначала находит точки, ближайшие к гиперплоскости (называемые опорными векторами, а затем проверяет, что они находятся дальше всего от гиперплоскости (т.е. граница равна максимум). Компоненты векторов признаков указывают позиции во входном пространстве, а это означает, что их группа принадлежности зависит от этих компонентов. Предположим, что отказ сетевого оборудования тесно связан с его состоянием, которое можно количественно оценить с помощью показателей, собранных во время работы оборудования системой мониторинга. Индикаторы здесь представляют физические параметры оборудования, такие как загруженность процессора, загруженность памяти, время отклика, внутренняя температура сетевого оборудования, влажность воздуха, индекс трафика на канале связи, и другие параметры, которые система мониторинга записывает в ежедневных операциях. Как правило, эти данные будут записываться системой в журналы памяти. Итак, при прогнозировании отказов сетевого оборудования используется значения показателей в качестве компонентов для составления вектора признаков. Каждому вектору признаков xn (n = 1, 2, …, N) соответствует метка в зависимости от его группы (отказ/поломка или нормальная). Данные с меткой ln = + 1 будут удовлетворять у(x)>0, а данные с меткой ln = -1 будут удовлетворять у(x)<0. Таким образом, расстояние по перпендикуляру от xn до гиперплоскости определяется выражением |**w**T·**x**n + b|/||**w**|| = ln·(**w**T·**x**n + b)/||**w**||. Целью алгоритма SVM является получение значений **w** и **b**, которые максимизируют расстояние, описываемое уравнением (3.27).

(3.27)

Решение этой задачи оптимизации предполагает ее преобразование в более удобную форму. Для устранения ограничений в процессе оптимизации используется метод множителей Лагранжа. Двойственная задача, полученная с использованием множителей Лагранжа, позволяет нам найти оптимальные значения «w» и «b», которые максимизируют разницу между классами. Тогда задача оптимизации выражается следующим образом:

(3.28а)

На оптимизацию накладываются ограничения, обеспечивающие неотрицательность множителей Лагранжа и выполнение условий Каруша-Куна-Таккера:

(3.28b)

(3.28c)

Решение этой двойственной формулировки позволяет нам определить оптимальные значения w и b, которые являются необходимыми для определения границы решения SVM. Вектор весов w получается, как линейная комбинация точек обучающих данных, взвешенных их соответствующими множителями Лагранжа:

(3.29a)

Затем смещение b определяется на основе опорных векторов:

(3.29b)

На данном этапе эти векторы признаков называются обучающими данными, а вычислительный процесс известен как процесс обучения. Как видно, выбор обучающих данных напрямую влияет на решение для w и b, а затем влияет на точность классификации. Для диагностики отказов оборудования с высокой точностью необходимо достаточное количество данных об отказах и нормальной работе.

На практике группы данных могут перекрываться, что приводит к переобучению и плохой обобщающей способности, если точно разделяются данные. В этом случае SVM использует переменные расслабления, позволяющие допустить ошибочную классификацию обучающих данных. Коэффициент ограничения «C» используется для контроля баланса между точностью и обобщающей способностью гиперплоскости. С введением ограничения "C" формула (3.28b) модифицируется как C ≥ ai ≥ 0, i = 1, …, N.

Вышеизложенное обсуждение основывается на предположении, что группы данных линейно разделимы; однако на практике это часто не так. Для решения нелинейной задачи вводится функция ядра, которая позволяет отображать сложные, линейно неразделимые данные из пространства признаков низкой размерности в пространство признаков более высокой размерности, где данные линейно разделимы. После этого преобразования можно использовать решение линейной задачи для решения нелинейной задачи в пространстве более высокой размерности.

Мы определяем данное входное пространство как "X" и преобразованное пространство признаков как "H". Функция ядра K(x,z) удовлетворяющая условию K(x,z)=ϕ(x)⋅ϕ(z), облегчает это преобразование. В данной двойственной формулировке задачи SVM заменяется скалярными произведениями в уравнениях на функции ядра, что приводит к новой оптимизационной задаче:

(3.30a)

(3.30b)

(3.30c)

Уравнение (3.29b) заменяется на:

(3.31)

Эта функция преобразует эти входные данные в пространство признаков более высокой размерности, где возможно линейное разделение. Внутреннее произведение в данном исходном пространстве таким образом преобразуется во внутреннее произведение в этом преобразованном пространстве признаков, согласно уравнениям (3.28а)–(3.28с). Определив подходящую функцию ядра можно эффективно решать задачи нелинейной классификации, используя уравнения (3.30a)–(3.30c).

Это преобразование с помощью ядра позволяет SVM эффективно справляться с задачами бинарной классификации, такими как определение рабочего состояния сетевого оборудования. В этом контексте SVM служит инструментом для интеллектуального распознавания образов, идентифицируя потенциальные отказы на основе операционных данных.

* 1. **Комбинированный метод DES+LR/RF/DT/SVM**

Алгоритм SVM, как и другие рассмотренные выше алгоритмы МО являются эффективным средством для использования показателей при оценке состояния оборудования, но они не способны выполнять функции прогнозирования самостоятельно. Как демонстрируется на рисунке 3.12, исторические данные собираются за период с t-n до t-1, при этом показатели на момент времени t устанавливаются на основе этих данных. Однако это не может считаться прогнозированием, поскольку состояние оборудования в момент времени t уже известно. Более целесообразным является применение другого алгоритма для предсказания показателей на будущий момент времени t + T, который затем используется для прогнозирования вероятности отказа оборудования в этот момент.

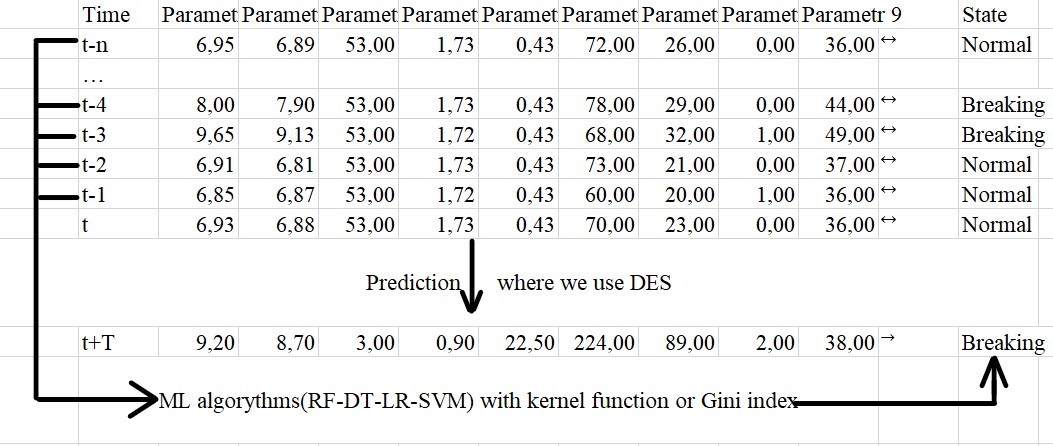


Рисунок 3.12 – Комбинированный метод алгоритмов DES+LR/RF/DT/SVM

Согласно написанному в разделе 1.3, было принято решение использовать метод двойного экспоненциального сглаживания для предсказания показателей параметров на будущий момент времени t + T.

Двойное экспоненциальное сглаживание улучшает традиционное экспоненциальное сглаживание за счёт включения двух ключевых компонентов: уровня и тренда временного ряда. Эта двухкомпонентная структура позволяет DES более чутко адаптироваться к внезапным сдвигам в данных, что является критически важным для точного отслеживания динамики временных рядов с резкими колебаниями.

Применение DES в данных предсказательных моделях особенно полезно в сценариях, где наблюдаются быстрые изменения показателей эффективности сети. Используя DES, можно генерировать более точные и своевременные прогнозы, которые необходимы для проактивного управления сетью и стратегий предотвращения отказов. Алгоритм DES, характеризующийся своей методикой слоистого сглаживания, применяет экспоненциальное сглаживание дважды: сначала к фактическим данным, а затем к сглаженным значениям. Этот процесс описывается в уравнениях:

(3.32a)

(3.32b)

Формула 3.32b представляет второй компонент (компонент тренда) алгоритма Двойного Экспоненциального Сглаживания (DES). Описание этой формулы следующее:

1. – Этот термин представляет оценочный компонент тренда временного ряда на момент времени t. Это результат этой части формулы, указывающий на сглаженную оценку тренда на текущем временном шаге.
2. *a*: Это параметр сглаживания для компонента тренда. Это значение между 0 и 1, которое определяет, сколько веса отдаётся наиболее недавнему наблюдению в временном ряду. Большее значение *a* придаёт больше веса недавним изменениям в тренде, делая алгоритм более чувствительным к новым трендам.
3. Этот термин является оценочным уровневым компонентом временного ряда на момент времени t, полученным из первого уравнения алгоритма DES. Он представляет сглаженную оценку значения серии на текущем временном шаге.
4. (1− *a*): Эта часть формулы придаёт вес предыдущей оценке тренда. Она дополняет параметр сглаживания a так, что сумма весов составляет 1. Это обеспечивает учёт всего диапазона прошлых данных с акцентом на наиболее последний тренд.
5. : Это оценочный компонент тренда с предыдущего временного шага. Он переносит ранее оценённый тренд в текущий расчёт, обеспечивая непрерывность в оценке тренда.

и используются для расчёта прогнозируемого значения на момент времени *t*+*T* согласно уравнениям (3.33a)-(3.33c)

(3.33a)

(3.33b)

(3.33c)

В этом уравнении *a*t рассчитывается с использованием текущей оценки уровня и тренда на момент времени t. Эта формула корректирует уровневый компонент, учитывая текущий тренд.

bt является компонентом тренда на момент времени t. Он рассчитывается как функция разницы между уровневым и трендовым компонентами, масштабируемая сглаживающим параметром *a*. Эта формула фиксирует скорость изменения уровневого компонента, что является индикатором тренда.

Следующим шагом в прогнозировании поломок является классификация спрогнозированных параметров методом DES. Для этого используется метод двойного экспоненциального сглаживания (DES) к каждому параметру, тем самым формируя и обновляя прогностическую кривую в соответствии с последними данными мониторинга. Для этого устанавливается, что , и применяются формулы (3.32a-b) для расчёта и тренда от t=1 до N для каждого параметра. Момент времени t=N отражает самые последние данные мониторинга. Имея значения для каждого параметра, можно предсказать значение признака на момент времени t=N+1 используя уравнения (3.33a-c).

Последним шагом является уже непосредственно сам процесс прогнозирования поломок сетевого оборудования. Ниже, в главе 4 будет описан процесс обучения алгоритмов МО для классификации поломок на основе данных. После обучения алгоритмов МО () для успешной классификации, применяются прогнозируемые значения в этих обученных алгоритмах. Это позволит делать прогноз состояния оборудования () на момент времени t=N+1, что является результатом данного метода прогнозирования.

(3.34)

**Выводы по разделу 3**

В данном разделе подробно рассматривается процесс создание моделей прогнозирования с использованием машинного обучения, включая все этапы проекта по моделированию. Эта методология представляет собой набор последовательных этапов предварительной обработки данных. На каждом из этих этапов было создано все более совершенные модели прогнозирования, заканчивая созданием наилучших моделей в рамках рассматриваемого сценария. Глава начинается с определений и требований, касающихся того, какими будут модели и как они будут оценены. Затем описывается структуры методологии и представляются результаты каждого из применённых этапов. Строится математическая модель для каждого выбранного алгоритма. Далее предлагается новый прогностический метод DES-ML, где DES используется для прогнозирования будущих значений каждого индикатора, а эти прогнозы затем подаются в модели машинного обучения (LR, RF, DT, SVM). Модели ML оценивают вероятность отказа оборудования в будущих временных точках на основе этих прогнозов. Этот синергетический подход объединяет преимущества DES в обработке временных рядов с трендами и мощные возможности классификации алгоритмов машинного обучения, предлагая комплексный инструмент для проактивного обслуживания и прогнозирования отказов в сетевом оборудовании.

1. **ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ ПО ПРИМЕНЕНИЮ DES-ML АЛГОРИТМОВ ПРОГНОЗИРОВАНИЯ НА ДАННЫЕ ИЗ ПРЕДПРИЯТИЯ**

**4.1 Построение и описание обучаемого алгоритма на прикладном уровне. DES-ML**

4.1.1 SVM

Как было написано ранее, по итогам предварительного моделирования нескольких алгоритмов были выбраны, как перспективные, 3-4 алгоритма: алгоритм случайных деревьев, алгоритм деревьев решений, алгоритм логистической регрессии и алгоритм SVM. Каждый из выбранных алгоритмов имел свои преимущества и недостатки.

Рассмотрим разработанный метод на алгоритме SVM. Построение алгоритма начинается с импортирования необходимых билбиотек:

* *pandas:* Используется для работы с данными в виде таблиц. Она обеспечивает структуры данных для эффективного хранения и манипуляции табличными данными;
* *numpy:* Эффективная библиотека для работы с массивами, включая математические, логические и статистические операции;
* *matplotlib и seaborn:* Для визуализации данных, включая построение гистограмм;
* *sklearn:* Обширная библиотека для машинного обучения, содержащая функции для разделения данных, обучения моделей, метрик оценки и многое другое.

Из библиотеки scikit-learn импортируются функции для разделения данных на обучающую и тестовую выборки (*train\_test\_split*), а также для проведения кросс-валидации (*cross\_val\_score*):

* *standardScaler:* инструмент для стандартизации признаков, что часто необходимо для многих алгоритмов машинного обучения;
* *SVM*: классификатор на основе метода опорных векторов (Support Vector Classifier), который является одним из видов машин опорных векторов (SVM);
* *make\_pipeline:* Импортирует утилиту для создания конвейера, что позволяет упростить процесс обучения и применения моделей;
* *accuracy\_score, classification\_report:* Импортирует метрики для оценки качества модели, включая точность (*accuracy\_score*) и полный отчёт по классификации (*classification\_report*), который предоставляет такие метрики, как точность, полнота и F1-мера для каждого класса;
* *if \_\_name\_\_ == '\_\_main\_\_':* Эта конструкция используется для обозначения основного блока кода, который будет выполнен, если скрипт запущен как самостоятельная программа, а не как модуль.

Далее идёт процесс загрузки и анализа данных:

Процесс загрузки входных данных в виде датафрейма Excel-файла инициируется командой *ds = pd.read\_excel(...).*

Затем следует процесс преобразование данных: «*ds['Breaking'] = ds['Breaking'].map({'Yes': 1, 'No': 0})».* Здесь происходит преобразование категориальной переменной '*Breaking*' в числовую (бинарную). Это необходимо для использования этой переменной в качестве целевого признака в моделях машинного обучения. «*Y = ds['Breaking']»*: Целевой признак (или метки) сохраняются в переменной Y для дальнейшего использования.

Дальше следует команда по визуализации данных:

В цикле «*for i in [...]*»: Итерация проходит по ряду признаков для создания гистограмм. Следом извлекаются значения признака для случаев, когда '*Breaking*' равно «*1*», и удаляются все пропущенные значения. Аналогично извлекаются значения признака для случаев, когда '*Breaking*' равно «*0*». Обязательно надо вычислить минимальные и максимальные значения для оси X гистограммы.

Командой «*sns.distplot(...)*» создаются гистограммы с использованием библиотеки Seaborn. Одна гистограмма для '*died*', другая для '*survived*', с разными цветами для наглядности, как показано на рисунке 4.1.

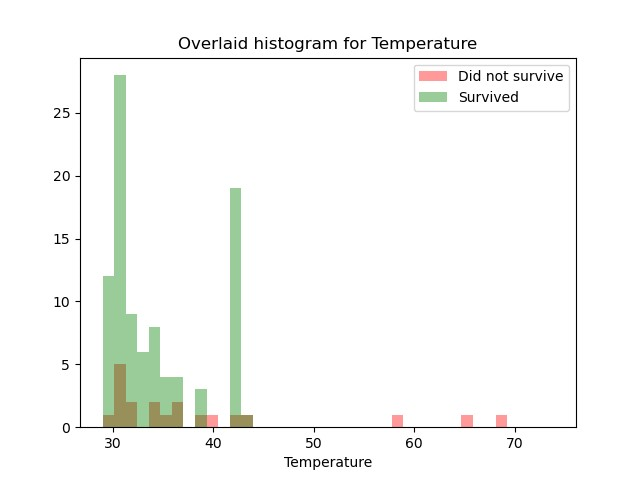


Рисунок 4.1 – Зависимость поломок от показателей температуры

Далее следуют команды по добавлению легенды для различия гистограмм, их названия и самого процесса отображения.

Следом идёт процесс по удалению выбранных столбцов из данных. Это происходит с использованием переменной '*cat\_feet*', которая содержит список категориальных признаков (features), которые необходимо удалить из набора данных перед обучением модели. В данном случае, список включает следующие столбцы: *'System uptime', 'Downtime', 'Breaking'*. Функционал этой переменной связан с последующим этапом кода, где используется метод *'drop'*, чтобы удалить указанные столбцы из набора данных:

(4.1)

Эта строка кода удаляет все столбцы, указанные в '*cat\_feet*', из *DataFrame 'ds'*. Как говорилось ранее в главе 2, удаление определенных столбцов может быть необходимым в машинном обучении, чтобы исключить лишние или нерелевантные признаки, которые могут негативно повлиять на качество обучения модели.

В данном случае, эти столбцы были исключены, так как они либо являются категориальными переменными, которые нужно по-другому обработать, либо они уже были использованы и преобразованы в другой части кода (например, столбец 'Breaking' был закодирован в бинарный формат).

Разделение данных происходит согласно команде *'train\_test\_split'*. Эта часть кода разделяет данные на обучающую и тестовую выборки (от 50% до 30%, в зависимости от количества данных, для тестирования). Это позволяет проверить производительность модели на неизвестных данных. Параметр «*random\_state=42*» обеспечивает воспроизводимость разделения. Результатом выполнения этой части кода являются четыре переменные:

1. X\_train: признаки обучающей выборки.
2. X\_test: признаки тестовой выборки.
3. Y\_train: целевые значения обучающей выборки.
4. Y\_test: целевые значения тестовой выборки.

Далее следует процесс создания конвейера (4.2), который объединяет два этапа обработки: стандартизацию признаков и классификацию с помощью метода опорных векторов (SVM):

(4.2)

*StandardScaler():* Это препроцессор, который стандартизирует признаки, удаляя среднее и масштабируя их до единичной дисперсии. Это помогает оптимизировать работу многих алгоритмов машинного обучения.

*SVM(gamma='scale', probability=True):* Это классификатор метода опорных векторов. Аргумент *gamma='scale'(4.3)* автоматически выбирает значение гамма, а *‘probability=True’* позволяет вычислять вероятности для каждого класса.

(4.3)

где *nfeatures –* это количество признаков;

*X.var –* дисперсия всех признаков. Использование *gamma='scale'* обычно предпочтительнее, чем использование фиксированного значения, поскольку это учитывает различный масштаб признаков в вашем наборе данных.

Конвейер (‘*pipeline’*) позволяет упростить рабочий процесс, соединяя несколько этапов обработки в одном объекте. В данном случае, он стандартизирует данные, а затем применяет классификатор SVM.

Вместе эти строки кода подготавливают данные и создают модель для обучения. После этого можно использовать метод ‘*fit’* для обучения модели на данных ‘*X\_train’* и ‘*Y\_train’*, а затем применять методы ‘*predict’* или ‘*predict\_proba’* для выполнения предсказаний на новых данных.

Затем идет сам процесс прогнозирования и оценка производительности модели. Производительность модели оценивается метриками *‘accuracy\_score’, ‘cross\_val\_score’,* и *‘classification\_report’.*

Далее производится 5-кратная кросс-валидация модели на обучающей выборке, выводятся результаты кросс-валидации и выводится среднее значение результатов кросс-валидации.

В самом конце идёт процесс вывода в окне полного отчёта о классификации, который включает в себя такие метрики как точность, полнота и F1-мера для каждого класса.

Как итог модели, если прогноз будет равен 1(сбой), выводится сообщение “Alarm!”, как показано на рисунке 4.2.

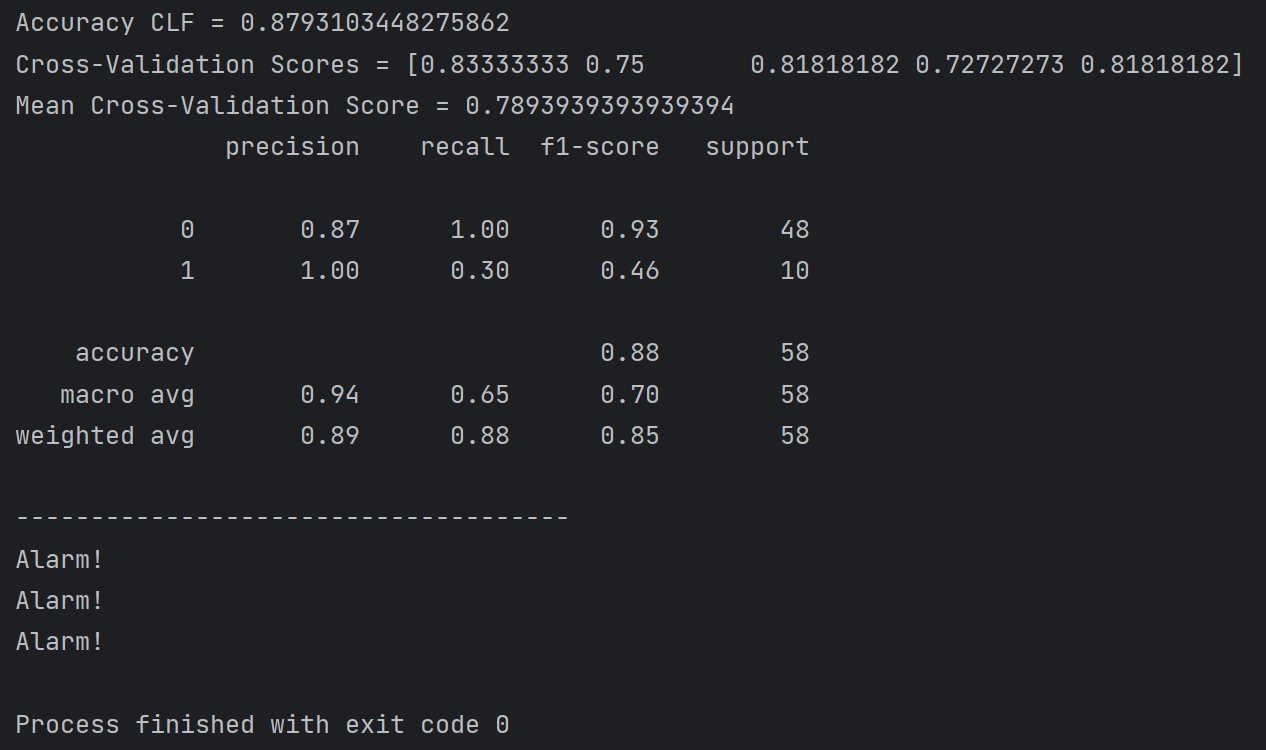


Рисунок 4.2 – Итоговый показатель модели на основе SVM метода

4.1.2 Описание алгоритма на основе метода деревьев решений и случайных деревьев (DT и RF)

Как и в предыдущей модели прогнозирования на основе алгоритма SVM, в начале идёт процесс импорта библиотек NumPy, Pandas, Pyplot и Seaborn для работы с данными в виде таблиц, её обработки и дальнейшей визуализации. Затем импортируется классификатор на основе случайных лесов из scikit-learn. Это ансамблевый метод, использующий множество деревьев решений для улучшения производительности модели. После идёт импорт логистической регрессии, которая является статистической моделью, используемой для анализа набора данных, в котором есть одна или несколько независимых переменных, влияющих на исход (обычно двоичный). Следом импортируем функций для создания пользовательских метрик (*make\_scorer*) и расчёта точности модели (*accuracy\_score*). Не следует забывать и про функции использующихся для разделения набора данных на обучающую и тестовую выборки (*train\_test\_split*) и для проведения кросс-валидации (*cross\_val\_score*). После импортируется классификатор на основе деревьев решений. Это модель предсказания значения целевой переменной на основе нескольких входных переменных.

Процесс загрузки датафрейма и его предварительная обработка идентична как у модели с SVM, то есть надо преобразовать категориальный признак 'Breaking' в бинарный формат. Если значение равно 'Yes', оно заменяется на 1, иначе на 0. Это необходимо для использования этого признака в моделях машинного обучения, которые требуют числовых входных данных. Затем указываются список признаков, которые будут удалены из DataFrame, и удаляются все строки, содержащие хотя бы одно пропущенное значение. Процесс удаления лишних столбцов, например столбца “Breaking” для получения строгой матрицы признаков также идентичен как и в модели SVM. Единственное отличие – это то, что независимые переменные (или признаки), на основе которых будет строиться модель, сохраняются в переменной «*X*»: «*X = df1*».

Далее следует процесс настройки визуализации данных, а именно – выбор стиля графиков Seaborn. В данном случае был выбран стиль «*ticks*». Это значит, что создаются попарные графики для всех признаков в DataFrame с использованием ядерной оценки плотности (KDE) для диагонали. Пример этих графиков приедены в главе 4.

Следующим этапом в данном алгоритме были этапы инициализации модели случайного леса с 100 деревьями, максимальной глубиной деревьев 100 и фиксированным начальным состоянием генератора случайных чисел (*random\_state*) для воспроизводимости результатов и инициализация модели дерева решений с критерием "энтропия" и максимальной глубиной 3. Количество числа разбиений для кросс-валидаций указывается 5. При увеличении этого параметра до 20, не выявлено качественных улучшений прогнозируемых моделей, поэтому 5-кратная кросс-валидация оставалась оптимальным выбором. Следует отметить что 5-кратная кросс-валидация делалась для каждой модели на основе МО без исключения. Затем следовал процесс оценки моделей и вывод результатов на экран, как показано на рисунке 4.3.

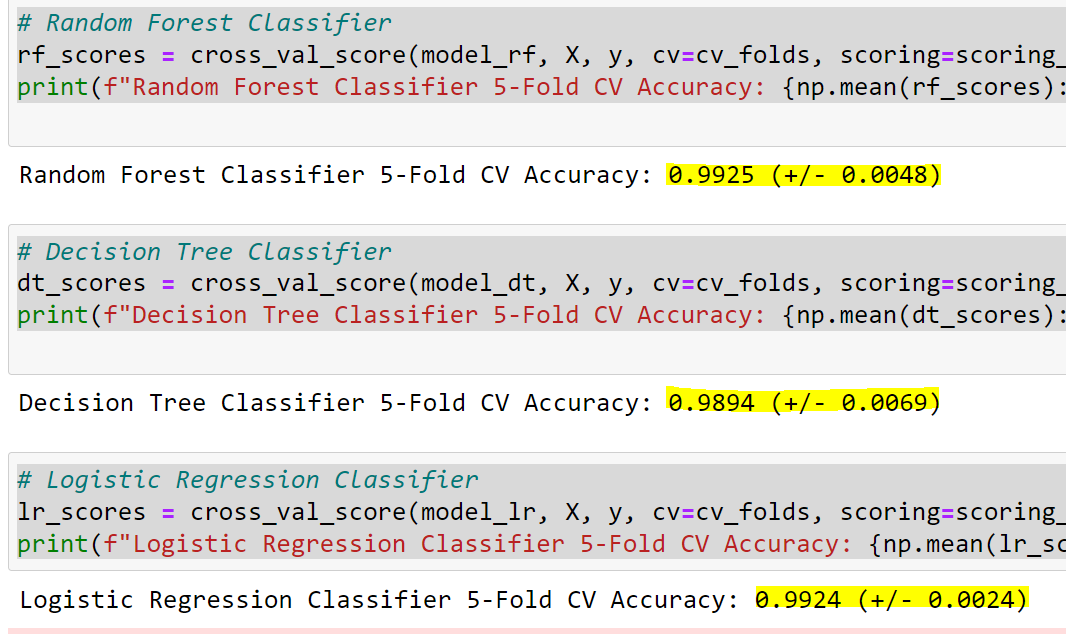


Рисунок 4.3 – Визуализация результатов моделей

Критерий энтропии является одним из методов, используемых для разделения узлов в деревьях решений и случайных лесах. Энтропия является мерой неопределённости или случайности в данных. В контексте деревьев решений, энтропия используется для оценки «беспорядка» в данных; чем выше энтропия, тем больше беспорядка и неопределённости в данных.

Формула (4.4) для расчёта энтропии узла *t* определяется следующим образом:

(4.4)

где «*с*» – количество классов в узле;

*pi* – вероятность выбора элемента класса *i* из узла *t*.

Когда дерево решений делает разбиение, оно стремится минимизировать энтропию в дочерних узлах. Идеальное разбиение будет иметь энтропию 0, что означает, что все значения принадлежат одному классу.

В данном коде была использована модель *DecisionTreeClassifier* с параметром *criterion="entropy"*. Это означает, что при построении дерева алгоритм будет использовать критерий энтропии для определения оптимальных разбиений узлов. Этот метод часто считается более робастным, особенно когда в данных много категориальных атрибутов.

* 1. **Сравнение результатов в ходе обучения моделей на основе МО**

После успешного разработанного алгоритма перед нами встала задача определить оптимальное количество данных, которое требуется для наиболее эффективного процесса обучения модели. Для достижения этой цели, основной набор данных был разделён на три отдельные выборки: первая содержала 500 записей, вторая увеличена до 1000 записей, и финальная выборка включала в себя 2500 записей. Стоит подчеркнуть, что независимо от размера выборки, количество параметров, используемых в алгоритме, оставалось постоянным и равнялось 14, однако некоторые параметры были проигнорированы алгоритмами, так как их весовые коэффициенты очень малы и граничили на уровне 0. Также все модели применяли пятикратную перекрёстную проверку.

На рисунке 4.4 можно наблюдать следующие результаты.

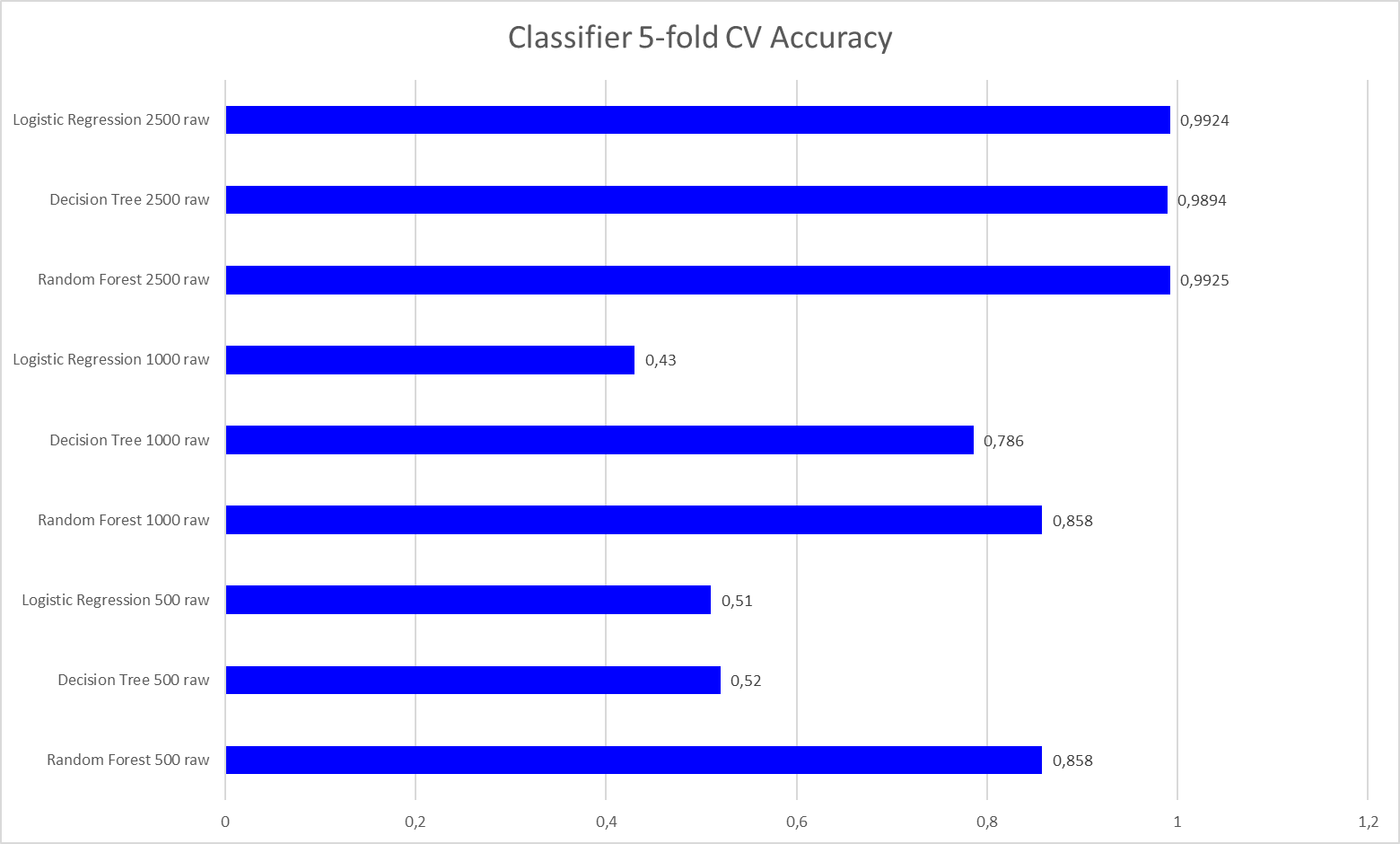


Рисунок 4.4 **–** Общая визуализация точности результатов прогнозирования

1. Для выборки в 500 строк:
   1. применяя метод логистической регрессии, была получена точность прогноза, равная 0,51;
   2. при использовании метода деревьев решений качество прогноза достигло 0,52;
   3. метод случайных деревьев показал более высокое качество прогноза, равное 0,858.
2. В случае выборки из 1000 строк данных:
   1. метод деревьев решений продемонстрировал увеличение качества прогноза до 0,786;
   2. напротив, метод логистической регрессии показал некоторое снижение точности прогноза, равное 0,43;
   3. метод случайных деревьев сохранил свою прошлую эффективность, равную 0,858.
3. И, наконец, для выборки размером в 2500 строк:
   1. метод логистической регрессии значительно улучшил свою эффективность, показывая качество прогноза равное 0,9924;
   2. метод деревьев решений также показал высокую точность прогноза, равную 0,9894;
   3. метод случайных деревьев выявил результат в 0,9925.

Анализируя данные, представленные на рисунке 4.4, можно сделать вывод о том, что с ростом объёма данных, используемых для обучения, точность прогнозирования также продолжает расти. Это подтверждает гипотезу о том, что больший объём обучающих данных может способствовать повышению точности модели.

В процессе исследования оптимального объёма данных для обучения модели, основанной на методе случайных деревьев решений, было установлено, что минимальное необходимое количество данных для обеспечения эффективного обучения и получения достоверных результатов находится в диапазоне от 500 до 1000 строк данных. Это интересное наблюдение, так как оно подчёркивает возможность достижения приемлемой точности прогноза даже при относительно небольшом объёме обучающего набора данных. Однако стоит отметить, что в случае других двух рассмотренных моделей для достижения удовлетворительной точности требуется значительно больший объём данных, а именно, более чем 2000 строк.

Также замечено, что с увеличением объёма данных, используемых для обучения, происходят изменения в распределении весов различных параметров в модели. Это весьма логичное наблюдение, так как больший объём данных предоставляет модели больше информации для выявления и учёта более тонких зависимостей между различными параметрами. Например, рассматривая модель, построенную на основе алгоритма деревьев решений, при использовании выборки из 500 строк данных, можно заметить, что некоторые параметры, такие как показатели температуры, индекс времени отклика и CPU 1, имеют больший вес и, следовательно, большую значимость по сравнению с другими параметрами. Эти наблюдения подтверждаются данными, представленными на рисунке 4.5, который детально иллюстрирует распределение весов этих параметров в рассматриваемой модели.

В ходе эксперимента, где количество обучающих данных было увеличено до 1000 строк, было выявлено, что модель, основанная на алгоритме деревьев решений, демонстрирует интересные изменения в весах отдельных параметров. Эти изменения показывают, как различные параметры становятся более или менее значимыми для модели в зависимости от объёма данных.

Конкретно, было отмечено, что вес параметра, связанного с температурными показателями, снизился с предыдущего значения в 0.48 до нового значения в 0.45. Несмотря на это уменьшение, данный параметр все ещё занимает первое место по значимости среди всех рассматриваемых параметров.

Далее, параметр, обозначенный как CPU 1, также показал изменение в своём весе, сократившись с 0.24 до 0.18. Этот параметр переместился на второе место по значимости в иерархии всех параметров, используемых в модели.

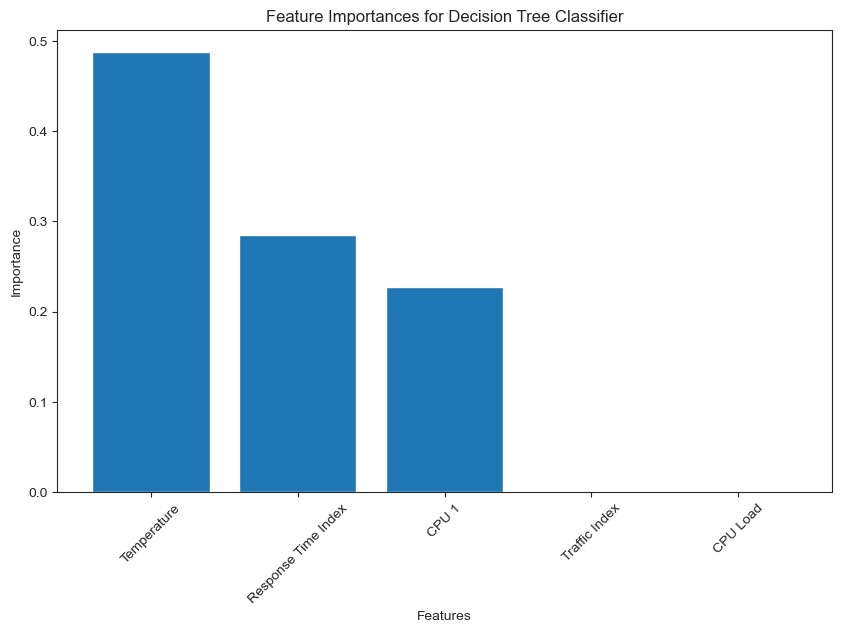


Рисунок 4.5 – Веса параметров при выборке в количестве 500 строк модели на основе метода деревьев решений

Третье по значимости место теперь занимает параметр, связанный с индексом времени отклика. Его вес уменьшился с 0.28 до 0.16, что говорит о снижении его относительной значимости для модели при увеличении объёма данных.

Кроме того, было замечено, что параметры, связанные с загруженностью процессора и индексом траффика, продемонстрировали увеличение своих весов: с 0 до 0.15 и с 0 до 0.06 соответственно. Эти изменения в весах параметров можно детально рассмотреть на рисунке 4.6, который визуализирует эти данные, позволяя наглядно увидеть динамику изменений.

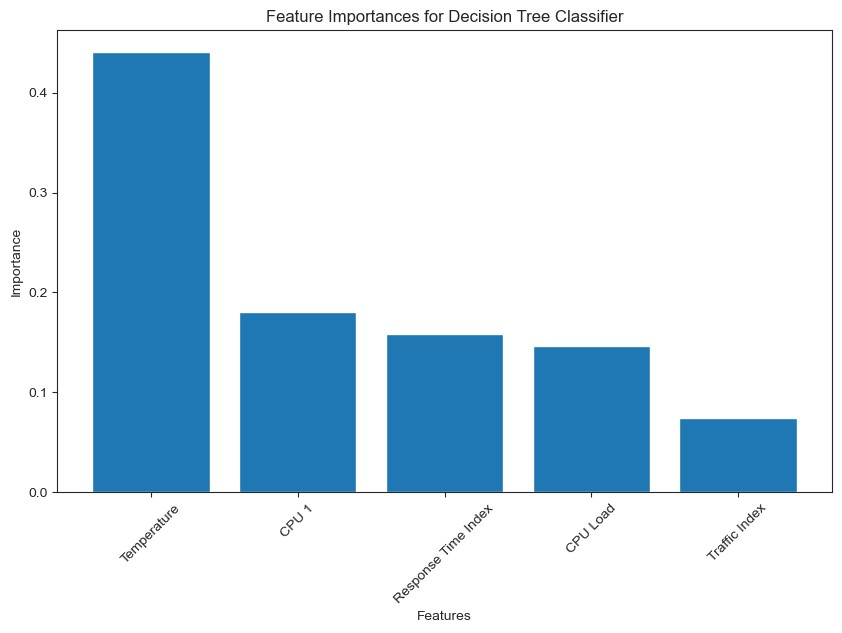


Рисунок 4.6 – Веса параметров при выборке в количестве 1000 строк модели на основе метода деревьев решений

После расширения обучающей выборки алгоритма до 2500 строк, при этом сохраняя константным коэффициент 5-кратной перекрёстной проверки, был обнаружен ряд интересных тенденций в распределении весов параметров.

Среди наиболее выдающихся изменений стоит отметить, что параметр, который отражает загруженность центрального процессора коммутатора (обозначенный как CPU Load), вышел на лидирующую позицию по значимости с индексом 0.29. Это демонстрирует, что при более крупной выборке этот параметр становится критически важным для этой модели.

Далее следуют параметры, связанные с индексом трафика и временем отклика. Их веса составляют приблизительно 0.23 и 0.185 соответственно. Это показывает, что эти два параметра также имеют большое значение для алгоритма при работе с более крупными датасетами.

Интересно также заметить, что в отличие от предыдущей выборки, состоящей из 1000 строк данных, некоторые параметры сместили свои позиции. Например, параметр, который отражает температурные показания коммутатора, переместился с лидирующей позиции на четвёртое место по значимости. А параметр CPU 1, который ранее занимал вторую позицию, теперь находится на последнем месте с весами 0.175 и 0.12 соответственно.

Эти наблюдения подтверждаются визуализацией на рисунке 4.7, который наглядно демонстрирует динамику изменений весов параметров в зависимости от размера обучающей выборки.

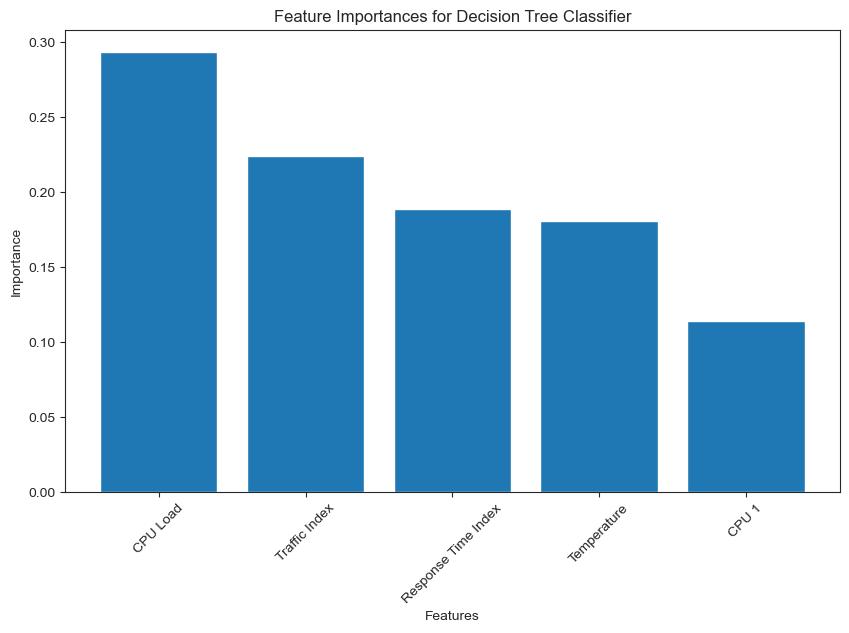


Рисунок 4.7 – Веса параметров при выборке в количестве 2500 строк модели на основе метода деревьев решений

Для более глубокого понимания работы различных моделей был проведён детальный анализ также и для модели, которая основана на применении метода случайных деревьев решений. Этот анализ ориентировался на выборку данных, в которой содержалось 500 строк параметров. Результаты этого анализа были представлены на рисунке 4.8. Этот график наглядно демонстрировал особенности работы данной модели.

Основываясь на полученных данных, можно заметить, что при достигнутой точности прогнозирования поломок на уровне 85.8% для модели на основе метода случайных деревьев решений, веса различных параметров распределены определенным образом. Так, параметр, который относится к индексу времени отклика, имеет вес равный 0.25, что делает его одним из наиболее важных. Затем следует параметр CPU 1 с весом 0.21. Температурный показатель коммутатора также является достаточно значимым с весом 0.20. При этом параметр, который характеризует загруженность центрального процессора коммутатора, имеет вес 0.18. Наконец, индекс траффика коммутатора в этой модели имеет вес 0.16.

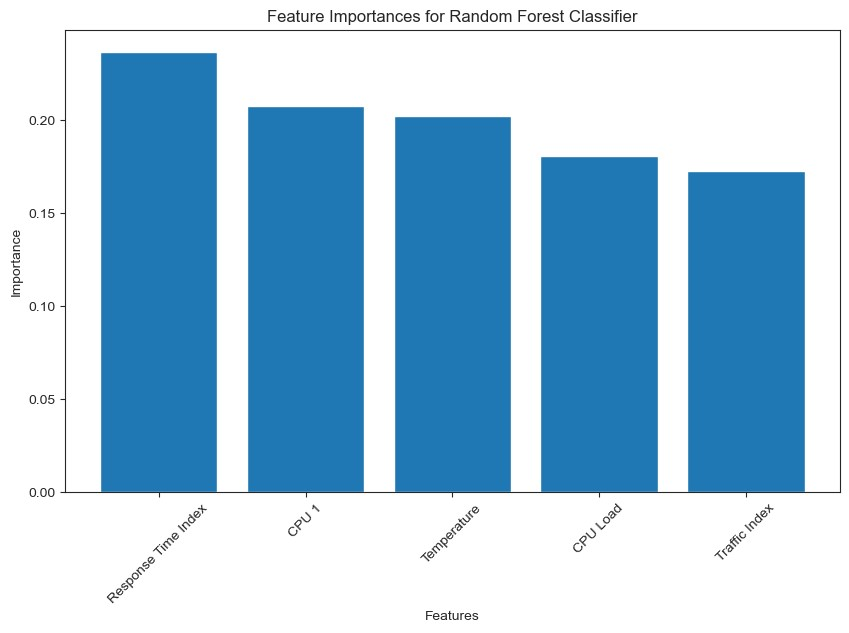


Рисунок 4.8 – Веса параметров при выборке в количестве 500 строк модели на основе метода случайных деревьев решений

Когда в рамках данной работы была рассмотрена расширенная выборка данных, включающая в себя до 1000 строк, стало ясно, что стоит отметить интересную особенность: несмотря на увеличение объёма данных, качественные характеристики по параметру точности прогнозирования модели остались без значительных изменений. Это может говорить о том, что объём данных в 1000 строк уже достаточен для стабильной работы модели, или же модель не способна выявить дополнительные закономерности в увеличенном объёме данных.

Тем не менее, интересно заметить, что, несмотря на отсутствие качественных улучшений в точности прогнозирования, произошли изменения в распределении весов параметров модели. Эти изменения были визуализированы на рисунке 4.9, что позволяет наглядно увидеть динамику изменений.

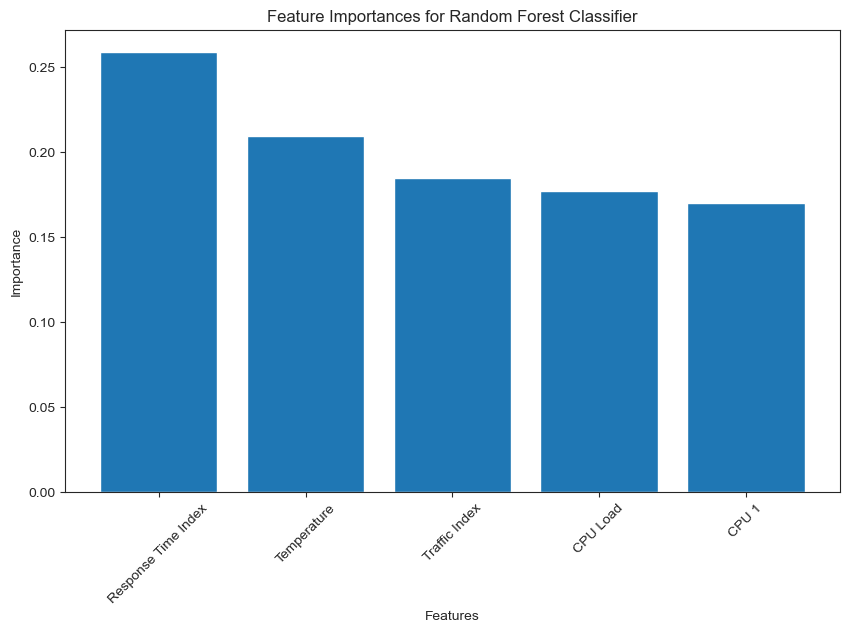


Рисунок 4.9 – Веса параметров при выборке в количестве 1000 строк модели на основе метода случайных деревьев решений

Более детально рассматривая эти изменения, можно заметить следующее:

1. Параметр, связанный с индексом времени отклика, увеличил свой вес и достиг значения 0.26. Это может свидетельствовать о том, что при большем объёме данных данный параметр становится более значимым для модели.
2. Вес параметра, который характеризует температуру, также показал рост и составил 0.21.
3. Не обошел стороной рост и параметр, связанный с индексом трафика, который увеличил свой вес до 0.19.
4. Однако, стоит отметить и те параметры, вес которых уменьшился. Так, параметр, отражающий загрузку центрального процессора коммутатора, показал снижение до 0.175.
5. Аналогичная тенденция наблюдается и у параметра CPU 1, чей вес уменьшился до 0.165.

Эти наблюдения демонстрируют, как даже при отсутствии качественных изменений в точности прогнозирования, расширение датасета может влиять на веса параметров, что в свою очередь может влиять на интерпретацию результатов и понимание взаимосвязей между данными.

Когда рассматривалась выборка данных, включающая в себя 2500 записей, стало очевидно, что модель, основанная на методологии случайных деревьев решений, проявила себя чрезвычайно эффективно, достигая поразительной точности прогноза в 99,25%. Это подтверждает высокую адаптивность и возможность модели корректно интерпретировать такой объём данных.

Далее, при анализе весов различных параметров, которые были использованы в этой модели, было выявлено несколько интересных наблюдений, которые также были визуализированы на рисунке 4.10 для наглядности и лучшего понимания:

1. Параметры, связанные с загрузкой центрального процессора коммутатора и индексом времени отклика, занимали лидирующие позиции в важности. Оба эти параметра имели веса, равные 0.22, что указывает на их ключевое значение для модели в этом конкретном контексте.
2. На третьем месте по важности оказался параметр, характеризующий индекс траффика. Его оценочный вес был установлен на уровне 0.19.
3. Немного уступая ему, другой важный параметр, обозначенный как CPU 1, демонстрировал вес в 0.185.
4. Наконец, параметр, отражающий температурные показатели коммутатора, на этот раз оказался на последнем месте с весом в 0.167.

Эти результаты подсвечивают важность каждого параметра для данной модели и позволяют получить глубокое понимание того, какие именно характеристики наиболее влияют на процесс прогнозирования в контексте использования метода случайных деревьев решений с таким количеством набора данных.

Тем самым, сравнивая веса параметров двух алгоритмов, на основе методов деревьев решений и методов случайных деревьев решений, можно определить что в обеих моделях параметр загруженности процессора сетевого оборудования является самым высоко оценённым параметром. На втором и третьем месте стоят параметры индекса траффика и индекса времени отклика сетевого коммутатора. Четвертый и пятый по важности параметры в обеих моделях были параметры температурного показателя и параметр CPU 1.

В рамках экспериментального исследования, было также рассмотрено использование алгоритма, основанного на логистической регрессии, в условиях аналогичных тем, что были использованы для других методов. Конкретно, была проведена пятикратная кросс-валидация и использовались три разных объёма данных: малый набор из 500 строк, средний объём данных в 1000 строк и наибольший объём в 2500 строк. Каждый из этих наборов данных включал в себя 14 различных параметров для анализа.

Когда было проведено тестирование алгоритма на основе метода логистической регрессии (LR) с использованием набора данных размером в 500 строк, то была получена довольно низкая точность прогнозирования поломок сетевого оборудования – всего 51%. Этот результат оказался наименьшим по сравнению с двумя другими методами, которые были рассмотрены в рамках данного исследования. Что касается важности или весов параметров для этой модели, они распределились следующим образом, что подтверждается данными на рисунке 4.11: параметр, измеряющий индекс времени отклика, имел наибольший вес среди всех параметров, составив 0.355. Непосредственно за ним шёл параметр, отражающий загрузку центрального процессора коммутатора, с весом 0.27. Третьим по важности был параметр, указывающий на индекс трафика коммутатора, его вес составил 0.1675. Оставшиеся параметры CPU 1 и показатели температуры имели веса 0.1075 и 0,1 соответственно.

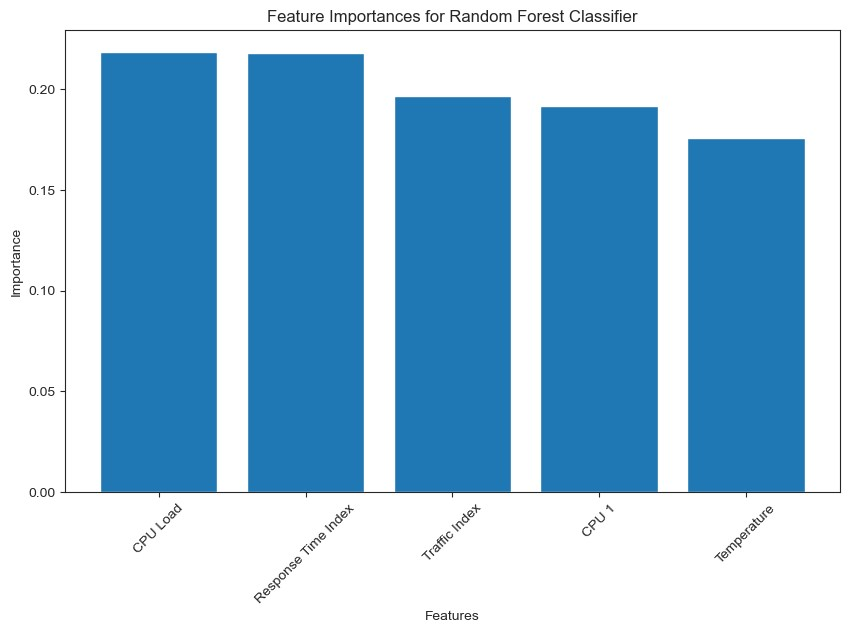


Рисунок 4.10 – Веса параметров при выборке в количестве 2500 строк модели на основе метода случайных деревьев решений (RF)

При продолжении данного экспериментального исследования, было принято решение увеличить объём используемого датасета до 1000 строк. Это позволило нам углубиться в анализ и получить ещё более детализированную картину о влиянии каждого из параметров на решение, принимаемое алгоритмом. Как результат такого увеличения объёма данных, наблюдалась заметная перемена в весах отдельных параметров:

1. Интересным образом, параметр, относящийся к температурным показателям, проявил себя как наиболее значимый, прыгнув с последней позиции прямо на первую. Его вес в данной модели составил 0.32.
2. Параметр, который ранее был на первом месте, а именно индекс времени отклика, теперь снизил свою важность, переместившись на второе место со значением веса в 0.26.
3. Далее, параметр, характеризующий загрузку центрального процессора коммутатора, занял третье место с весом 0.205.
4. Параметр индекса трафика также продемонстрировал себя как достаточно важный, его вес оказался равен 0.195.
5. Отдельно стоит упомянуть параметр CPU 1, который, кажется, утратил свою значимость в этой модели, так как его вес уменьшился до всего 0.02.

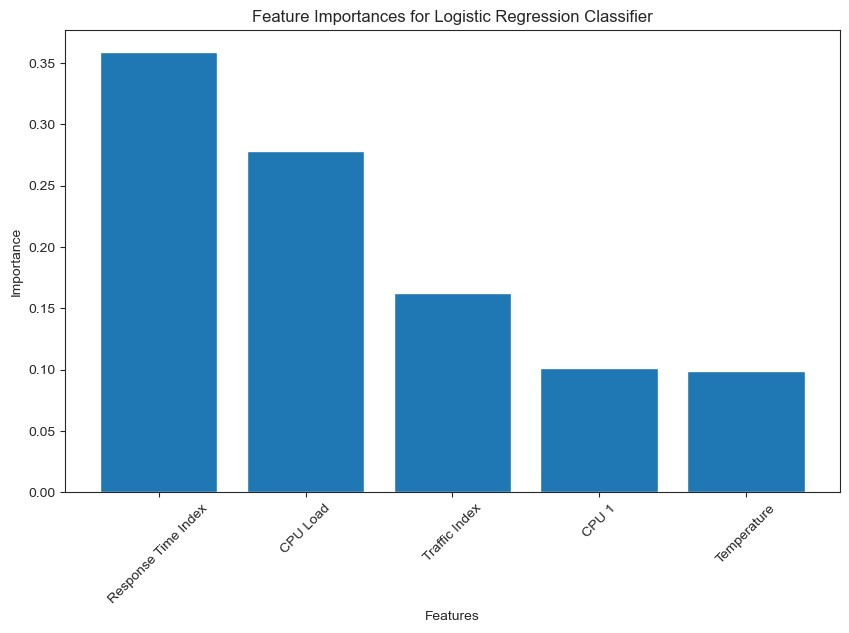


Рисунок 4.11 – Веса параметров при выборке в количестве 500 строк модели на основе метода логистической регрессии

Все эти детали и динамика изменения весов параметров подробно отражена на рисунке 4.12.

Продолжая экспериментальное исследование, был осуществлён следующий логический шаг – увеличение объёма датасета до значительных 2500 строк записей. Этот объём данных соответствует ежедневным записям семи коммутаторов, собранным за целый год их непрерывной работы. Подобное увеличение позволило нам ещё глубже погрузиться в характеристики и взаимосвязи данных, что несомненно повлияло на результаты моделирования. Иллюстрация этих данных показана на рисунке 4.13.

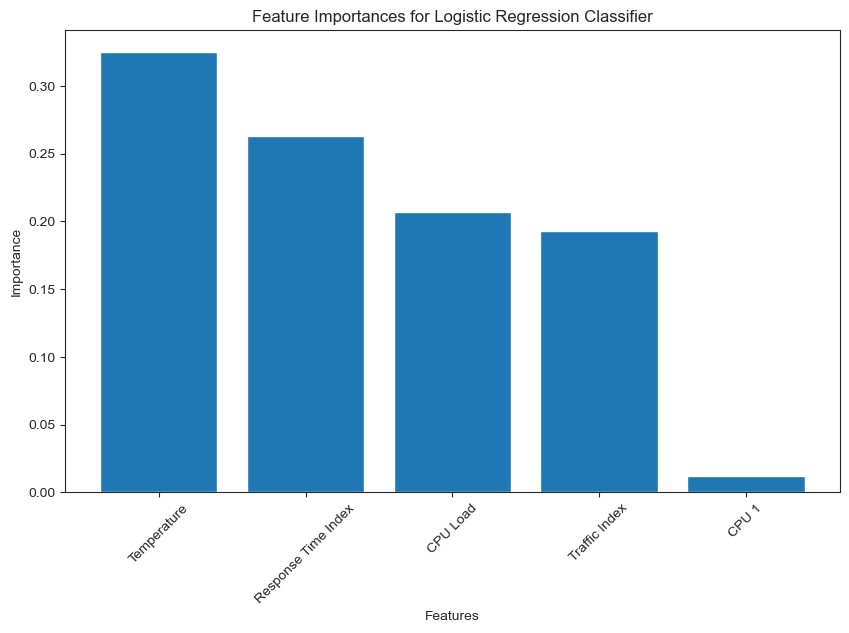


Рисунок 4.12 – Веса параметров при выборке в количестве 1000 строк модели на основе метода логистической регрессии

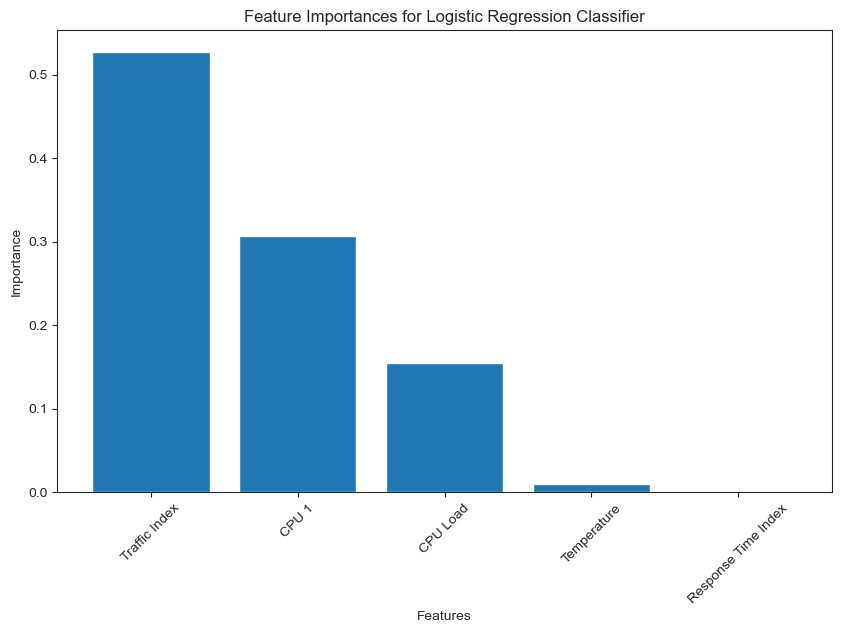


Рисунок 4.13 – Веса параметров при выборке в количестве 2500 строк модели на основе метода логистической регрессии

В ходе анализа, используя модель, основанную на методе логистической регрессии, были выявлены следующие характеристики весов параметров:

1. На первом месте по значимости оказался параметр, характеризующий индекс трафика. Этот параметр имел наивысший вес, равный 0.53, что подчёркивает его критическую важность в контексте этого алгоритма.
2. Следующим по важности оказался параметр CPU 1, который показал вес в 0.3. Это высокое значение говорит о том, что данная переменная имеет существенное влияние на результаты модели.
3. Параметр, отражающий загрузку процессора коммутатора, занял следующее место с весовым показателем 0.15.
4. Параметр температурных показаний, кажется, стал менее значимым, его вес установился на уровне примерно 0.02.
5. Параметр индекса времени отклика, возможно, оказался наименее важным для этой модели, так как его вес составил 0.

Также стоит подчеркнуть, что показатель точности (или accuracy) этой модели на уровне 99,24% говорит о высоком качестве прогнозирования, что, несомненно, свидетельствует о успешной работе модели при таком объёме данных.

В контексте обсуждения временных затрат, связанных с обработкой данных, следует уделить внимание следующим моментам. Все три модели, которые были в центре данного внимания, демонстрировали разные методологические подходы: одна была создана на основе традиционного метода деревьев решений, в то время как другая применяла метод случайных деревьев решений, а третья – метод логистической регрессии. Несмотря на их различия, все три модели показали сравнимые показатели производительности. Рассматривая сценарий в условиях "среды А", где проводились эксперименты с алгоритмами, эти модели потребовали всего лишь 2-3 секунды реального времени для обработки массива данных, включающего в себя 2500 записей, и это с учётом проведения пятикратной перекрёстной проверки.

Однако, переходя к обсуждению "среды В", стоит отметить, что, хотя исходные условия оставались схожими, время обработки увеличивалось и составляло до 6 секунд. Это даёт понять, что разные условия эксплуатации могут оказывать заметное влияние на производительность модели.

Принимая во внимание технические характеристики устройства, на котором проводились испытания, стоит подчеркнуть ключевую роль таких компонентов, как центральный процессор, размер оперативной памяти и особенности выбранной файловой системы. Эти параметры не просто составляют базовую инфраструктуру для работы алгоритмов, но и напрямую влияют на их эффективность. С учётом этого можно предположить, что качество и производительность используемых ресурсов будут иметь прямое отношение к времени, которое потребуется алгоритмам для обработки данных. Таким образом, оптимизация этих ресурсов может значительно ускорить процесс прогнозирования и сделать его более точным.

На завершающем этапе исследования было выполнено подробное сравнение трёх описанных ранее моделей алгоритмов с четвертой моделью, которая была построена на основе метода SVM. Исследуя алгоритм, использующий метод SVM, можно отметить, что условия тестирования были аналогичными условиям, при которых тестировались другие три модели. В частности, использовались наборы данных из 1000 и 2500 записей, 14 различных параметров, пятикратная кросс-валидация, а также проведение экспериментов в двух различных средах разработки: "А" и "В".

С точки зрения качественного показателя, а именно, метрики точности (accuracy), модель, основанная на методе SVM, показала результат в 87% при использовании набора данных из 1000 записей (см. рисунок 4.14) и 94% при наборе данных из 2500 записей (см. рисунок 4.15). Следует подчеркнуть, что этот результат оказался ниже по сравнению с результатами, которые были получены другими моделями для прогнозирования сбоев сетевого оборудования.

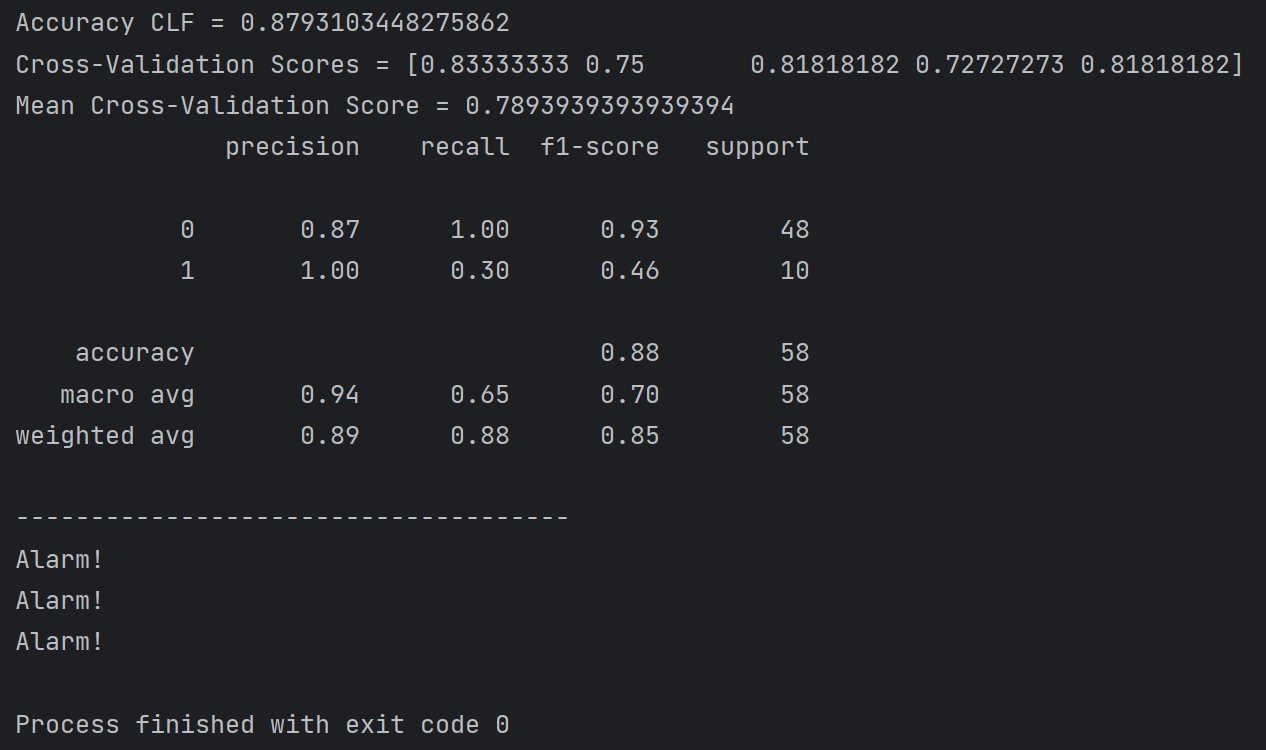


Рисунок 4.14 – Качественные показатели модели на основе метода SVM с набором данных в 1000 строк

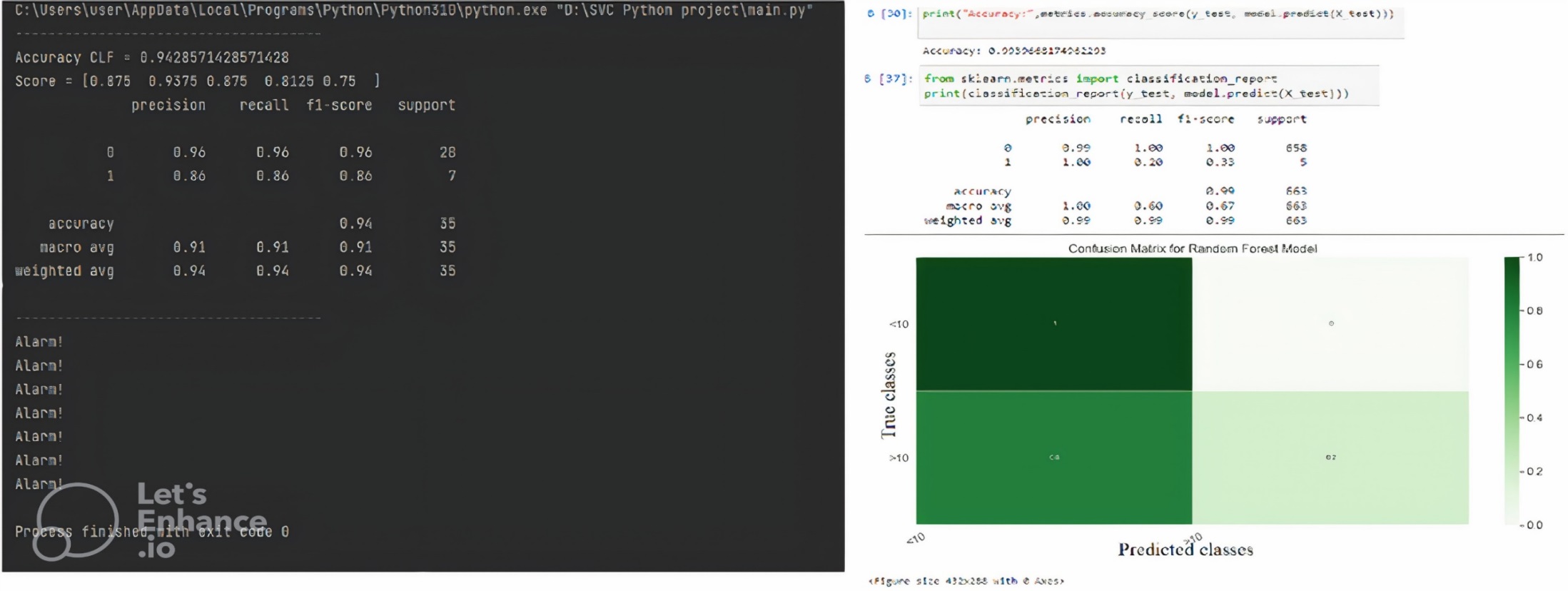


Рисунок 4.15 – Качественные показатели модели на основе метода SVM с набором данных в 2500 строк

С точки зрения времени, необходимого для прогнозирования, модель на основе метода SVM также показала наиболее продолжительное время среди всех рассмотренных моделей. В частности, в среде "А" время прогнозирования составило в среднем от 12 до 14 секунд, тогда как в среде "В" этот показатель варьировался от 17 до 19 секунд.

Таким образом, несмотря на то что модель на основе метода SVM показала наименьшую точность прогнозирования и самое длительное время работы среди всех рассматриваемых моделей, следует учесть одну важную особенность. А именно, метод SVM, являясь подклассом метода SVM, обычно обладает высокой устойчивостью к переобучению, особенно когда применяются методы регуляризации. Этот аспект может быть критически важным в определенных условиях и должен быть учтён при выборе модели для конкретных задач прогнозирования.

**4.3 Сравнение результатов гибридной модели DES-ML(RF/LR/DT/SVM)**

Как говорилось в разделе 3, для прогнозирования тренда каждого параметра данных на момент () был применён метод двойного экспоненциального сглаживания. Результаты прогноза тренда были следующими: при попытке выявить тренд параметра «Temperature» (рисунок 4.16), метод DES показал, что разность значений между предсказанными и наблюдаемыми значениями не велик, и колеблется от 1(min) до 10 (max)значений. Красная линия показывает сглаженный тренд, который отражает общее направление и динамику изменений температуры. Из графика видно, что модель DES эффективно выявляет основные тенденции в данных, позволяя лучше понять общее направление изменений температуры. Средний абсолютный процент ошибок (MAPE) равен примерно 5.25%. Это указывает на то, что в среднем предсказания отклоняются от фактических значений на 5.25%.

Средний абсолютный процент ошибок (MAPE) измеряет относительную точность модели и рассчитывается как среднее абсолютных процентов ошибок по всем наблюдениям и высчитывается следующим образом:

(4.5)

где n это количество наблюдений;

– предсказанные значения.

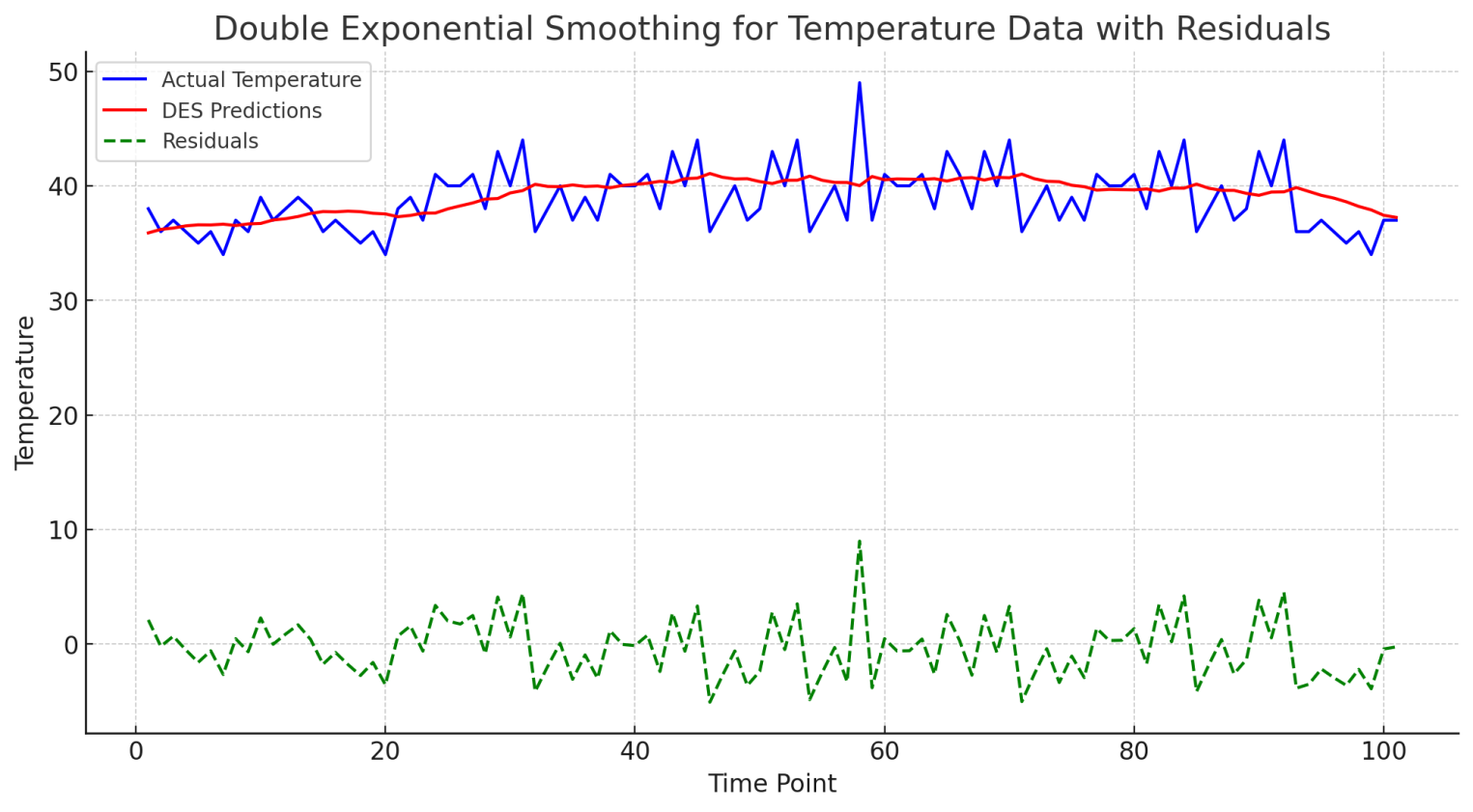


Рисунок 4.16 – Прогноз показателей параметра «Тemperature» методом DES

RMSE, или Среднеквадратичная ошибка (Root Mean Square Error), измеряет среднее значение квадратов ошибок, то есть разности между предсказанными значениями модели и фактическими значениями данных и его формула (4.6):

(4.6)

В данном случае, показатель RMSE приблизительно равна 2.56. Это говорит о том, что в среднем предсказания данной модели отклоняются от фактических значений на 2.56 единицы температуры, что является очень неплохим результатом в данном случае.

Далее DES был применён на параметр «CPU Load» для годичной записи (365 записей), и как показывает график, DES успешно справляется с прогнозированием трендов этого параметра (рисунок 4.17). Среднеквадратичная ошибка (RMSE) в данном случае в среднем, отклоняются от фактических значений на 0.2486 единицы. Средний абсолютный процент ошибок (MAPE) равен 1.1490%. Другими словами, метод DES для параметра «CPU Load» показывал очень неплохие результаты.

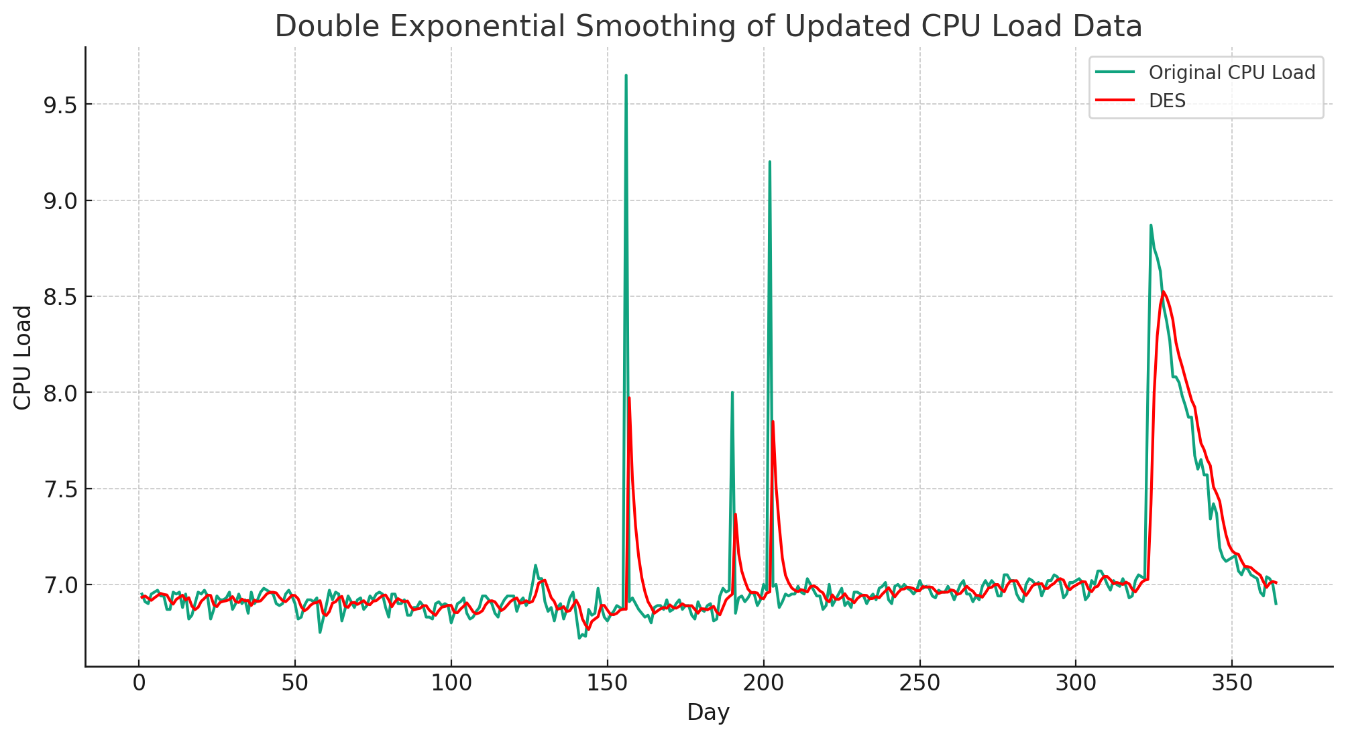
****

Рисунок 4.17 – Прогноз показателей параметра «CPU Load» методом DES

Далее представлены результаты применения метода DES на параметры «Traffic index» и «Response time index» на рисунках 4.18 и 4.19 соответственно.

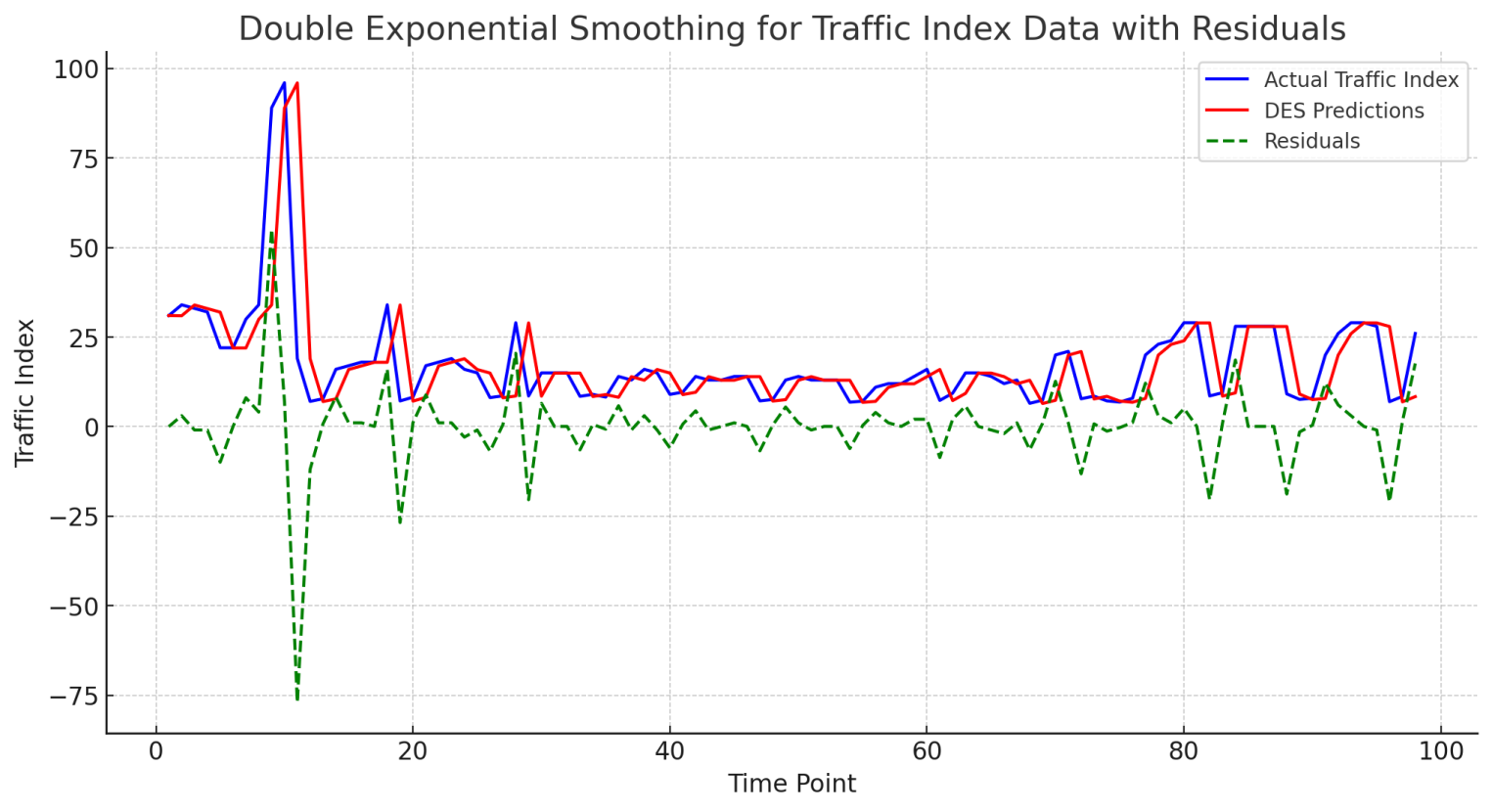


Рисунок 4.18 – Прогноз показателей параметра «Traffic index» методом DES.

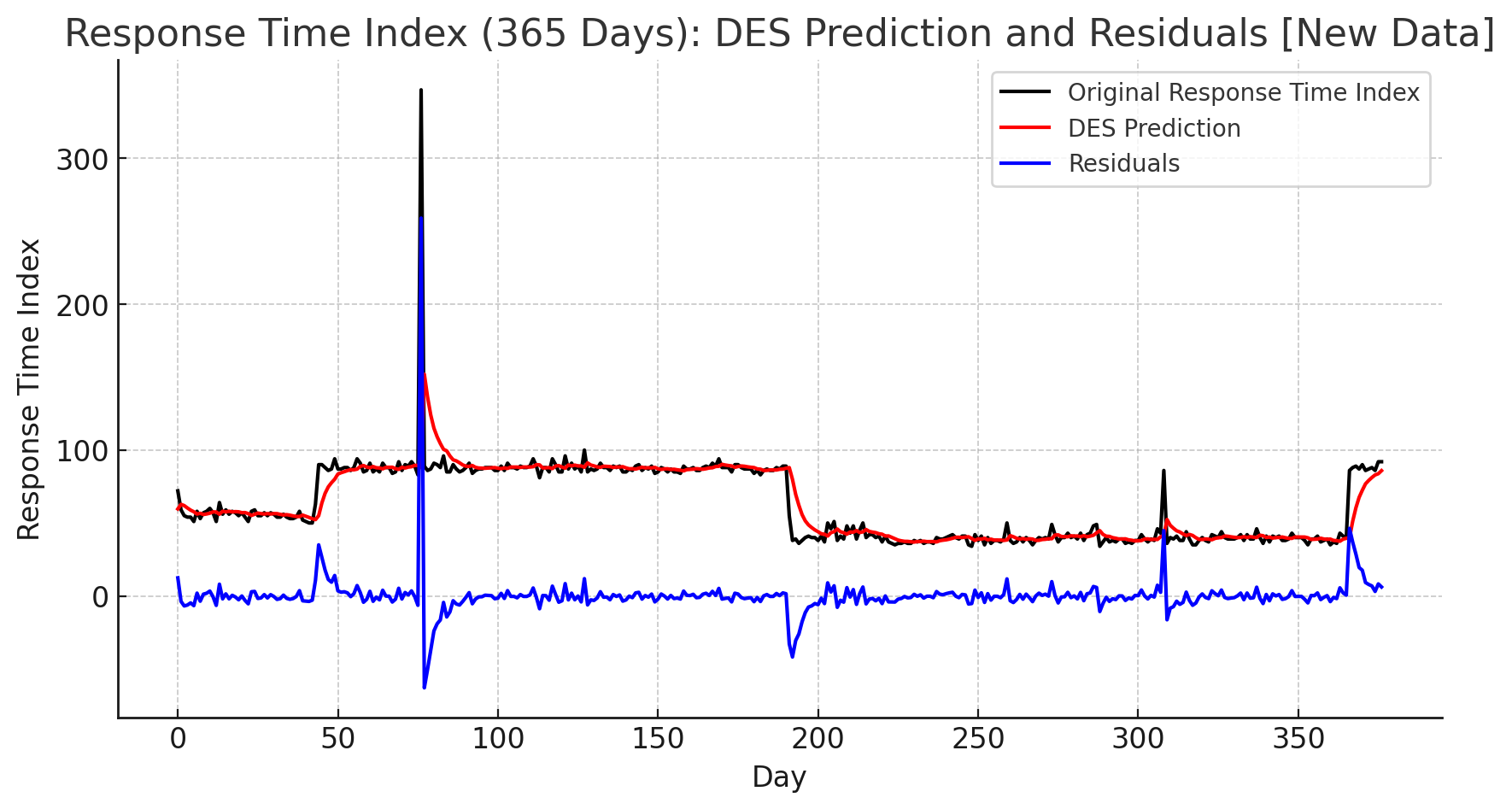


Рисунок 4.19 – Прогноз показателей параметра «Response time index» методом DES

Как видно из представленных (рисунки 4.16, 4.17, 4.18, 4.19) визуализации работы двойного экспоненциального сглаживания по каждому из приведённых в пример параметров, при плавном изменении параметров, результат прогнозирования имеет высокую точность. Однако, при резком изменении показателей, точность данных на короткое время снижается. Основной причиной этого является недостаточное количество точек данных, из-за чего алгоритм не может гладко соответствовать изменениям. Тем не менее, DES все же демонстрирует высокую точность предсказаний. А показатели MAPE и RMSE для этих параметров были не столь высоки, что свидетельствует о том, что модель в целом хорошо справляется с прогнозированием, хотя и с некоторыми отклонениями от фактических значений.

После метода DES, как и говорилось в разделе 3, были применены обученные модели на основе алгоритмов МО в спрогнозированные данные для классификации потенциальных поломок. Были получены следующие результаты: С качественной точки зрения, особенно с точки зрения показателей точности, модель на основе DES-SVM продемонстрировала точность 87% с набором данных из 1000 записей и 94% с набором данных из 2500 записей. Важно отметить, что эти результаты были ниже по сравнению с результатами, достигнутыми другими моделями прогнозирования отказов сетевого оборудования.

С точки зрения времени, необходимого для прогнозирования, модель на основе DES-SVM также показала самую большую продолжительность среди всех рассмотренных моделей. В частности, в «Среде А» время прогнозирования в среднем составляло от 12 до 14 секунд, а в «Среде Б» – от 17 до 19 секунд.

Самым эффективным по показателям времени обучения и точности результата были комбинированные методы DES-DT и DES-RF. В последнем методе, можно ограничиться обучением на менее чем 2000 записей данных для сокращения времени обучения.

Но несмотря на то, что модель на основе DES-SVM демонстрирует самую низкую точность прогнозирования и наибольшее время обработки среди рассмотренных моделей, крайне важно учитывать ключевую характеристику. Метод SVM обычно демонстрирует высокую устойчивость к переобучению, особенно когда применяются методы регуляризации. Этот аспект может быть критически важным при определенных условиях и его следует учитывать при выборе модели для конкретных задач прогнозирования (рисунок 4.20).

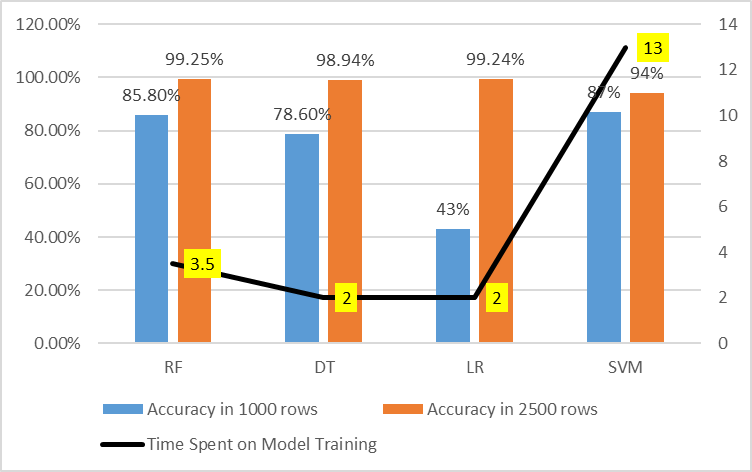


Рисунок 4.20 – Точность моделей и время, затраченное на обучение моделей DES-ML

**4.4 Оптимизация принятий решений в ЛВС**

В предшествующих главах было описано работа гибридной прогностической системы основанной на двойном экспоненциальном сглаживании который прогнозирует тренд параметров (), а обученные алгоритмы МО () классифицируют предсказанные тренды и дают прогноз () на наличие и отсутствие поломок на ближайшее N+1 время. Далее, полученный прогноз надо применить в процессе формирования ремонтных работ, что является задачей оптимизации процесса принятий решений. Эту задачу можно определить как условные вероятности правильного или корректного прогнозирования (4.1) и пропуска прогноза (4.2):

(4.1)

(4.2)

Поскольку соответствующие одному и тому же условию *A*1 решения и являются взаимоисключающими, то + =1

Качественным показателем обнаружения при отсутствии неисправности являются условная вероятность ложной тревоги или, иными словами, ложного прогноза поломки (4.3):

(4.3)

и вероятность правильно не спрогнозированной поломки (4.4):

(4.4)

причём .

В результате прогнозирования должно быть выдано решение о ремонте за N+1 время или отсутствии ремонтных работ. Решение может быть принято при двух взаимно исключающих условиях [8, 10]:

* условие *A*1 – «поломка спрогнозирована за N+1 время»;
* условие*A*0 – «поломки не будет за N+1 время», которое при выработке решения неизвестно.

За счёт потенциальных помех и флюктуаций прогнозирующей системы, каждому условию могут соответствовать два вида решения:

1. решение – «поломка спрогнозирована, принято решение к ремонту за N+1 время»;
2. решение – «поломки нет, соответственно, бездействие».

Третьего решения после завершения процесса прогнозирования не должно быть.

При прогнозировании поломок за N+1 время возможны четыре сценария:

1. сценарий – «спрогнозированная поломка за N+1 время обнаружена»;
2. сценарий – «спрогнозированная поломка за N+1 время пропущена»;
3. сценарий – «спрогнозированная поломка за N+1 время не состоялась, то есть ложная тревога»;
4. сценарий – «правильный прогноз корректной работы оборудования, то есть, корректный прогноз отсутствия поломок за N+1 время»

Перечисленным сценариям соответствуют четыре вероятности совмещения событий, сумма которых равна единице:

(4.5)

Каждому ошибочному сценарию поставим в соответствие некоторую цену – например, стоимость ошибки *riki =1000 USD/час; k =1*. Для безошибочных сценариев эту стоимость условимся считать приблизительно равной нулю *r*11 *=r*00 *≈0,* так как допускается возможность преждевременной (N+1) реакции обслуживающего технического персонала, нивелирующая негативные последствия. Тогда систему прогнозирования поломок можно характеризовать средней стоимостью (математическое ожидание стоимости) ошибочных сценариев:

(4.6)

Лучшей из сравниваемых систем прогнозирования поломок можно тогда считать систему, удовлетворяющую критерию минимума этой стоимости, иначе критерию риска.

Используя приведённые соотношения (4.1)–(4.4), выражение (4.6) для средней стоимости ошибки можно представить в виде:

(4.7)

или, после замены и простых преобразований, получается:

(4.8)

где .

При этом, критерий оптимальности прогнозирования по минимуму среднего риска сводится к так называемому весовому критерию

(4.9)

Последний показывает что совокупность требований повышения условной вероятности правильного прогноза поломки и понижения условной вероятности ложного прогноза поломки стремится к увеличению взвешенной разности . Множитель , называемый весовым множителем, зависит от соотношения стоимостей ошибок каждого вида и вероятностей наличия или отсутствия поломок в прогнозируемом объекте.

**4.5 Оптимизация плановых профилактических и экстренных ремонтных работ**

Программные продукты АСУ технического обслуживания и ремонтных работ, предлагаемые рынком, реализуют различные принципы и методики планирования планово-профилактических работ. В частности, возможно:

– планирование по установленным нормам периодичности и продолжительности эксплуатационного периода;

– планирование по наработке;

– планирование по заявкам на ремонт.

Разобьём процесс формирования графика плановых профилактических работ на предприятиях на несколько этапов: построение исходного графика; оптимизация графика; корректировка графика при поступлении новых заявок на ремонтные работы от предиктивной ИС.

Для реализации первого этапа выберем модель с учетом установленных норм периодичности:

; (4.10)

; (4.11)

где – номер устройства профилактической работы;

*j* – номер профилактической работы для рассматриваемого устройства;

*tн i j* – дата начала очередной *j-*ой профилактической работы *i*-го устройства;

– дата планируемого завершения очередной *j-*ой профилактической работы *i*-го устройства;

– дата предыдущей (j-1)-ой профилактической работы *i*-го устройства;

– норма времени для данного типа профилактической работы *i*-го устройства;

*i*-го устройства;

– коэффициент использования *i*-го устройства профилактической работы, определяющий степень нагруженности данного устройства.

Здесь следует уточнить математическое описание, включив в него элементы, касающиеся экстренных ремонтных работ спрогнозированной предиктивной системой, блок схема которой показана на рисунке 4.21. Эти работы часто требуют немедленного внимания и могут изменять планирование или требовать дополнительных ресурсов.

Для этого, добавим переменную – обозначающую наличие экстренной работы для устройства 𝑖 в период *V*. Она принимает значения или «1» – «требуется экстренный ремонт» или «0» – «не требуется проводить экстренные ремонтные работы».

Для учета возможного влияния экстренных работ на планирование, необходимо включить проверку условия наличия экстренных работ и их влияние на расписание профилактических работ.

Этим самым, мы совершенствуем формулу (4.10) следующим образом:

(4.12)

где – это добавочное время, необходимое для выполнения экстренных ремонтных работ, если они требуются.

Для каждого периода и каждого устройства проверяется , и в случае необходимости экстренного ремонта, время, затрачиваемое на эти работы, добавляется к общему времени планирования.

После расчёта времени начала каждой работы и времени завершения , ресурсы должны быть пересмотрены и оптимизированы с учетом возможного добавления экстренных работ.

В зависимости от наличия экстренных работ и остатка ресурсов, профилактические работы могут быть отложены или перепланированы, чтобы учесть приоритет и срочность ремонтов. Бок-схема системы расчёта профилактических работ устройств показана на рисунке 4.22.

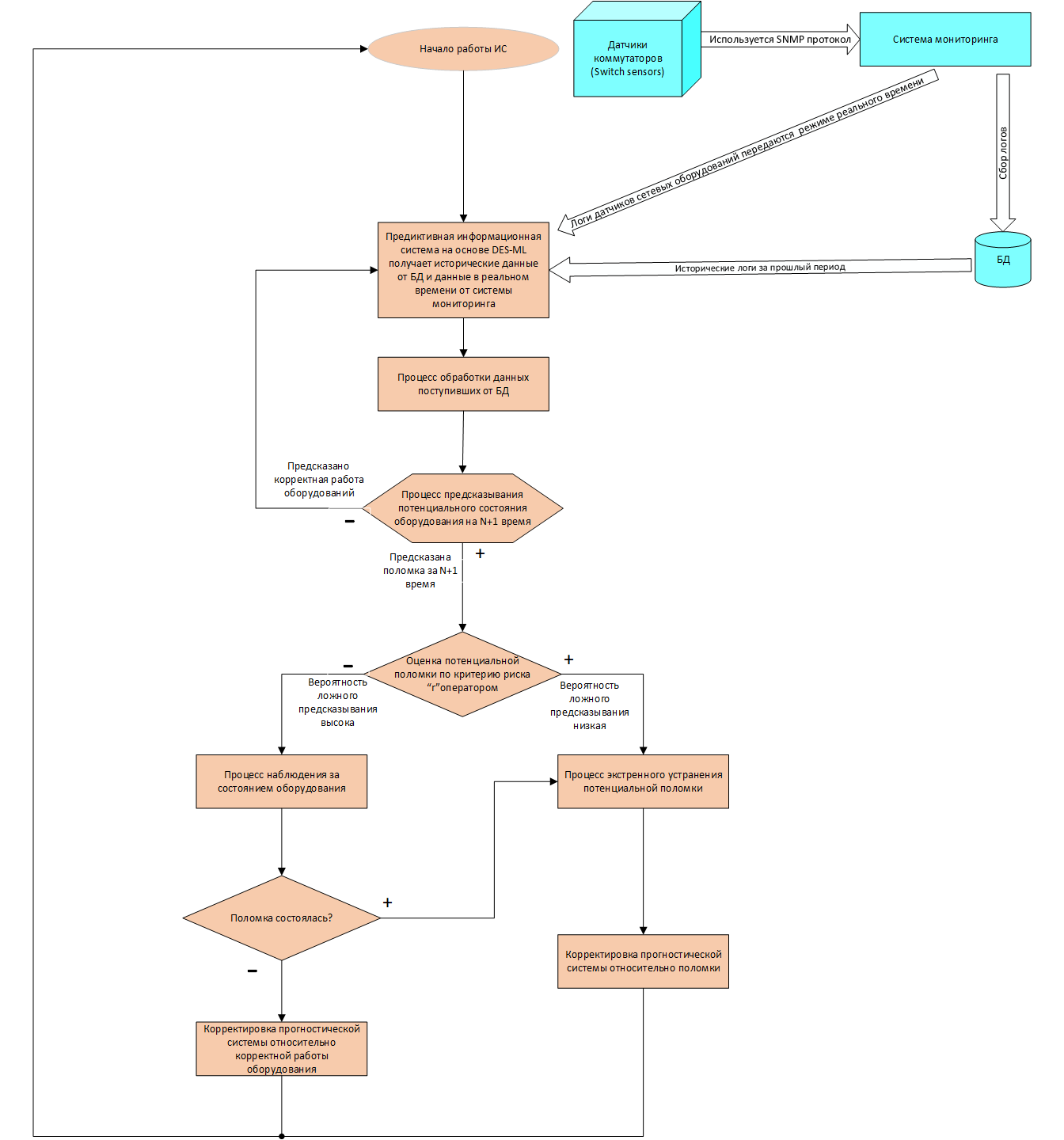


Рисунок 4.21 – Блок-схема предиктивной системы

Большинство функций информационной системы автоматизировано, однако, планирование ремонтов и ресурсов ведётся обслуживающим персоналом подразделений в ручном режиме. Это требует определенного времени и квалификации; не исключает ошибки и накладки при составлении планов; допускает их неоптимальность. Авторами предлагается алгоритм построения и оптимизации графиков ремонтных работ с учетом экстренных ремонтных работ, который может быть использован как основа системы поддержки принятия решений для главных специалистов с последующей её интеграцией в гипотетическую информационную систему предприятия "Планирование ремонтов".

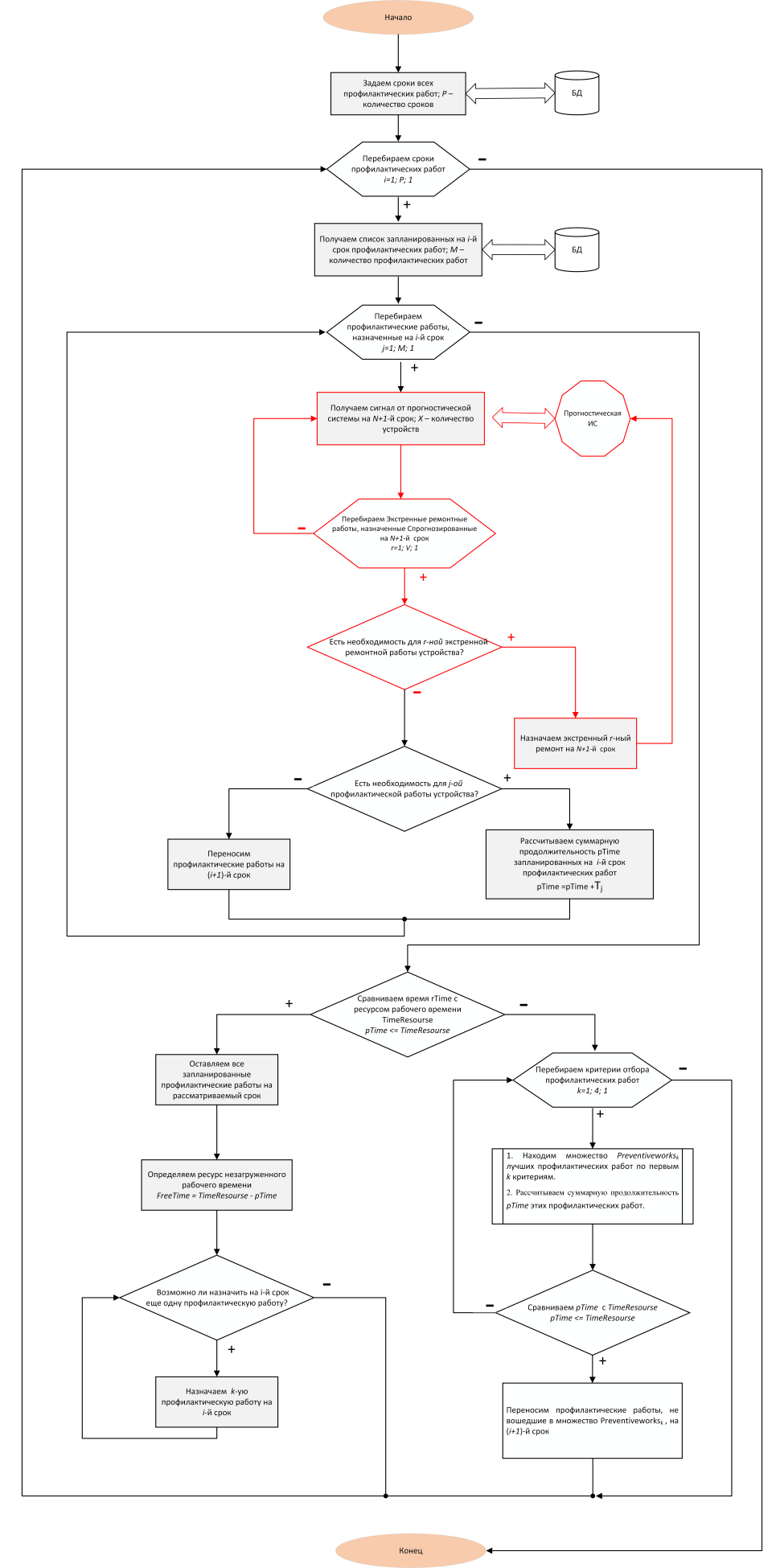


Рисунок 4.22 – Блок-схема системы расчёта профилактических работ устройств

**ЗАКЛЮЧЕНИЕ**

При сравнении деревьев решений, логистической регрессии, случайного леса и метода опорных векторов (SVM) в контексте прогнозирования сбоев в ЛВС, важно учитывать специфические характеристики проблемы и вовлеченных данных.

В сценарии прогнозирования сбоев в ЛВС может быть важно понимать причины предсказанного сбоя, например, какие сетевые параметры наиболее критичны для определения результата. Деревья решений предлагают более интерпретируемую модель, так как структура дерева ясно показывает связи между входными характеристиками и предсказанным сбоем. В отличие от этого, SVM более сложны и затрудняют понимание связей между входными характеристиками и предсказанным результатом.

Сетевые параметры часто демонстрируют сложные, нелинейные связи с событиями сбоя. SVM могут обрабатывать нелинейные данные с использованием "kernel trick", который позволяет модели находить разделяющую границу в пространстве большей размерности. Деревья решений/случайный лес также могут обрабатывать нелинейные данные, но для этого им может потребоваться более сложная структура дерева. Логистическая регрессия также может быть полезным инструментом для прогнозирования сбоев коммутаторов в ЛВС. Она особенно полезна, когда задача классификации линейная или приближенно линейная, что может быть актуально для определённых сценариев в сетевых системах.

Модели прогнозирования сбоев в ЛВС должны быть устойчивы к переобучению, так как переобученные модели не будут хорошо обобщаться на новые, невидимые сетевые события. SVM, как правило, более устойчивы к переобучению, особенно при использовании методов регуляризации. Деревья решений более склонны к переобучению, хотя такие методы, как обрезка дерева и установка ограничений на глубину дерева, помогают смягчить эту проблему. В свою очередь, логистическая регрессия может не справиться с задачами, где присутствуют сложные, нелинейные зависимости между признаками и метками классов. В таких случаях методы, такие как SVM с использованием ядер или деревья решений, могут оказаться более подходящими.

В крупных сетях с множеством устройств и точек данных масштабируемость может быть проблемой. Деревья решений обычно быстрее обучаются и более эффективно обрабатывают большие наборы данных. SVM могут быть вычислительно затратными, особенно при работе с большими наборами данных или использовании сложных ядер.

Данные мониторинга сети могут содержать пропущенные значения из-за сбоев устройств или проблем с коммуникацией. Деревья решений эффективнее обрабатывают пропущенные данные, используя методы замены или методы внесения пропусков в процессе обучения. SVM, как правило, требуют полных данных или внесения пропусков перед обучением. Для обработки пропущенных данных логистическая регрессия, как правило, требует предварительной импутации, что может внести дополнительную неопределенность в модель.

Для прогнозирования сбоев в ЛВС может быть необходимо определить наиболее важные сетевые параметры, влияющие на события сбоя. Деревья решений внутренне выполняют отбор признаков, выбирая наиболее важные признаки для разделения данных на каждом узле. SVM не имеют встроенного отбора признаков, но их можно комбинировать с методами отбора признаков, такими как рекурсивное исключение признаков (RFE).

В заключение, выбор между деревьями решений, случайными деревьями, логистической регрессией и SVM для прогнозирования сбоев в ЛВС зависит от конкретной проблемы, набора данных и желаемых свойств модели. Деревья решений предлагают большую интерпретируемость, способны обрабатывать пропущенные данные и, как правило, быстрее обучаются, в то время как SVM более устойчивы к переобучению и эффективнее моделируют сложные, нелинейные связи, в отличии от логистической регрессии.

И хотя модели на основе SVM и случайных деревьев могут иметь свои преимущества, модель на основе деревьев решений кажется более эффективной в данном конкретном контексте, как с точки зрения вычислительной эффективности, так и точности прогнозирования. Таким образом, для промышленных масштабных приложений на основе предоставленной информации более подходящим выбором может быть модель на основе деревьев решений.

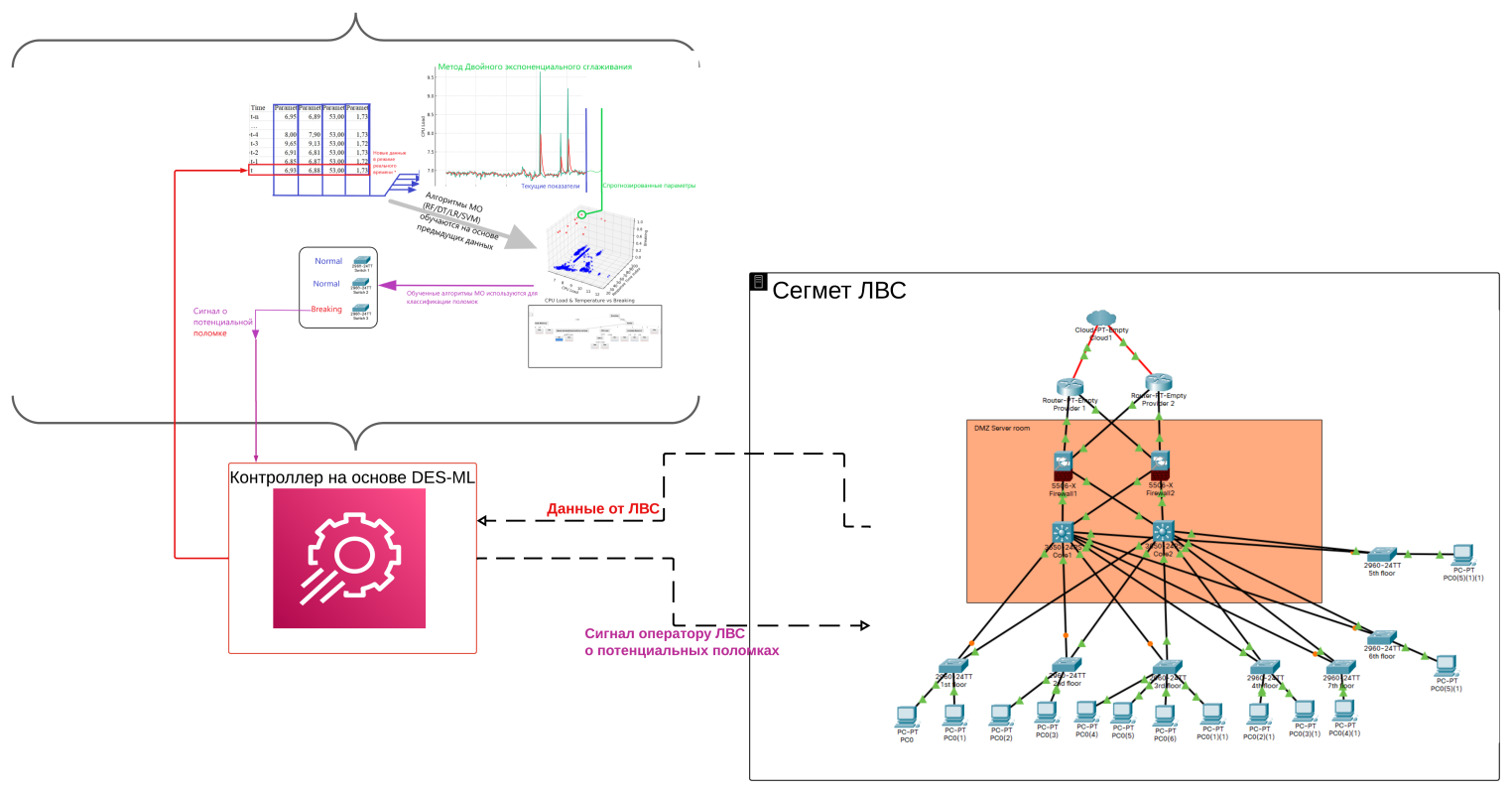


Рисунок 4.23 **–** Концепт предиктивной системы DES-ML на предприятии

В качестве заключения, хочет подчеркнуть, что концепт на основе DES-ML показанная на рисунок 4.23 и бизнес-процесс показанная на рисунке 4.24 может повысить отказоустойчивость ЛВС в целом. Но следует отметить и то, что такая концепция является, пожалуй, самой затратной как в финансовом, так и в профессиональном плане. Нужно будет развёртывать дополнительные аппаратные устройства хранилищ данных, систем прогнозирования. Нужно будет повысить квалификацию оператора ЛВС для эффективной работы с такими системами помощи оператору. Но все же теоретическая выгода и преимущество такой концепции неоспорима, так как сбои активных оборудовании будут предсказываться по неочевидным признакам системой предсказывания на базе гибридного алгоритма машинного обучения, что позволит среагировать на опережение оператору ЛВС, тем самым уменьшив время простоя ЛВС.

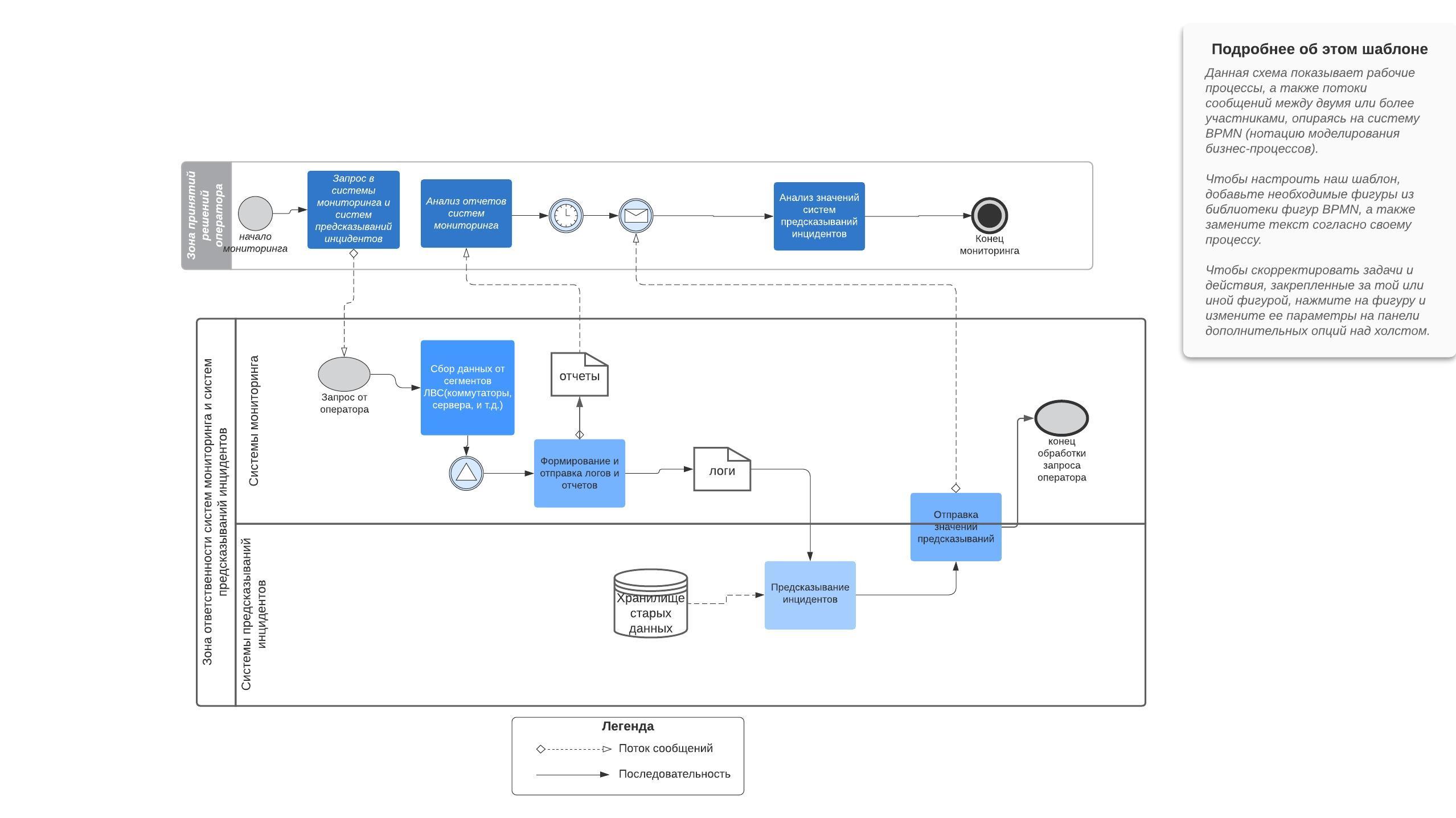
****

Рисунок 4.24 **–** Процесс мониторинга состояния ЛВС оператором посредством систем мониторинга и систем прогнозирования

Следует подчеркнуть, что вышесказанная концепция целесообразна лишь в тех случаях, когда организации требуется безотказно функционирующая система ЛВС и она способна в долгосрочном планировании извлечь выгоду от применяемой технологии.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The age of analytics: competing in a data-driven world // <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-age>. 10.11.2023.
2. Elliot S. DevOps and the cost of downtime: Fortune 1000 best practice metrics quantified // <https://whitepapers.theregister.com/paper/view/4077/devops>. 10.11.2023.
3. Frawley W.J., Piatetsky-Shapiro G., Matheus C.J. Knowledge discovery in databases: An overview // AI magazine. – 1992. – Vol. 13, №3. – P. 57-70.
4. Fayyad U.M. et al. Advances in Knowledge Discovery and Data Mining. – Menlo Park, 1996. – 611 р.
5. Wirth R., Hipp J. CRISP-DM: Towards a standard process model for data mining // Proceed. of the 4th internat. conf. on the practical applications of knowledge discovery and data mining. – Manchester, 2000. – Р. 29-40.
6. Mount J., Zumel N. Practical data science with R. – NY.: Simon and Schuster, 2019. – 568 p.
7. Fayyad U., Piatetsky-Shapiro G., Smyth P. The KDD process for extracting useful knowledge from volumes of data // Communications of the ACM. – 1996. – Vol. 39, №11. – P. 27-34.
8. Hevner A.R. et al. Design science in information systems research // Management Information Systems Quarterly. – 2008. – Vol. 28, №1. – P. 75-105.
9. Duenas J.C. et al. Applying event stream processing to network online failure prediction // IEEE Communications Magazine. – 2018. – Vol. 56, №1. – P. 166-170.
10. González J.M.N., Jiménez J.A., López J.C.D. Optimizing failure prediction time windows through genetic algorithms and random forests // IEEE Access. – 2018. – Vol. 6. – P. 58307-58323.
11. Gonzalez J.M.N. et al. Root cause analysis of network failures using machine learning and summarization techniques // IEEE Communications Magazine. – 2017. – Vol. 55, №9. – P. 126-131.
12. Kotsiantis S.B. et al. Supervised machine learning: A review of classification techniques // Emerging artificial intelligence applications in computer engineering. – 2007. – Vol. 160, Issue 1. – P. 3-24.
13. Sutton R.S. Learning to predict by the methods of temporal differences // Machine learning. – 1988. – Vol. 3. – P. 9-44.
14. Muggleton S., King R.D., Stenberg M.J. E. Protein secondary structure prediction using logic-based machine learning // Protein Engineering, Design and Selection. – 1992. – Vol. 5, №7. – P. 647-657.
15. Carpenter G.A. A self-organizing neural network for supervised learning, recognition, and prediction // IEEE Communications Magazine. – 1992. – Vol. 30, №9. – Р. 38-49.
16. Chen J.H., Asch S.M. Machine learning and prediction in medicine–beyond the peak of inflated expectations // The New England journal of medicine. – 2017. – Vol. 376, №26. – P. 2507-2509.
17. Weng S.F. et al. Can machine-learning improve cardiovascular risk prediction using routine clinical data? // PLoS ONE. – 2017. – Vol. 12, №4. – P. e0174944.
18. Athey S. The impact of Machine Learning on Economics // In book: The economics of artificial intelligence: An agenda. – Chicago, 2018. – P. 507-547.
19. Asim K.M. et al. Earthquake magnitude prediction in Hindukush region using machine learning techniques // Natural Hazards. – 2017. – Vol. 85. – P. 471-486.
20. Kong Q. et al. Machine learning in seismology: Turning data into insights // Seismological Research Letters. – 2019. – Vol. 90, №1. – P. 3-14.
21. Sasisekharan R. et al. Data mining and forecasting in large-scale telecommunication networks // IEEE expert. – 1996. – Vol. 11, №1. – P. 37-43.
22. Challagulla V.U.B. et al. Empirical assessment of machine learning based software defect prediction techniques // International Journal on Artificial Intelligence Tools. – 2008. – Vol. 17, №02. – P. 389-400.
23. Liang Y. et al. Bluegene/L Failure Analysis and Prediction Models // Procced. internat. conf. on Dependable Systems and Networks (DSN'06). – Philadelphia, 2006. – P. 425-434.
24. Lima R.F., Pereira A.C.M. A fraud detection model based on feature selection and undersampling applied to web payment systems // Procced. IEEE/WIC/ACM internat. conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT). – Singapore, 2015. – P. 219-222.
25. Salfner F., Lenk M., Malek M. A survey of online failure prediction methods // ACM Computing Surveys (CSUR). – 2010. – Vol. 42, №3. – P. 1-42.
26. Alonso J., Belanche L., Avresky D. R. Predicting software anomalies using machine learning techniques // Procced. IEEE 10th internat. sympos. on network computing and applications. – Cambridge, 2011. – P. 163-170.
27. Omar S., Ngadi A., Jebur H. H. Machine learning techniques for anomaly detection: an overview // International Journal of Computer Applications. – 2013. – Vol. 79, №2. – P. 33-41.
28. Hamerly G. et al. Bayesian approaches to failure prediction for disk drives // ICML. – 2001. – Vol. 1. – P. 202-209.
29. Pelaez A. et al. Online failure prediction for hpc resources using decentralized clustering // Procced. 21st internat. conf. on High Performance Computing (HiPC). – Goa, 2014. – P. 1-9.
30. Chigurupati A., Thibaux R., Lassar N. Predicting hardware failure using machine learning // Procced. 2016 Annual Reliability and Maintainability sympos. (RAMS). – Tucson, 2016. – P. 1-6.
31. Weiss G. Predicting telecommunication equipment failures from sequences of network alarms // In book: Handbook of Knowledge Discovery and Data Mining. – Oxford, 2002. – Р. 891-896.
32. Shatnawi M., Hefeeda M. Real-time failure prediction in online services // Procced. IEEE conf. on Computer Communications (INFOCOM). – Hong Kong, 2015. – P. 1391-1399.
33. Weiss G.M., Hirsh H. Learning to predict rare events in categorical time-series data // International Conference on Machine Learning. – 1998. – Vol. WS-98-07. – P. 83-90.
34. Agarwal V. et al. Discovering rules from disk events for predicting hard drive failures // Procced. internat. conf. on Machine Learning and Applications. – Miami, 2009. – P. 782-786.
35. Borkowski M. et al. Event-based failure prediction in distributed business processes // Information Systems. – 2019. – Vol. 81. – P. 220-235.
36. Kotsiantis S.B., Kanellopoulos D., Pintelas P.E. Data preprocessing for supervised leaning // International Journal of computer science. – 2006. – Vol. 1, №2. – P. 111-117.
37. Rowe J.W. A study of the path availability improvements in a large network for various route diversity protection strategies, using failure event simulation // Procced. IEE national conf. on Telecommunications. – NY., 1989. – P. 167-172.
38. Bazargan M., McGrath R.N. Discrete event simulation to improve aircraft availability and maintainability // Procced. Annual Reliability and Maintainability sympos. – Tampa, 2003. – P. 63-67.
39. Hua B. et al. Extracting rare failure events in composite system reliability evaluation via subset simulation // IEEE Transactions on Power Systems. – 2014. – Vol. 30, №2. – P. 753-762.
40. Hebert J. Predicting rare failure events using classification trees on large scale manufacturing data with complex interactions // Procced. IEEE internat. conf. on big data (big data). – Washington, 2016. – P. 2024-2028.
41. Vilalta R., Ma S. Predicting rare events in temporal domains // Procced. IEEE internat. conf. on Data Mining. – Maebashi CityE, 2002. – P. 474-481.
42. Akidau T., Chernyak S., Lax R. Streaming systems: the what, where, when, and how of large-scale data processing. – Sebastopol, 2018. – 352 p.
43. Gwadera R., Atallah M.J., Szpankowski W. Reliable detection of episodes in event sequences // Knowledge and Information Systems. – 2005. – Vol. 7. – P. 415-437.
44. Sahoo R.K. et al. Critical event prediction for proactive management in large-scale computer clusters // Proceed. of the ninth ACM SIGKDD internat. conf. on Knowledge discovery and data mining. – Washington, 2003. – PС. 426-435.
45. Fu S., Xu C.Z. Exploring event correlation for failure prediction in coalitions of clusters // Proceed. of the 2007 ACM/IEEE conf. on Supercomputing. – Reno, 2007. – P. 1-12.
46. Li J. et al. Hard drive failure prediction using decision trees // Reliability Engineering & System Safety. – 2017. – Vol. 164. – P. 55-65.
47. Chuah E. et al. Linking resource usage anomalies with system failures from cluster log data // Procced. IEEE 32nd internat. sympos. on Reliable Distributed Systems. – Braga, 2013. – P. 111-120.
48. Fulp E.W., Fink G.A., Haack J.N. Predicting Computer System Failures Using Support Vector Machines // Procced. conf. 1st USENIX Workshop on the Analysis of System Logs (WASL, 2008). – San Diego, 2008. – P. 1-9.
49. Li J. et al. Hard drive failure prediction using classification and regression trees // Procced. 44th Annual IEEE/IFIP internat. conf. on Dependable Systems and Networks. – Atlanta, 2014. – P. 383-394.
50. Weiss G.M. Timeweaver: A genetic algorithm for identifying predictive patterns in sequences of events // Proceed. of the 1st Annual conf. on Genetic and Evolutionary Computation. – Orlando, 1999. – P. 718-725.
51. Weiss G.M., Hirsh H. Learning to predict extremely rare events // <https://aaai.org/papers/WS00-05-013-learning-to-predict-extremely-rare>. 10.11.2023.
52. Yang W. et al. Hard drive failure prediction using big data // Procced. IEEE 34th sympos. on Reliable Distributed Systems Workshop (SRDSW). – Montreal, 2015. – P. 13-18.
53. Kholidy H. A. et al. Online risk assessment and prediction models for Autonomic Cloud Intrusion srevention systems // Procced. IEEE/ACS 11th internat. conf. on Computer Systems and Applications (AICCSA). – Doha, 2014. – P. 715-722.
54. Dangut M.D., Skaf Z., Jennions I.K. An integrated machine learning model for aircraft components rare failure prognostics with log-based dataset // ISA transactions. – 2021. – Vol. 113. – P. 127-139.
55. Yu L. et al. Practical online failure prediction for blue gene/p: Period-based vs event-driven // Procced. IEEE/IFIP 41st internat. conf. on Dependable Systems and Networks Workshops (DSN-W). – Hong Kong, 2011. – P. 259-264.
56. Wang Z. et al. Failure prediction using machine learning and time series in optical network //Optics Express. – 2017. – Т. 25. – №. 16. – С. 18553-18565.
57. Wiyanti W. Effectiveness of Single and Double Exponential Smoothing: SES, ARRSES and Holt’s Linear for Time Series Data Prediction with Trend and Non-seasonal Characteristic (Covid-19 Vaccinate Case) // Jurnal Matematika, Statistika dan Komputasi. – 2023. – Vol. 20, №1. – P. 52-64.
58. Shabir F. et al. Implementation Of The Double Exponential Smoothing Method In Determining The Planting Time In Strawberry Plantations // Proceed. of the 1st Jakarta internat. conf. on Multidisciplinary Studies Towards Creative Industries (JICOMS 2022). – Jakarta, 2022. – P. 50.
59. Yousuf M.U., Al-Bahadly I., Avci E. Wind speed prediction for small sample dataset using hybrid first-order accumulated generating operation-based double exponential smoothing model // Energy Science & Engineering. – 2022. – Vol. 10, №3. – P. 726-739.
60. Sabarina A.M., Rustamaji H.C., Himawan H. Prediction of Drug Sales Using Methods Forecasting Double Exponential Smoothing (Case Study: Hospital Pharmacy of Condong Catur) // Telematika: Jurnal Informatika dan Teknologi Informasi. – 2021. – Vol. 18, №1. – P. 106-117.
61. Haixiang G. et al. Learning from class-imbalanced data: Review of methods and applications // Expert systems with applications. – 2017. – Vol. 73. – P. 220-239.
62. Beyan C., Fisher R. Classifying imbalanced data sets using similarity based hierarchical decomposition // Pattern Recognition. – 2015. – Vol. 48. – P. 1653-1672.
63. Khalilia M., Chakraborty S., Popescu M. Predicting disease risks from highly imbalanced data using random forest // BMC medical informatics and decision making. – 2011. – Vol. 11. – P. 1-13.
64. Liu Y., An A., Huang X. Boosting prediction accuracy on imbalanced datasets with SVM ensembles // Procced. Advances in Knowledge Discovery and Data Mining: 10th Pacific-Asia conf. (PAKDD). – Singapore, 2006. – P. 107-118.
65. Lingden P. et al. A novel modified undersampling (MUS) technique for software defect prediction // Computational Intelligence. – 2019. – Vol. 35, №4. – P. 1003-1020.
66. Weiss G. M., Provost F. The effect of class distribution on classifier learning: an empirical study // <https://www.researchgate.net/publication/2364670>. 11.11.2024.
67. Japkowicz N. Why Question Machine Learning Evaluation Methods // Why Question Machine Learning Evaluation Methods? (An illustrative review of the shortcomings of current methods) // https://aaai.org/papers/WS06-06-003-why. 11.10.2023.
68. Branco P. et al. A survey of predictive modeling on imbalanced domains // ACM computing surveys (CSUR). – 2016. – Vol. 49, №2. – P. 1-50.
69. Loyola-González O. et al. Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases // Neurocomputing. – 2016. – Vol. 175. – P. 935-947.
70. Davis J., Goadrich M. The relationship between Precision-Recall and ROC curves // Proceed. of the 23rd internat. conf. on Machine learning. – NY., 2006. – P. 233-240.
71. Weiss G. M., Hirsh H. Learning to Predict Rare Events in Event Sequences // KDD. – 1998. – Vol. 98. – P. 359-363.
72. Zhang S. et al. Syslog processing for switch failure diagnosis and prediction in datacenter networks // Proccedl. IEEE/ACM 25th internat. sympos. on Quality of Service (IWQoS). – Vilanova i la Geltrú, 2017. – P. 1-10.
73. Shah V.H. Machine learning techniques for stock prediction // Foundations of Machine Learning| Spring. – 2007. – Vol. 1, №1. – P. 6-12.
74. Shen S., Jiang H., Zhang T. Stock Market Forecasting Using Machine Learning Algorithms // <https://pdf4pro.com/view/stock-market-forecasting>. 10.11.2023.
75. Suchatpong T., Bhumkittipich K. Hard Disk Drive failure mode prediction based on industrial standard using decision tree learning // Procced. 2014 11th internat. conf. on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). – Nakhon Ratchasima, 2014. – P. 1-4.
76. Jhaveri S. et al. Success prediction using random forest, catboost, xgboost and adaboost for kickstarter campaigns // Proccae. 3rd internat. conf. on Computing Methodologies and Communication (ICCMC). – Erode, 2019. – P. 1170-1173.
77. Cai Z. et al. Modelling of failure prediction Bayesian network based on fault tree analysis // Procced. 2010 IEEE 17th internat. conf. on Industrial Engineering and Engineering Management. – Xiamen, 2010. – P. 937-941.
78. Yin J., Dong F., Wang N. Modified gram-schmidt algorithm for extreme learning machine // Procced. 2009 2nd internat. sympos. on Computational Intelligence and Design (IEEE). – Changsha, 2009. – P. 517-520.
79. Zhang G. A modified svm classifier based on rs in medical disease prediction // Procced. 2009 2nd internat. sympos. on Computational Intelligence and Design (IEEE). – Changsha, 2009. – P. 144-147.
80. Ma Y., Zhai D. A modified multi-output support vector regression machine based on data dependent kernel function // Procced. 2009 internat. Joint conf.e on Computational Sciences and Optimization. – Sanya, 2009. – P. 939-941.
81. Chandrasekar P. et al. Improving the prediction accuracy of decision tree mining with data preprocessing // Procced. 2017 IEEE 41st annual computer software and applications conf. (COMPSAC). – Turin, 2017. – P. 481-484.
82. Liu W. et al. Empirical studies of a two-stage data preprocessing approach for software fault prediction // IEEE Transactions on Reliability. – 2015. – Vol. 65, №1. – P. 38-53.
83. Lo S. The effects of feature selection and model selection on the correctness of classification // Procced. 2010 IEEE internat. conf. on Industrial Engineering and Engineering Management. – Macao, 2010. – P. 989-993.
84. Pirbazari A.M., Chakravorty A., Rong C. Evaluating feature selection methods for short-term load forecasting // Procced. 2019 IEEE internat. conf. on Big Data and Smart Computing (BigComp). – Kyoto, 2019. – P. 1-8.
85. Das A., Mueller F., Rountree B. Aarohi: Making real-time node failure prediction feasible // Procced. 2020 IEEE internat. Parallel and Distributed Processing sympos. (IPDPS). – New Orleans, 2020. – P. 1092-1101.
86. Ghosh T. et al. Real time failure prediction of load balancers and firewalls // Prossed. IEEE internat. conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). – Chengdu, 2016. – P. 822-827.
87. Liang Y. et al. Failure prediction in ibm bluegene/l event logs // Procced. 7th IEEE internat. conf. on Data Mining (ICDM 2007). – Omaha, 2007. – P. 583-588.
88. Zheng Z. et al. A practical failure prediction with location and lead time for blue gene/p // Procced. internat. conf. on Dependable Systems and Networks Workshops (DSN-W). – Chicago, 2010. – P. 15-22.
89. Chawla N.V. et al. SMOTE: synthetic minority over-sampling technique // Journal of artificial intelligence research. – 2002. – Vol. 16. – P. 321-357.
90. Pears R., Finlay J., Connor A.M. Synthetic Minority over-sampling technique (SMOTE) for predicting software build outcomes // https://arxiv.org/abs/1407.2330. 10.11.2023.
91. Irrera I., Vieira M. A practical approach for generating failure data for assessing and comparing failure prediction algorithms // Procced. 2014 IEEE 20th Pacific Rim internat. sympos. on Dependable Computing. – Singapore, 2014. – P. 86-95.
92. Duraes J.A., Madeira H.S. Emulation of software faults: A field data study and a practical approach // Ieee transactions on software engineering. – 2006. – Vol. 32, №11. – P. 849-867.
93. Misra J. Distributed discrete-event simulation // ACM Computing Surveys (CSUR). – 1986. – Vol. 18, №1. – P. 39-65.
94. Fujimoto R.M. Parallel discrete event simulation // Communications of the ACM. – 1990. – Vol. 33, №10. – P. 30-53.
95. Franta W.R., Maly K. An efficient data structure for the simulation event set // Communications of the ACM. – 1977. – Vol. 20, №8. – P. 596-602.
96. Siebers P. O. et al. Discrete-event simulation is dead, long live agent-based simulation! //Journal of Simulation. – 2010. – Т. 4. – №. 3. – С. 204-210.
97. Koulinitch A.S., Sheremetov L.B. Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor // Expert Systems with Applications. – 1998. – Vol. 15, №3-4. – P. 295-307.
98. Arsene O., Dumitrache I., Mihu I. Expert system for medicine diagnosis using software agents //Expert Systems with Applications. – 2015. – Vol. 42, №4. – P. 1825-1834.
99. Macal C. M., North M. J. Tutorial on agent-based modeling and simulation // Proceed. of the Winter Simulation conf. – Orlamdo, 2005. – P. 2-15.
100. Allan R.J. et al. Survey of agent based modelling and simulation tools: technical report. – NY.: Science & Technology Facilities Council, 2010. – 48 p.
101. Gaber C. et al. Synthetic logs generator for fraud detection in mobile transfer services // Procced. 2013 internat. conf. on Collaboration Technologies and Systems (CTS). – San Diego, 2013. – P. 174-179.
102. Kavak H. et al. Big data, agents, and machine learning: towards a data-driven agent-based modeling approach // Proceed. of the Annual Simulation sympos. – Baltimore, 2018. – P. 1-12.
103. Ashraf A.W.U., Budka M., Musial K. Simulation and augmentation of social networks for building deep learning models // <https://paperswithcode.com.> 10.11.2023.
104. Augustijn E.W. et al. Machine learning to derive complex behaviour in agent-based modellzing // Procced. 2020 internat. conf. on Computer Science and Software Engineering (CSASE). – Duhok (Iraq), 2020. – P. 284-289.
105. Fuller D.B., de Arruda E.F. et al. Learning-agent-based simulation for queue network systems // Journal of the Operational Research Society. – 2020. – Vol. 71, №11. – P. 1723-1739.
106. Lundin E., Kvarnström H., Jonsson E. A synthetic fraud data generation methodology // Information and Communications Security: 4th internat. conf. (ICICS 2002). – Singapore, 2002. – P. 265-277.
107. Pereda M., Santos J.I., Galán J.M. A brief introduction to the use of machine learning techniques in the analysis of agent-based models. – Cham: Springer, 2017. – P. 179-186.
108. Batata O., Augusto V., Xie X. Mixed machine learning and agent-based simulation for respite care evaluation // Procced. 2018 Winter Simulation conf. (WSC). – Gothenburg, 2018. – P. 2668-2679.
109. Pyle D. Data preparation for data Mining. – San Diego: Morgan Kaufmann, 1999. – 540 p.
110. Salfner F. Predicting failures with hidden Markov models // Proceed. of 5th European Dependable Computing Conference (EDCC-5). – Budapest, 2005. – P. 41-46.
111. Watanabe Y. et al. Online failure prediction in cloud datacenters by real-time message pattern learning // Procced. 4th IEEE internat. conf. on cloud computing technology and science proceedings. – Taipei, 2012. – P. 504-511.
112. Baldoni R. et al. On-line failure prediction in safety-critical systems // Future Generation Computer Systems. – 2015. – Vol. 45. – P. 123-132.
113. Zhang S. et al. Prefix: Switch failure prediction in datacenter networks // Proceedings of the ACM on Measurement and Analysis of Computing Systems. – 2018. – Vol. 2, №1. – P. 1-29.
114. Matthews B.W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme // Biochimica et Biophysica Acta (BBA)-Protein Structure. – 1975. – Vol 405, №2. – P. 442-451.
115. Bergstra J., Bengio Y. Random search for hyper-parameter optimization // Journal of machine learning research. – 2012. – Vol. 13, №2. – P. 281-305.
116. He H. et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning // Procced. 2008 IEEE internat. joint conf. on neural networks (IEEE world congr. on computat. intelligence). – Hong Kong, 2008. – P. 1322-1328.
117. El Emam K., Mosquera L., Hoptroff R. Practical synthetic data generation: balancing privacy and the broad availability of data. – Sebastopol*:* O'Reilly Media, 2020. – 175 p.
118. Odersky M. et al. The Scala language specification. Version 2.8. – Lausanne, 2013. – 191 p.
119. Odersky M., Spoon L., Venners B. Programming in scala. – Mountain View: Artima Inc, 2008. – 736 p.
120. Zaharia M. et al. Spark: Cluster computing with working sets // [https://shortscience.org/paper?bibtexKey=conf/hotcloud/ZahariaCFSS10&](https://shortscience.org/paper?bibtexKey=conf/hotcloud/ZahariaCFSS10&a). 10.11.2023.
121. Armbrust M. et al. Spark sql: Relational data processing in spark // Proceed. of the 2015 ACM SIGMOD international conference on management of data. – Melbourne, 2015. – P. 1383-1394.
122. Armbrust M. et al. Structured streaming: A declarative api for real-time applications in apache spark // Proceed. of the 2018 internat. conf. on Management of Data. – Houston, 2018. – P. 601-613.
123. Meng X. et al. Mllib: Machine learning in apache spark // The Journal of machine learning research. – 2016. – Vol. 17, №1. – P. 1235-1241.
124. Reback J. et al. pandas-dev/pandas: Pandas 1.0. 5 // https://zenodo.org/records/3898987. 10.11.2023.
125. McKinney W. et al. Data structures for statistical computing in python // Proceedings of the 9th Python in Science Conf. – 2010. – Vol. 445, №1. – P. 51-56.
126. Pedregosa F. et al. Scikit-learn: Machine learning in Python // The Journal of machine Learning research. – 2011. – Vol. 12. – P. 2825-2830.
127. Wold S., Esbensen K., Geladi P. Principal component analysis // Chemometrics and intelligent laboratory systems. – 1987. – Vo. 2, №1-3. – P. 37-52.
128. Blei D.M., Ng A.Y., Jordan M.I. Latent dirichlet allocation // Journal of machine Learning research. – 2003. – Vol. 3. – P. 993-1022.
129. Shvachko K. et al. The hadoop distributed file system // 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). – Incline Village, 2010. – P. 1-10.
130. Subramanian J., Simon R. Overfitting in prediction models–is it a problem only in high dimensions? // Contemporary clinical trials. – 2013. – Vol. 36, №2. – P. 636-641.
131. Rodriguez J.D., Perez A., Lozano J.A. Sensitivity analysis of k-fold cross validation in prediction error estimation // IEEE transactions on pattern analysis and machine intelligence. – 2009. – Vol. 32, №3. – P. 569-575.
132. Nelder J.A., Wedderburn R.W.M. Generalized linear models // Journal of the Royal Statistical Society Series A: Statistics in Society. – 1972. – Vol. 135, №3. – P. 370-384.
133. Alpaydin E. Introduction to machine learning. – Cambridge: MIT press, 2020. – 712 p.
134. Boser B.E., Guyon I.M., Vapnik V.N. A training algorithm for optimal margin classifiers // Proceed. of the 5th annual workshop on Computational learning theory. – NY., 1992. – С. 144-152.
135. Arar Ö. F., Ayan K. A feature dependent Naive Bayes approach and its application to the software defect prediction problem // Applied Soft Computing. – 2017. – Vol. 59. – P. 197-209.
136. Feng X. et al. Prediction of slope stability using naive Bayes classifier // KSCE Journal of Civil Engineering. – 2018. – Vol. 22. – P. 941-950.
137. Granik M., Mesyura V. Fake news detection using naive Bayes classifier // Procced. 2017 IEEE first Ukraine conf. on electrical and computer engineering (UKRCON). – Kyiv, 2017. – P. 900-903.
138. Breiman L. et al. Classification and regression trees. – Routledge, 2017. – London, 368 p.
139. Breiman L., Cutler R.A. Random Forests // Machine learning. – 2001. – Vol. 45. – P. 5-32.
140. Friedman J.H. Stochastic gradient boosting // Computational statistics & data analysis. – 2002. – Vol. 38, №4. – P. 367-378.
141. Schapire R.E. The strength of weak learnability // Machine learning. – 1990. – Vol. 5. – P. 197-227.
142. Węglarczyk S. Kernel density estimation and its application // ITM web of conferences. – 2018. – Vol. 23, №2. – Р. 00037-1-00037-8.
143. Nwana H.S. Software agents: An overview // The knowledge engineering review. – 1996. – Vol. 11, №3. – P. 205-244.
144. Hewitt C., Bishop P., Steiger R. A universal modular actor formalism for artificial intelligence // Proceed. of the 3rd internat. joint conf. on Artificial intelligence. – Stanford, 1973. – P. 235-245.

**ПРИЛОЖЕНИЕ A**

Код процесса преобразования и дальнейшей работы алгоритмов:

*Модель на основе SVM*

“import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report

from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Load the dataset

ds = pd.read\_excel('/mnt/data/New\_data\_1.xlsx')

# Convert 'Breaking' to numerical (binary) values

ds['Breaking'] = ds['Breaking'].map({'Yes': 1, 'No': 0})

Y = ds['Breaking']

# Drop the specified columns from the dataset

columns\_to\_drop = ['System uptime', 'Downtime', 'Available Memory', 'CPU Load Index', 'Alarms', 'CPU 1']

X = ds.drop(columns=['Breaking'] + columns\_to\_drop)

# Data Splitting

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, random\_state=42, test\_size=0.3)

# Model Pipeline

clf = make\_pipeline(StandardScaler(), SVC(gamma='scale', probability=True))

# Model Training

clf.fit(X\_train, Y\_train)

# Model Prediction

clf\_pred = clf.predict(X\_test)

# Model Evaluation

clf\_accur = accuracy\_score(Y\_test, clf\_pred)

print('-------------------------------------')

print(f'Accuracy CLF = {clf\_accur}')

# Cross-Validation (5-Fold)

cross\_val\_scores = cross\_val\_score(clf, X\_train, Y\_train, cv=5)

print(f'Cross-Validation Scores = {cross\_val\_scores}')

print(f'Mean Cross-Validation Score = {np.mean(cross\_val\_scores)}')

print(classification\_report(Y\_test, clf\_pred))

print('-------------------------------------')

# Function to apply DES

def apply\_des(series, alpha=0.5, beta=0.5):

model = ExponentialSmoothing(series, trend='add', seasonal=None)

fit = model.fit(smoothing\_level=alpha, smoothing\_slope=beta)

forecast = fit.forecast(1)

return forecast[0]

# Function to classify new data

def classify\_new\_data(new\_data, model, des\_features):

# Apply DES to each feature in new\_data

des\_values = {}

for feature in des\_features:

des\_values[f'{feature}\_DES'] = apply\_des(new\_data[feature])

# Create DataFrame for new DES values

des\_df = pd.DataFrame([des\_values])

# Standardize and classify using the trained model

des\_df\_scaled = model.named\_steps['standardscaler'].transform(des\_df)

prediction = model.named\_steps['svc'].predict(des\_df\_scaled)

return prediction

# Convert new\_data to DataFrame

new\_data\_df = pd.DataFrame(new\_data)

# Define the features to apply DES

des\_features = ['System uptime', 'Downtime', 'CPU Load', 'CPU 1',

'Available Memory 1 (Processor)', 'Percent Available Memory 1 (Processor)',

'Available Memory 2', 'Percent Available Memory',

'Response Time Index', 'Traffic Index', 'Alarms', 'Temperature']

# Classify new data

predicted\_state = classify\_new\_data(new\_data\_df, clf, des\_features)

print(f'Predicted state: {"Breaking" if predicted\_state[0] == 1 else "Not Breaking"}')

*Модель на основе деревьев решений (DT), логистической регрессии и случайных деревьев (RF):*

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report

from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Load the dataset

ds = pd.read\_excel('/mnt/data/New\_data\_1.xlsx')

# Convert 'Breaking' to numerical (binary) values

ds['Breaking'] = ds['Breaking'].map({'Yes': 1, 'No': 0})

Y = ds['Breaking']

# Drop the specified columns from the dataset

columns\_to\_drop = ['System uptime', 'Downtime', 'Available Memory', 'CPU Load Index', 'Alarms', 'CPU 1']

X = ds.drop(columns=['Breaking'] + columns\_to\_drop)

# Data Splitting

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, random\_state=42, test\_size=0.3)

# Define models

models = {

"Decision Tree": DecisionTreeClassifier(),

"Logistic Regression": LogisticRegression(max\_iter=1000),

"Random Forest": RandomForestClassifier(n\_estimators=100)

}

for model\_name, model in models.items():

# Model Pipeline

clf = make\_pipeline(StandardScaler(), model)

# Model Training

clf.fit(X\_train, Y\_train)

# Model Prediction

clf\_pred = clf.predict(X\_test)

# Model Evaluation

clf\_accur = accuracy\_score(Y\_test, clf\_pred)

print(f'--- {model\_name} ---')

print(f'Accuracy = {clf\_accur}')

# Cross-Validation (5-Fold)

cross\_val\_scores = cross\_val\_score(clf, X\_train, Y\_train, cv=5)

print(f'Cross-Validation Scores = {cross\_val\_scores}')

print(f'Mean Cross-Validation Score = {np.mean(cross\_val\_scores)}')

print(classification\_report(Y\_test, clf\_pred))

print('-------------------------------------')

# Function to apply DES

def apply\_des(series, alpha=0.5, beta=0.5):

model = ExponentialSmoothing(series, trend='add', seasonal=None)

fit = model.fit(smoothing\_level=alpha, smoothing\_slope=beta)

forecast = fit.forecast(1)

return forecast[0]

# Function to classify new data

def classify\_new\_data(new\_data, model, des\_features):

# Apply DES to each feature in new\_data

des\_values = {}

for feature in des\_features:

des\_values[f'{feature}\_DES'] = apply\_des(new\_data[feature])

# Create DataFrame for new DES values

des\_df = pd.DataFrame([des\_values])

# Standardize and classify using the trained model

des\_df\_scaled = model.named\_steps['standardscaler'].transform(des\_df)

prediction = model.named\_steps[type(model.named\_steps['standardscaler']).\_\_name\_\_.lower()].predict(des\_df\_scaled)

return prediction

# Convert new\_data to DataFrame

new\_data\_df = pd.DataFrame(new\_data)

# Define the features to apply DES

des\_features = ['Percent Available Memory 1 (Processor)', 'Available Memory 2', 'Percent Available Memory',

'Response Time Index', 'Traffic Index', 'Temperature']

# Classify new data with each model

for model\_name, model in models.items():

print(f'--- {model\_name} ---')

predicted\_state = classify\_new\_data(new\_data\_df, clf, des\_features)

print(f'Predicted state: {"Breaking" if predicted\_state[0] == 1 else "Not Breaking"}')

Таблица данных рисунка А.1

|  |  |
| --- | --- |
| Года | Количество публикации |
| 1980-84 | 249 |
| 1985-89 | 1570 |
| 1990-94 | 5110 |
| 1995-99 | 13000 |
| 2000-04 | 29400 |
| 2005-09 | 73600 |
| 2010-14 | 237000 |
| 2015-20 | 232000 |