Министерство науки и высшего образования Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт

Высшая школа теоретической механики и математической физики

Работа допущена к защите

Директор ВШТМиМФ, д.ф.-м.н., чл.-корр. РАН

_____ А. М. Кривцов

«___»_____2022г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Многокритериальный поиск эффективных управленческих решений для повышения выживаемости и ускорения роста компаний в изменяющихся рыночных условиях методами анализа данных и машинного обучения

по направлению (специальности)

01.04.03 Механика и математическое моделирование

Направление (профиль)

01.04.03_03 Механика и цифровое производство

Выполнил

студент гр. 5040103/00301 А. В. Минина

Руководитель

к.ф.-м.н., доцент А. А. Лукашин

Консультант С. Кортманн

Санкт-Петербург

2022

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
Физико-механический институт
Высшая школа теоретической механики и математической физики

УТВЕРЖДАЮ
Директор ВШТМиМФ, д.ф.-м.н., чл.-корр. РАН

А.М. Кривцов
« » 2022 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы

студенту Мининой Анне Валерьевне, группы 5040103/00301
фамилия, имя, отчество (при наличии), номер группы

1. Тема работы: методами машинного обучения создать модель, позволяющую выделять наиболее ценные атрибуты управленческих решений, приводящих к повышению выживаемости и ускорению роста компаний в условиях изменяющихся внешних факторов на основе анализа данных о деятельности компаний и альянсов, исследовать работу модели на реальных данных.

2. Срок сдачи студентом законченной работы: 01.06.2022.

3. Исходные данные по работе: научные статьи по теме работы, данные из Bureau van Dijk, которые отражают характеристики деятельности компаний на мировом рынке, данные по альянсам, которые отражают совокупные характеристики альянсов из нескольких компаний, учет относительного влияния множества независимых внешних переменных, применение методов машинного обучения для нахождения наилучших рабочих параметров модели.

4. Содержание работы (перечень подлежащих разработке вопросов):

Анализ существующих экономических бизнес-стратегий для поиска эффективных управленческих решений в изменяющихся рыночных условиях.

Анализ исходных данных, исследование влияния независимых внешних переменных на характеристики выживаемости и роста компаний и альянсов, выявление наиболее важных факторов.

Построение модели для алгоритма машинного обучения, осуществляющего выявление наиболее эффективных деревьев решений, используя соотношение данных по деятельности отдельных компаний и выживаемости альянсов.

Программная реализация алгоритма машинного обучения.

Исследование работы алгоритма на реальных данных.

5. Перечень графического материала (с указанием обязательных чертежей): не предусмотрено

6. Консультанты по работе: С. Кортманн

7. Дата выдачи задания 13.05.2022.

Руководители ВКР _____ А.А. Лукашин

(подпись) инициалы, фамилия

Задание принял к исполнению 13.05.2022.

(дата)

Студент _____ А. В. Минина

(подпись) инициалы, фамилия

СОДЕРЖАНИЕ

РЕФЕРАТ

На 142 с., 65 рисунков, 5 таблиц, 1 приложение

КЛЮЧЕВЫЕ СЛОВА: СЛУЧАЙНЫЙ ЛЕС, МОСЛ, МНОЖЕСТВЕННАЯ ЛИНЕЙНАЯ РЕГРЕССИЯ, МАШИННОЕ ОБУЧЕНИЕ, ДЕРЕВЬЯ РЕШЕНИЙ, МОДЕЛЬ, PYTHON, PANDAS

Тема выпускной квалификационной работы: «Многокритериальный поиск эффективных управленческих решений для повышения выживаемости и ускорения роста компаний в изменяющихся рыночных условиях методами анализа данных и машинного обучения».

Данная работа посвящена исследованию зависимости выживаемости альянсов от различных бизнес-факторов и построению алгоритма множественного оптимизированного случайного лесы для выявления этих зависимостей. Задачи, которые решались в ходе исследования:

1. Анализ существующих экономических бизнес-стратегий для поиска эффективных управленческих решений в изменяющихся рыночных условиях.
2. Анализ исходных данных, исследование влияния независимых внешних переменных на характеристики выживаемости и роста компаний и альянсов, выявление наиболее важных факторов.
3. Построение модели для алгоритма машинного обучения, осуществляющего выявление наиболее эффективных деревьев решений, используя соотношение данных по деятельности отдельных компаний и выживаемости альянсов.
4. Программная реализация алгоритма машинного обучения.
5. Исследование работы алгоритма на реальных данных.

Работа проведена на базе двух главных источников – датасета с информацией об альянсах и группе датасетов из Бюро ван Дайк. Был проведен анализ данных, показывающий наглядно влияние бизнес-стратегий на компании в разные временные промежутки. Анализ проводился методами множественной линейной регрессии и модификацией алгоритма случайного леса.

В результате был проанализирован рынок бизнес-стратегий, выделены ключевые слова и рассчитана модель взаимоотношений SIC-кодов. Построена модель множественной линейной регрессии и выявлены наилучшие параметры по p-значениям. Разработан алгоритм и модель множественного случайного леса. На основании проведенных исследований были выведены независимые переменные, оказывающие сильное влияние на выживаемость компаний. Также продемонстрировано, что МОСЛ показывает не только важные переменные, но и их взаимосвязи.

ABSTRACT

142 pages, 65 figures, 5 tables, 1 appendix.

KEY WORDS: RANDOM FOREST, MORF, MULTIPLE LINEAR REGRESSION, MACHINE LEARNING, DECISION TREES, MODEL, PYTHON, PANDAS

The subject of the graduate qualification work is « Multi-criteria search for effective management solutions to increase the survival and accelerate the growth of companies in changing market conditions by using data analysis and machine learning methods»

The given work is devoted to studying the dependence of the survival of alliances on various business factors and the construction of an algorithm for multiobjective optimized random forests to identify these dependencies. The research set the following goals:

1. Analysis of existing economic business strategies to find effective management solutions in changing market conditions.
2. Analysis of initial data, research of the influence of independent external variables on the characteristics of survival and growth of companies and alliances, identification of the most important factors.
3. Building a model for a machine learning algorithm that identifies the most effective decision trees using the ratio of data on the activities of individual companies and the survival of alliances.
4. Software implementation of the machine learning algorithm.
5. Research the work of the algorithm on real data.

The work was made with two main sources - a dataset with information about alliances and a group of datasets from the Bureau van Dijk. Data analysis was made, which clearly showed the impact of business strategies on companies in different time periods. The analysis was carried out using multiple linear regression methods and a modification of the random forest algorithm.

As a result, the market of business strategies was analyzed, keywords were identified, and a model of the relationship of SIC-codes was calculated. A model of multiple linear regression was built and the best parameters for p-values were identified. An algorithm and a model for a multiobjective optimized random forest have been developed. Based on the conducted research, independent variables were derived that have a strong impact on the survival of companies. It has also been demonstrated that MORF shows not only important variables, but also their relationships.

ИСПОЛЬЗУЕМЫЕ ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

SIC-коды - standard industrial classification code или код стандартной промышленной классификации

НИОКР – научно-исследовательские и опытно-конструкторские работы

ИИ – искусственный интеллект

NLP – natural language processing или обработка естественного языка

MSE – mean squared error или среднеквадратическая ошибка

IQR – interquartile range или межквартильный диапазон

MAE – mean absolute error или средняя абсолютная ошибка

IoT – internet of things или интернет вещей

МОСЛ – множественный оптимизированный случайный лес

БвД – Бюро ван Дайк

Введение

Современный мир маркетинга и бизнеса развивается с невероятной скоростью. Каждый день небольшие компании и компании-конгломераты объединяются друг с другом в альянсы или партнерства для достижения поставленных целей и задач. Однако, при объединении нужно учитывать много факторов, начиная с бизнес-стратегий каждого и заканчивая экономическими подходами. Очень важно понимать, что именно в наибольшей степени влияет на выживаемость партнерств и альянсов, чтобы с самого «старта» обратить внимание именно на эти аспекты. «Ручная» аналитика бизнес-процессов в компаниях является долгим и дорогостоящим процессом, требующим очень большой концентрации и глубоких знаний в разных областях.

С целью уменьшения трудовых и финансовых затрат, а также изучения современного бизнес-рынка и экономического влияния была сформулирована задача по созданию программы, которая позволит выделить наиболее важные признаки, влияющие на выживаемость компаний методами машинного обучения. Были предоставлены данные по 120 альянсам, состоящим из компаний или ИП различного уровня. Каждый альянс содержит в себе следующую информацию:

- Компании и ИП, которые входят в состав альянса;
- Страны, в которых альянсы ведут свою деятельность;
- Год основания альянса;
- Описание деятельности альянса;
- SIC-коды каждой компании, присутствующей в альянсе;
- SIC-коды целого альянса.

Также для анализа были взяты данные из базы Бюро ван Дайк.

Целью данной работы является создать модель, позволяющую выделять наиболее ценные атрибуты управленческих решений, приводящих к

повышению выживаемости и ускорению роста компаний в условиях изменяющихся внешних факторов на основе анализа данных о деятельности компаний и альянсов методами машинного обучения, а также исследовать работу модели на реальных данных.

Для достижения поставленной цели были решены следующие промежуточные задачи:

- Анализ текстовых данных методами обработки естественного языка;
- Работа с различными инструментами визуализации;
- Разработка модели для расчета зависимостей SIC-кодов;
- Разработка программного кода для объединения разных наборов данных;
- Выбор алгоритма машинного обучения для первоначального анализа;
- Доработка существующего алгоритма машинного обучения, опираясь на цель работы и исходные данные;
- Создание моделей машинного обучения и оценка их работы;
- Анализ итоговых результатов и сравнение двух алгоритмов.

В первой главе работы рассматриваются актуальные для анализа бизнес-стратегии и их особенности.

Во второй главе обозреваются методы анализа и визуализации данных, изученных и использованных в данной работе, а также сравнение этих методов с аналогами. Также идет разбор нескольких методов машинного обучения и все аспекты, необходимые для построения модели.

В третьей главе реализован разведочный анализ данных, построение моделей машинного обучения, работа данных моделей на пред обработанных данных и обзор результатов предсказания моделей.

ГЛАВА 1. Экономические бизнес-стратегии и их особенности

1.1.Вертикальная интеграция

В мире бизнес-стратегий вертикальная интеграция является популярным, сложным, дорогим и рискованным методом. Основная цель – это рост бизнеса, прибыли и укрепление позиций на рынке. Предполагается расширение компании путем включения в состав бизнес-единиц, оперирующих на различных этапах технологической цепочки.

Причины для внедрения вертикальной интеграции могут быть следующие:

- Наблюдается «провал» вертикального рынка, то есть рынок слишком рискованный и ненадежный;
- Основная рыночная власть не у компании «головы», а у компаний из «смежных звеньев»;
- Высокие входные барьеры и ценовая дискриминация в разных рыночных сегментах даст компании рыночную власть;
- Развитие рынка еще продолжается, и компания нуждается в «интегрировании вперед», или рынок находится в упадке, и производственные звенья терпят потери [40].

Данный вид интеграции помогает компаниям предотвратить банкротство, повысить эффективность производства и т. д. Для каждой отрасли задачи будут отличаться, но процесс останется неизменным. Вертикальная интеграция решает следующие задачи:

- Защита от конкурентов;
- Увеличение сбыта и прибыли;
- Усиление влияния компании на рынке;
- Сокращение размеров производства;
- Контроль каналов сбыта;
- Уменьшение транзакционных задержек [47].

В более узком понятии стратегия вертикальной интеграции реализуется в двух направлениях: обратная интеграция и прямая интеграция. Более подробно данные методы отображены на рисунке 1.

Рис. 1. Прямая и обратная вертикальная интеграция [44]

Однако, данный подход имеет свои отрицательные и положительные стороны. Так как вертикальная интеграция подразумевает объединение последовательных производственных процессов, то можно выделить следующие положительные стороны:

- Увеличение доходности;
- Укрепление репутации;
- Увеличение качества производимых товаров;
- Сокращение расходов предприятия;
- Повышение объема продаж;

- Конкурентоспособность.

К недостаткам можно отнести следующие пункты:

- Увеличение постоянных расходов;
- Снижение производственной гибкости;
- Затраты на юридический контроль производства;
- Уменьшение восприимчивости к покупательскому спросу [47].

1.2.Горизонтальная интеграция

Объединение фирм, чьи продукты и уровень деятельности одинаков – это подход горизонтальной интеграции. Однако, компании могут варьироваться в различных рыночных направлениях. Объединение компаний разного «калибра» делает фирму гораздо более конкурентноспособной и дает перспективы существенного увеличения дохода. Главными предпосылками для выбора метода горизонтальной интеграции могут служить:

- Быстрый рост в отрасли производства компании;
- Усиление основных конкурентных преимуществ;
- Избыток трудовых и финансовых ресурсов (это позволит управлять расширившейся компанией);
- Необходимость устранить товар, который является близким заменителем;
- У конкурента, являющегося целью покупки – значительный дефицит финансовых ресурсов [20].

Данный вид интеграции успешно снижает конкуренцию между фирмами на рынке. Это происходит, потому что при объединении продуктов производители могут создать монополию. Однако, стоит отметить, что может произойти и обратный эффект – олигополия, если на рынке все еще есть

достаточно независимых производителей. Путь развития горизонтальный интеграции достаточно прост и отображен на рисунке 2.

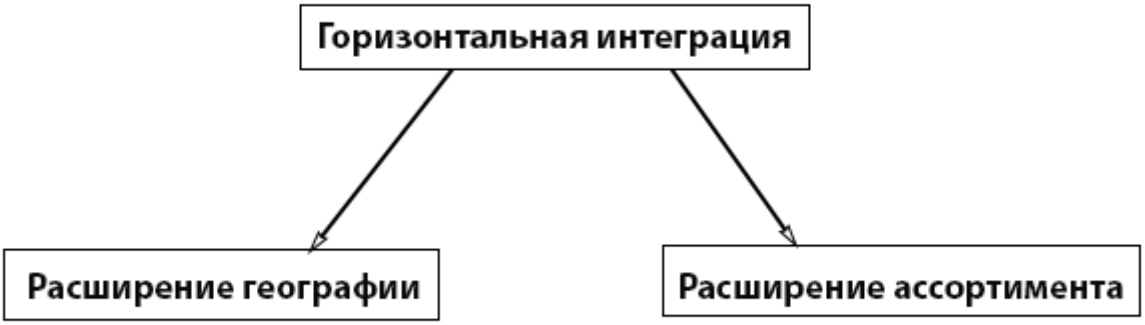

Рис. 2. Принцип горизонтальной интеграции [46]

К положительному влиянию горизонтальной интеграции на судьбу компании можно отнести:

- Увеличение доли компании на рынке;
- Выгода от увеличения клиентской базы;
- Увеличение дохода компании;
- Выгода от снижения конкуренции в своей области [45].

Несмотря на большое количество положительных сторон, у горизонтальной интеграции есть недостатки для рынка, особенно если интеграция прошла успешно:

- Существенное усложнение общего корпоративного управления;
- Трудности с получением синергетического эффекта – обеспечение экономии общекорпоративных издержек, оптимизация закупок, маневренность финансовых потоков;
- Проблема «продуктового каннибализма»;
- Затруднение распространения и продаж разных продуктов на различных рыночных сегментах;
- Тщательная проверка со стороны государственных органов.

1.3.Открытые бизнес-модели

По мере того, как открытые бизнес-модели, в которых знания, компетенции и ресурсы внешних фирм интегрируются в «бизнес-модель» компании, компании с традиционными закрытыми бизнес-моделями стремятся определить, стоит ли им последовать примеру «открытых» компаний и когда. Понимая предшественников открытых бизнес-моделей, внутренние и внешние факторы которых поощряют или требуют перехода к открытости, компании могут принимать более обоснованные решение о том, следует ли изменять и адаптировать свои бизнес-модели.

Закрытые бизнес-модели все еще доминируют в мире бизнеса, но все большую и большую популярность приобретает возможность открытой коллаборации и обмена между партнерами с помощью бизнес-моделей.

Вопрос для руководителей корпораций заключается в том, когда предпринять шаги именно в этом направлении и как именно. Изучение фирм с открытой бизнес-моделью может помочь ответить на этот вопрос, выявив предшественников открытых бизнес-моделей: внешние и внутренние факторы, способствующие открытости. В ходе многочисленных тематических исследований были выявлены следующие факторы:

1) Несоответствие бизнес-модели.

   Это происходит, когда элементы бизнес-модели — ценностное предложение для клиентов, процессы и модель доходов — не согласованы. Швейцарская компания Buehler, лидер мирового рынка машин для пищевой промышленности, хотела конкурировать на развивающихся рынках, но ее опыта в области машин для пищевой промышленности было недостаточно; ей требовалась фирма-партнер, обладающая опытом в области питательных веществ и производственными ноу-хау, чтобы разработать бизнес-модель, которая работала на развивающихся рынках.

2) Необходимо создать и зафиксировать новую ценность.

У британского фармацевтического производителя Shire не было огромных ресурсов для исследований и разработок, необходимых для конкуренции с Big Pharma. Он по-прежнему мог идти в ногу со временем в отрасли, используя свои партнерские отношения для приобретения новых идей, ноу-хау и технологий. Результат: поток инновационных продуктов.

3) Предыдущий опыт сотрудничества.

   Это ключ к тому, чтобы открытые бизнес-модели работали. SAP — это одна из компаний, в которой тесное сотрудничество со своими партнерами, включая более 3000 реселлеров и 1700 сервисных партнеров, является стандартным.

4) Открытые шаблоны бизнес-моделей.

   Есть ли примеры в других отраслях, которым можно подражать? Набег Procter & Gamble на открытые бизнес-модели был вызван примерами таких крупных компаний, как Eli Lilly и IBM.

5) Конвергенция отрасли.

   Иногда границы между отраслями могут стать размытыми (либо в результате появления новых технологий, либо в результате выхода на рынок мощной корпорации). Автомобилестроение и высокие технологии объединились, когда BMW и высокотехнологичная компания Immersion объединились для создания революционной бортовой системы управления BMW iDrive.

Эти пять предпосылок предлагают рекомендации относительно того, когда и в какой степени следует вводить открытость. Однако первым шагом является понимание четырех уровней открытых бизнес-моделей. Бизнес-модели открытых НИОКР и открытых инноваций ограничены функцией НИОКР компании. Разница в том, что при открытом НИОКР открытость с партнерами играет второстепенную роль; с открытыми инновациями фирма сильно зависит от открытости [20].

1.4.Иерархические цепи

Чтобы разработать успешную стратегию цепочки поставок, фирма должна иметь эффективную структуру и методологию для разработки стратегии. Кроме того, эта структура и методология должны гарантировать, что фирма сможет выбрать те проекты, которые будут наилучшим образом поддерживать ее стратегию цепочки поставок.

Формулирование, внедрение и поддержание продуманной комплексной стратегии цепочки поставок — серьезная задача для фирм, конкурирующих в современной глобальной экономике. Более того, этот процесс планирования, определения приоритетов и реализации стратегий и проектов для обеспечения конкурентоспособности цепочки поставок никогда не заканчивается. Фирмы должны постоянно оценивать эффективность и действенность своей текущей цепочки поставок, определять следующие шаги, необходимые для удовлетворения будущих требований, и реализовывать конкретные проекты, которые наилучшим образом достигают их целей. Короче говоря, процесс разработки стратегии и планирования продолжается и должен иметь целью постоянное совершенствование.

Чтобы упростить процесс разработки и реализации планов цепочки поставок, фирма должна иметь эффективную основу для организации процессов планирования и распределения ресурсов.

Руководство по управлению цепочками поставок определило, что необходимо разработать стратегии цепочек поставок, сосредоточенные на шести ключевых областях:

- Потребитель;
- Клиенты;
- Правительство;
- Общественная ответственность;
- Финансовое благополучие;
- Поставщики и партнеры;

- Внутренние операции и сотрудники [32].

Существует четыре основных направления в мире иерархических цепей:

1) Ориентация на клиента.

   Эта стратегия цепочки поставок фокусируется в первую очередь на потребностях и предпочтениях клиентов. Чтобы обеспечить клиентоориентированность, владельцам бизнеса необходимо повысить точность прогнозирования спроса. Это позволит компаниям подготовить свои продукты и уровни запасов для удовлетворения прогнозируемых будущих потребностей. Предприятия могут определять тенденции спроса и прогнозировать продажи, используя программное обеспечение для управления запасами с возможностями прогнозирования. Эти инструменты записывают данные о продажах и запасах для предоставления отчетов в режиме реального времени. Используя этот путь, организации могут улучшить свои общие финансовые показатели и обеспечить превосходное обслуживание клиентов. Постоянный мониторинг спроса также позволит владельцам бизнеса улучшить сотрудничество со своими поставщиками и эффективно обрабатывать новые запасы.

2) Бизнес-прогнозирование

   Время от времени будут возникать небольшие сбои, такие как неисправность машины и задержка доставки из-за погодных условий или пробок. Предприятия могут предотвратить развитие этих проблем, быстро решая их с помощью новейших технологий прогнозирования. Например, компании обычно используют цифровых двойников для прогнозной оценки своих продуктов и оборудования. Цифровые двойники — это технологические решения, которые могут прогнозировать, когда могут возникнуть перебои в обслуживании машин, и могут заранее предупреждать

пользователей о любых возможных простоях. Эти инструменты обычно используют большие данные, искусственный интеллект (ИИ), Интернет вещей (IoT) и машинное обучение, чтобы получить представление о производительности производства и эксплуатации. Инвестируя в профилактическое обслуживание, предприятия могут получить полное представление обо всей своей работе, от оборудования, продуктов до активов. Они также могут получить представление об улучшениях продукта и о том, как минимизировать затраты, обеспечивая при этом положительное качество обслуживания клиентов.

3) Интеллектуальная автоматизация.

Предприятия из всех отраслей интегрировали автоматизацию в управление цепочками поставок. Это можно увидеть в виде автономных вилочных погрузчиков, дронов, доставляющих товары клиентам, и программных решений, упрощающих управление складскими помещениями. Производственные предприятия максимизируют потенциал своей цепочки поставок, используя интеллектуальную автоматизацию, которая представляет собой решение, использующее искусственный интеллект, роботизированные процессы и другие методы разработки программного обеспечения для автоматизации сложных бизнес-операций. Производители используют этот инструмент для массового производства настраиваемых продуктов, отвечающих требованиям потребителей. В частности, интеллектуальная автоматизация позволяет компаниям иметь гибкие производственные ячейки, которые могут создавать индивидуальные изделия вместо того, чтобы зависеть от непрерывного производства или сборочной линии, которая дублирует продукт. Это выгодно для предприятий, потому что они могут иметь конкурентное преимущество в предоставлении

инвентаря, который соответствует тому, что хотят клиенты. Следовательно, это повысит их продажи и доходы. Гибкость автоматизации также дает предприятиям возможность производить продукцию по запросу, а не создавать сразу большие партии. Это снизит затраты и защитит размер прибыли.

4) Общая видимость.

Чтобы правильно управлять цепочкой поставок, владельцы бизнеса должны иметь полное представление о каждой роли и процессе, которые с ней связаны. Руководителям необходимо не только следить за производством, перемещением и доставкой продукции, но и часто оценивать данные, связанные с продажами, и прогнозировать возможные сбои в цепочке поставок. Полная прозрачность цепочки поставок в режиме реального времени позволит руководству выявлять любые аномалии и быстро вносить коррективы в запасы или производственные процессы. Помимо повышения оперативности бизнеса, руководители смогут минимизировать риски и расходы. Для обеспечения видимости важно, чтобы управленческие команды тщательно общались и сотрудничали со всеми своими партнерами, включая складские помещения, дистрибьюторов и поставщиков. Компании могут внедрять новейшие технологии управления бизнесом, такие как программное обеспечение для управления запасами. Эти решения будут предоставлять отчеты в режиме реального времени об истории продаж и оповещения, которые будут информировать авторизованных пользователей об их операциях. [34]

1.5.Бизнес-экосистемы

Описать термин «экосистема» лучше всего, как структуру согласования многостороннего набора партнеров, которым необходимо

взаимодействовать для реализации основного ценностного предложения» [1]. Это предложение не может быть представлено одним независимым участником, поскольку оно опирается на набор дополнительных возможностей и ресурсов, обычно распределенных между несколькими участниками [3]. Таким образом, участники экосистемы в значительной степени зависят друг от друга, и успешные экосистемы требуют, чтобы их участники соблюдали баланс между совместным созданием ценности и конкурентным присвоением ценности [2]. Ключевой особенностью бизнес-экосистем является то, что ни один из участников не обладает всеми необходимыми дополнительными возможностями и/или ресурсами для самостоятельного предоставления ценностного предложения [4].

Достаточно необычным является процесс проектирования бизнес-экосистемы. Если проектирование стратегии традиционной бизнес-модели похоже на планирование и строительство дома, проектирование экосистемы больше похоже на разработку целого жилого района: более сложное, больше участников для координации, больше уровней взаимодействия и непредвиденных возникающих результатов.

Бизнес-экосистемы, как и жилые районы, нельзя полностью спланировать и спроектировать. Эта адаптивность, на самом деле, является одной из их основных сильных сторон. Тем не менее, есть некоторые ключевые варианты проектирования, которые нужно сделать правильно, чтобы увеличить шансы на успех. Эти варианты не являются независимыми; они должны согласовываться друг с другом и предлагать согласованную общую конфигурацию. Проблему проектирования бизнес-экосистемы можно решить, ответив на шесть последовательных вопросов:

- Какую проблему в компании нужно решить?
- Кто должен быть частью вашей экосистемы?
- Какой должна быть начальная модель управления вашей экосистемой?
- Как вы можете оценить ценность вашей экосистемы?

- Каким образом можно решить проблему «курицы и яйца» во время запуска стратегии?
- Как можно обеспечить эволюционируемость и долгосрочную жизнеспособность экосистемы [35]?

Бизнес-экосистемы имеют существенные отличия от стратегий, описанных в предыдущих подпунктах. Основными отличиями являются:

- Модульность.

  В бизнес-экосистемах различные компоненты предложения разрабатываются независимо, но функционируют как единый организм. Во многих случаях покупатель может выбрать один из компонентов или способов их сочетаний.
- Кастомизация.

  Участники экосистемы, как правило, адаптируются под нее и стремятся к взаимной совместимости.
- Принцип многосторонних отношений.

  Все игроки в экосистеме взаимосвязаны, их отношения сложно разложить на отдельные взаимодействия.
- Координация.

  В отличие от модели вертикальной интеграции или цепочек поставок, бизнес-экосистемы не полностью контролируются иерархически. Тем не менее существует особый механизм координации, например, через стандарты, правила или процессы — помимо рыночных механизмов [9].

Существует два основных типа бизнес-экосистем:

1) Экосистема решений - участники создают или предоставляют пользователю продукт за счет координации разных компаний.
2) Экосистема транзакций - связка участников и потребителей через общую (как правило, цифровую) платформу [7].

Разработка бизнес-экосистемы — это серьезное решение. Шесть шагов и лежащие в их основе взаимозависимые варианты проектирования,

несомненно помогают. Бизнес-экосистема, хорошо спроектированная таким образом, может создать целые новые отрасли или существенно изменить и преобразовать существующие отрасли. В то же время важно признать, что бизнес-экосистемы нельзя полностью спланировать. Проект экосистемы должен обеспечивать наличие основ и избегать стратегических ошибок, но он также должен оставлять место для творчества, случайных открытий и новых потребностей клиентов. Экосистемы, которые будут успешными в долгосрочной перспективе, должны быть адаптируемыми и готовыми изменять свою структуру в ожидании изменений на рынках, технологиях, правилах и общественных настроениях. Они также должны быть готовы принять случайность непреднамеренных и непредвиденных возможностей [35].

ГЛАВА 2. Аналитические модели и машинное обучение

2.1 Основные понятия

Машинное обучение — это отрасль искусственного интеллекта (ИИ) и компьютерных наук, которая фокусируется на использовании данных и алгоритмов для имитации того, как обычно обучаются люди, постепенно повышая точность обучения и полученных результатов [25]. Актуальность данной сферы растет с каждым днем. Основная цель машинного обучения — это метод решения проблемы, для поиска которой алгоритм ИИ анализирует свои обучающие данные. Как только алгоритм находит свою целевую функцию, эту функцию можно использовать для прогнозирования результатов.

В данной работе применяются различные методы машинного обучения, цель которых заключается в выведении общих зависимостей, исходя из набора частных наблюдений. В работе применяются такие методы машинного обучения, как:

- обработка естественного языка (NLP) для обнаружения закономерностей слов и обработки письменной, а иногда и устной человеческой речи;
- обучение с «с учителем» или контролируемое обучение, делящееся на две категории: регрессия и классификация. В данном исследовании используется метод регрессии;
- ансамблевые алгоритмы, такие как деревья решений и случайный лес.

Целью работы алгоритма машинного обучения является частично или полностью автоматизировать решение сложных аналитических задач на основе построенных причинно-следственных связей.

2.2 Обработка данных

Подготовка исходных данных и их обработка – это один из самых сложных шагов в проектах, связанных с машинным обучением и искусственным интеллектом. Качественная подготовка данных может привести к созданию более точных и эффективных алгоритмов, упрощая при этом переход к новым аналитическим задачам, адаптацию при изменении точности модели и экономию значительного времени и усилий ученых, занимающихся данными, и бизнес-пользователей в будущем.

На этапе обработки данных очень важно знание того, что конкретно нужно предсказать. Этот фактор помогает определить, какие переменные важны или нет с самого начала. Формулируя задачу, необходимо провести исследование данных и сделать вывод, к какой категории относится решаемая задача: классификации, кластеризации, регрессии или ранжирования.

- Классификация подразумевает собой бинарные ответы алгоритма в виде «да» и «нет». Также существует много классовая классификация, которая подразумевает несколько верных ответов. Очень важно заранее определить для алгоритма правильные ответы, чтобы он мог научиться распознавать нужную закономерность.
- Кластеризация – это метод машинного обучения, включающий в себя группировку по точкам данных. Основная задача такого алгоритма состоит в том, чтобы определить каждую целевую переменную в определенный «кластер» или группу.
- Регрессия является алгоритмом, который можно обучить для прогнозирования входных данных в действительные числа.
- Ранжирование — это класс алгоритмических методов, которые применяют контролируемое машинное обучение для решения проблем ранжирования в релевантности поиска. Другими словами, это то, что заказывается по результатам запроса.

2.2.1 Обработка текстовых значений

Текстовые символы, которыми общаются между собой люди, являются непригодными для построения моделей машинного обучения, так как в первоначальном виде ИИ не может распознать его. Обычный человеческий текст содержит в себе строки с пунктуационными символами, словами в разнообразных формах и падежах, а также союзами и артиклями. Из-за такого содержимого есть много отдельных текстовых единиц, но нет единой структуры. Таким образом, без предварительной обработки, выявить различные признаки из исходного текста не представляется возможным.

Машинное обучение для обработки естественного языка (NLP) и текстовой аналитики включает использование алгоритмов машинного обучения и «узкого» искусственного интеллекта (ИИ) для понимания смысла текстовых документов. Этими документами может быть практически все, что содержит текст: комментарии в социальных сетях, онлайн-обзоры, ответы на опросы, даже финансовые, медицинские, юридические и нормативные документы. По сути, роль машинного обучения и ИИ в обработке естественного языка и текстовой аналитике заключается в улучшении, ускорении и автоматизации основных функций текстовой аналитики и функций НЛП, которые превращают этот неструктурированный текст в полезные данные и идеи.

Текстовые данные требуют особого подхода в машинном обучении. Это связано с тем, что текстовые данные могут иметь сотни тысяч слов и фраз, но могут быть очень разреженными. Например, в русском языке около 300 000 общеупотребительных слов. Но любое сообщение содержит всего несколько десятков из них. Это отличается от чего-то наподобие видеоконтента, где очень высокая размерность, но есть куча и куча данных для работы, поэтому он не такой разреженный.

Самые популярные алгоритмы по работе с естественным языком:

- Метод опорных векторов;
- Байесовская сеть;
- Принцип максимума энтропии;
- Условные случайные поля;
- Нейронные сети/глубокое обучение [13].

Обычно, процесс обработки текстовых данных состоит из обязательных пунктов, необходимых для корректного распознавания данных.

2.2.1.1 Токенизация

Токенизация — это метод разбиения фрагмента текста на небольшие блоки, называемые токенами. Маркером может быть слово, часть слова или просто символы, такие как пунктуация. Таким образом, токенизацию можно условно разделить на три группы:

- Токенизация слов;
- Токенизация на под слова;
- Частичная токенизация слов (символы n-грамм).

Результатом токенизации текста является преобразование текстовой строки в список слов, из которого эта строка состояла. Токенизация текста включает в себя несколько этапов:

1. Приведение исходного текста к нижнему регистру
2. Замена пунктуационных символов пробелами
3. Объявление слов отдельными токенами

В русском и некоторых других языках можно определять окончание предложения, когда есть определенные пунктуационные знаки, например точка. Однако, точка не всегда может указывать на окончание предложения, так как она используется также и при сокращении слов. В таком случае, при обработке текста нужно учитывать таблицу сокращений, тогда шанс неверной расстановки границ приложений будет крайне мал. Существует большое

количество уже реализованных библиотек для данных операций, так что самостоятельно разрабатывать алгоритм не придется

2.2.1.2. Лемматизация и стемминг текста

Как правило, тексты содержат в себе однокоренные слова, разные грамматические формы, большое количество синонимов. Лемматизация и стемминг преследуют цель привести все встречающиеся словоформы к одной, нормальной словарной форме.

Лемматизацию и стемминг можно отнести к частному случаю нормализации, но оба понятия имеют разный смысл:

- Стемминг, безусловно, является более простым из двух подходов. С помощью основы слова сокращаются до своих словесных основ. Основа слова не обязательно должна быть тем же корнем, что и морфологический корень на основе словаря, это просто равная или меньшая форма слова.
- Лемматизация преобразует слово в его лемму, однако необходимо знать его часть речи. Это требует дополнительных вычислительных лингвистических мощностей, таких как тегирование частей речи. Это позволяет ему делать более качественные разрешения.

Поскольку стемминг обычно основан на эвристике, он далек от совершенства. На самом деле, он обычно страдает, в частности, от двух проблем: чрезмерного и недостаточного развития.

Чрезмерное развитие происходит, когда слишком много слов обрезается. Это может привести к бессмысленным основам, где все значение слова потеряно или запутано. Или это может привести к тому, что слова будут разрешены к одним и тем же основам, хотя, вероятно, этого не должно быть.

Недостаточное развитие — противоположная проблема. Это происходит, когда есть несколько слов, которые на самом деле являются формами друг друга. Было бы хорошо, если бы они все резолвились в один стебель, но, к сожалению, этого не происходит. Применение новых правил и эвристик может быстро выйти из-под контроля. Решение одной или двух проблем с чрезмерным или недостаточным стеммингом может привести к появлению еще двух. Создание хорошего алгоритма стемминга — тяжелая работа [24].

Примеры:

1. Слово «плохой» – это лемма для слова «хуже». Стеммер не увидит эту связь, так как слова не однокоренные и нужно сверяться с источниками.

2. Слово учиться – это базовая форма слова обучаться. В данном случае проблему решат и стемминг, и лемматизация.

2.2.1.3. Стоп-слова

Стоп-слова – это нежелательные слова или словосочетания, которые обычно убирают во время обработки текстовых данных. При обработке текста методами машинного обучения стоп-слова могут оказывать негативное влияние на выборку и исказить ее значения, потому от них необходимо избавляться, но делать это корректно.

Чаще всего, стоп-слова представляют из себя текстовые символы и словосочетания, которые не несут в себе смысловой нагрузки. Универсального списка стоп-слов не существует, наборы варьируются от случая к случаю.

В NLTK (библиотека на языке python, которая используется в данной работе) есть предустановленный список стоп-слов. Он содержит в себя наиболее часто употребляемые нерелевантные слова.

Всегда можно проанализировать и самостоятельно составить список конкретных стоп-слов, которые необходимо убрать конкретно из нужных нам данных [24].

2.2.1.4 Регулярные выражения

Регулярное выражение (regexp, regex) – это специально закодированные текстовые строки, используемые в качестве шаблонов для сопоставления наборов строк. Например:

- . – любой символ, кроме перевода строки;
- \w – один символ;
- \d – одна цифра;
- \s – один пробел;
- \W – один НЕсимвол;
- \D – одна НЕцифра;
- \S – один НЕпробел;
- [abc] – находит любой из указанных символов match any of a, b, or c;
- [^abc] – находит любой символ, кроме указанных;
- [a-g] – находит символ в промежутке от a до g.

Регулярные выражения играют жизненно важную роль в ваших повседневных задачах кодирования, а также являются сверхмощным инструментом, когда речь идет о очистке данных, интеллектуальном анализе данных и других операциях, которые слишком обширны для жесткого кода. Создается специальный шаблон регулярного выражения, который скользит по всему тексту, и нам возвращается местоположение или значение соответствующей части.

Сопоставление шаблонов регулярных выражений состоит из двух частей:

1. Правильный шаблон: нужно выяснить, что мы действительно хотим сопоставить.
2. Правильная функция: это связано с позицией. Хотим ли мы соответствовать шаблону в начале или где-либо еще. Если мы хотим разделить и заменить, мы также должны изменить нашу функцию re [39].

2.2.1.5 Синтаксический анализ текста

2.2.1.5.1 Мешок слов

Модель «мешка слов», или сокращенно BoW (Bag of Words), — это алгоритма, который превращает произвольный текст в векторы фиксированной длины, подсчитывая, сколько раз появляется каждое слово.

Этот процесс часто называют векторизацией. Этот подход простой и гибкий, и его можно использовать множеством способов для извлечения признаков из документов. Пример мешка слов показан на рисунке 3.

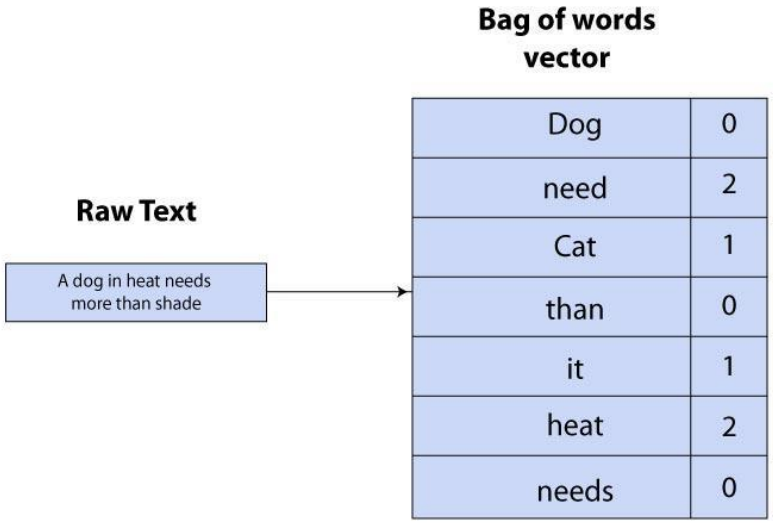

Рис. 3. Пример «мешка слов» [11]

Сам «мешок слов» представляет из себя текст, который содержит в себе количество слов в документе. Это включает в себя два пункта:

- Словарь известных слов;
- Мера присутствия известных слов.

Его называют «мешком» слов, так как любая информация о порядке или структуре слов в документе отбрасывается. Модель разбирает только то, встречаются ли нужные слова в документе, а где именно они встречаются в документе - нет [14].

2.2.1.5.2 N-граммы

В отличии от «мешка слов» N-граммы способны учитывать порядок слов, а это позволяет расширять признаковое пространство. По этой причине более простые модели могут помогать в поиске более сложных моделей и закономерностей, если сравнивать это с «мешком слов».

Существуют буквенные и словесные N-граммы. Словесные – это наборы из n-идущих токенов подряд. Условно, n-граммы можно разделить на следующие:

- Униграммы – наборы, состоящие из одного токена;
- Биграммы – наборы, состоящие из двух токенов, идущих подряд;
- Триграммы – наборы, состоящие из трех токенов, идущих подряд.

Если рассматривать данные методы на примере фразы «использование алгоритмов машинного обучения», то результат будет:

- Униграммы: использование, алгоритмов, машинного, обучения;
- Биграммы: использование алгоритмов, машинного обучения и т. д.

В зависимости от выбора параметра n можно получить как огромное число признаков, так и свести всё к тому, что каждый текст будет сам по себе

являться отдельным признаком (что будет означать подгонку под исходную выборку).

В буквенных n-граммах в качестве токенов рассматриваются буквы. Остальной процесс выделения униграмм, биграмм и т. д. аналогичен словесным биграммам. Также можно обратить внимание на расширенную версию n-грамм, а именно k-skip-n-граммы. Это наборы из n токенов, где между соседними токенами должно быть не более чем k токенов. В качестве примера можно рассмотреть 1-skip-2- граммы. Тогда для приведенной ранее фразы получим следующий результат: обработка языка, естественного методами, языка машинного, методами обучения [14].

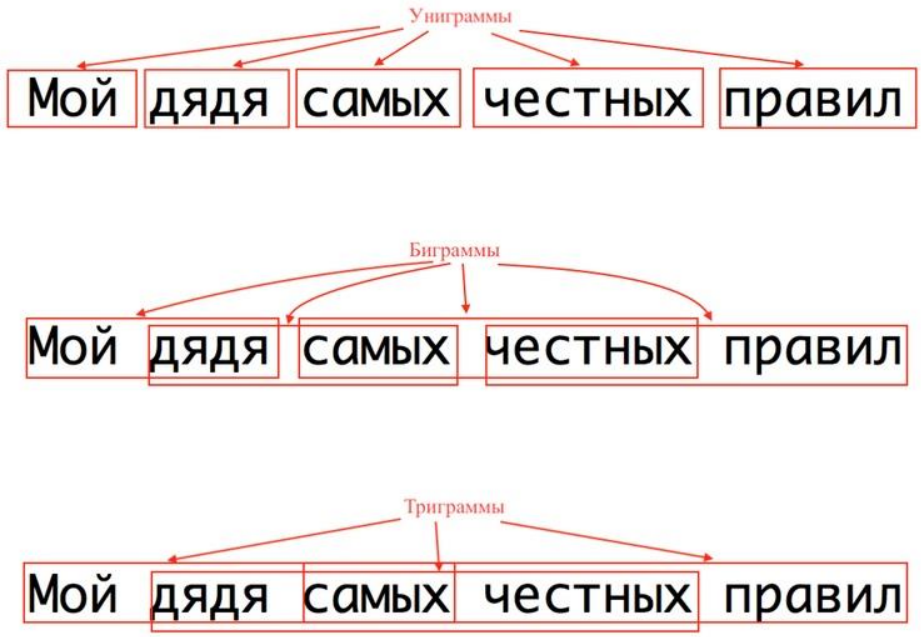

Рис. 4. Пример разных n-грамм [8]

2.2.2 Обработка выбросов

Обычно выброс — это что-то лишнее или то, что отличается от большинства. Некоторые статистики определяют выбросы как «имеющие основное поведение, отличное от остальных данных». В качестве альтернативы выброс — это точка данных, удаленная от других точек.

Крайне важно не путать это определение с определением несбалансированного набора данных, хотя в обоих определениях есть некоторое сходство. Для полноты стоит отметить, что несбалансированный набор данных с точки зрения машинного обучения — это когда одна метка класса имеет гораздо меньше выборок по сравнению с другой меткой класса.

На рисунке 5 видно, что точки выборки в зеленом цвете близки друг к другу, тогда как две точки выборки в красном находятся далеко друг от друга. Эти красные точки выборки являются выбросами.

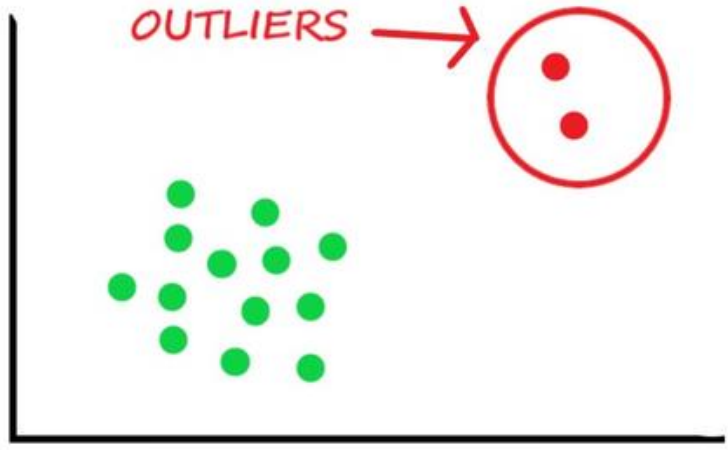

Рис. 5. Outliers или «выбросы» отмечены красным цветом

Выбросы делятся на три категории:

- Точечные или глобальные выбросы: наблюдения, аномальные по отношению к большинству наблюдений в объекте. Иначе говоря, точка данных считается глобальным выбросом, если ее значение выходит далеко за пределы всего набора данных, в котором она находится.
- Контекстные (условные) выбросы: наблюдения, считающиеся аномальными в конкретном контексте. Точка данных считается контекстуальным выбросом, если ее значение значительно отличается от остальных точек данных в том же контексте. Важно обратить внимание, что это означает, что одно и то же значение не может считаться выбросом, если оно встречается в другом контексте. Если мы ограничим наше обсуждение данными временных рядов, «контекст» почти всегда будет временным, потому что данные временных рядов представляют собой записи определенного количества во времени. Поэтому неудивительно, что контекстуальные выбросы распространены в данных временных рядов. Значения контекстной аномалии не выходят за пределы нормального глобального диапазона, но являются ненормальными по сравнению с сезонным паттерном.
- Коллективные выбросы: набор наблюдений, аномальных, но кажущихся близкими друг к другу, потому что все они имеют одинаковую аномальную ценность.
- Подмножество точек данных в наборе данных считается аномальным, если эти значения как совокупность значительно отклоняются от всего набора данных, но значения отдельных точек данных сами по себе не являются аномальными ни в контекстуальном, ни в глобальном смысле. В данных временных рядов это может проявляться в виде нормальных пиков и спадов, происходящих за пределами временных рамок, когда эта

сезонная последовательность является нормальной, или в виде комбинации временных рядов, которые находятся в состоянии выброса как группа.

Существуют различные методы для работы с выбросами, которые можно сгруппировать в два широких подхода, а именно методы на уровне алгоритма и на уровне данных.

Первый направлен на изменение алгоритма обучения, чтобы он справлялся с набором данных, и, как известно, это требует относительно высоких вычислительных затрат.

Последний не зависит от классификатора и относительно прост в применении, поскольку фокусируется на методах предварительной обработки данных. Например, чтобы справиться с выбросами, некоторые исследователи идентифицируют и удаляют их полностью, в то время как другие контролируют количество удаляемых выбросов [41].

Большинство алгоритмов машинного обучения преследует цель достичь минимального значения метрик. Чаще всего самым распространенным вариантом метрики ошибки является среднеквадратическая ошибка. Квадрат ошибки предсказания на выбросах может многократно превышать аналогичные значения для остальных данных, из-за чего алгоритм машинного обучения, стремясь уменьшить среднеквадратическую ошибку, будет подстраиваться под конкретные значения выбросов, игнорируя основной массив данных. Если обучение проходит на небольшом наборе данных, то описанная выше проблема наиболее характерна, так как при «выбросах» будет сильно искажена работа алгоритмов.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n}, \quad (1)$$

, где MSE - среднеквадратическая ошибка,

y_i – значение целевого признака,

$\widehat{y_i}$ – предсказанное значение целевого признака,

n – количество примеров.

Помимо вышеперечисленных техник для очищения данных от выбросов, существуют более простые подходы. Для распознавания (и последующей очистки) можно использовать:

- Визуализацию;
- Математический подход.

Когда речь идет о технике визуализации, это не означает обнаружение невооруженным глазом. Конечно, это возможно, но только в том случае, если выборочные данные немногочисленны и сильно отличаются друг от друга. Например, когда вам показывают изображения животных и нужно определить лишнего. Но реальные точки данных не такие, они собраны в миллионах, возможно, миллиардах записей.

Box Plot или блочная диаграмма — это графическое отображение для описания распределения данных. На блочных диаграммах используется медиана, а также нижняя и верхняя квартили. Выброс можно легко обнаружить с помощью диаграммы, где любая точка выше или ниже усов представляет собой выброс. Это также известно как «одномерный метод», так как здесь мы используем анализ выбросов одной переменной.

Выбросы также можно обнаружить с помощью гистограммы, где основная часть наблюдений находится с одной стороны, а несколько наблюдений появляются вдали от основной группы, они представляют собой выбросы.

Их также можно обнаружить с помощью точечной диаграммы, которая помогает понять степень связи между двумя числовыми переменными, и любое наблюдение, которое далеко от нормальной связи, является выбросом.

В данной работе используется именно метод визуализации и построения графиков. С помощью точечных диаграмм определяются аномальные точки, которые подразумеваются, как выбросы [41]. На рисунках 6, 7 и 8 изображены Box Plot, гистограмма и точечная диаграмма с выбросами.

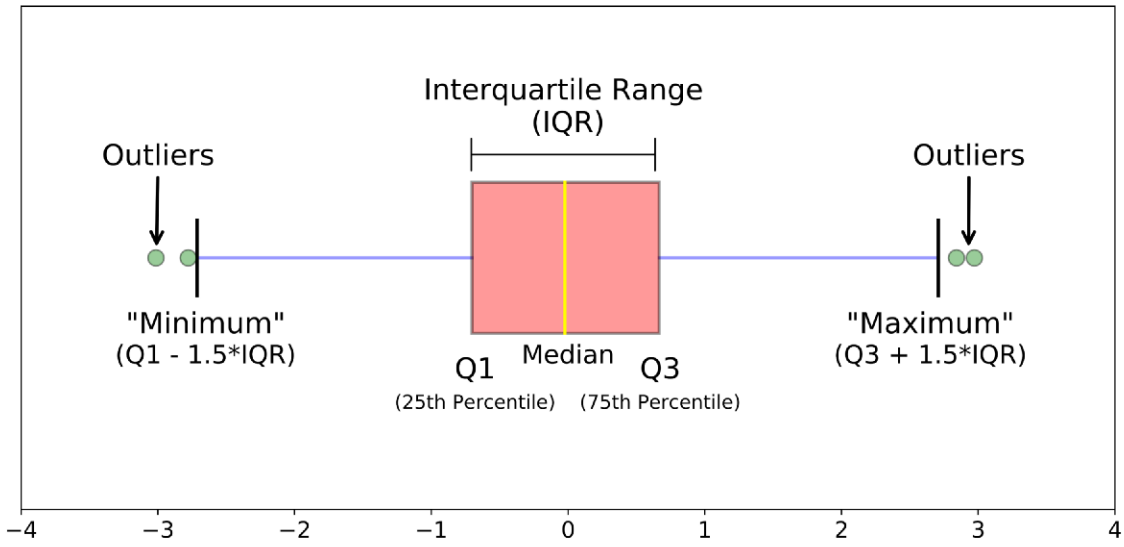

Рис. 6. Box Plot, где outliers являются выбросами [21]

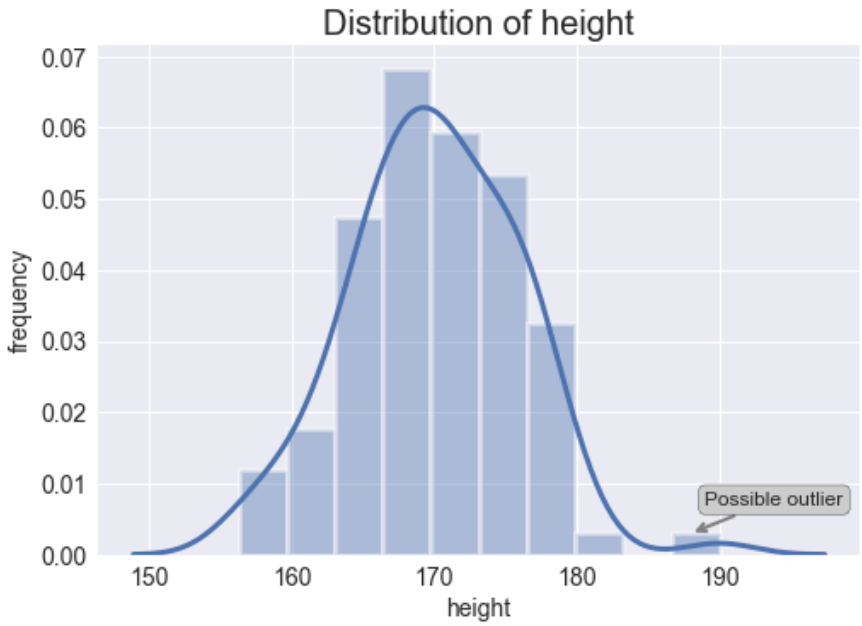

Рис. 7. Гистограмма с предположительным выбросом [33]

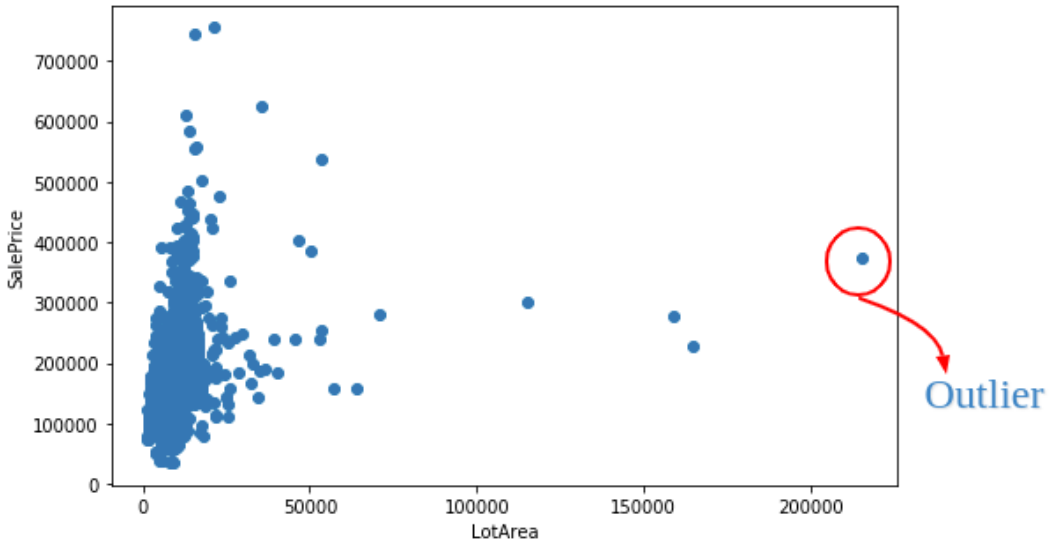

Рис. 8. Точечная диаграмма с выбросами [26]

С помощью математического подхода выбросы можно обнаружить двумя способами:

1. Z-оценка. Это мера относительного разброса наблюдаемого или измеренного значения, которая показывает, сколько стандартных отклонений составляет его разброс относительного среднего значения. Это безразмерный статистический показатель, используемый для сравнения значений разной размерности или шкалой измерений [52]. Суть, стоящая за Z-оценкой, заключается в том, чтобы описать любую точку данных, найдя их связь со стандартным отклонением и средним значением группы точек данных. Z-оценка находит распределение данных, где среднее значение равно 0, а стандартное отклонение равно 1, то есть нормальное распределение. При расчете Z-показателя нужно изменить масштаб и центрировать данные, найти точки данных, которые слишком далеки от нуля. Эти точки данных, находящиеся слишком далеко от нуля, будут рассматриваться как выбросы. В большом количестве случаев пороговое значение составляет 3 или

-3, т. е. если значение Z-оценки больше 3 или меньше -3, эта точка данных будет идентифицирована как выброс.

2. IQR (межквартильный диапазон) — это разница между третьим и первым квартилями распределения (или 75-й процентиль минус 25-й процентиль). Это мера того, насколько широко наше распределение, поскольку этот диапазон содержит половину точек набора данных. Очень полезно составить представление о форме распределения. Например, это ширина блоков на блочной диаграмме.

$$IQR = Q_3 - Q_1, \quad (2)$$

Другими словами, IQR — это первый квартиль, вычтенный из третьего квартиля; эти квартили можно четко увидеть на диаграмме данных. Это мера дисперсии, аналогичная стандартному отклонению или дисперсии, но гораздо более устойчивая к выбросам. IQR чем-то похож на Z-оценку с точки зрения поиска распределения данных и последующего сохранения некоторого порога для выявления выбросов [41].

2.2.3 Стандартизация и нормализация данных

На практике часто встречаются разные типы переменных в одном и том же наборе данных. Существенной проблемой является то, что диапазон переменных может сильно различаться. Использование исходной шкалы может придать больший вес переменным с большим диапазоном. Чтобы решить эту проблему, необходимо применить технику масштабирования признаков к независимым переменным или признакам данных на этапе предварительной обработки данных.

Результатом стандартизации (или нормализации Z-оценки) является то, что функции будут масштабироваться так, чтобы среднее значение и стандартное отклонение были равны 0 и 1 соответственно.

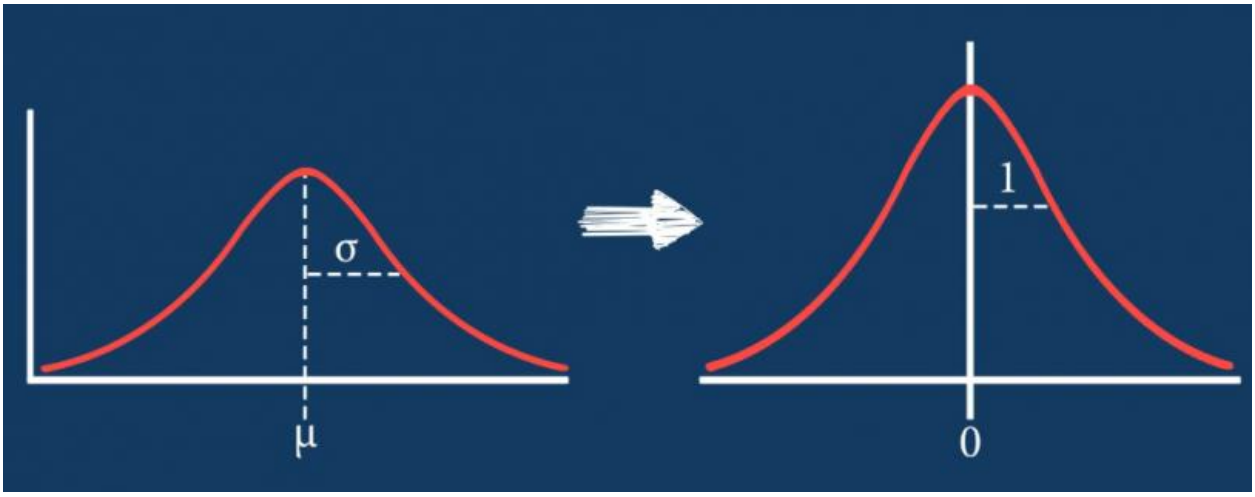

Рис. 9. Распределение до и после стандартизации [43]

$$X_{stand} = \frac{x - mean(x)}{standart\ deviation(x)}, \quad (3)$$

Концепция стандартизации появляется, когда непрерывные независимые переменные измеряются в разных масштабах. Этот метод заключается в повторном масштабировании значения признаков со значением распределения от 0 до 1. Он полезен для алгоритмов оптимизации, таких как градиентный спуск, которые используются в алгоритмах машинного обучения, которые взвешивают входные данные (например, регрессия и нейронные сети). Масштабирование также используется для алгоритмов, использующих измерения расстояния, например K-Nearest-Neighbours (KNN). Это означает, что эти переменные не дают равного вклада в анализ. Например, мы проводим анализ сегментации клиентов, в котором мы пытаемся сгруппировать клиентов на основе их однородных (похожих) характеристик.

- Алгоритмы машинного обучения, такие как линейная регрессия, логистическая регрессия, нейронная сеть и т. д., которые

используют градиентный спуск в качестве метода оптимизации, требуют масштабирования данных.

- Алгоритмы расстояния, такие как KNN, K-means и SVM, больше всего зависят от набора параметров. Это связано с тем, что «за кулисами» они используют расстояния между точками данных для определения их сходства. Нужно масштабировать данные, прежде чем использовать алгоритм на основе расстояния, чтобы все функции в равной степени влияли на результат.
- Древовидные алгоритмы довольно нечувствительны к масштабированию признаков. Дерево решений просто разбивает узел на основе одной функции. Дерево решений разбивает узел на функцию, которая увеличивает однородность узла. Это разделение функции не зависит от других функций. Таким образом, влияние остальных признаков на расщепление практически отсутствует. Именно это делает их инвариантными к масштабу признаков.

Основные методы стандартизации:

1) Z-score standardization

Стандартизация Z-оценки — один из самых популярных методов нормализации данных. В этом случае масштабируется исходная переменная, чтобы получить среднее значение, равное нулю, и стандартное отклонение, равное единице.

Математически масштабированная переменная будет вычисляться путем вычитания среднего значения исходной переменной из необработанного значения, а затем деления его на стандартное отклонение исходной переменной.

$$X_{stand} = \frac{x - mean(x)}{standart\ deviation(x)}, \quad (3)$$

2) Max-Min

Это метод масштабирования, при котором значения сдвигаются и масштабируются таким образом, что в итоге они оказываются в диапазоне от 0 до 1.

$$X_{norm} = \frac{x - \min(x)}{max(x) - \min(x)}, \quad (4)$$

3) Standard Deviation Method

В данном методе делится каждое значение на стандартное отклонение. Идея состоит в том, чтобы иметь одинаковую дисперсию, но разные средние значения и диапазоны.

$$X_{stand} = \frac{x}{stdev(x)}, \quad (5)$$

4) Range Method

В этом методе делится каждое значение на диапазон. В этом случае средние значения, дисперсии и диапазоны переменных все еще различаются, но, по крайней мере, диапазоны, вероятно, будут более похожими.

$$X_{stand} = \frac{x}{max(x) - \min(x)}, \quad (6)$$

Пример данных до и после нормализации изображен на рисунках 10 и 11.

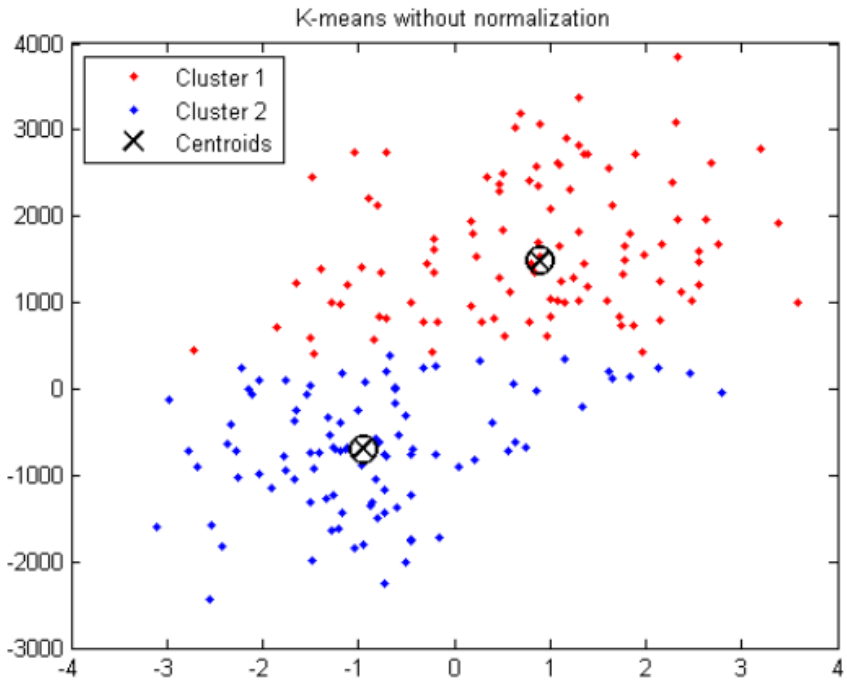

Рис. 10. Распределение данных до нормализации [43]

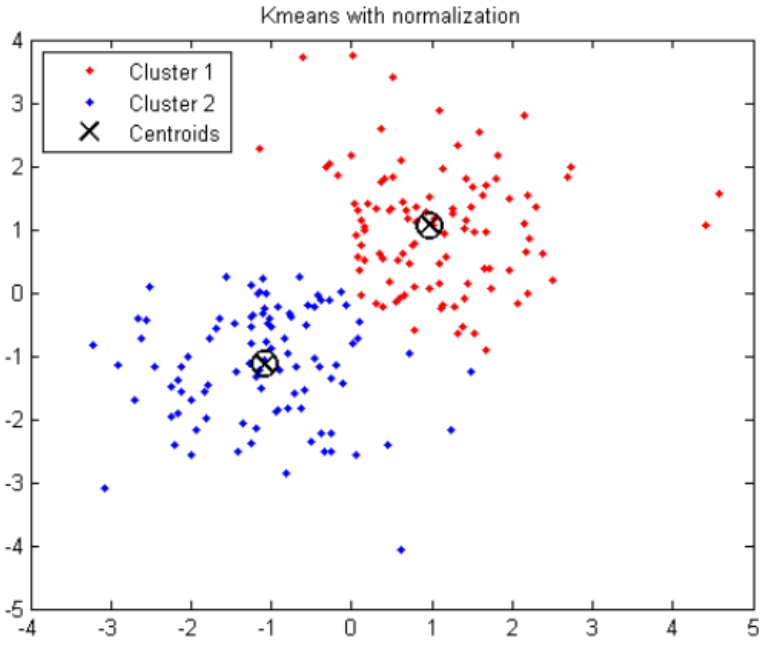

Рис. 11. Распределение данных после нормализации [43]

Различия между стандартизацией и нормализацией данных:

- Нормализация полезна, когда известно, что распределение данных не соответствует распределению Гаусса. Это может быть полезно в алгоритмах, которые не предполагают никакого распределения данных, таких как K-ближайшие соседи и нейронные сети.
- Стандартизация, может быть полезна в случаях, когда данные следуют распределению Гаусса. Однако это не обязательно истина. В отличие от нормализации, стандартизация не имеет ограничивающего диапазона. Таким образом, даже если в данных есть выбросы, стандартизация не повлияет на них.
- Выбор использования нормализации или стандартизации будет зависеть от проблемы и используемого алгоритма машинного обучения.
- Не существует жесткого правила, указывающего, когда следует нормализовать или стандартизировать данные. Всегда можно начать с подгонки модели к необработанным, нормализованным и стандартизированным данным и сравнить производительность для достижения наилучших результатов.
- Хорошей практикой является подгонка масштабирующего устройства к обучающим данным, а затем его использование для преобразования данных тестирования. Это позволит избежать утечки данных в процессе тестирования модели. Кроме того, обычно не требуется масштабирование целевых значений [43].

2.3 Корреляция и таблицы корреляций

Корреляция объясняет, как одна или несколько переменных связаны друг с другом. Эти переменные могут быть функциями входных данных, которые использовались для прогнозирования целевой переменной.

Корреляция - статистический метод, который определяет, как одна переменная перемещается/изменяется по отношению к другой переменной. Это дает представление о степени взаимосвязи двух переменных. Это мера двумерного анализа, которая описывает связь между различными переменными. В большинстве случаев полезно выражать один предмет с точки зрения его отношений с другими.

Основными аспектами корреляционного анализа являются:

1. Если две переменные тесно связаны (коррелируют), то можно предсказать одну переменную по другой.
2. Корреляция играет жизненно важную роль в определении важных переменных, от которых зависят другие переменные.
3. Корреляция используется в качестве основы для различных методов моделирования.
4. Правильный корреляционный анализ приводит к лучшему восприятию данных.
5. Корреляция способствует пониманию причинно-следственной связи (если таковая имеется).

Существует три корреляционных состояния: положительная, отрицательная и полное отсутствие корреляции.

Положительная корреляция: две функции (переменные) могут положительно коррелировать друг с другом. Это означает, что, когда значение одной переменной увеличивается, значение другой переменной (других) также увеличивается [42].

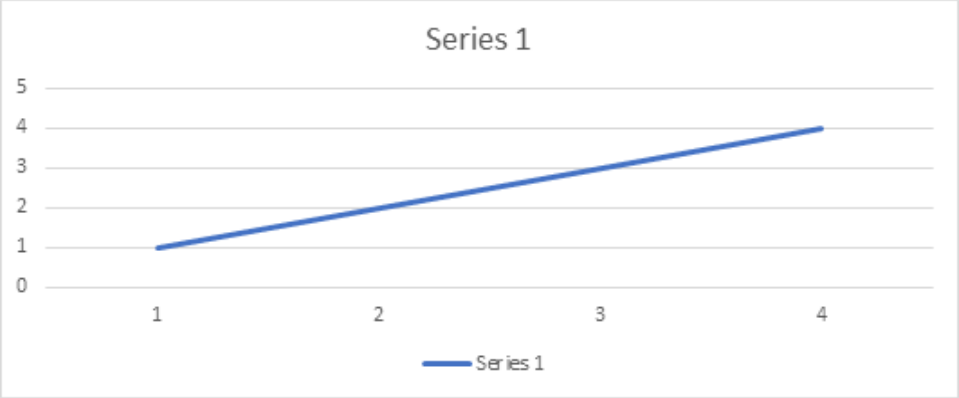

Рис. 12. Положительная корреляция

Отрицательная корреляция: две функции (переменные) могут иметь отрицательную корреляцию друг с другом. Это означает, что, когда значение одной переменной увеличивается, значение другой переменной (переменных) уменьшается.

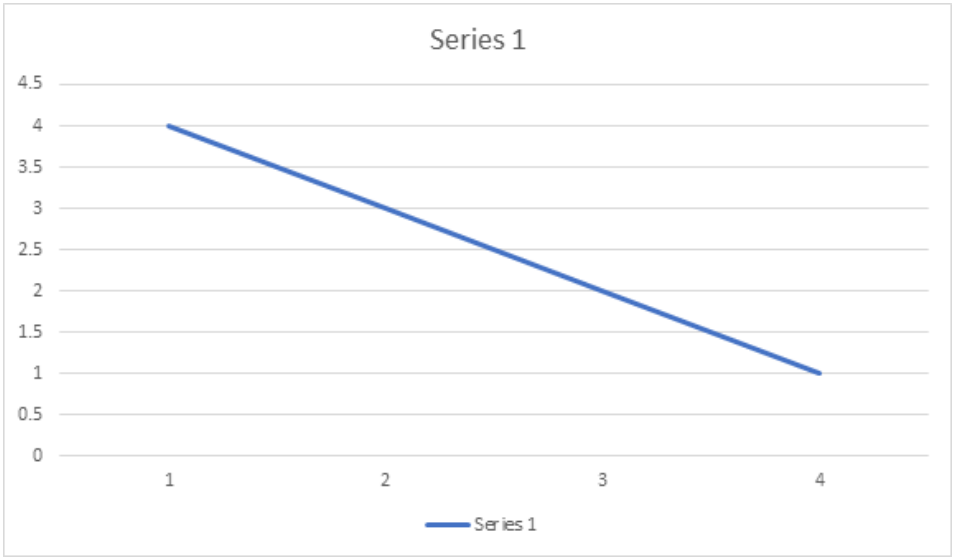

Рис. 13. Отрицательная корреляция

Отсутствие корреляции: две функции (переменные) не коррелируют друг с другом. Это означает, что, когда значение одной переменной увеличивается или уменьшается, значение другой переменной (переменных) не увеличивается или уменьшается.

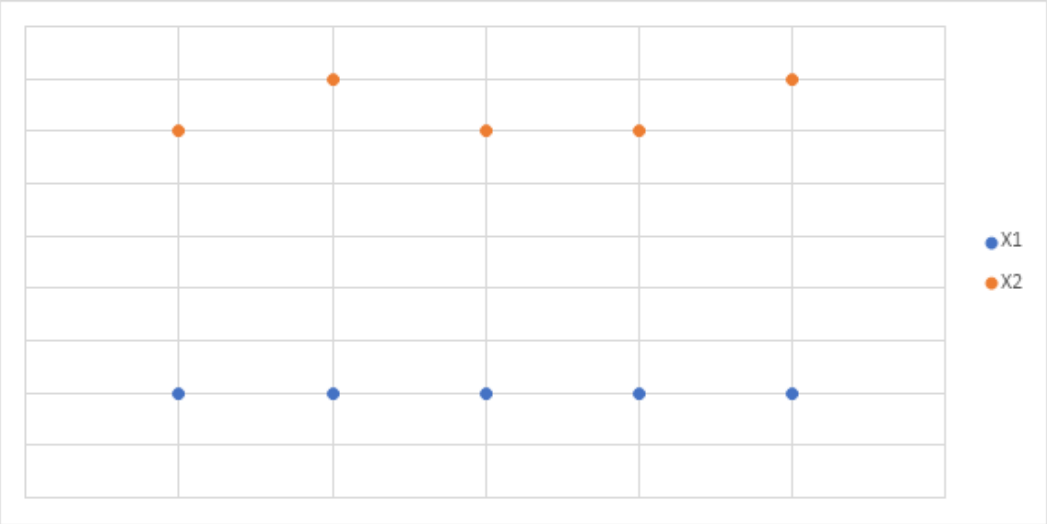

Рис. 14. Отсутствие корреляции [42]

Вычислить корреляцию можно несколькими путями:

1) Корреляция Пирсона

Коэффициент корреляции Пирсона является мерой силы линейной связи между двумя переменными и обозначается r. По сути, корреляция Пирсона пытается провести линию наилучшего соответствия данных двух переменных. Коэффициент корреляции Пирсона r указывает, насколько далеко все эти точки данных находятся от этой линии наилучшего соответствия.

- В коэффициенте корреляции Пирсона переменные могут измеряться в совершенно разных единицах. Например, можно соотнести рост человека с его весом. Он разработан таким образом, что единица измерения не может повлиять на изучение ковариации.
- Коэффициент корреляции Пирсона (r) представляет собой безразмерную меру корреляции и не изменяется в зависимости от происхождения или измерения сдвига шкалы.

- При этом не принимается во внимание, была ли переменная классифицирована как зависимая или независимая переменная. Он обрабатывает все переменные одинаково. Можно было бы захотеть выяснить, связаны ли результаты игры в баскетбол с ростом человека. Но если выяснить, определялся ли рост человека его игрой в баскетбол (что не имеет смысла), результат будет таким же.

Рассчитать коэффициент корреляции Пирсона можно по следующей формуле:

$$\rho = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2 (Y_i - \bar{Y})^2}}, \quad (7)$$

, где X_i и Y_i – переменные,

$\bar{X}$ и $\bar{Y}$ – среднее значение.

2) Корреляция Спирмена

Коэффициент корреляции Спирмена — это непараметрическая мера силы и направления связи, которая существует между двумя переменными, измеряемыми по крайней мере по порядковой шкале. Его обозначает символ rs или ρ. Рассчитать коэффициент корреляции Спирмена можно по формуле:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}, \quad (8)$$

, где n – общее количество наблюдений,

$d_i = (X_i - Y_i)$ – X_i и Y_i – наблюдения.

Корреляция Спирмена определяет силу и направление монотонной связи между двумя переменными, а не силу и направление линейной связи между двумя переменными, которую определяет корреляция Пирсона [12].

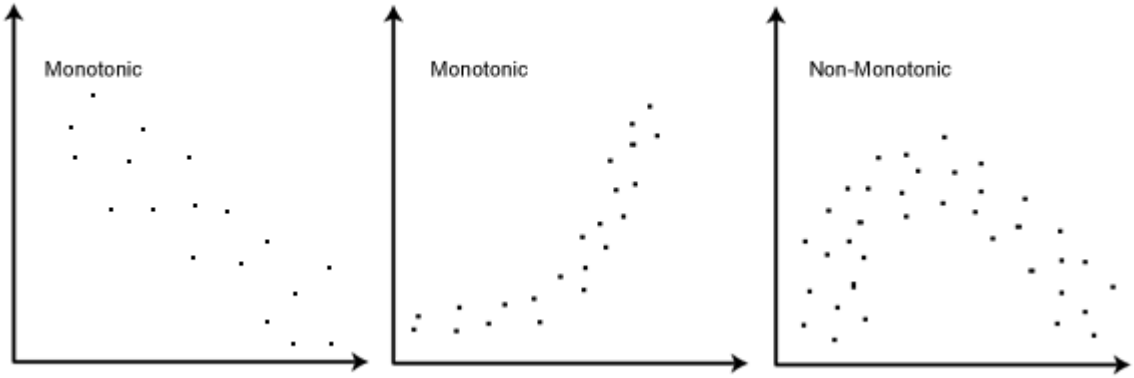

Рис. 15. ρ с монотонным графиком

В данной работе для исследования переменных была использована корреляция Спирмена, так как корреляция Пирсона очень чувствительна к любым «выбросам». В реальных данных достаточно много выбросов, так что корреляция Пирсона невозможна.

3) Корреляция Кендалла

Коэффициент корреляции Кендалла — это непараметрическая мера отношений между столбцами ранжированных данных. Коэффициент корреляции Кендалла возвращает значение от 0 до 1, где:

0 - нет отношений,

1 - идеальные отношения.

Отличительная особенность в том, что корреляция также может давать отрицательные значения (от -1 до 0). В отличие от линейного графика, отрицательное отношение не имеет большого значения для ранжированных столбцов (кроме того, что вы, возможно, поменяли местами столбцы), поэтому нужно удалить отрицательный знак, когда интерпретируются результаты корреляции Кендалла. Формула данного коэффициента корреляции:

$$Kendall's\ Tau = \left(\frac{C - D}{C + D}\right), \quad (9)$$

, где C — количество согласованных пар,

D — количество несогласованных пар [12].

В машинном обучении результаты корреляции чаще всего отображаются в виде тепловой карты с коэффициентами корреляции между переменными по выбранной формуле.

Рис. 16. Пример отображения корреляции в машинном обучении

Чаще всего, карту корреляций строят перед регрессионным анализом. Ключевая цель регрессионного анализа в машинном обучении — изолировать взаимосвязь каждой независимой переменной и зависимой переменной. Таким образом, изменение одной независимой переменной не должно влиять ни на какие другие переменные в данных. Однако, когда независимые переменные коррелированы, это указывает на то, что изменения одной переменной связаны со сдвигами другой переменной. По мере того, как серьезность мультиколлинеарности увеличивается, увеличиваются и эти проблемные

эффекты. Таким образом, во время подгонки модели небольшое изменение одной переменной может привести к значительному колебанию выходных данных модели. Однако эти проблемы затрагивают только те независимые переменные, которые коррелируют.

Решение проблемы мультиколлинеарности зависит от:

1. Серьезность проблем возрастает со степенью мультиколлинеарности. Поэтому, если у есть умеренная мультиколлинеарность, возможно, ничего менять не следует.
2. Мультиколлинеарность влияет только на определенные независимые переменные, которые коррелируют друг с другом. Поэтому, если для интересующих независимых переменных мультиколлинеарность отсутствует, то может не потребоваться ее разрешение.
3. Если одна из коллинеарных функций не имеет большого значения для предсказания или классификации алгоритмов машинного обучения на основе расстояния, можно отказаться от анализа переменной [12].

2.4 Концепты линейной регрессии и множественной линейной регрессии

Линейная регрессия – это один из базовых алгоритмов машинного обучения. Обычная линейная регрессия имеет одну зависимую переменную и только одну независимую переменную. Модель линейной регрессии можно описать формулой:

$$\hat{y} = \sum_{i=1}^{n} x_i w_i + b_0 \,, \tag{10}$$

, где $\hat{y}$ – предсказываемое значение,

x_i – значение признака,

w_i – коэффициент (вес) признака,

b_0 – смещение,

n – количество признаков.

Множественная линейная регрессия делает еще один шаг вперед, и вместо одной будет две или более независимых переменных. Уравнение множественной линейной регрессии со свободным членом и k-независимыми переменными (факторами) выглядит следующим образом:

$$\hat{y} = b_0 + b_1 x_1 + \cdots + b_k x_k, \quad (11)$$

, где $\hat{y}$ – зависимая переменная,

b_0 – пересечение оси зависимой переменной,

x_1 – x_n – независимые переменные в наборе данных,

b_1 – b_n – параметры коэффициентов, которые модель также будет настраивать,

n – количество признаков в наборе данных.

Модель множественной линейной регрессии способна анализировать взаимосвязь между несколькими независимыми переменными и одной зависимой переменной.

При использовании средств машинного обучения на основе регрессии можно сгруппировать (как минимум) три разные проблемные области:

1. Количественная оценка отношений: известно, что с помощью корреляции можно найти силу (выраженную в виде числа от 0 до 1) и направление (положительное или отрицательное) отношения между двумя переменными. Но множественная регрессия идет еще дальше и фактически дает количественную оценку этой взаимосвязи.
2. Прогноз: машинное обучение — это о прогнозировании. Поскольку регрессия способна количественно определять

отношения, можно использовать эту возможность для решения задач прогнозирования.

3. Прогнозирование. Третьей проблемной областью множественной регрессии является прогнозирование с помощью анализа временных рядов. Векторная авторегрессия и регрессия панельных данных — это два мощных метода прогнозирования, в которых используются принципы множественной регрессии.

Несколько дополнительных моментов, о которых следует помнить при построении модели линейной регрессии для решения реальных задач:

- При выборе признаков нужно проверять корреляцию между зависимой и каждой независимой переменной отдельно. Если они не коррелированы, удалить переменную из модели;
- Проверка на мультиколлинеарность, связь между независимыми переменными. Удалить коррелированные переменные, чтобы избежать переобучения модели;
- Есть несколько способов выбрать переменные для модели. Прямой отбор и обратное исключение — два из них. Как следует из названий, в этом процессе добавляется или удаляется одна переменная за раз и проверяется производительность режима;
- Часто используется R^2 для оценки производительности модели, но существуют и другие показатели, такие как AIC, BIC, p-значение и т. д. [10]

В данной работе для отбора наиболее важных и ценных параметров использовался метод множественной линейной регрессии. При оценке производительности и важности переменных принималось во внимание p-значение.

2.5 Эффекты взаимодействия в линейной регрессии

Эффект взаимодействия присутствует и в статистике, и в маркетинге. В маркетинге это же понятие называют эффектом синергии. Эффект взаимодействия означает, что два или более объединенных признаков/переменных оказывают значительно большее влияние на признак по сравнению с суммой отдельных переменных. Этот эффект важно понимать в регрессии, когда есть цель изучить влияние нескольких переменных на одну целевую переменную.

Рассмотрим на примере обычного уравнения линейной регрессии:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \varepsilon, \quad (12)$$

Здесь нужно найти линейную связь между независимыми переменными (X_1 и X_2) с зависимой переменной Y, а ε — необратимая ошибка. Чтобы проверить, существует ли какая-либо значимая статистическая связь между предиктором и переменными отклика, проводится проверка гипотез. Если проводится этот тест для предикторной переменной X_1, будет две гипотезы:

- Нулевая гипотеза (H_0): нет никаких взаимоотношений между X_1 и Y ($b_1 = 0$)
- Альтернативная гипотеза (H_1): есть взаимоотношения между X_1 и Y ($b_1 \neq 0$)

Затем решается, следует ли отклонить нулевую гипотезу на основе p-значения. P-значение — это вероятность результатов теста при условии, что нулевая гипотеза верна.

Например, если получается ненулевое значение β_1 в результатах теста, это указывает на то, что существует связь между X_1 и Y. Но если значение p велико, это указывает на высокую вероятность того, что могло бы получиться ненулевое значение для β_1, даже если нулевая гипотеза действительно верна.

В таком случае нельзя отвергнуть нулевую гипотезу и сделать вывод об отсутствии связи между предиктором и переменной отклика. Но если p-значение низкое (обычно пороговое значение p-значения считается равным 0,05), то даже небольшое ненулевое значение β_1 указывает на значительную связь между предиктором и переменной отклика.

Если мы приходим к выводу, что существует связь между X_1 и Y, считается, что на каждую единицу увеличения X_1 Y увеличивается/уменьшается на β_1 единиц. В приведенном выше линейном уравнении предполагается, что влияние X_1 на Y не зависит от X_2. Это также называется аддитивным предположением в линейной регрессии.

Но может быть ситуация, что влияние X_1 на Y также зависит от X_2. Такие отношения наблюдаются во многих бизнес-задачах. Рассмотрим, на примере окупаемости яблок на рынке для двух разных типов яблок. Уравнение линейной регрессии для этого примера будет таким:

$$ROI = b_0 + b_1 * apple_1 + b_2 * apple_2 + \varepsilon, \quad (13)$$

В данном примере есть вероятность, что прибыль будет больше, если мы инвестируем в оба типа яблок частично, а не в один полностью. Например, если у нас есть 1000 единиц денег для инвестирования, вложение 500 единиц денег в оба типа яблок может привести к большей прибыли по сравнению с полной инвестицией 1000 единиц в любой из типов яблок. В таком случае отношение apple1 к рентабельности яблочной продукции будет зависеть от apple2. Это соотношение может быть включено в уравнение 13 следующим образом:

$$ROI = b_0 + (b_1 + b_3 * apple_2) * apple_1 + b_2 * apple_2 + \varepsilon, \quad (14)$$

В (формуле (14)) включено «взаимодействие» между apple1 и apple2 для прогнозирования общего дохода от яблочной продукции. Можно включить такие взаимодействия для любого уравнения линейной регрессии:

$$\hat{y} = b_0 + (b_1 + b_3 * x_2)x_1 + b_2x_2 + \varepsilon, \quad (15)$$

(Формулу(15)) также можно переписать в следующем виде:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_2x_1 + \varepsilon, \quad (16)$$

Здесь b_3 — коэффициент члена взаимодействия. Опять же, чтобы проверить наличие эффекта взаимодействия в регрессии, проводится проверка гипотезы и проверяется значение p для коэффициента (в данном случае b_3) [29].

Определить, есть ли взаимодействия между переменными можно и графическим путем. График взаимодействия представляет собой линейный график, показывающий наличие или отсутствие взаимодействия между независимыми переменными. Чтобы создать график взаимодействия, делаются следующие действия:

- Отображается зависимая переменная на вертикальной оси (т. е. на оси Y); и независимая переменная на горизонтальной оси (т. е. оси X);
- Строятся средние значения зависимой переменной отдельно для каждого уровня потенциально взаимодействующей переменной;
- Соединяются средние значения, создав отдельные линии для каждого уровня взаимодействующей переменной.

Затем, чтобы понять потенциальные эффекты взаимодействия, сравниваются линии на графике взаимодействия:

- Если прямые параллельны, взаимодействия нет;
- Если прямые не параллельны, то взаимодействие есть.

Наблюдение за наличием эффектов взаимодействия может сыграть важную роль в построении надежной модели обучения. Мало того, всегда нужно проверять статистическую значимость условий взаимодействия, прежде чем включать их в модель [36].

2.6 Метрики оценки качества регрессии

Существуют различные метрики оценки качества регрессии, в зависимости от поставленной задачи каждая обладает своими недостатками и преимуществами.

2.6.1 Среднеквадратическая ошибка (Mean Squared Error)

Среднеквадратическая ошибка (MSE), пожалуй, самая простая и наиболее распространенная функция потерь. Чтобы вычислить MSE, берется разница между предсказаниями модели и истинным значением, возводится ее в квадрат и усредняется по всему набору данных.

MSE никогда не будет отрицательная, так как всегда ошибка возводится в квадрат. MSE формально определяется следующим уравнением:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \bar{y})^2\,, \tag{17}$$

, где N — количество образцов, которые мы тестируем.

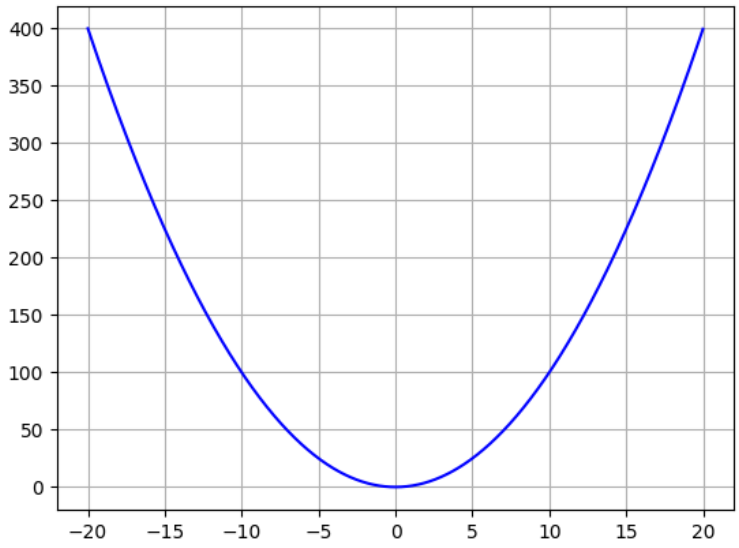

Рис. 17. Функция потерь MSE

Преимущества: MSE отлично подходит для обеспечения того, чтобы обученная модель не имела прогнозов с большими ошибками, поскольку MSE придает больший вес этим ошибкам из-за возведения в квадрат части функции.

Недостатки: если модель делает один очень плохой прогноз, возведение в квадрат части функции увеличивает ошибку. Тем не менее, во многих практических случаях нет особой заботы об этих выбросах и идет стремление к более всесторонней модели, которая достаточно хорошо работает в большинстве случаев.

2.6.2 Средняя абсолютная ошибка (Mean Absolute Error)

Средняя абсолютная ошибка (MAE) немного отличается по определению от MSE, но, что интересно, обладает почти точно противоположными свойствами. Чтобы рассчитать MAE, берется разница между предсказаниями модели и реальностью, применяется абсолютное значение к этой разнице, а затем усредняете ее по всему набору данных.

MAE, как и MSE, никогда не будет отрицательным, поскольку в этом случае всегда берется абсолютное значение ошибок. MAE формально определяется следующим уравнением:

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \bar{y_J}\right|, \tag{18}$$

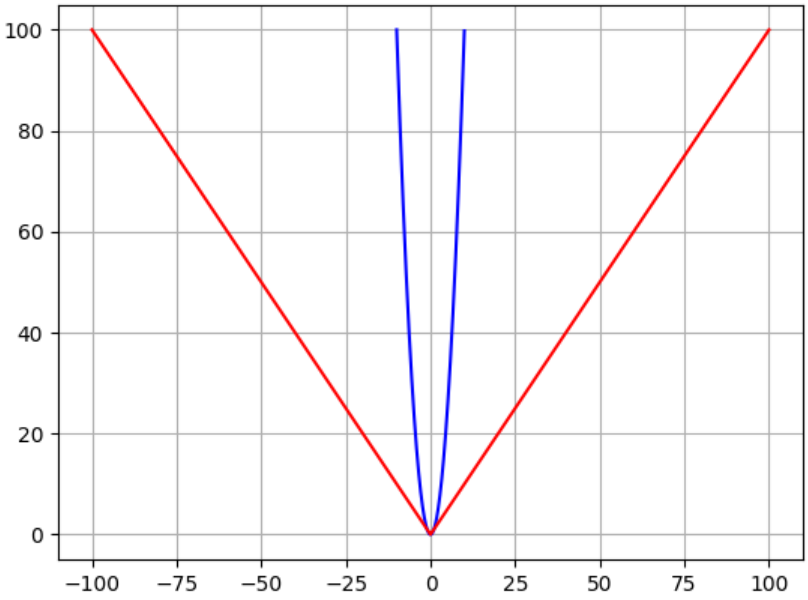

Рис. 18. Функции потерь MAE (красная) и MSE (синяя)

Преимущества: Прелесть MAE в том, что его преимущество напрямую перекрывает недостаток MSE. Поскольку берется абсолютное значение, все ошибки будут взвешены по одной и той же линейной шкале. Таким образом, в отличие от MSE, не будет придаваться слишком большое значение выбросам, а функция потерь обеспечивает общую и даже меру того, насколько хорошо работает модель.

Недостатки: если действительно заботиться о прогнозах, выпадающих из модели, то MAE не будет столь эффективным. Большие ошибки, возникающие из-за выбросов, в итоге взвешиваются точно так же, как и

меньшие ошибки. Это может привести к тому, что модель будет отличной большую часть времени, но время от времени будет давать несколько очень плохих прогнозов.

2.6.3 Функция потерь Хьюбера (Huber Loss)

MSE наиболее подходит для изучения выбросов, а MAE — для их игнорирования. Но есть метод, который находится где-то «посередине».

Функция потерь Хьюбера предлагает лучшее из обоих миров, уравновешивая MSE и MAE вместе. Можно определить его формулу, используя следующий кусочек функции:

$$L_\delta\big(y, f(x)\big) = \begin{cases} \frac{1}{2}\big(y - f(x)\big)^2 & \text{для } |y - f(x)| \le \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{в противном случае} \end{cases}, \quad (19)$$

По сути, это уравнение говорит следующее: для значений потерь меньше дельты используйте MSE; для значений потерь больше дельты используйте MAE. Это эффективно сочетает в себе лучшее из обоих миров от двух функций потерь.

Использование MAE для больших значений потерь уменьшает вес, который придается выбросам, так что по-прежнему получается всесторонняя модель. В то же время используется MSE для меньших значений потерь, чтобы поддерживать квадратичную функцию вблизи центра. Это приводит к увеличению значений потерь, если они больше 1. Как только потери для этих точек данных опускаются ниже 1, квадратичная функция уменьшает их вес, чтобы сосредоточить обучение на точках данных с более высокой ошибкой.

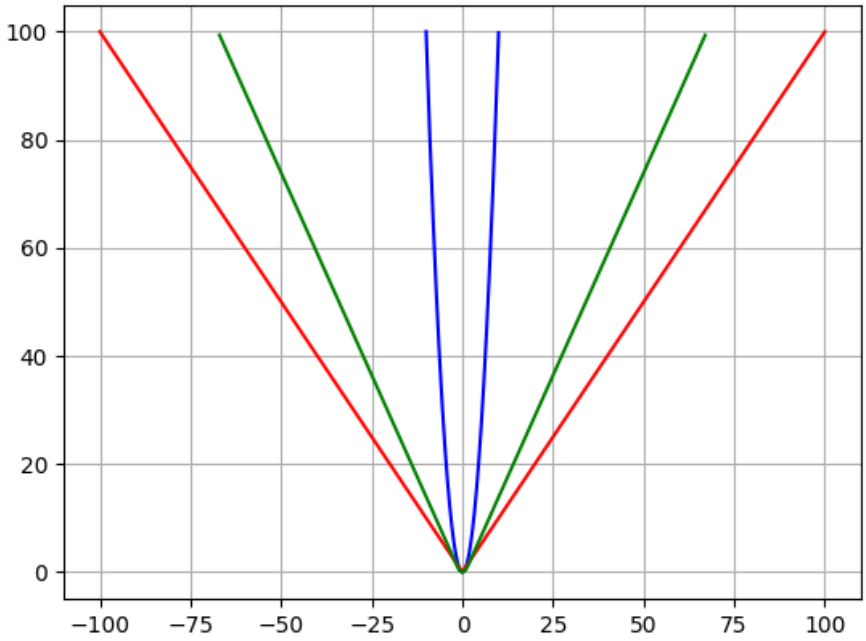

Рис. 19. Функции потерь MAE (красная), MSE (синяя), Huber (зеленая)

Можно использовать функцию потерь Хьюбера каждый раз, когда нужен баланс между приданием выбросам некоторого веса, но не слишком большого. В случаях, когда выбросы очень важны, следует использовать MSE. В тех случаях, когда выбросы совершенно неважны, следует использовать MAE [38].

2.7 Алгоритмы машинного обучения

Существует множество различных алгоритмов машинного обучения, в основы которых входят разнообразные подходы, зависящие от цели поставленной задачи. В данной работе для решения поставленной задачи были использованы алгоритмы на основе деревьев решений. По этой причине разберем сущность подобных алгоритмов и что лежит в их основе.

2.7.1 Ансамблевые методы машинного обучения

Ансамблевый метод — это метод машинного обучения, который объединяет несколько базовых моделей для создания одной оптимальной прогностической модели. Чтобы лучше понять это определение, нужно вернуться к конечной цели машинного обучения и построения

При обсуждении ансамблевых методов стоит начать с дерева решений. Дерево решений определяет прогностическую ценность на основе ряда вопросов и условий. Например, простое дерево решений на рисунке 20 определяет, должен ли человек играть на улице или нет. Дерево учитывает несколько погодных факторов, и с учетом каждого фактора либо принимает решение, либо задает другой вопрос. В этом примере каждый раз, когда будет пасмурно, будем играть на улице. Однако, если идет дождь, должен ли подставиться вопрос, ветрено или нет? Если будет ветер, играть не получится. Но если нет ветра, то против прогулки нет никаких факторов.

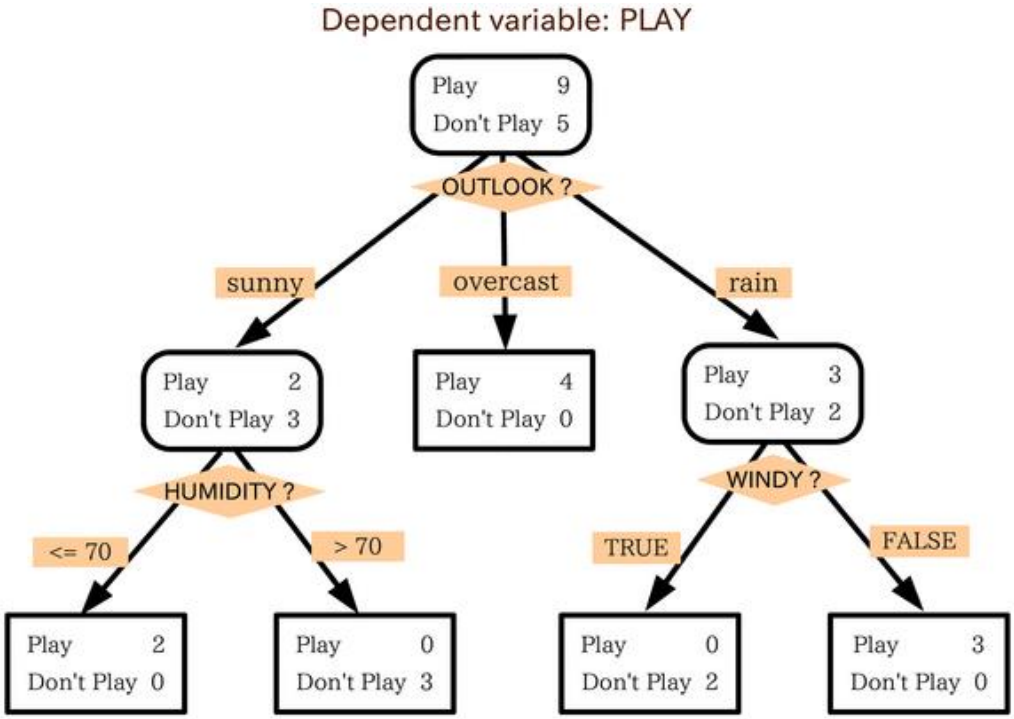

Рис. 20. Пример дерева решений

Деревья решений также могут решать количественные задачи в том же формате. В дереве на рисунке 21, например, хочется узнать, стоит ли инвестировать в коммерческую недвижимость.

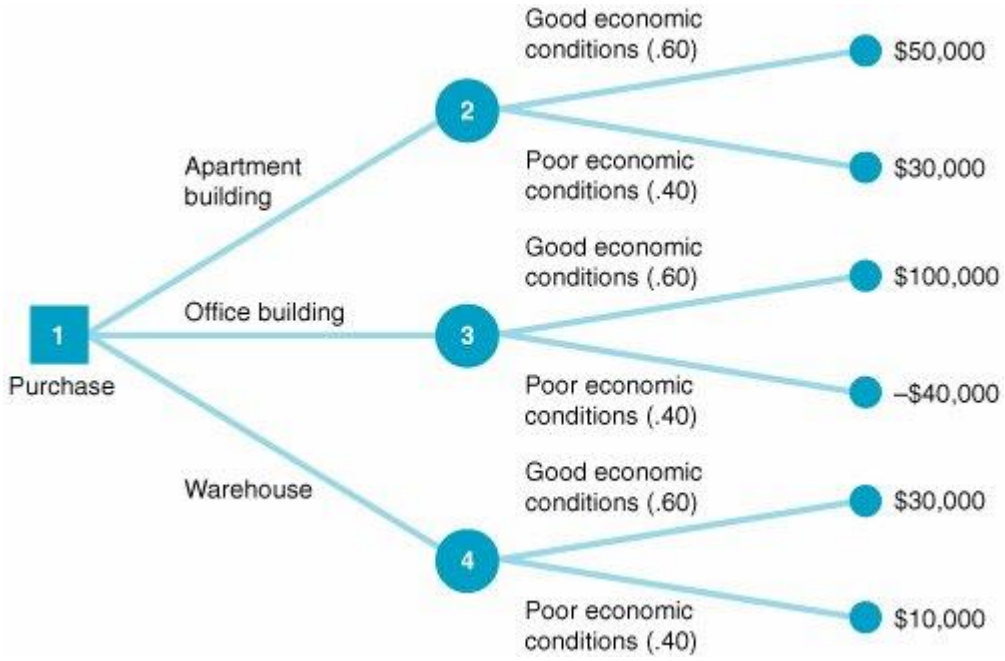

Рис. 21. Пример дерева решений

При создании деревьев решений должны учитываться несколько факторов:

- На каких функциях мы принимаем решения?
- Каков порог классификации каждого вопроса, как ответа «да» или «нет»?

Обращаясь к дереву решений на рисунке 20, что, если хочется задать вопрос, есть ли у нас друзья, с которыми можно поиграть, или нет? Если есть друзья, будем играть каждый раз. Если нет, могли бы продолжать задавать себе вопросы о погоде. Добавляя дополнительный вопрос, мы надеемся лучше определить классы «Да» и «Нет».

Вот где «на сцену» выходят ансамблевые алгоритмы. Вместо того, чтобы просто полагаться на одно дерево решений и надеяться, что было принято правильное решение при каждом разделении, ансамблевые методы позволяют принять во внимание выборку деревьев решений, рассчитать, какие функции использовать или вопросы, которые нужно задать при каждом

разделении, и сделать окончательный вывод на основе агрегированных результатов выборочных деревьев решений.

Виды ансамблевых алгоритмов можно разделить следующим образом:

1. Бэггинг (Bootstrap Aggregating).

   Бэггинг получил свое название, потому что он сочетает в себе бутстрэппинг и агрегацию, чтобы сформировать одну ансамблевую модель. При наличии выборки данных извлекается несколько подвыборок с бутстрэппингом. Дерево решений формируется для каждой из загруженных подвыборок. После формирования каждого дерева решений подвыборки, алгоритм используется для агрегирования деревьев решений для формирования наиболее эффективного предиктора. Более детально этот процесс изображен на рисунке 22.

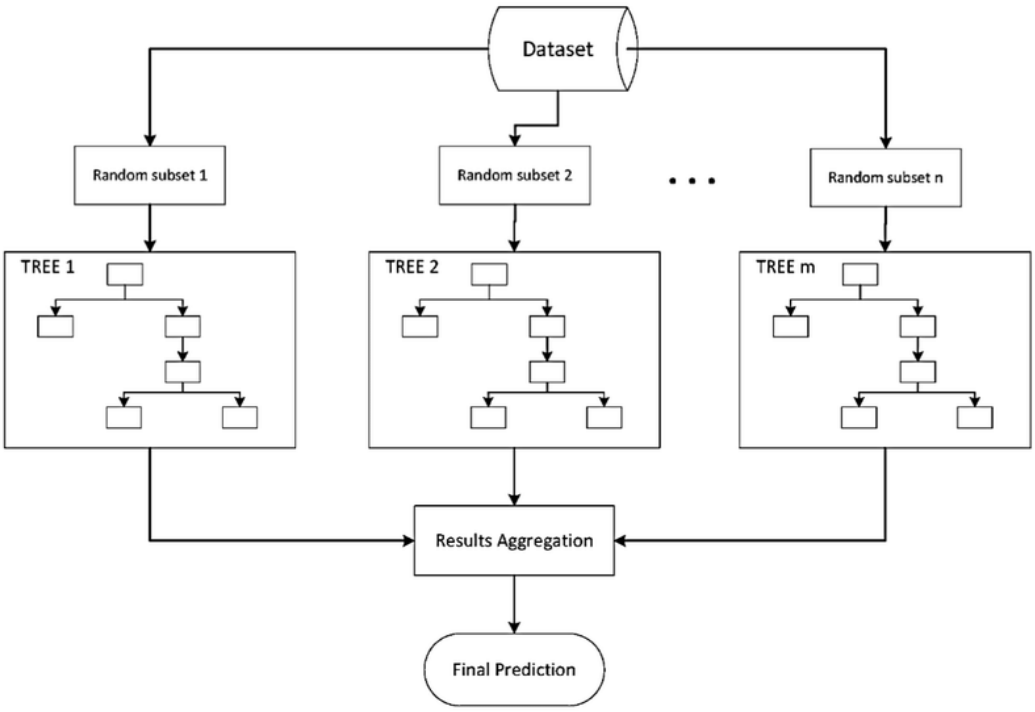

Рис. 22. Дерево решений формируется для каждой загруженной выборки, результаты каждого дерева агрегируются, чтобы получить самый надежный и точный предиктор

2. Модели случайного леса.

Случайный лес можно рассматривать как бэггинг с небольшой настройкой.

При принятии решения о том, где разделить и как принимать решения, «бэггинговые деревья решений» имеет полный набор функций на выбор. Таким образом несмотря на то, что выборки с бутстрэппингом могут немного отличаться, данные в основном будут разбиваться на одни и те же переменные для каждой модели.

В противовес деревьям решений, модели случайного леса решают, где разделить выборку, на основе случайного выбора признаков. Вместо того, чтобы разбивать на одинаковые переменные в каждом узле, модели случайного леса реализуют уровень дифференциации, поскольку каждое дерево будет разбиваться на основе разных переменных. Этот уровень дифференциации обеспечивает больший ансамбль для агрегирования, следовательно, дает более точный предиктор. Такая модель изображена на рисунке 23.

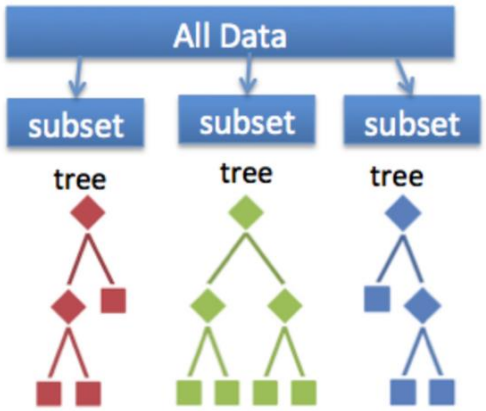

Рис. 23. Разбиение на леса при «случайном лесе»

Как и в случае бэггинга, бутстрэппинги извлекаются из большего набора данных. На каждой подвыборке формируется дерево решений. Однако

дерево решений разделено на разные переменные (на рисунке 23 переменные представлены фигурами).

Цель любой задачи машинного обучения — найти единую модель, которая лучше всего предскажет желаемый результат. Вместо того, чтобы создавать одну модель и надеяться, что эта модель является лучшим/самым точным предсказателем, который можно сделать, ансамблевые методы учитывают множество моделей и усредняют эти модели для получения одной окончательной модели. Важно отметить, что деревья решений — не единственная форма ансамблевых методов, а просто самая популярная и актуальная на сегодняшний день в науке о данных [30].

2.7.2 Случайный лес (Random Forest)

Случайный лес — это метод машинного обучения, который используется для решения задач регрессии и классификации. Он использует ансамблевое обучение, которое представляет собой метод, объединяющий множество классификаторов для решения сложных задач.

Алгоритм случайного леса состоит из множества деревьев решений. «Лес», сгенерированный алгоритмом случайного леса, обучается с помощью бэггинга. Бэггинг — ансамблевый метаалгоритм, повышающий точность алгоритмов машинного обучения. Алгоритм случайного леса устанавливает результат на основе прогнозов деревьев решений. Он прогнозирует, взяв среднее или среднее значение выходных данных различных деревьев. Увеличение количества деревьев повышает точность результата. Случайный лес устраняет ограничения алгоритмов деревьв решений. Это уменьшает переоснащение наборов данных и повышает точность. Он генерирует прогнозы, не требуя множества конфигураций в пакетах (например, scikit-learn).

Случайный лес имеет определенные преимущества:

- Он более точный, чем алгоритм на обычных деревьях решений;

- Он обеспечивает эффективный способ обработки отсутствующих данных;
- Он может дать разумный прогноз без настройки гиперпараметров;
- Он решает проблему переобучения в деревьях решений;
- В каждом дереве случайного леса подмножество признаков выбирается случайным образом в точке разделения узла.

2.7.2.1 Как работает случайный лес

Деревья решений — это строительные блоки алгоритма случайного леса. Дерево решений — это метод поддержки принятия решений, который формирует древовидную структуру. Дерево решений состоит из трех компонентов: узлов решений, конечных узлов и корневого узла. Алгоритм дерева решений делит обучающий набор данных на ветви, которые далее разделяются на другие ветви. Эта последовательность продолжается до тех пор, пока не будет достигнут листовой узел. Листовой узел не может быть отделен дальше. Узлы в дереве решений представляют атрибуты, которые

используются для прогнозирования результата. Узлы решений обеспечивают связь с листьями. На рисунке 24 показаны три типа узлов в дереве решений.

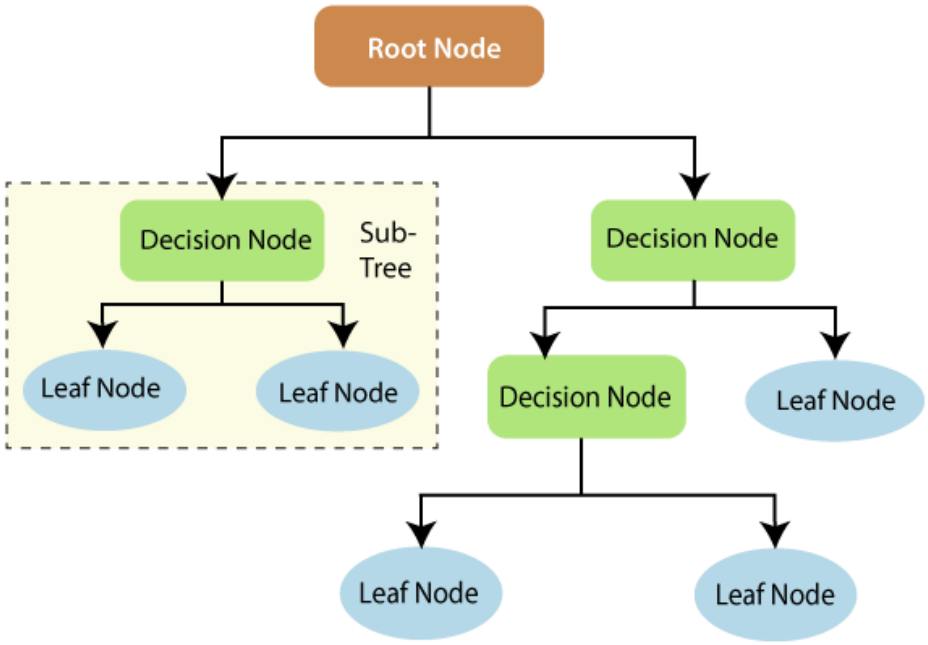

Рис. 24. Три типа узлов в деревьях решений

Теория информации может предоставить больше информации о том, как работают деревья решений. Энтропия и прирост информации являются строительными блоками деревьев решений. Обзор этих фундаментальных концепций улучшит наше понимание того, как строятся деревья решений.

Энтропия — это метрика для расчета неопределенности. Прирост информации является мерой того, как снижается неопределенность в целевой переменной при заданном наборе независимых переменных.

Концепция получения информации включает использование независимых переменных (признаков) для получения информации о целевой переменной (классе). Энтропия целевой переменной (Y) и условная энтропия Y (при заданном X) используются для оценки прироста информации. В этом случае условная энтропия вычитается из энтропии Y. Получение информации используется при обучении деревьев решений. Это помогает уменьшить неопределенность в этих деревьях. Высокий информационный прирост означает, что высокая степень неопределенности (информационная энтропия)

была устранена. Энтропия и прирост информации важны при разделении ветвей, что является важным действием при построении деревьев решений.

Основное различие между алгоритмом дерева решений и алгоритмом случайного леса заключается в том, что в последнем установление корневых узлов и разделение узлов выполняется случайным образом. Случайный лес использует метод мешков для генерации требуемого прогноза.

Бэггинг предполагает использование разных выборок данных (данные для обучения), а не только одной выборки. Набор обучающих данных содержит наблюдения и функции, которые используются для прогнозирования. Деревья решений дают разные результаты в зависимости от обучающих данных, подаваемых в алгоритм случайного леса. Эти результаты будут ранжированы, и в качестве окончательного результата будет выбран самый высокий результат.

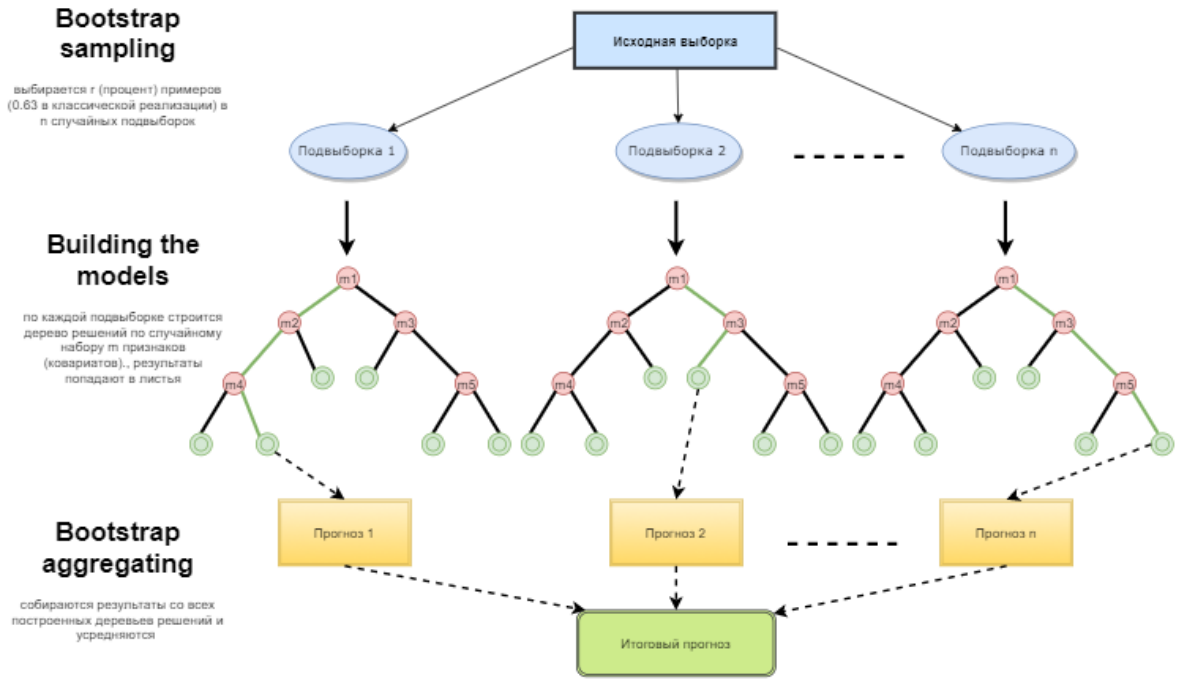

Рис. 25. Схема функционирования случайного леса [6]

2.7.2.2 Классификация в «случайном лесе»

Классификация в случайных лесах использует ансамблевую методологию для достижения результата. Данные обучения передаются для

обучения различных деревьев решений. Этот набор данных состоит из наблюдений и признаков, которые будут выбраны случайным образом во время разделения узлов.

Система лесов опирается на различные деревья решений. Каждое дерево решений состоит из узлов решений, конечных узлов и корневого узла. Листовой узел каждого дерева — это конечный результат, полученный этим конкретным деревом решений. Выбор окончательного результата осуществляется по системе мажоритарного голосования. В этом случае выход, выбранный большинством деревьев решений, становится окончательным выходом системы тропических лесов. На рисунке 25 показан простой классификатор случайного леса.

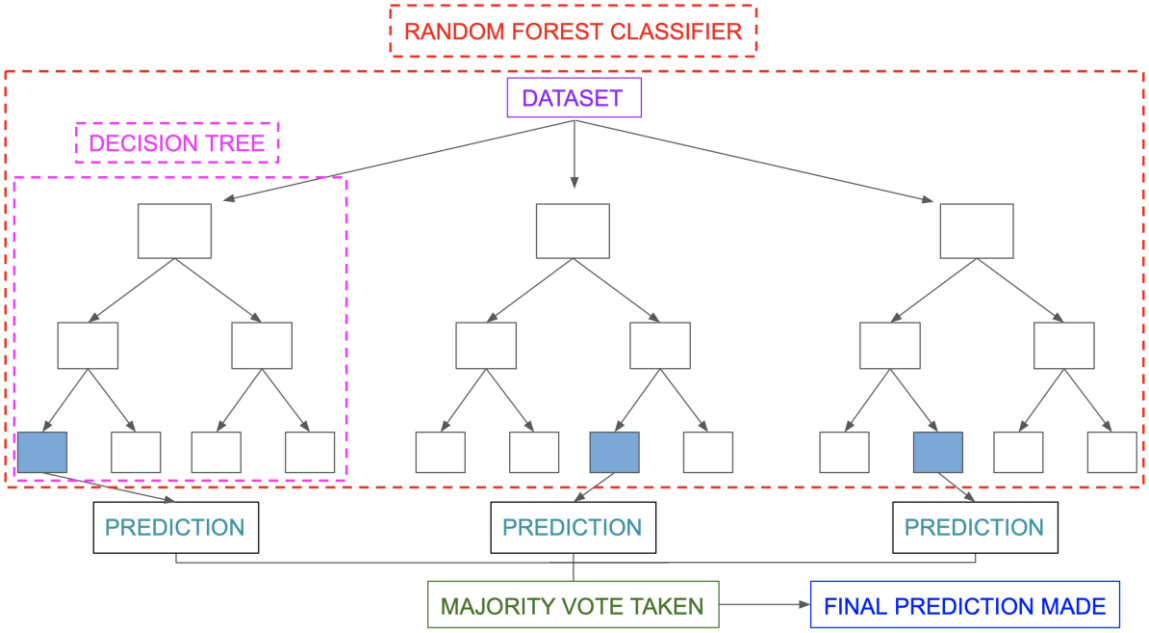

Рис. 25. Простой классификатор случайного леса

2.7.2.3 Регрессия в «случайном лесе»

Регрессия — это еще одна задача, выполняемая алгоритмом случайного леса. Регрессия случайного леса следует концепции простой регрессии. Значения зависимых (признаков) и независимых переменных передаются в модели случайного леса.

Можно запускать регрессии случайного леса в различных программах, таких как SAS, R и python. В регрессии случайного леса каждое дерево дает определенный прогноз. Средний прогноз отдельных деревьев является результатом регрессии. Это противоречит классификации случайных лесов, результат которой определяется режимом класса деревьев решений.

Хотя регрессия случайного леса и линейная регрессия следуют одной и той же концепции, они различаются по функциям. Функция линейной регрессии $y = bx + c$, где y — зависимая переменная, x — независимая переменная, b — параметр оценки, а c — константа. Функция сложной регрессии случайного леса подобна черному ящику.

Плюсы использования алгоритма случайного леса:

- Он может выполнять как задачи регрессии, так и задачи классификации;
- Случайный лес дает хорошие прогнозы, которые легко понять;
- Он может эффективно обрабатывать большие наборы данных;
- Алгоритм случайного леса обеспечивает более высокий уровень точности прогнозирования результатов по сравнению с алгоритмом дерева решений.

Минусы алгоритма случайного леса:

- При сборке случайного леса для вычисления требуется больше ресурсов;
- Это требует большего времени по сравнению с алгоритмом решений [31].

2.7.3 Многоцелевой оптимизированный случайный лес (МОСЛ)

Многоцелевой Оптимизированный Случайный Лес (МОСЛ) (или Multiobjective Optimized Random Forest model analysis (MORF)) – это алгоритм,

подход которого основан на оптимизированном и пост обработанном случайном лесе. Данный метод может хорошо обрабатывать достаточно большие наборы данных, объяснять влияние относительной силы независимых переменных и оптимизирован для уменьшения дисперсии без увеличения смещения. Имеется в виду дилемма смещения-дисперсии, которая подразумевает собой поиск удачного баланса между этими величинами. Всегда хотелось бы получить и малую дисперсию, и малое смещение, но выходит обратная картина – увеличивается смещение – уменьшается дисперсия, и наоборот.

В качестве основы для работы был выбран метод «Случайный лес», потому что он основан на деревьях решений, которые могут идентифицировать несколько конфигураций причинных условий и предоставлять способ пост обработки результирующей модели.

Они могут представить краткие изложения различных комбинаций условий, которые приводят к желаемым результатам в сложных обстоятельствах.

Если говорить об актуальности разработанного метода, то разница между существующими подходами и нашим подходом состоит в том, что в данным подходе мы «двигаемся дальше», используя метод ансамбля «Случайного Леса». Таким образом, у нас есть несколько случайно (или не случайно, зависит от настроек) построенных деревьев решений, которые мы используем для отсечения и анализа всех полученных решений. Возможно, таким образом мы справляемся с проблемой комбинаторного рода, результаты становятся менее хрупкими, и можно увидеть эффект относительно сильны наших независимых переменных.

Модели случайного леса были выбраны в качестве основы, так как их можно тестировать. Эти тесты исследуют не только то, произошел ли прогнозируемый результат при наличии ожидаемых атрибутов, но также и то, не произошел ли прогнозируемый результат при отсутствии ожидаемых атрибутов.

Процедура начальной загрузки приводит к повышению производительности модели, поскольку она уменьшает дисперсию модели без увеличения смещения. Это означает, что, хотя прогнозы одного дерева очень чувствительны к шуму в его обучающей выборке, среднее значение многих деревьев нет, если деревья не коррелированы. Простое обучение множества деревьев на одном обучающем наборе даст сильно коррелированные деревья (или даже одно и то же дерево много раз, если алгоритм обучения детерминистический); Начальная выборка — это способ упорядочить деревья, показывая им разные обучающие наборы.

В данной работе используется модифицированный алгоритм обучения по деревьям, который выбирает для каждого разделения кандидатов в процессе обучения случайное подмножество признаков. Этот процесс иногда называют «пакетированием функций». Причиной этого является корреляция деревьев в обычной выборке начальной загрузки: если один или несколько признаков являются очень сильными предикторами для переменной отклика (целевой результат), эти признаки будут выбраны во многих B-деревьях, что приведет к тому, что они станут коррелированными.

Была выявлена роль дисперсии в ожидаемой ошибке обобщения модели. В свете этих результатов разумный подход к уменьшению ошибки обобщения, таким образом, будет состоять в снижении дисперсии прогноза при условии, что соответствующее смещение можно сохранить на том же уровне или не увеличить слишком сильно. Как оказалось, ансамблевые методы представляют собой прекрасный простой способ реализовать именно эту функцию. В частности, основной принцип ансамблевых методов, основанных на рандомизации, заключается во введении случайных возмущений в процедуру обучения, чтобы создать несколько различных моделей из одного обучающего набора L, а затем объединить прогнозы этих моделей для формирования прогноза ансамбля.

Более детализировано алгоритм можно описать так:

1. Данные разбиваются на тренировочный и тестовый датасеты.

2. «Случайный лес» обучается на тренировочном датасете.
3. Каждое дерево в «лесу» разбивается в независимые кейсы:

a) Рейтинг рассчитывается для каждого узла (=condition) в дереве по формуле:

$$Condition\ rate = 1 - \log_{10}(residual\ points), \quad (20)$$

$$residual\ points == \begin{cases} condition\ threshhold, & \text{если } operator = " < " \\ 10 - condition\ threshhold, & \text{если } operator = " > " \end{cases}, \quad (21)$$

Таким образом, если вышли результаты “>2” или “<8”, то рейтинг будет низким, а в обратном случае высоким. Увеличение рейтинга не является линейным: шаг от “< 2” до “< 3” больше, чем шаг от “< 7” до “< 8”.

b) Рейтинг рассчитывается для каждой ветви (= любой путь, начинающийся с root=case) в дереве:

$$Branch\ rate = \\ = \frac{correct\ predictions * \log(1 + all\ predictions)}{all\ predictions} * \sum \frac{condition\ rate}{conditions\ count}, \quad (22)$$

Таким образом, это зависит от [case coverage] и [case precision], а также от показателей всех состояний, включенных в case. Получается, что наиболее эффективные случаи получают более высокую скорость. [Case coverage] зависимость нелинейная, шаг от 5 до 10 выборок больше, чем шаг от 45 до 50.

c) Для каждого случая выбирается исключение. Ветвь исключения включает основную ветвь и приводит к противоположной категории. То есть, если основная ветвь имеет категорию «высокая», то ее ветвь-исключение должна иметь категорию «низкая», и наоборот. Исключение делается только в том случае, если общий рейтинг ветвей main+exception выше, чем рейтинг самой основной ветви.

4. Условия во всех случаях фильтруются: те, у которых слишком много residual points (например, «> 2»), отбрасываются, потому что они «слишком безопасны» и бессмысленны.
5. Дублирующиеся условия (такие как «FOO > 3.5» и «FOO > 5») объединяются внутри кейса.
6. Кейсы фильтруются: некоторые со значениями “<2” отбрасываются, потому что они слишком «общие».
7. Кейсы сортируются по рейтингу.
8. Для каждого дерева хранятся 3 лучших результата.
9. F1-мера рассчитывается с использованием всех случаев вместе в обучающих и тестовых наборах.

а) Каждый образец в наборе оценивается по каждому случаю. Первый совпадающий случай используется для классификации выборки.

10. F1-мера начального «леса» рассчитывается на обучающих и тестовых выборках.
11. Точность (precision) и охват (coverage) рассчитываются для каждого случая отдельно на обучающих и тестовых наборах.

+ «Чистота» - являются ли случаи, найденные в конце ветви, все атрибутами одного типа (т. е. чистыми) или смесью видов.

+ «Охват» - какая доля всех существующих положительных случаев приходится на конец какой-либо конкретной ветки. В упражнениях по интеллектуальному анализу данных ветви с низким охватом часто «сокращаются», то есть удаляются из модели, чтобы уменьшить сложность модели (и, таким образом, помочь повысить ее обобщаемость).

Результаты анализа изображены на рисунке 26:

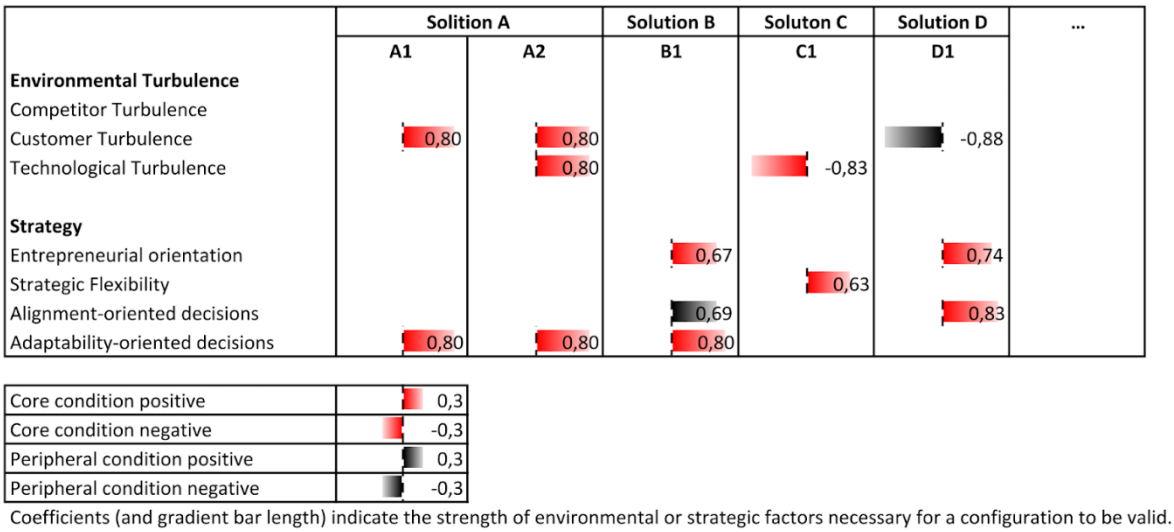

Рис. 26. Результаты МОСЛ анализа

Результаты на примере рисунка 26 показывают, что предпринимательская ориентация представляет собой агрессивную стратегическую позицию в случае турбулентности конкурентов (решение А). Утверждается, что, оказывая положительное влияние на организационный рост в целом, ценность предпринимательской стратегической позиции особенно высока в случае появления в отрасли новых конкурентов или адаптации существующих конкурентов своих конкурентных стратегий. Решения, ориентированные на согласование, представляют собой необходимое (но недостаточное) условие для организационного роста во всех решениях. Хотя решения, ориентированные на согласование, оказывают прямое и негативное влияние на организационный рост, высшие менеджеры должны подчеркивать их, чтобы обеспечить дополнительные динамические и предпринимательские способности, которые впоследствии приводят к организационному росту. Следует избегать принятия решений, ориентированных на адаптивность, в экстремальных условиях, т. е. в средах с высокой турбулентностью (высокая турбулентность конкурентов, клиентов и технологий) и в очень стабильных средах (низкая турбулентность конкурентов, клиентов и технологий) (решение С5 и С6). Этот вывод

напоминает склонность топ-менеджеров чрезмерно подчеркивать адаптацию (чрезмерное исследование) вместо того, чтобы внутренне согласовываться в особенно турбулентных условиях. Во всех конфигурациях (кроме крайне турбулентной среды) хотя бы одно из трех измерений турбулентности внешней среды является ключевым условием организационного роста. Этот вывод подтверждает важность среды как фактора непредвиденных обстоятельств при анализе факторов, ведущих к организационному росту. В конце концов, результаты показывают, что решения, ориентированные на амбидекстрию, в первую очередь приводят к организационному росту в слегка турбулентной среде. И решения, ориентированные на адаптивность, и ориентированные на выравнивание, являются высокими только в средах с высоким уровнем турбулентности в одном измерении.

2.7.4 Проблемы переобучения моделей

Переобучение — распространенное явление, на которое следует обращать внимание при обучении модели машинного обучения. Переобучение происходит, когда модель уделяет слишком много внимания конкретным деталям набора данных, на котором она обучалась. В частности, модель улавливает закономерности, характерные для наблюдений в обучающих данных, но не распространяется на другие наблюдения. Модель способна делать хорошие прогнозы на основе данных, на которых она была обучена, но не может делать хорошие прогнозы на данных, которые она не видела во время обучения.

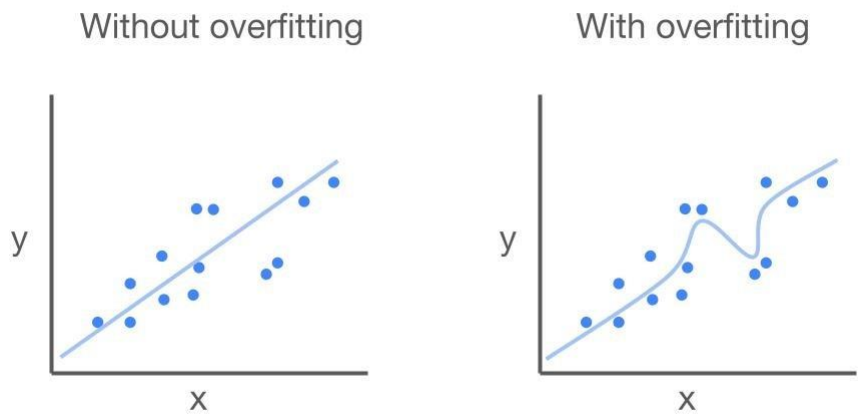

Рис. 27. Пример данных с переобучением и без

Переобучение является проблемой, потому что модели машинного обучения обычно обучаются с целью делать прогнозы на невидимых данных. Модели, которые соответствуют своему набору обучающих данных, не могут делать хорошие прогнозы на новых данных, которые они не видели во время обучения, поэтому они не могут делать прогнозы на невидимых данных.

Если планируется использовать модель машинного обучения для прогнозирования невидимых данных, всегда следует проверять, не подходит ли модель к обучающим данным. Чтобы проверить, подходит ли модель к обучающим данным, нужно обязательно разделить свой набор данных на обучающий набор данных, который используется для обучения вашей модели, и тестовый набор данных, который вообще не затрагивается во время обучения модели. Таким образом, будет доступный набор данных, который модель вообще не видела во время обучения, который вы можете использовать для оценки того, переоснащается ли ваша модель.

Как правило, нужно выделять около 70% данных для набора обучающих данных и 30% данных для набора тестовых данных. Только после того, как модель обучится на обучающем наборе данных, оптимизируется и будут подобраны наилучшие гиперпараметры, можно использовать тестовый набор данных. В этот момент можно использовать модель для

прогнозирования как тестовых данных, так и обучающих данных, а затем сравнивать показатели производительности тестовых и обучающих данных.

Если модель подходит для обучающих данных, будет заметно, что показатели производительности на обучающих данных намного лучше, чем показатели производительности на тестовых данных.

Есть несколько путей постараться избежать переобучения моделей.

Первый способ предотвратить и уменьшить переобучение — это упростить модель. Более сложные модели с большей вероятностью переобучатся, поэтому, если есть простой способ уменьшить сложность модели, это отличное начало. Точный путь, который выберется для упрощения модели, будет зависеть от типа используемой модели, но вот несколько примеров шагов, которые можно предпринять:

- Уменьшить количество переменных в модели. Простой способ уменьшить сложность модели — уменьшить количество переменных, которые используются в модели.
- Использовать более простую модель. Другой вариант — поменять используемую модель на более простую. Как правило, будет лучше использовать модель с меньшим количеством параметров, которые необходимо оценить. Например, модели линейной или логистической регрессии.
- Настройка гиперпараметров. Во многих случаях можно уменьшить сложность, не меняя тип используемой модели, используя другие гиперпараметры. Например, если используется древовидная модель, такая, как случайный лес, можно уменьшить максимальную глубину дерева модели. Модель случайного леса, состоящая из 20 деревьев глубиной всего в 1 уровень, будет иметь гораздо меньшую сложность, чем модель, состоящая из 1000 деревьев глубиной до 100 уровней.

Другой вариант, когда видно, что модель начинает переобучаться — это внести изменения в обучающие данные, которые используются для

обучения модели. Вот несколько примеров изменений, которые можно внести в обучающие данные, чтобы уменьшить переоснащение:

- Попробовать использовать больше данных. Первый вариант — просто использовать больше данных для обучения модели. Модели, обученные на небольших наборах данных, как правило, с большей вероятностью переобучаются, чем модели, обученные на больших наборах данных.
- Дополнять свои данные. Если у нет большого количества данных для обучения, можно дополнить имеющиеся данные, чтобы создать дополнительные примеры, которые можно использовать во время обучения. Увеличение данных — это процесс, с помощью которого применяются преобразования к существующим обучающим данным для создания новых синтетических данных, которые можно использовать при обучении. Это распространено в таких областях, как обработка и анализ изображений.
- Выборка более разнообразных данных. Иногда просто выборки большего количества данных недостаточно. Возможно, придется переосмыслить стратегию, которая используется для выборки данных, чтобы получить более разнообразный набор данных в дополнение к выборке большего количества точек данных. Скорее всего, это так, если большое количество примеров в вашей обучающей выборке идентичны или почти идентичны.

Третий вариант, который должен помочь предотвратить переобучение модели машинного обучения, — это настроить процедуру, используемую для обучения модели. Существует множество различных типов модификаций, которые можно внести в процедуру обучения модели, чтобы смягчить последствия переобучения.

- Регуляризация (L1, L2 и т. д.). Регуляризация — еще один распространенный выбор для предотвращения переобучения

модели. Общая идея этого метода заключается в том, что добавляется штраф, который увеличивается пропорционально размеру каждого из коэффициентов модели. Это приводит к уменьшению коэффициентов модели ближе к нулю. Когда коэффициент для данного признака приближается к нулю, этот признак эффективно удаляется из модели, что снижает сложность модели. Этот метод широко используется в самых разных моделях машинного обучения.

- Ранняя остановка. Ранняя остановка — это метод, который обычно используется для сокращения продолжительности обучения нейронных сетей. Когда нейронные сети обучаются, они обычно многократно повторяют один и тот же набор обучающих примеров. Эти модели могут стать более подходящими, поскольку они видят одни и те же примеры снова и снова. При ранней остановке обучение модели приостанавливается в определенных контрольных точках, чтобы оценить, насколько модель улучшилась с момента последней контрольной точки. Могут применяться различные правила остановки, но общая идея заключается в том, что если производительность модели значительно не улучшилась или если производительность ухудшилась, то процедура обучения останавливается досрочно.
- Исключение или дропаут. Введение отсева — еще один метод, который применяется к нейронным сетям для уменьшения переобучения. В этом методе подмножество узлов в сети выбирается так, чтобы их выходные данные отбрасывались или игнорировались в течение части процесса обучения. Это эффективно разрывает связь между этим узлом и любыми нижестоящими узлами для этой части процесса обучения. Этот метод предназначен для имитации усреднения модели (или

объединения прогнозов нескольких моделей с различной архитектурой). Различные связи разрываются на разных этапах процесса обучения, поэтому эффективная архитектура модели меняется на протяжении всего процесса обучения [16].

2.7.4.1 Переобучение моделей случайных лесов

В целом случайные леса гораздо реже переобучаются, чем другие модели, потому что они состоят из множества слабых классификаторов, которые обучаются совершенно независимо на совершенно разных подмножествах обучающих данных.

Случайные леса — отличный вариант для использования, если нужно обучить быструю модель, которая вряд ли переобучится. При этом возможно, что в некоторых случаях модель случайного леса может переобучиться, поэтому вам все равно следует следить за переобучением при обучении моделей случайного леса.

Чтобы избежать переобучения моделей случайного леса, следует придерживаться следующих пунктов:

- Уменьшить глубину дерева. Если есть опасения, что модель случайного леса переобучается, первое, что нужно сделать, это уменьшить глубину деревьев в модели случайного леса. Различные реализации моделей случайного леса будут иметь разные параметры, которые контролируют это, но обычно будет параметр, который явно контролирует количество ветвей в глубину, которые может получить дерево, количество разбиений дерева или минимальный размер узлов. Уменьшение сложности модели, как правило, улучшает проблемы переобучения, а уменьшение глубины дерева — это самый простой способ уменьшить сложность в случайных лесах.

- Уменьшить количество переменных, отобранных при каждом разделении. Также можно уменьшить количество переменных, рассматриваемых для каждого разбиения, чтобы внести больше случайности в модель. Чтобы сделать шаг назад, каждый раз, когда в дереве создается разбиение, берется подмножество переменных, и только эти переменные считаются переменными, по которым выполняется разбиение. Если рассматриваются все или большинство переменных при каждом разбиении, все деревья могут выглядеть одинаково, потому что выбраны одни и те же разбиения для одних и тех же переменных. Если рассматривается меньшее подмножество переменных при каждом разбиении, деревья с меньшей вероятностью будут выглядеть одинаково, потому что маловероятно, что одни и те же переменные были доступны для рассмотрения при каждом разбиении.
- Использовать больше данных. Всегда можно попробовать увеличить размер набора данных. Переобучение более вероятно, когда сложные модели обучаются на небольших наборах данных, поэтому увеличение размера набора данных может помочь [17].

ГЛАВА 3. Решение аналитических задач и применение модификации методов машинного обучения для анализа данных

3.1 Разведочный анализ данных

Для построения и работы с моделями машинного обучения очень важно правильно подготовить и обработать данные. В качестве данных в данной работе выступают следующие датасеты:

- Датасет с альянсами, сформированными из нескольких компаний разной величины, содержащий в себе различную информацию о данных компаниях и об альянсе в целом;
- Датасеты из Бюро ван Дайк (Bureau van Dijk или BvD), содержащие в себе информацию о компаниях из альянсов с 1990 по 2021 годы. Бюро ван Дайк – это крупная сеть, специализирующаяся на наборе данных о компаниях.

Первым этапом является обработка данных из датасета с альянсами. Сет содержит в себе информацию о годах возникновения альянса, описание деятельности альянса, SIC-коды альянсов и т. д. Другими словами, там содержатся как числовые и текстовые данные, так и бинарные, и формат даты/времени. В первую очередь нужно выяснить, какие из альянсов действуют по исследуемым бизнес-стратегиям. Для этот следует проанализировать деятельность альянсов и найти ключевые слова, которые будут очевидно указывать на принадлежность или отсутствие бизнес-стратегии.

3.1.1 Разведочный анализ данных

3.1.1.1 Работа с датасетом по альянсам

Обработка текста состоит из нескольких пунктов, которые были описаны в главе 2. В данной работе для получения альянсов, в которых

используются нужные бизнес-стратегии, используются средства библиотеки NLTK на языке python. Данная библиотека идеально подходит для обработки естественного языка. Основные модули, которые поддерживает NLTK перечислены в таблице 1.

Таблица 1

Содержание библиотеки NLTK [50]

NLTK модули	**Функциональность**
nltk.corpus	Корпус
nltk.tokenize, nltk.stem	Токенизация и стемминг
nltk.collocations	Коллокации или выражения нескольких слов, которые обычно встречаются одновременно. С помощью этой библиотеки можно строить t-тесты, критерий хи-квадрата.
nltk.tag	N-граммы, откат, HMM, TnT
nltk.classify, nltk.cluster	Деревья решений, наивный байес, метод k-средних
nltk.chunk	Регулярные выражения, n-граммы
nltk.parsing	Парсинг
nltk.sem, nltk.interence	Семантические интерпретации
nltk.metrics	Метрики оценивания
nltk.probability	Вероятность и оценка
nltk.app, nltk.chat	Приложения

В рамках работы используются библиотеки для стемминга, токенизации, обработки стоп-слов и построения n-грамм.

Чтобы отобрать нужные нам данные, нужно выделить ключевые слова, характерные для каждой из бизнес-стратегии. Далее проводится обработка и

анализ данных ключевых слов в тексте для поиска важных альянсов и отсева ненужных для дальнейшей работы данных. Ключевые слова, отобранные для каждой бизнес-стратегии отображены в таблице 2.

Таблица 2

Ключевые слова по каждой бизнес-стратегии

Бизнес-стратегия	**Ключевые слова (включая их синонимы и однокоренные)**
Вертикальная интеграция	Information, patent
Горизонтальная интеграция	Properties, plant
Открытые бизнес-модели	Equipment, leadership, open source
Иерархические цепи	System, convergence
Бизнес-экосистемы	Data, platforms, network

Для работы с текстом из колонки с описанием деятельности компаний были убраны стандартные стоп-слова, а также те, которые были выделены в ходе анализа, как «ненужные», «не несущие полезную информацию, но часто встречающиеся», была проведена лемматизация и парсинг с помощью регулярных выражений.

```
In [26]: words

Out[26]: ['joint',
          'venture',
          'manufacture',
          'transmission',
          'controlsnfluid',
          'level',
          'indicator',
          'parking',
          'brake',
          'control',
          'cable',
          'modulatorsnshifters',
          'pedal',
          'control',
          'cable',
          'throttle',
          'control',
          'system',
          'collaborate',
          'accelerate',
```

Рис. 28. Пример результата обработки и лемматизации текста

Для выявления наиболее частых словосочетаний был использован метод n-грамм. Самым оптимальным оказался подход из выборок 3 слов.

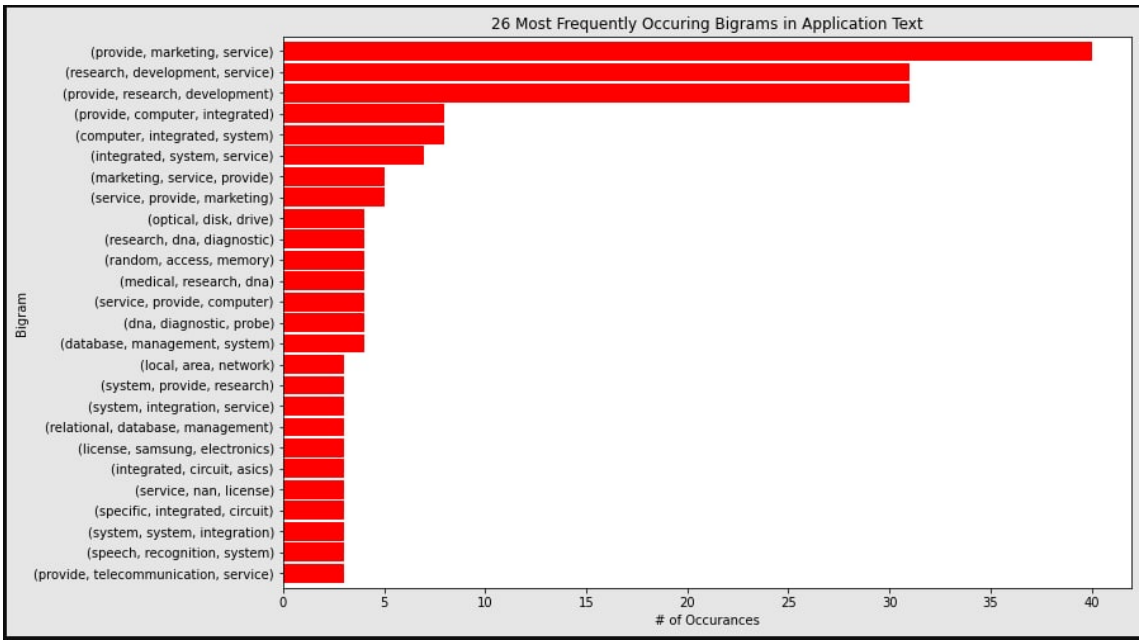

Рис. 29. Результат 3-граммы в исследуемом тексте

В ходе анализа было выявлено, что часто встречаются нужные нам ключевые слова, такие как «integrated», «relational», «network», «platform» и другие. Для отбора нужных признаков был проведен анализ каждого ключевого слова из таблицы 2 и частота его появления. Ниже представлены результат исследования ключевых слов в тексте с 1990 по 2021 год.

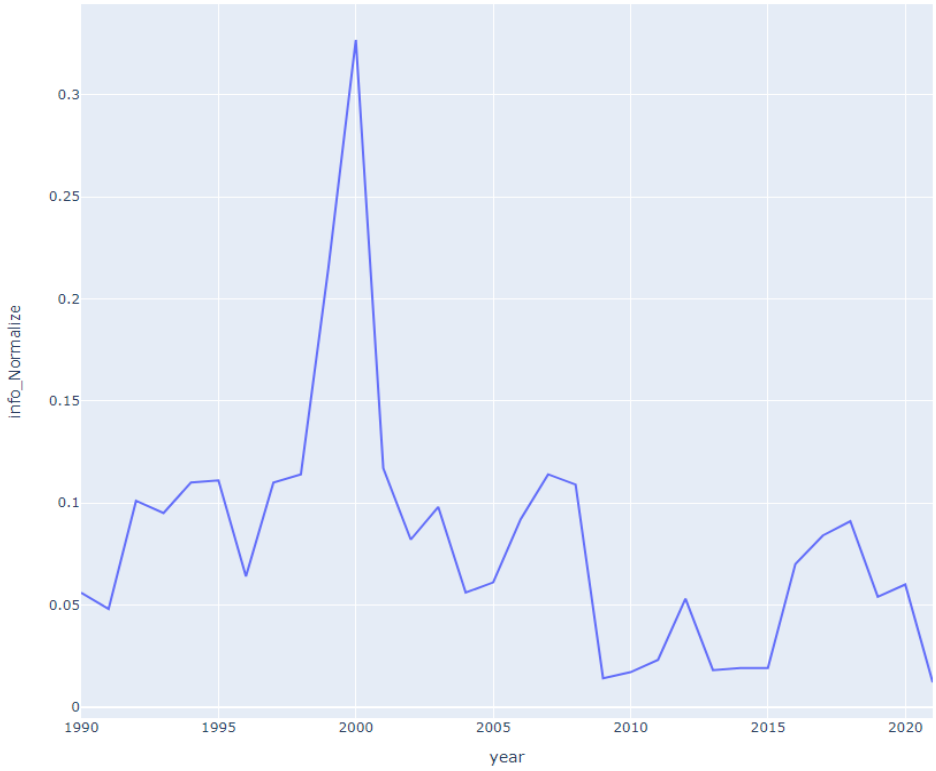

Рис. 30. Статистика слова information

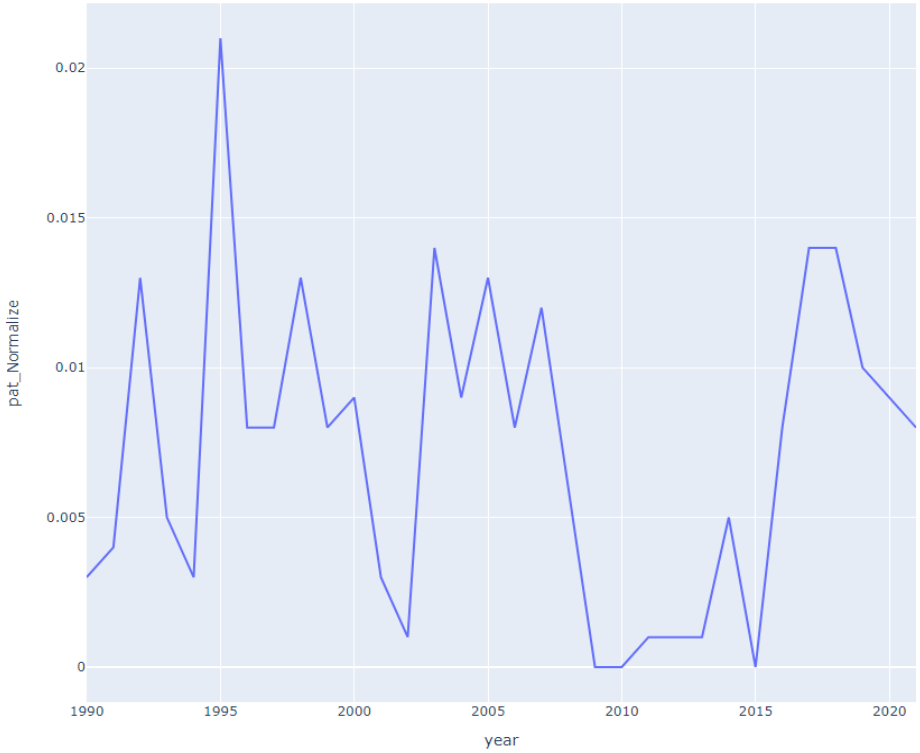

Рис. 31. Статистика слова patent

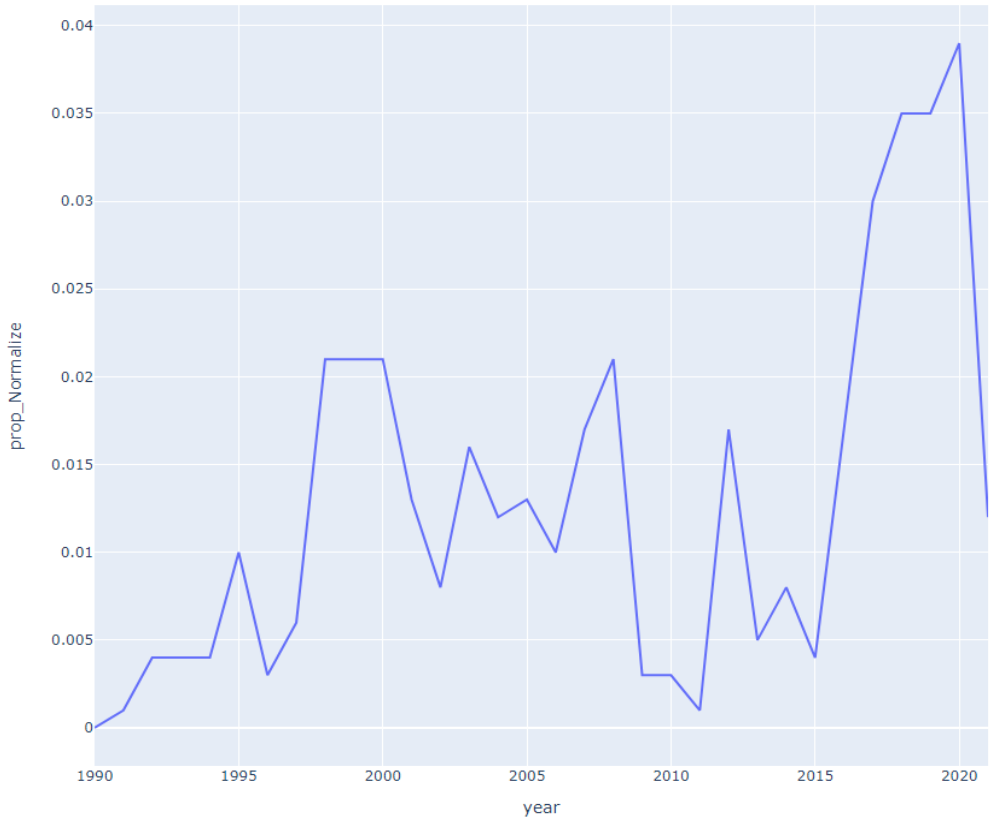

Рис. 32. Статистика слова properties

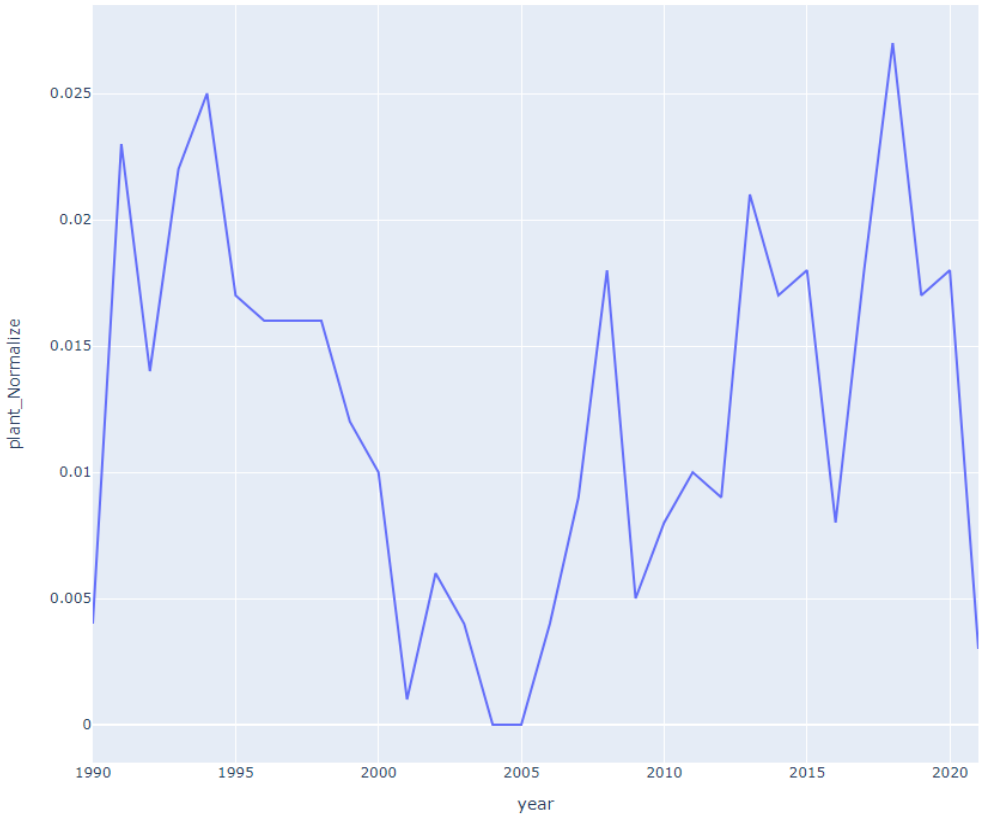

Рис. 33. Статистика слова plant

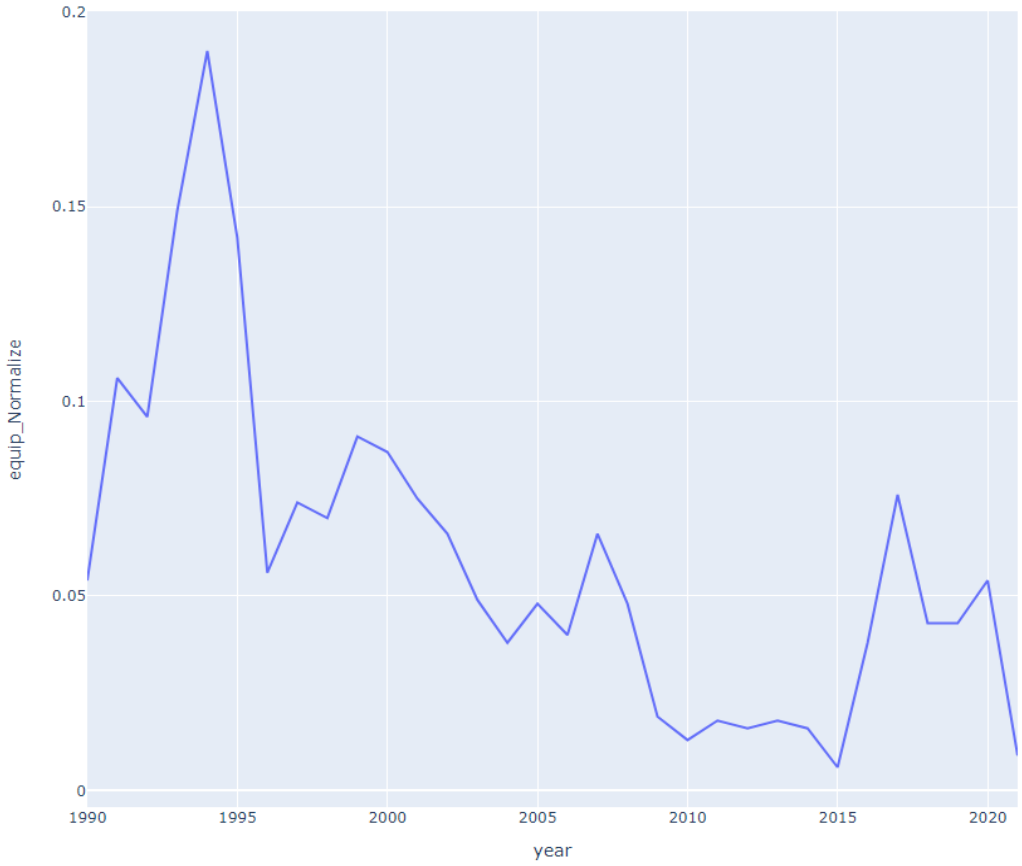

Рис. 34. Статистика слова equipment

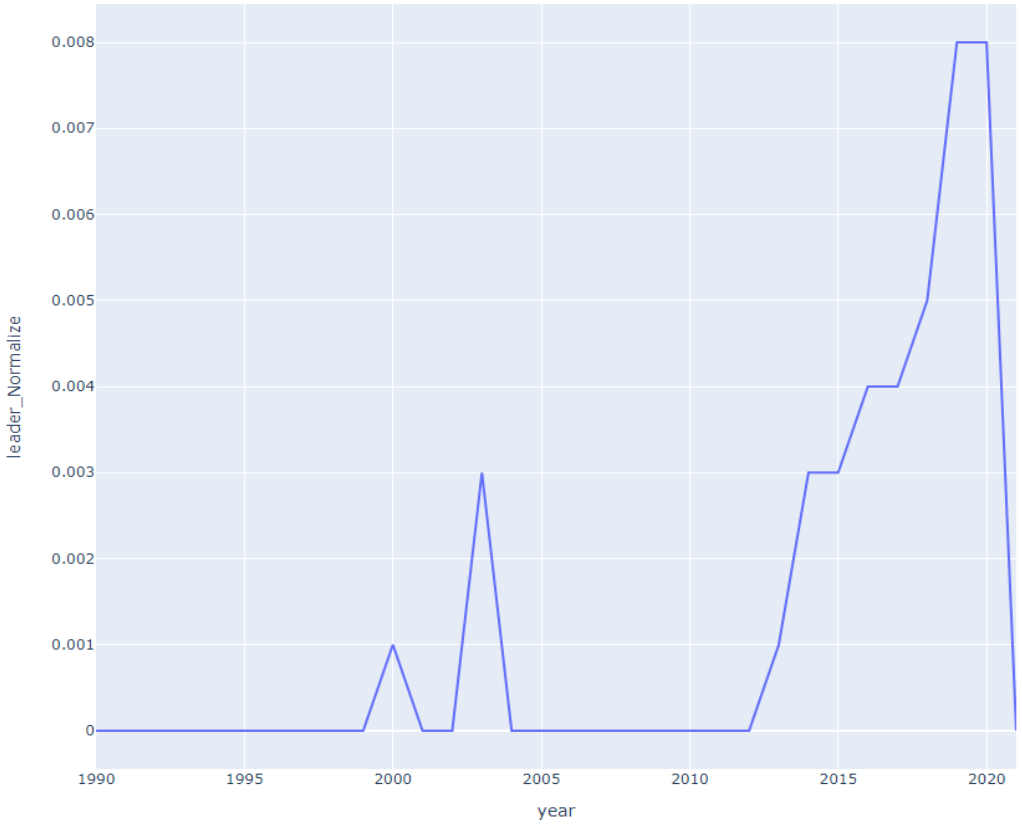

Рис. 35. Статистика слова leadership

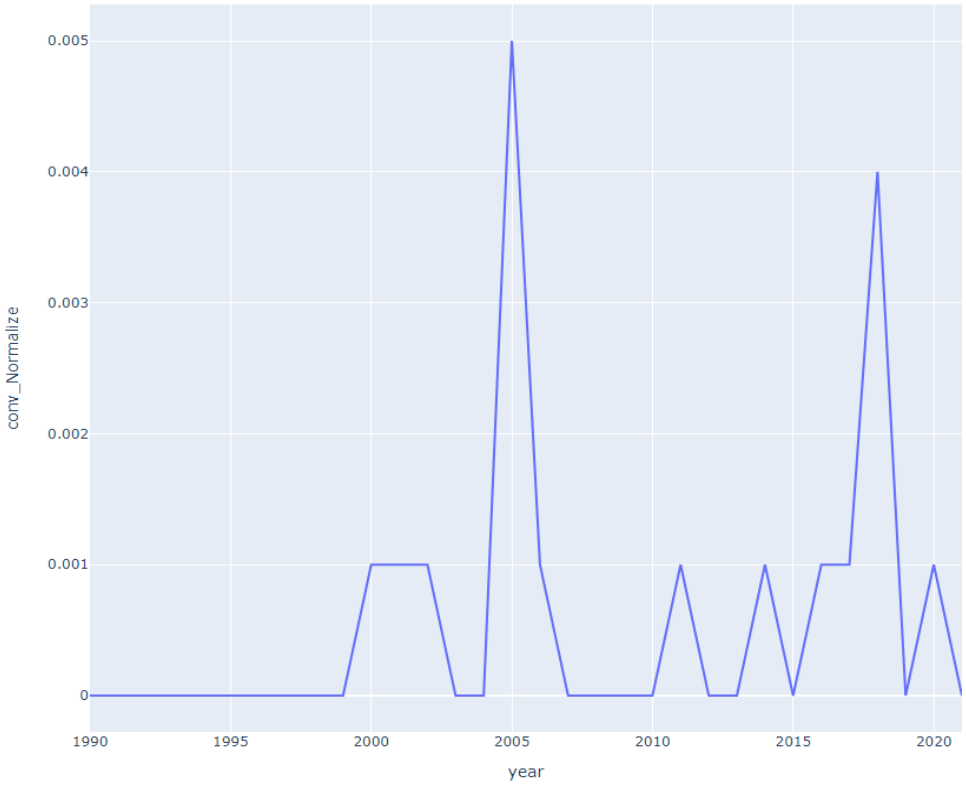

Рис. 36. Статистика слова convergence

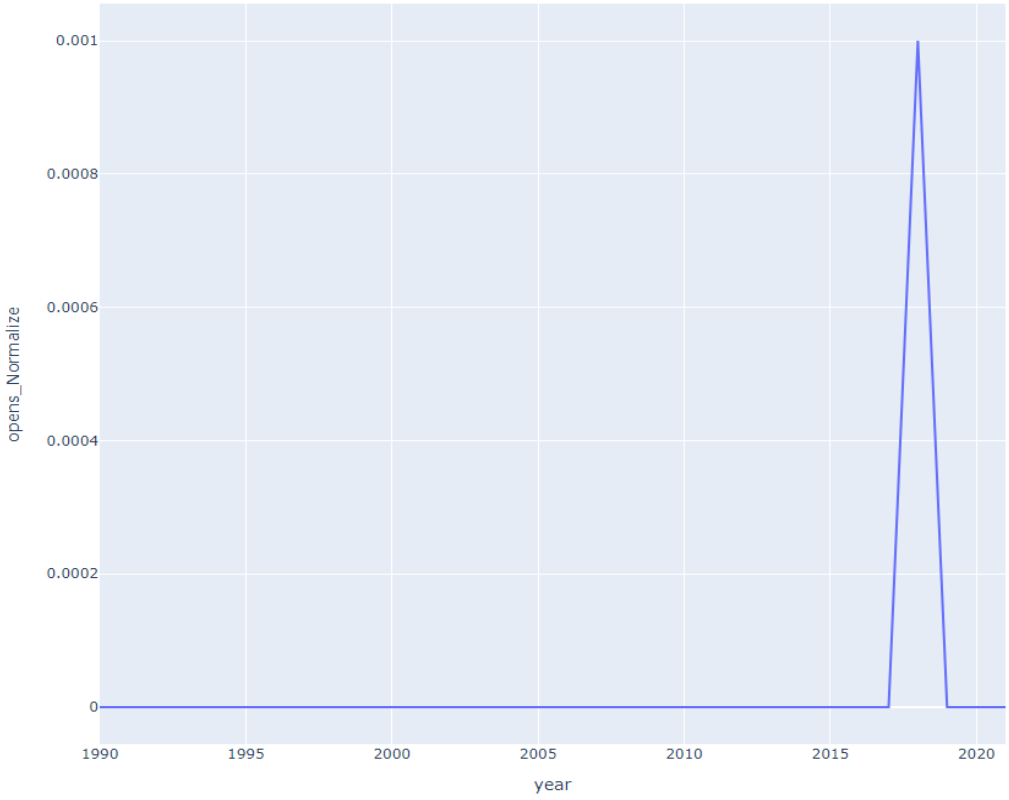

Рис. 37. Статистика слова open source

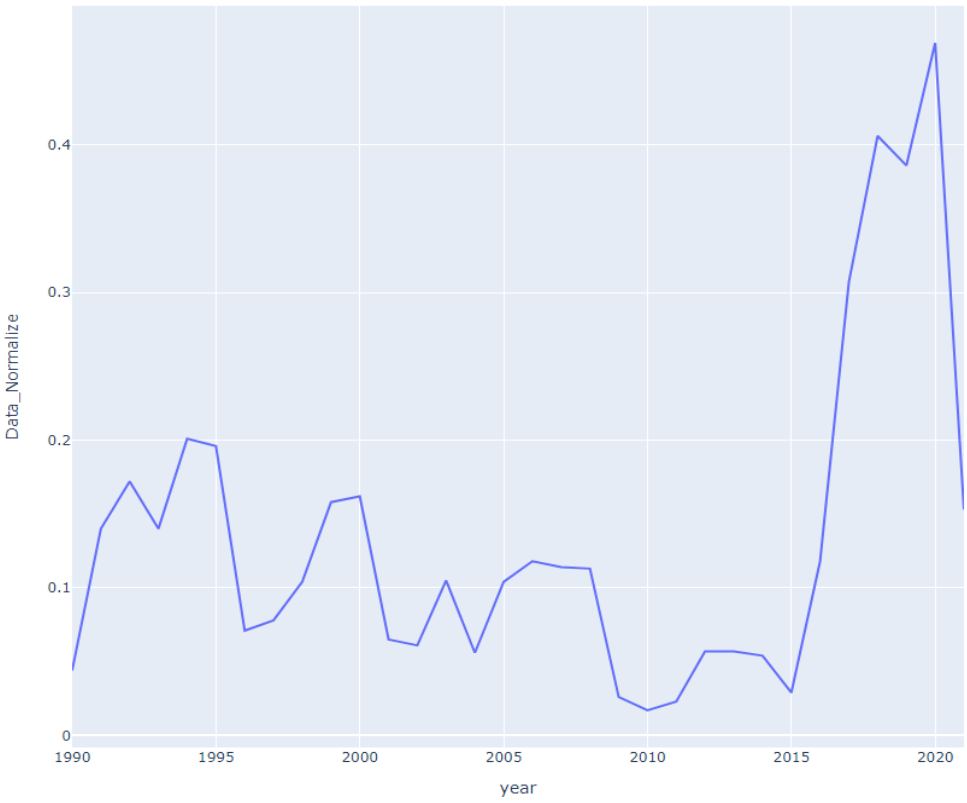

Рис. 38. Статистика слова data

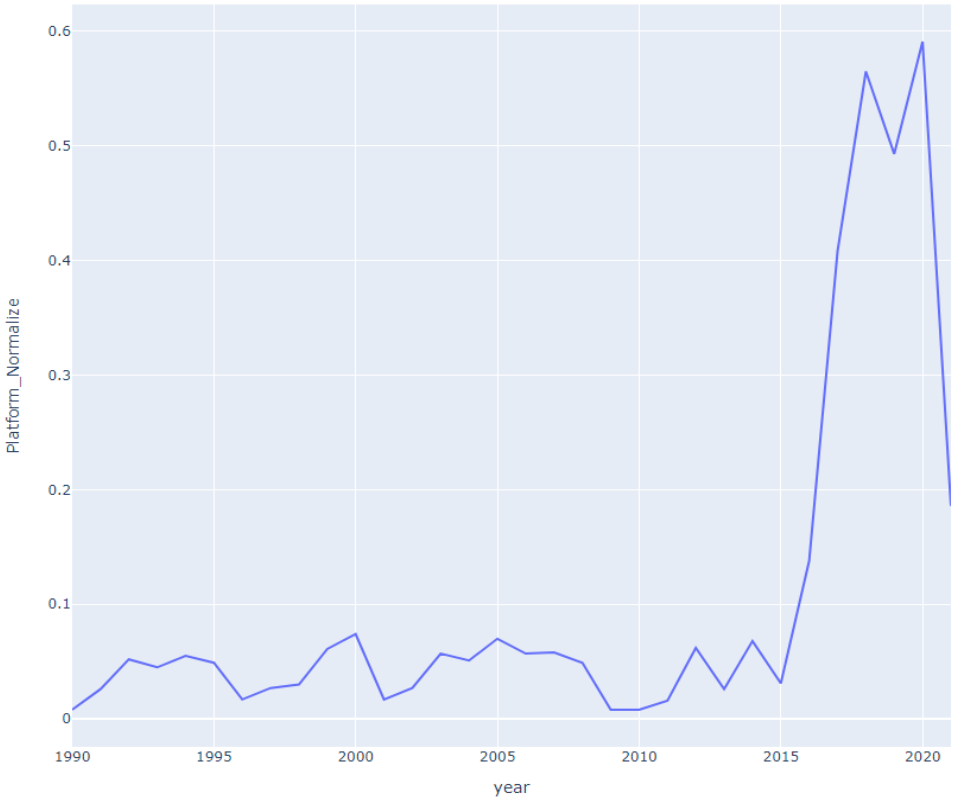

Рис. 39. Статистика слова platform

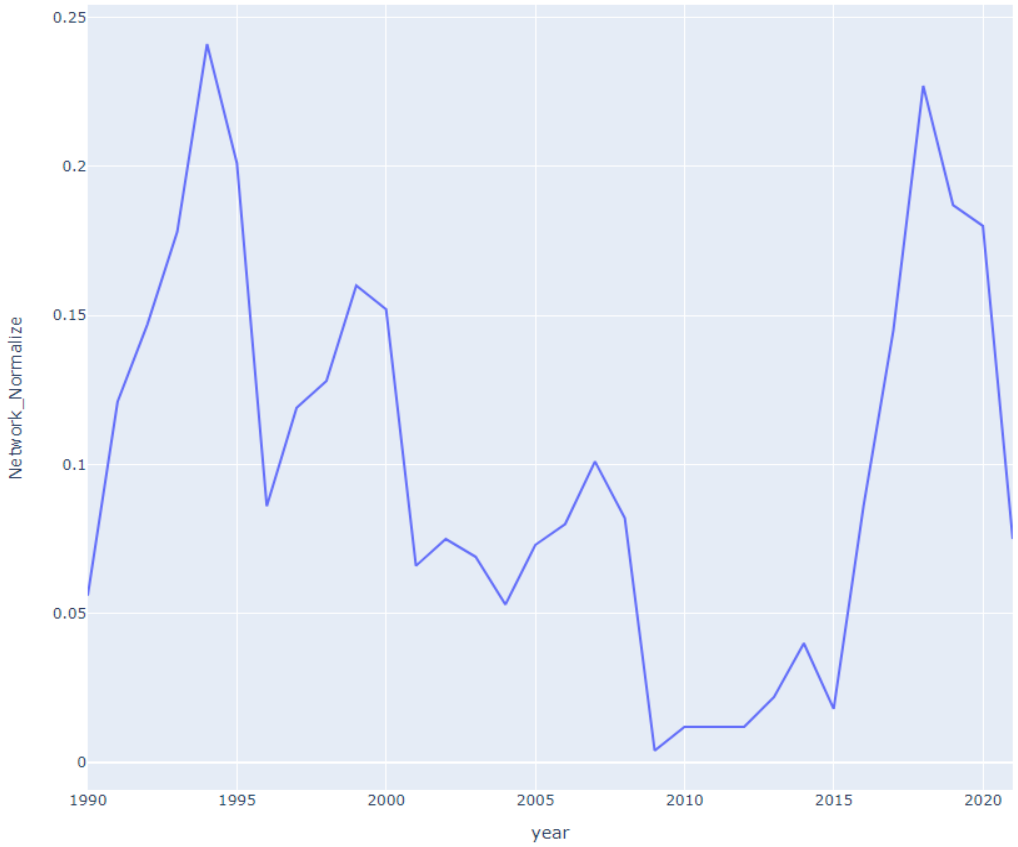

Рис. 40. Статистика слова network

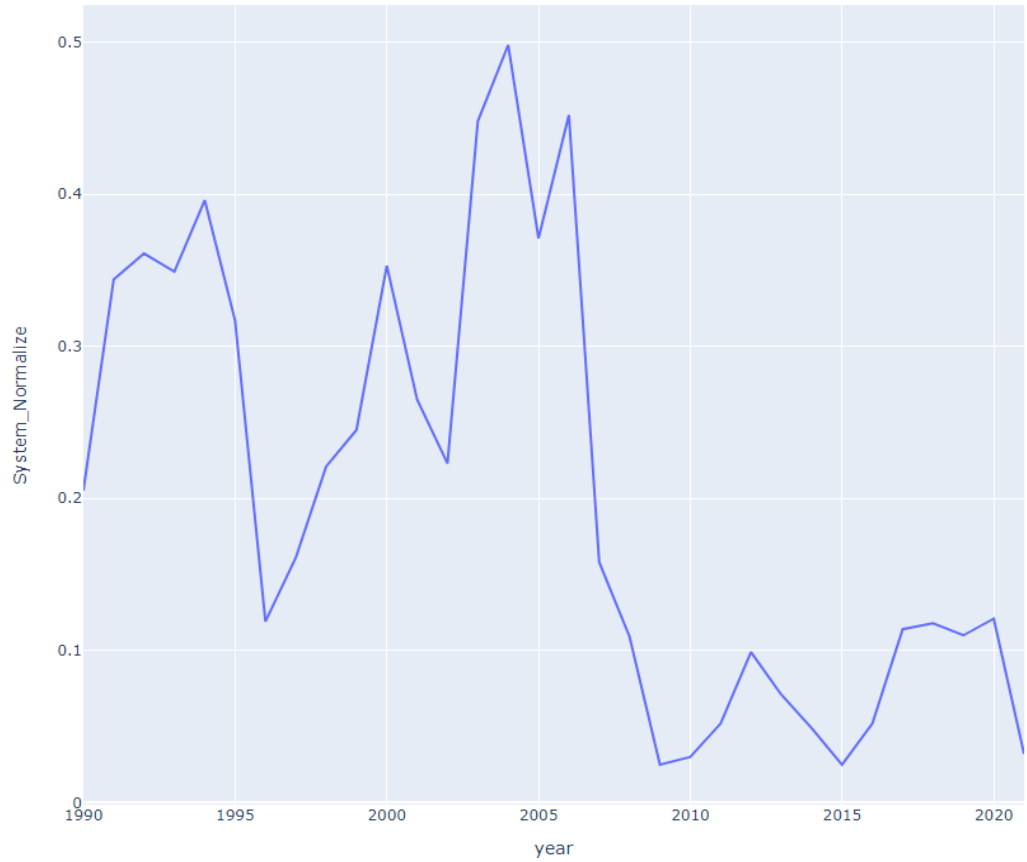

Рис. 41. Статистика слова system

Из рисунков 30–41 можно сделать вывод, что все ключевые слова встречаются в датасете. Это означает, что по каждой из бизнес-стратегий есть альянсы для анализа данных. Также, стоит обратить внимание на распространение слов по годам, так как это показывает, в какие года были наиболее или наименее распространены разные бизнес-стратегии.

Благодаря полученным результатам можно отсеять ненужные альянсы и оставить только статистически значимые. Однако, слова – это не единственный важный фактор. Достаточно важным является индустрия каждого альянса, принадлежность. За эту характеристику отвечают SIC-коды альянсов. Так как каждый альянс формирует новую команду или компанию, которая занимается своей деятельностью, у каждого альянса есть свой собственный SIC-код, невзирая на отдельные сферы деятельности каждого участника этого альянса. Для изучения влияния SIC-кодов на распространения компаний по годам была выбрана метрика «площади треугольника». Идея состоит в том, чтобы:

1. Для построения треугольников из кодов был выбран метод Евклидова расстояния. Стандартная формула Евклидова расстояния выглядит так:

$$distance(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}, \quad (23)$$

   Для вычисления координат используется остаток от деления и целая часть от деления.
2. Вычисляем периметры треугольников по следующей формуле:

$$P = \frac{1}{2}A + B + C, \quad (24)$$

3. Находим площади треугольников по формуле Герона:

$$S = \sqrt{P(P - A)(P - B)(P - C)}, \quad (25)$$

В результате получается значение площади треугольника для каждого альянса на основе их SIC-кодов. В качества итоговой обработки кодов и альянсов был построен график по годам, где в качестве оси X – вычисленные

площади треугольников, а оси Y – результат обработки ключевых слов. Внешний вид графика изображен на рисунке 42.

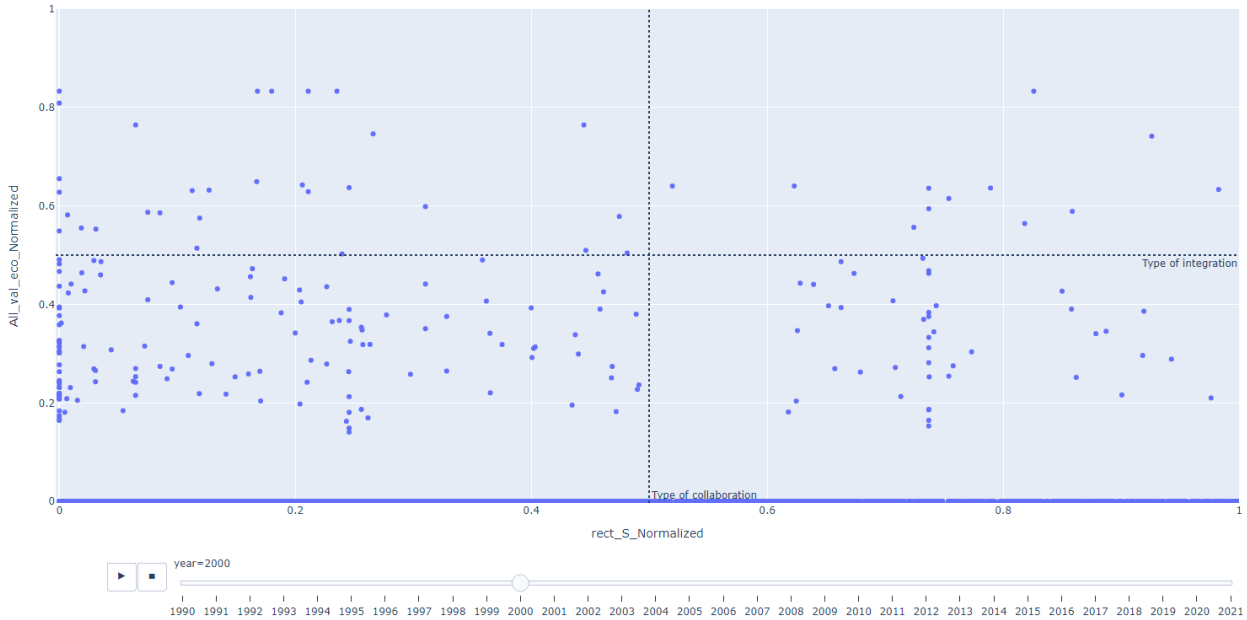

Рис. 42. Пример отображения итогового графика по альянсам

3.1.1.2 Работа с датасетом Бюро Ван Дайк

При выгрузке данных из БвД были использованы характеристики компаний, отображенные в таблице 3.

Таблица 3

Значения, используемые из БВД

DATE OF INCORPORATION (dateinc)
CATEGORY OF THE COMPANY (category_of_company)
COUNTRY (country)
FIXED ASSETS (fias)
INTANGIBLE FIXED ASSETS (ifas)
TANGIBLE FIXED ASSETS (tfas)
OTHER FIXED ASSETS (ofas)
CURRENT ASSETS (cuas)
STOCK (stok)
OTHER CURRENT ASSETS (ocas)
CASH & CASH EQUIVALENT (cash)
TOTAL ASSETS (toas)
SHAREHOLDERS FUNDS (shfd)
CAPITAL (capi)
OTHER SHAREHOLDERS FUNDS (osfd)

NON-CURRENT LIABILITIES (ncli)
LONG TERM DEBT (ltdb)
OTHER NON-CURRENT LIABILITIES (oncl)
PROVISIONS (prov)
CURRENT LIABILITIES (culi)
TOTAL SHAREH. FUNDS & LIAB. (tshf)
WORKING CAPITAL (wkca)
NET CURRENT ASSETS (ncas)
ENTERPRISE VALUE (enva)
NUMBER OF EMPLOYEES (empl)
OPERATING REVENUE (TURNOVER) (opre)
SALES (turn)
COSTS OF GOODS SOLD (cost)
GROSS PROFIT (gros)
OTHER OPERATING EXPENSES (oope)
OPERATING P/L [=EBIT] (oppl)
FINANCIAL REVENUE (fire)
FINANCIAL EXPENSES (fiex)
FINANCIAL P/L (fipl)
P/L BEFORE TAX (plbt)
TAXATION (taxa)
P/L AFTER TAX (plat)
EXTR. AND OTHER REVENUE (exre)
EXTR. AND OTHER EXPENSES (exex)
EXTR. AND OTHER P/L (extr)
P/L FOR PERIOD [=NET INCOME] (pl)
EXPORT REVENUE (expt)
MATERIAL COSTS (mate)
COSTS OF EMPLOYEES (staf)
DEPRECIATION & AMORTIZATION (depr)
INTEREST PAID (inte)
RESEARCH & DEVELOPMENT EXPENSES (rd)
CASH FLOW (cf)
ADDED VALUE (av)
EBITDA (ebta)
ROE USING P/L BEFORE TAX (%) (rshf)
ROCE USING P/L BEFORE TAX (%) (rcem)
ROA USING P/L BEFORE TAX (%) (rtas)
ROE USING NET INCOME (%) (roe)
ROCE USING NET INCOME (%) (roce)
ROA USING NET INCOME (%) (roa)
PROFIT MARGIN (%) (prma)
GROSS MARGIN (%) (grma)

EBITDA MARGIN (%) (etma)
EBIT MARGIN (%) (ebma)
CASH FLOW / OPERATING REVENUE (%) (cfop)
ENTERPRISE VALUE / EBITDA (X) (evet)
MARKET CAP / CASH FLOW FROM OPERATIONS (X) (mkcf)
NET ASSETS TURNOVER (X) (nat)
INTEREST COVER (X) (ic)
STOCK TURNOVER (X) (stot)
COLLECTION PERIOD (DAYS) (coll)
CREDIT PERIOD (DAYS) (crpe)
EXPORT REVENUE / OPERATING REVENUE (%) (exop)
R&D EXPENSES / OPERATING REVENUE (%) (rdop)
CURRENT RATIO (X) (curr)
LIQUIDITY RATIO (X) (liqr)
SHAREHOLDERS LIQUIDITY RATIO (X) (shlq)
SOLVENCY RATIO (ASSET BASED) (%) (solr)
SOLVENCY RATIO (LIABILITY BASED) (%) (soll)
GEARING (%) (gear)
PROFIT PER EMPLOYEE (TH) (ppe)
OPERATING REVENUE PER EMPLOYEE (TH) (tpe)
COSTS OF EMPLOYEES / OPERATING REVENUE (%) (sct)
AVERAGE COST OF EMPLOYEE (TH) (ace)
SHAREHOLDERS FUNDS PER EMPLOYEE (TH) (sfpe)
WORKING CAPITAL PER EMPLOYEE (TH) (wcpe)
TOTAL ASSETS PER EMPLOYEE (TH) (tape)
ESTIMATED OPERATING REVENUE (df_oprev)
ESTIMATED EMPLOYEES (df_employees)
OPERATING REVENUE ORIGINAL RANGE VALUE (op_rev_original_range_value)
EMPLOYEES ORIGINAL RANGE VALUE (emp_orig_range_value)
NUMBER OF MONTHS - from cashflow data (cashflow_nr_months)
AUDIT STATUS - from cashflow data (cashflow_audstatus)
ACCOUNTING PRACTICE - from cashflow data (cashflow_accpractice)
ORIGINAL UNITS - from cashflow data (cashflow_orig_units)
ORIGINAL CURRENCY - from cashflow data (cashflow_orig_currency)
EXCHANGE RATE FROM ORIGINAL CURRENCY - from cashflow data (cashflow_exchrate)
NET INCOME (onet)

Также при выгрузке данных были учтены годы оснований компаний, так как очень важно, чтобы использовались корректные данные правильных

компаний. Для объединения выгруженных данных с таблицей по альянсам использовались нормализованные имена компаний и годы их основания.

Для нормализации текстовых имен компаний произвелась обработка текстовых данных:

- Удаление знаков препинания;
- Приведение всех букв к одинаковому регистру;
- Удаление аббревиатур в наименованиях компаний (ltd, llc и т.д.);
- Проверка на целостность имен.

Так как цель данной работы состоит в определении, какие факторы наиболее и наименее влияют на выживаемость компаний, важно определить, как переменные из таблицы 3 коррелируют с целевой переменной выживаемости. Для построения карты корреляций использовался подход корреляции по Спирмену, так как реальные данные могут содержать много выбросов, что не является проблемой для данного подхода. Полученные результаты корреляций можно наблюдать в таблице 4.

Таблица 4

Корреляции признаков с целевой переменной Survival

Features	**SURVIVAL**
EXTR. AND OTHER EXPENSES (exex)	-0,5
EXPORT REVENUE (expt)	-0,498461648
MATERIAL COSTS (mate)	-0,288837988
STOCK (stok)	-0,201345289
FINANCIAL P/L (fipl)	-0,193585344
CURRENT RATIO (X) (curr)	-0,192950038

EXTR. AND OTHER REVENUE (exre)	-0,184812331
CAPITAL (capi)	-0,143597174
NET ASSETS TURNOVER (X) (nat)	-0,116886114
NUMBER OF EMPLOYEES (empl)	-0,114579103
LIQUIDITY RATIO (X) (liqr)	-0,095931546
CREDIT PERIOD (DAYS) (crpe)	-0,089260557
PROVISIONS (prov)	-0,088539669
EXTR. AND OTHER P/L (extr)	-0,084216159
WORKING CAPITAL PER EMPLOYEE (TH) (wcpe)	-0,072323952
SHAREHOLDERS LIQUIDITY RATIO (X) (shlq)	-0,059667824
WORKING CAPITAL (wkca)	-0,040303887
SOLVENCY RATIO (ASSET BASED) (%) (solr)	-0,02717188
OTHER NON-CURRENT LIABILITIES (oncl)	-0,021092579
OTHER CURRENT ASSETS (ocas)	-0,015203632
CURRENT LIABILITIES (culi)	-0,010052143
NET CURRENT ASSETS (ncas)	-0,001515786
CURRENT ASSETS (cuas)	-7,4244E-05
EXPORT REVENUE / OPERATING REVENUE (%) (exop)	0,00494751
TANGIBLE FIXED ASSETS (tfas)	0,007994333

CASH & CASH EQUIVALENT (cash)	0,012851667
TOTAL ASSETS (toas)	0,014212816
TOTAL SHAREH. FUNDS & LIAB. (tshf)	0,014212816
SHAREHOLDERS FUNDS PER EMPLOYEE (TH) (sfpe)	0,015217565
DEPRECIATION & AMORTIZATION (depr)	0,022810576
COSTS OF EMPLOYEES / OPERATING REVENUE (%) (sct)	0,030740848
DATE OF INCORPORATION (dateinc)	0,032546639
SOLVENCY RATIO (LIABILITY BASED) (%) (soll)	0,05070787
OTHER FIXED ASSETS (ofas)	0,051704313
SHAREHOLDERS FUNDS (shfd)	0,055882512
INTANGIBLE FIXED ASSETS (ifas)	0,057117748
OTHER SHAREHOLDERS FUNDS (osfd)	0,062139102
GEARING (%) (gear)	0,063665248
AVERAGE COST OF EMPLOYEE (TH) (ace)	0,071924313
NON-CURRENT LIABILITIES (ncli)	0,07780716
LONG TERM DEBT (ltdb)	0,080994763
INTEREST PAID (inte)	0,085035775
COSTS OF GOODS SOLD (cost)	0,088475612
OPERATING REVENUE (TURNOVER) (opre)	0,088895763

FIXED ASSETS (fias)	0,097080571
TOTAL ASSETS PER EMPLOYEE (TH) (tape)	0,104013843
FINANCIAL REVENUE (fire)	0,110909766
OPERATING REVENUE PER EMPLOYEE (TH) (tpe)	0,149064474
COLLECTION PERIOD (DAYS) (coll)	0,152380258
EXCHANGE RATE FROM ORIGINAL CURRENCY – from cashflow data (cashflow_exchrate)	0,167771024
TAXATION (taxa)	0,179918374
OTHER OPERATING EXPENSES (oope)	0,212562558
FINANCIAL EXPENSES (fiex)	0,21618293
CASH FLOW (cf)	0,220312742
GROSS MARGIN (%) (grma)	0,222541443
EBITDA MARGIN (%) (etma)	0,223728897
EBITDA (ebta)	0,226492642
SALES (turn)	0,227842468
CASH FLOW / OPERATING REVENUE (%) (cfop)	0,233902611
STOCK TURNOVER (X) (stot)	0,23432909
GROSS PROFIT (gros)	0,241876682
ROCE USING P/L BEFORE TAX (%) (rcem)	0,2742159

P/L AFTER TAX (plat)	0,280205351
OPERATING P/L [=EBIT] (oppl)	0,283913829
P/L BEFORE TAX (plbt)	0,284831368
PROFIT PER EMPLOYEE (TH) (ppe)	0,29497267
R&D EXPENSES / OPERATING REVENUE (%) (rdop)	0,310252155
ROE USING P/L BEFORE TAX (%) (rshf)	0,315991952
ROE USING NET INCOME (%) (roe)	0,31664706
P/L FOR PERIOD [=NET INCOME] (pl)	0,317896018
EBIT MARGIN (%) (ebma)	0,323384786
ADDED VALUE (av)	0,324602186
PROFIT MARGIN (%) (prma)	0,33156018
ROCE USING NET INCOME (%) (roce)	0,347667643
TLG_sum(Sum of Top-Line Growth values)	0,350129864
ENTERPRISE VALUE (enva)	0,363322722
RESEARCH & DEVELOPMENT EXPENSES (rd)	0,37248835
ROA USING P/L BEFORE TAX (%) (rtas)	0,37986231
ROA USING NET INCOME (%) (roa)	0,385920026
INTEREST COVER (X) (ic)	0,389881482
NET INCOME (onet)	0,39404824

ENTERPRISE VALUE / EBITDA (X) (evet)	0,578122663
MARKET CAP / CASH FLOW FROM OPERATIONS (X) (mkcf)	0,637027463
SURVIVAL	1

Исходя из результатов в таблице 4 можно сделать вывод, что нет переменных, которые бы сильно коррелировали с целевой переменной и привели бы к мультиколлинеарности. Ни одна из переменных не превышает пороговое значение 0.7. Некоторые переменные отрицательно коррелированы, но сильной отрицательной корреляции нет.

3.1.1.3 Оценка линейности параметров

Для более глубокой оценки взаимосвязи переменных и лучшей оценки параметров модели, в качестве математического анализа подходит построение множественной линейной регрессии. Однако, чтобы регрессия была корректной и имела физический смысл важно, чтобы каждая переменная в расчете регрессии имела линейную зависимость с целевой переменной. Чтобы найти подходящие переменные нужно провести оценку линейности.

В оценке на линейность участвовали все переменные из таблицы 1. Наиболее подходящие параметры, имеющие тенденции к линейной зависимости изображены на рисунках 43–52 в таблице 5.

Таблица 5

Переменные, имеющие тенденцию к линейной зависимости

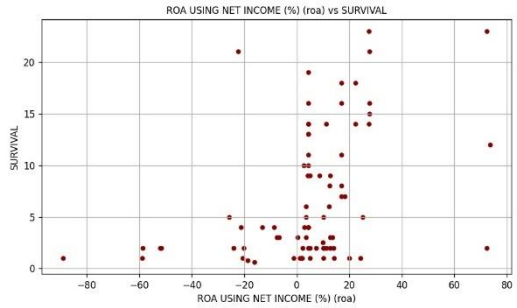

Рис. 43. roa vs survival

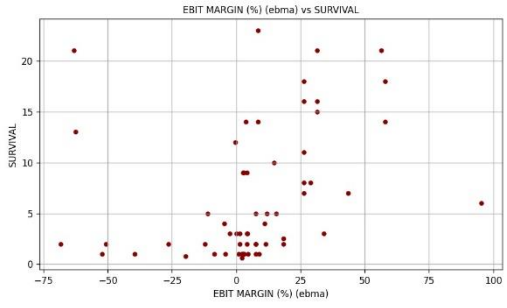

Рис. 44. ebma vs survival

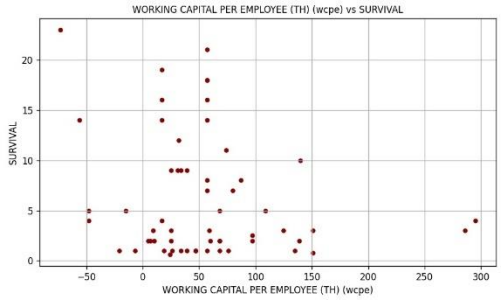

Рис. 45. wcpe vs survival

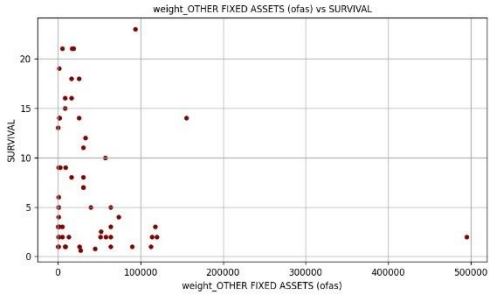

Рис. 46. ofas vs survival

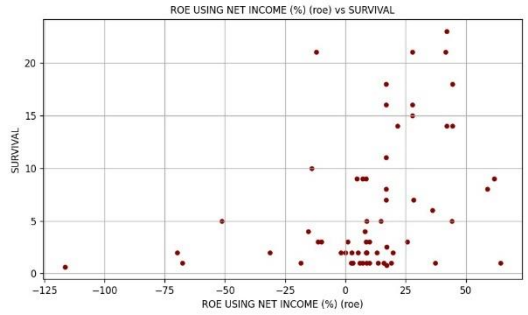

Рис. 47. roe vs survival

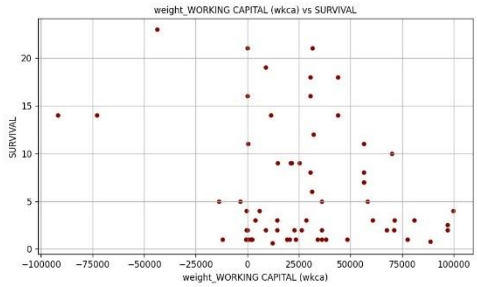

Рис. 48. wkca vs survival

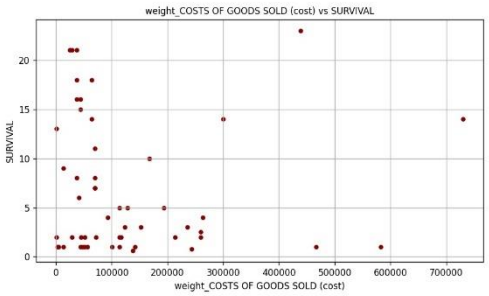

Рис. 49. cost vs survival

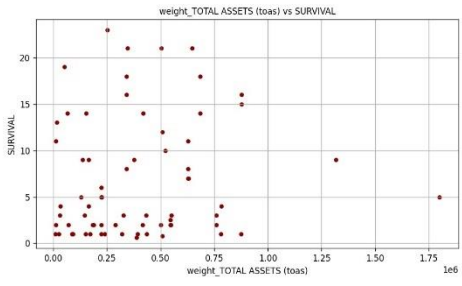

Рис. 50. toas vs survival

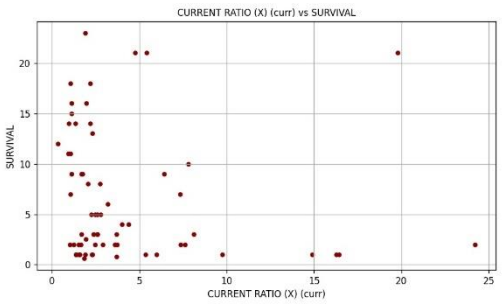

Рис. 51. Current ratio vs survival

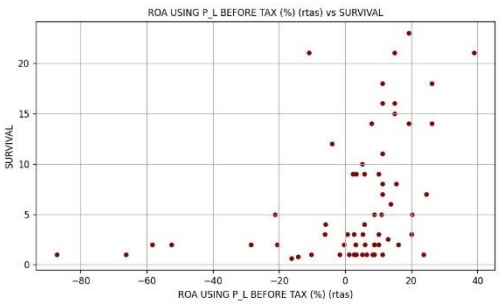

Рис. 52. rtas vs survival

3.2 Решение задач множественной линейной регрессии

Учитывая весь предыдущий анализ данных, оценку линейности переменных и подбор параметров были подобраны оптимальные переменные, которые дали наилучший результат при множественной линейной регрессии. Математическая модель данной регрессии можно записать следующим образом:

$$Survival = 2.1246 + 1.4599 * ROA\ Using\ Net\ Income + 0.5630 * Ebit\ Margin + 0.9786 * Cash\ Ratio + 1.2470 * Working\ Capital\ Per\ Employee - 1.5204 * weight\ Other\ Fixed\ Assets + 1.1087 * weight\ Provisions - 1.2670 * weight\ Working\ Capital + 1.2827 * weight\ Costs\ Of\ Goods\ Sold - 0.7511 * weight\ Total\ Assets - 2.1811 * Operating\ cash\ flow\ ratio - 1.3903 * Current\ ratio + e, (26)$$

Каждую переменную можно расшифровать следующим образом:

ROA Using Net Income - термин рентабельность активов (ROA) относится к финансовому коэффициенту, который показывает, насколько прибыльна компания по отношению к ее совокупным активам [23].

Ebit Margin - это финансовый коэффициент, который измеряет прибыльность компании, рассчитанную без учета влияния процентов и налогов [48].

Cash Ratio - мера ликвидности компании. Он специально рассчитывает отношение общей суммы денежных средств и их эквивалентов компании к ее текущим обязательствам [27].

Working Capital Per Employee - может быть связан с оборотным капиталом с точки зрения человеческих активов и затрат, связанных с работниками [15].

Other Fixed Assets - могут включать здания, компьютерное оборудование, программное обеспечение, мебель, землю, машины и транспортные средства [28].

Provisions - представляют собой средства, отложенные компанией для покрытия ожидаемых убытков в будущем. Другими словами, резерв представляет собой обязательство с неопределенными сроками и суммой [49].

Working Capital - представляет собой разницу между текущими активами компании, такими как денежные средства, дебиторская задолженность/неоплаченные счета клиентов, запасы сырья и готовой продукции, и ее текущими обязательствами, такими как кредиторская задолженность и долги [19].

Costs Of Goods Sold - это общая сумма, уплаченная бизнесом в качестве затрат, непосредственно связанных с продажей продукции [37].

Total Assets - относятся к общей сумме активов, принадлежащих физическому или юридическому лицу. Активы представляют собой объекты экономической ценности, которые расходуются с течением времени, чтобы принести выгоду владельцу. Если владельцем является бизнес, эти активы обычно регистрируются в бухгалтерских записях и появляются в балансе бизнеса [51].

Operating cash flow ratio - это показатель того, сколько раз компания может погасить текущие долги денежными средствами, полученными в течение одного и того же периода [22].

Current ratio - это коэффициент ликвидности, который измеряет способность компании погасить краткосрочные обязательства или обязательства со сроком погашения в течение одного года [18].

3.2.1 Нормализация данных и очистка от выбросов

Исходные данные имеют очень большую размерность и отображаются в экспоненциальном формате. Для корректной работы алгоритма множественной линейной регрессии и правильной оценки результатов необходимо нормализовать данные. Для нормализации использовался инструмент библиотеки scikit learn – MinMaxScaler.

Для нормализации будет использоваться конфигурация масштабирования значений в диапазоне от 1 до 7. Сначала экземпляр MinMaxScaler определяется с гиперпараметрами по умолчанию. После определения вызывается функция fit_transform() и передается в набор данных, чтобы создать преобразованную версию изначального набора данных. На рисунке 53 отображен пример использования этой функции.

```
scaler=MinMaxScaler(feature_range=(1,7))

df_research['SURVIVAL'] = scaler.fit_transform(df_research[['SURVIVAL']])
```

Рис. 53. Пример использования MinMaxScaler

Также, для увеличения линейности и точности отработки модели, необходимо очистить ключевые данные от выбросов. Как было указано в главе 2, для очистки от выбросов использовалась оценка точечных диаграмм и удаление данных, которые сильно «отбивались» от общей выборки.

3.2 2 Построение регрессионной модели с помощью statsmodel

Statsmodels — это модуль Python, который предоставляет классы и функции для оценки множества различных статистических моделей, а также

для проведения статистических тестов и исследования статистических данных. Для каждого оценщика доступен обширный список результатов статистики. Результаты проверяются на соответствие существующим статистическим пакетам, чтобы убедиться, что они верны.

Данный модуль также содержит возможность построения регрессии. Для нашей цели был взят МНК – метод наименьших квадратов (по-английски OLS).

Результаты множественной линейной регрессии с описанными ранее параметрами представлены на рисунке 54.

OLS Regression Results

Dep. Variable:	SURVIVAL	R-squared:	0.815
Model:	OLS	Adj. R-squared:	0.713
Method:	Least Squares	F-statistic:	8.015
Date:	Mon, 30 May 2022	Prob (F-statistic):	3.61e-05
Time:	03:45:34	Log-Likelihood:	-36.846
No. Observations:	32	AIC:	97.69
Df Residuals:	20	BIC:	115.3
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.1246	1.923	1.105	0.282	-1.888	6.137
ROA USING NET INCOME (%) (roa)	1.4599	0.360	4.056	0.001	0.709	2.211
EBIT MARGIN (%) (ebma)	0.5630	0.278	2.026	0.056	-0.017	1.143
Cash ratio	0.9786	0.432	2.264	0.035	0.077	1.880
WORKING CAPITAL PER EMPLOYEE (TH) (wcpe)	1.2470	0.371	3.361	0.003	0.473	2.021
weight_OTHER FIXED ASSETS (ofas)	-1.5204	0.325	-4.675	0.000	-2.199	-0.842
weight_PROVISIONS (prov)	1.1087	0.533	2.078	0.051	-0.004	2.222
weight_WORKING CAPITAL (wkca)	-1.2670	0.414	-3.062	0.006	-2.130	-0.404
weight_COSTS OF GOODS SOLD (cost)	1.2827	0.291	4.415	0.000	0.677	1.889
weight_TOTAL ASSETS (toas)	-0.7511	0.359	-2.093	0.049	-1.500	-0.002
Operating cash flow ratio	-2.1811	0.565	-3.863	0.001	-3.359	-1.003
Current ratio	-1.3903	1.037	-1.341	0.195	-3.554	0.773

Omnibus:	0.432	Durbin-Watson:	2.372
Prob(Omnibus):	0.806	Jarque-Bera (JB):	0.511
Skew:	0.243	Prob(JB):	0.774
Kurtosis:	2.616	Cond. No.	133.

Рис. 54. Множественная линейная регрессия

Оценивая результаты полученной регрессии, можно сделать следующие выводы:

- Пять переменных имеют отрицательный коэффициент. Это значит, что чем выше значение целевой переменной, тем ниже значения этих предикторов;

- Шесть переменных имеют положительный коэффициент. Это значит, что чем выше значение целевой переменной, тем выше значения этих предикторов;
- Рассматривая P-value, который является достаточно важным показателем регрессии, можно утверждать, что почти все наши переменные достаточно значимы по отношению к целевой переменной. Чем меньше P value в нашей модели, тем больше шанс, что независимые переменные влияют на целевую переменную.

3.2.3 Построение модели с эффектами взаимодействия

Для построения эффектов взаимодействия были взяты две переменные с наилучшими P-значениями:

- ROA Using Net Income;
- Working Capital per Employee;

Проверка взаимодействия осуществлялась умножением каждой из этих величин на другие независимые переменные, чтобы позже проверить, являются ли какие-нибудь из них значимыми по отношению к целевой переменной выживаемости.

Наилучший результат показали два эффекта взаимодействия:

- Working Capital per Employee * Cash ratio
- Working Capital per Employee * Current ratio

На рисунке 55 отображена регрессия с эффектами взаимодействия.

OLS Regression Results

Dep. Variable:	SURVIVAL	R-squared:	0.887
Model:	OLS	Adj. R-squared:	0.805
Method:	Least Squares	F-statistic:	10.83
Date:	Mon, 30 May 2022	Prob (F-statistic):	5.49e-06
Time:	04:47:44	Log-Likelihood:	-29.019
No. Observations:	32	AIC:	86.04
Df Residuals:	18	BIC:	106.6
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	8.0070	2.592	3.089	0.006	2.562	13.452
ROA USING NET INCOME (%) (roa)	0.8242	0.358	2.305	0.033	0.073	1.575
EBIT MARGIN (%) (ebma)	0.8359	0.254	3.291	0.004	0.302	1.369
Cash ratio	1.3766	0.388	3.546	0.002	0.561	2.192
WORKING CAPITAL PER EMPLOYEE (TH) (wcpe)	-1.7182	1.034	-1.661	0.114	-3.892	0.455
weight_OTHER FIXED ASSETS (ofas)	-1.0900	0.304	-3.582	0.002	-1.729	-0.451
weight_PROVISIONS (prov)	0.8907	0.558	1.597	0.128	-0.281	2.062
weight_WORKING CAPITAL (wkca)	-0.6683	0.385	-1.736	0.100	-1.477	0.140
weight_COSTS OF GOODS SOLD (cost)	0.5397	0.333	1.620	0.123	-0.160	1.240
weight_TOTAL ASSETS (toas)	-0.9301	0.315	-2.955	0.008	-1.591	-0.269
Operating cash flow ratio	-2.0060	0.471	-4.258	0.000	-2.996	-1.016
Current ratio	-4.2543	1.450	-2.934	0.009	-7.300	-1.208
wcpe * Cash ratio	-1.1108	0.546	-2.033	0.057	-2.259	0.037
wcpe * Current ratio	3.8069	1.140	3.340	0.004	1.413	6.201

Omnibus:	3.850	Durbin-Watson:	2.106
Prob(Omnibus):	0.146	Jarque-Bera (JB):	2.385
Skew:	-0.500	Prob(JB):	0.303
Kurtosis:	3.888	Cond. No.	275.

Рис. 55. Множественная линейная регрессия с эффектами взаимодействия

Итоговая математическая модель множественной линейной регрессии вместе с эффектами взаимодействия выглядит так :

$$Survival = 8.0070 + 0.8242 * ROA\ Using\ Net\ Income + 0.8359 * Ebit\ Margin + 1.3766 * Cash\ Ratio - 1.7182 * Working\ Capital\ Per\ Employee - 1.0900 * weight\ Other\ Fixed\ Assets + 0.8907 * weight\ Provisions - 0.6683 * weight\ Working\ Capital + 0.5397 * weight\ Costs\ Of\ Goods\ Sold - 0.9301 * weight\ Total\ Assets - 2.0060 * Operating\ cash\ flow\ ratio - 4.2543 * Current\ ratio - 1.1108 * (wcpe * Cash\ ratio) + 3.8069 * (wcpe * Current\ ratio) + e, (27)$$

Оценивая результаты итоговой модели регрессии, можно сделать следующие выводы:

- Семь переменных имеют отрицательный коэффициент. Это значит, что чем выше значение целевой переменной, тем ниже значения этих предикторов;
- Шесть переменных имеют положительный коэффициент. Это значит, что чем выше значение целевой переменной, тем выше значения этих предикторов;
- Рассматривая P-value, который является достаточно важным показателем регрессии, можно утверждать, что почти все наши переменные достаточно значимы по отношению к целевой переменной, учитывая и эффекты взаимодействия

Таблица с данными параметрами будет использоваться для анализа методом случайного леса.

3.3 Реализация алгоритма МОСЛ

Подход для решения поставленной задачи должен обеспечивать:

1. Эквифинальность (изучено несколько возможных решений);
2. Конъюнктурность (решение должно опираться на несколько (> 1) независимых переменных;
3. Экономия (насколько просто каждое полученное решение);
4. F1 - score (метрика, основанная на точности и полноте).

Для построения деревьев решений используется следующий алгоритм:

Для каждого дерева решений рассчитывается “потомки” - первое - "значение меньше порога", второе - "значение больше порога":

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}, \quad (28)$$

Важность каждого значения для дерева решений:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}, \quad (29)$$

Нормализация значений от 0 до 1

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j}, \quad (30)$$

Вычисление среднего значения по всем деревьям:

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T}, \quad (31)$$

Оценка критерия качества выполняется следующим образом:

$$coverage = \left(1.0 * \frac{length\ of\ results}{length\ of\ X}\right), \quad (32)$$

, где length of results - длина массива результатов по предсказаниям
length of X -метрика, заданная пользователем

$$precision = \left(1.0 * \frac{length\ of\ matching\ results}{length\ of\ results}\right), \quad (33)$$

, где length of matching results - длина массива результата сравнения результатов

$$rate = coverage * precision * parsimony^{\frac{1}{expressiveness\ rate}}, \quad (34)$$

- Parsimony и Expressiveness rate задаются пользователем

3.3.1 Принцип работы программы

1. Пользователь задает необходимые настройки исходя из результирующего набора данных:
   - max_value = 7 #максимальное значение, используемое в тренировочном сете данных
   - min_value = 1 #минимальное значение, используемое в тренировочном сете данных
   - y_threshold = 3 #порог, разделяющий верхнее и нижнее целевые значения
   - intersection_threshold = 1 #порог для классификации случая как другого
   - min_conditions = 4 #минимальное количество колонок для каждого кейса
   - results_count = 10 #сколько максимально вывести деревьев в итоге
   - parsimony_rate = 0.4 #степень экономии в рейте
   - expressiveness_rate = 0.8 #степень выразительности в рейте
2. Программа выбирает кейсы из датасетов и ищет наилучшие деревья решений по заданным математическим моделям;
3. Деревья решений выводятся в формате dot и визуализируются средством graph viz;
4. Также результаты выводятся в текстовом формате doc.

3.3.2 Отображение результатов в формате doc

Ориентируясь на настройки, указанные в предыдущем подпункте, в алгоритм был «запущен» датасет, состоящий из переменных. Результат в формате DOC выглядит, как строки:

From tree 75: If ROA USING NET INCOME (%) (roa) is (> 1.72) and EBIT MARGIN (%) (ebma) is (< 4.81) and weight_WORKING CAPITAL (wkca) is (> 2.72) and weight_COSTS OF GOODS SOLD (cost) is (< 2.96) then value is Low

	precision	coverage	rate
Train	0.88	0.43	2.01
Test	0.12	0.57	

From tree 83: If weight_WORKING CAPITAL (wkca) is (> 3.88) and Cash ratio is (> 1.01) and weight_WORKING CAPITAL (wkca) is (< 6.7) and weight_PROVISIONS (prov) is (< 1.23) then value is Low

	precision	coverage	rate
Train	0.45	0.36	0.27
Test	0.25	0.29	

From tree 18: If WORKING CAPITAL PER EMPLOYEE (TH) (wcpe) is (< 3.92) and weight_COSTS OF GOODS SOLD (cost) is (< 1.57) and Cash ratio is (< 4.85) and WORKING CAPITAL PER EMPLOYEE (TH) (wcpe) is (> 3.14) then value is High

	precision	coverage	rate
Train	0.75	0.07	0.27
Test	0.0	0.0	

From tree 28: If Cash ratio is (> 1.0) and EBIT MARGIN (%) (ebma) is (< 5.31) and weight_WORKING CAPITAL (wkca) is (> 2.55) and Cash ratio is (< 3.02) then value is Low

	precision	coverage	rate
Train	0.74	0.41	0.18

	precision	coverage	rate
Test	0.12	0.57	

From tree 68: If weight_PROVISIONS (prov) is (< 1.21) and ROA USING NET INCOME (%) (roa) is (> 2.64) and weight_WORKING CAPITAL (wkca) is (< 5.74) and ROA USING NET INCOME (%) (roa) is (< 5.5) then value is Low

	precision	coverage	rate
Train	0.78	0.16	0.03
Test	0.0	0.21	

From tree 41: If weight_PROVISIONS (prov) is (< 1.21) and WORKING CAPITAL PER EMPLOYEE (TH) (wcpe) is (< 4.31) and Cash ratio is (< 4.39) and EBIT MARGIN (%) (ebma) is (> 4.69) then value is High

	precision	coverage	rate
Train	1.0	0.25	3.09
Test	0.5	0.14	

From tree 17: If Cash ratio is (< 3.86) and weight_PROVISIONS (prov) is (< 1.21) and weight_COSTS OF GOODS SOLD (cost) is (< 2.03) and weight_COSTS OF GOODS SOLD (cost) is (> 1.44) then value is High

	precision	coverage	rate
Train	0.88	0.14	1.57
Test	0.25	0.29	

From tree 90: If EBIT MARGIN (%) (ebma) is (< 5.31) and weight_WORKING CAPITAL (wkca) is (> 3.0) and weight_WORKING

CAPITAL (wkca) is (< 6.96) and weight_COSTS OF GOODS SOLD (cost) is (> 2.03) then value is Low

	precision	coverage	rate
Train	1.0	0.16	0.55
Test	0.33	0.21	

From tree 15: If EBIT MARGIN (%) (ebma) is (< 5.31) and weight_WORKING CAPITAL (wkca) is (< 6.96) and WORKING CAPITAL PER EMPLOYEE (TH) (wcpe) is (< 4.18) and weight_COSTS OF GOODS SOLD (cost) is (< 3.29) then value is High

except for cases weight_WORKING CAPITAL (wkca) is (> 6.01) where value is Low

	precision	coverage	rate
Train	0.54	0.62	0.16
Test	0.67	0.64	

From tree 94: If ROA USING NET INCOME (%) (roa) is (< 5.86) and weight_WORKING CAPITAL (wkca) is (< 5.82) and Cash ratio is (< 2.52) and weight_PROVISIONS (prov) is (> 1.24) then value is Low

	precision	coverage	rate
Train	0.75	0.14	0.04
Test	0.67	0.21	

Как можно заметить, алгоритм строит деревья таким количеством переменных, которые и были указаны в min_conditions. Также, алгоритм указывает, при каких «симбиозах» переменных и на каком пороге выживаемость будет ниже или выше.

3.3.3 Отображение результатов в формате dot

Результаты, аналогичные формату doc, также представляются в формате dot средством утилиты graphviz. Graphviz представляет собой пакет инструментов с открытым исходным кодом, инициированный AT&T Labs Research для рисования графиков, указанных в языковых сценариях DOT, имеющих расширение имени файла «gv». Он также предоставляет библиотеки для программных приложений для использования инструментов.

На рисунках 56–65 изображены деревья решений, аналогичные тем, что представлены в формате doc.

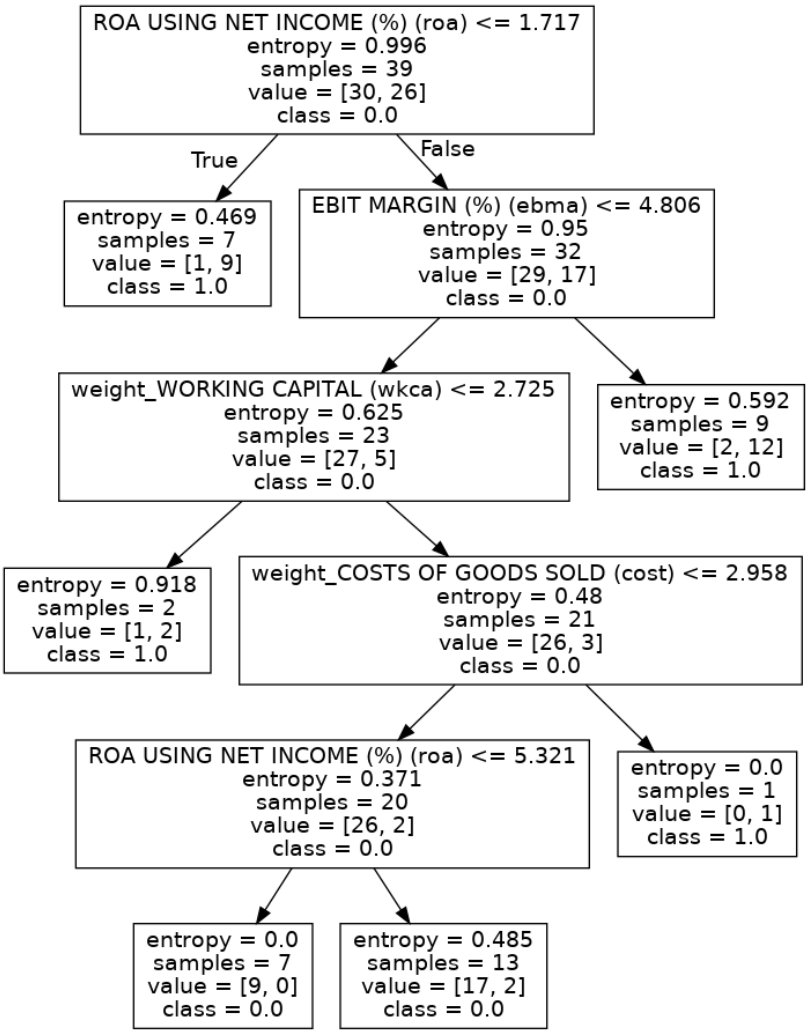

Рис. 56. Дерево 75

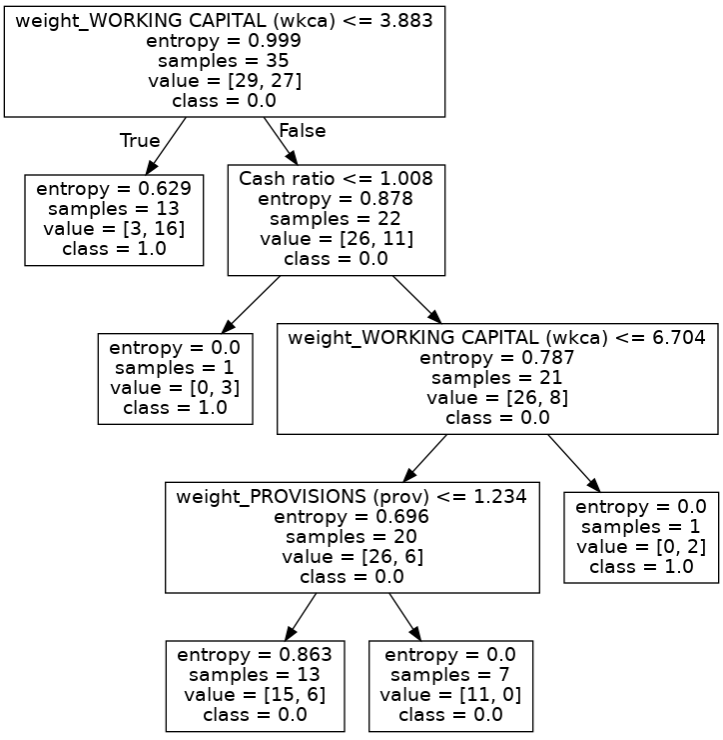

Рис. 57. Дерево 83

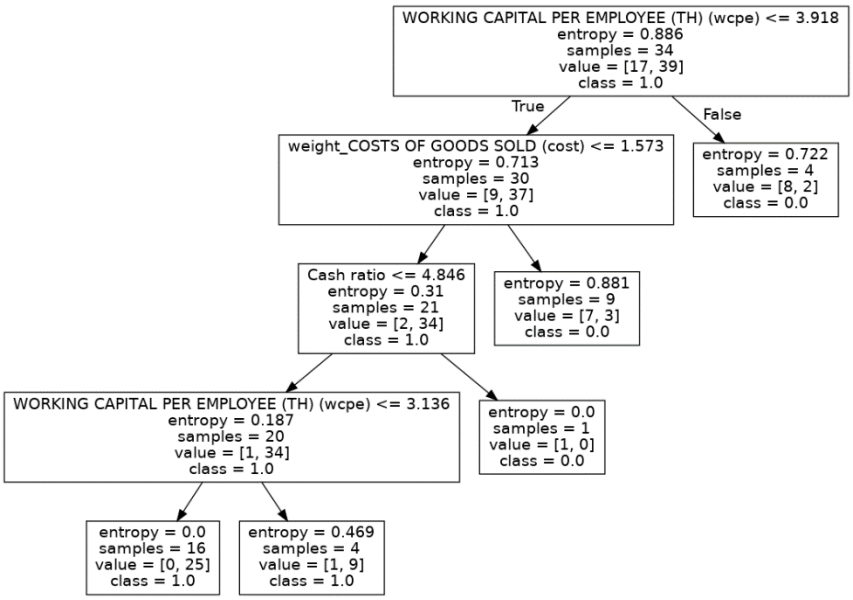

Рис. 58. Дерево 18

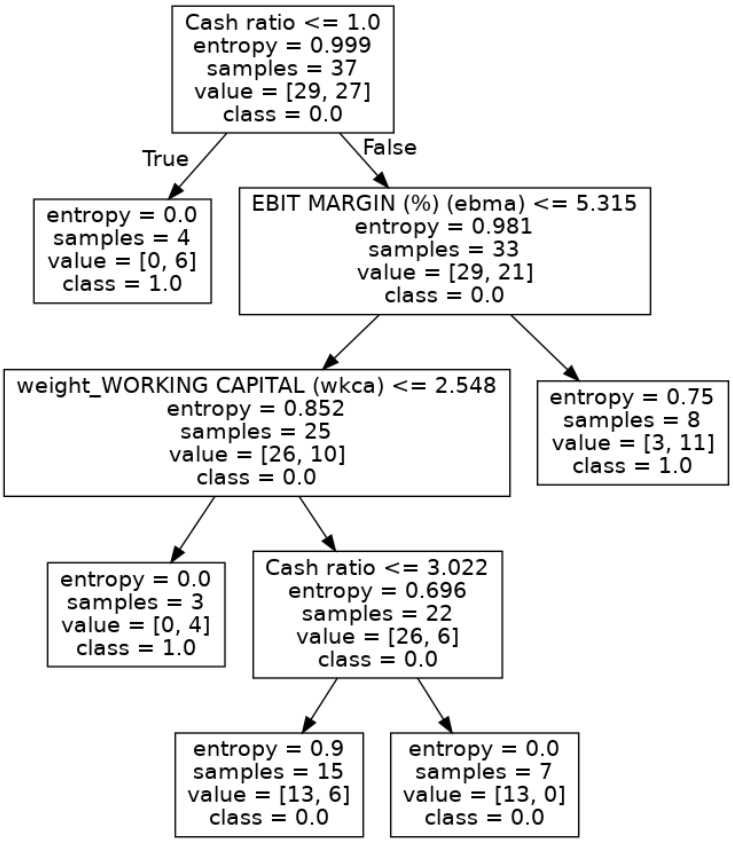

Рис. 59. Дерево 28

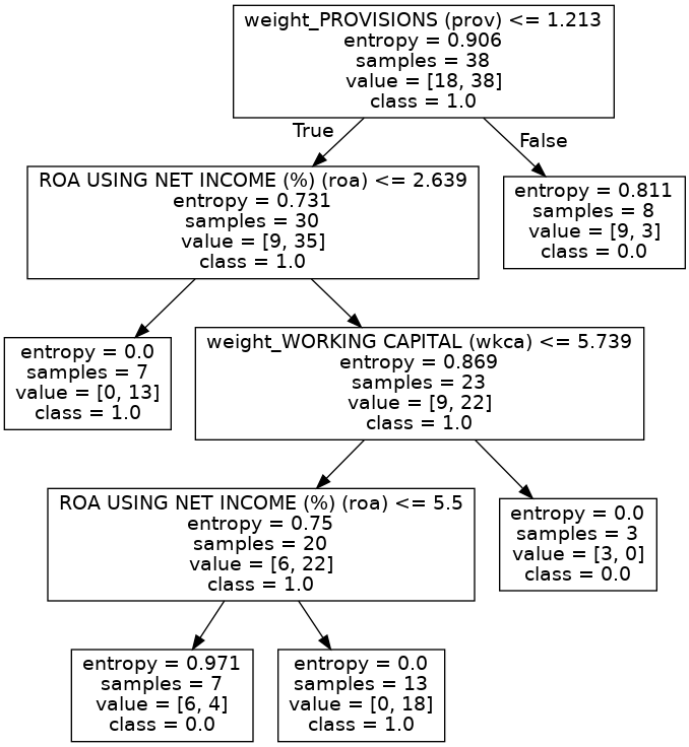

Рис. 60. Дерево 68

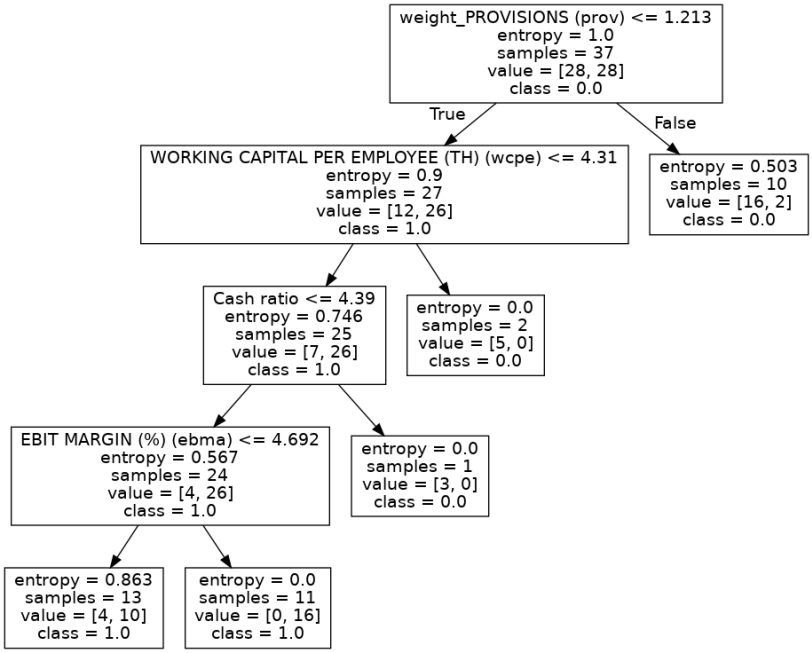

Рис. 61. Дерево 41

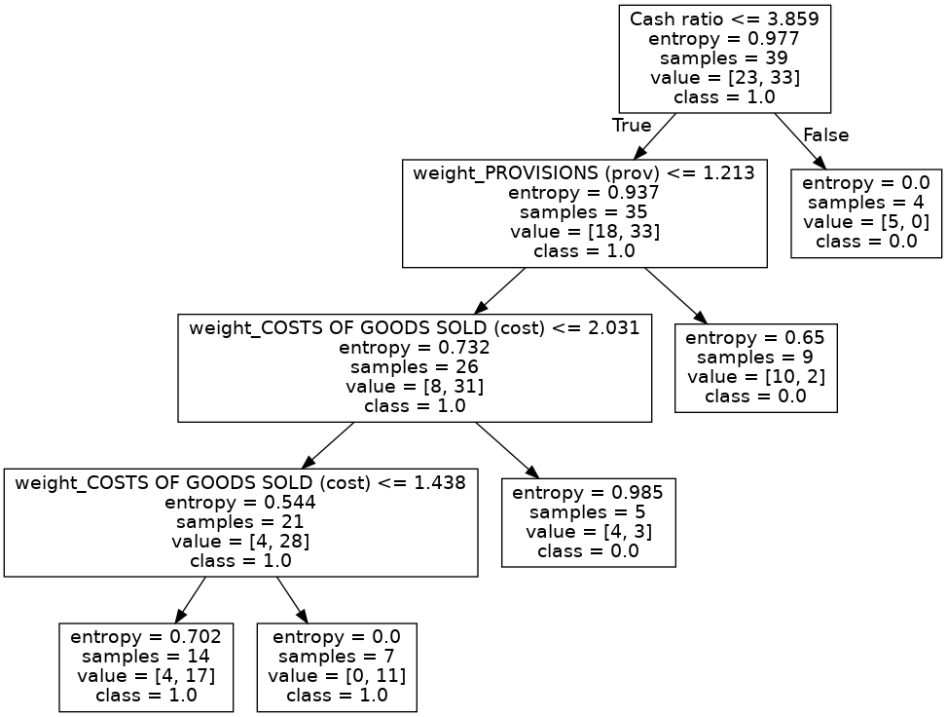

Рис. 62. Дерево 17

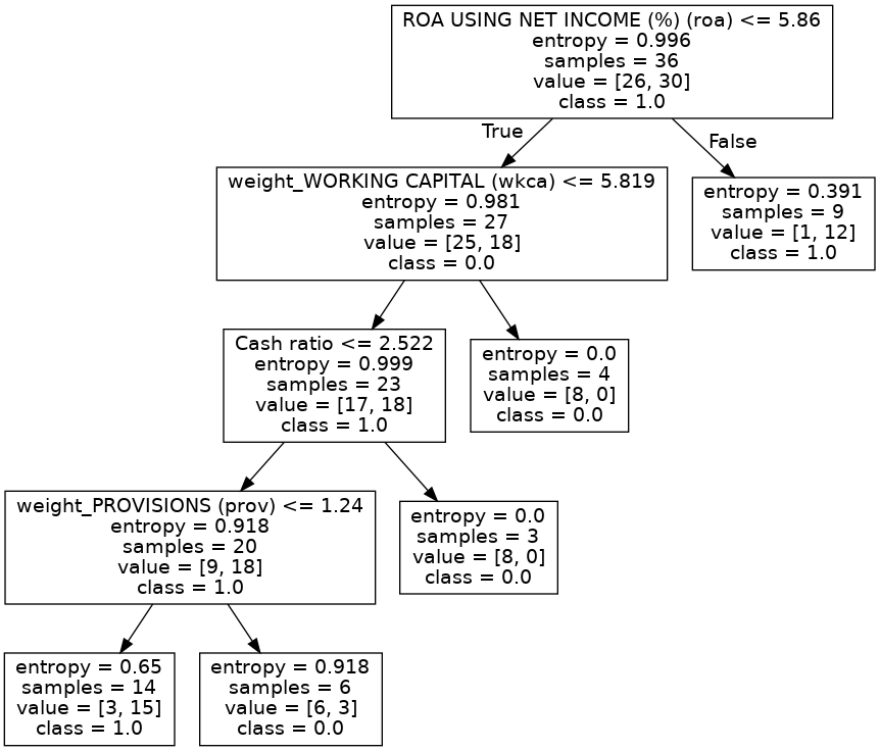

Рис. 63. Дерево 94

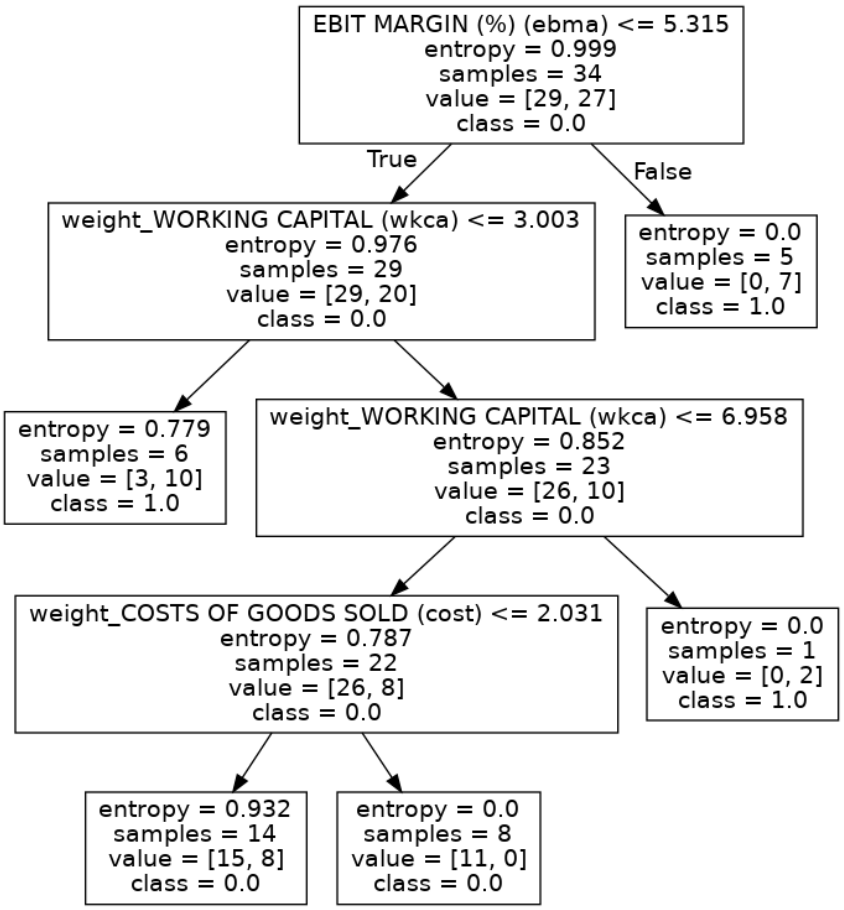

Рис. 64. Дерево 90

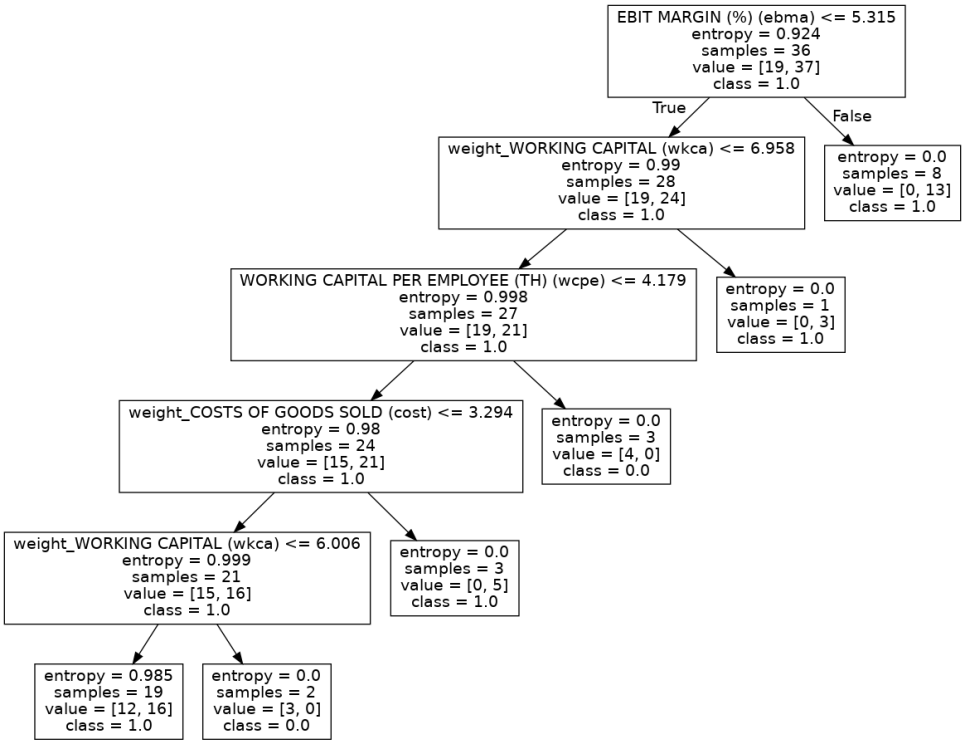

Рис. 65. Дерево 15

Выводы по главе 3

В подпункте 3.3.2 и на рисунках 56–65 представлены результаты работы алгоритма МОСЛ на исходных данных по альянсам и компаниям. Данные деревья указывают не только на статистически значимые переменные, которые влияют на выживаемость, но и показывают, при каком численном пороге и в каких совместных вариациях эти переменные оказывают на выживаемость хороший эффект - повышая ее, и обратный эффект – уменьшая её. С технической точки зрения алгоритм отрабатывает верно, исполняя все указанные настройки корректно и без нареканий.

Также по такой же результативной таблице данных, которая была использована в работе с алгоритмом МОСЛ, была построена множественная линейная регрессия. После полной обработки данных и анализа, итоговую формулу множественной линейной регрессии, включающую в себя важные параметры, влияющие на выживаемость, можно наблюдать под номером 26.

Сравнивая результаты множественной линейной регрессии и алгоритма МОСЛ выходит, что предсказание статистически важных переменных для выживаемости альянсов можно назвать успешным. Это объясняется большим совпадением переменных, которые оказались значимы в обоих анализах данных.

Существенное влияние на оба алгоритма влияет «состояние» данных. Изначально данные «сырые», с большим количеством пропусков или неточностей. Также, во время анализа путем множественной линейной регрессии нужно учитывать, что рассмотрение результатов «в группе» может существенно отличаться от обычной линейной регрессии каждого параметра отдельно. Стоит отметить, что алгоритм МОСЛ ни разу не выдал никакие исключения, что говорит о возможности обработки больших данных, невзирая на остаточные выбросы. Также он нашел зависимости и с другими переменными, не вошедшими в множественную линейную регрессию, что может являться хорошим знаком при поиске зависимостей.

Заключение

Для решения поставленной задачи был создан программный продукт на основе метода «случайного леса» для определения наиболее важных независимых переменных в датасетах, влияющих прямым образом на целевую переменную. Также было реализовано:

- МОСЛ не только выбирает важнейшие переменные, но и комбинирует их между собой, если это возможно, выдавая пороговые значения влияния независимых переменных на зависимую;
- Был определен оптимальный метод анализа и обработки данных в условиях большого количества «зашумлений» и пропусков в исходных данных;
- Алгоритм множественной линейной регрессии помог подтвердить успешную отработку алгоритма МОСЛ на реальных данных. Результаты алгоритма несущественно отличаются, на это могла повлиять «зашумленность» данных;
- Данный алгоритм можно использовать на любых датасетах, где целью является определение влияния независимых переменных на целевую переменную. Однако, для использования алгоритма нужно нормализовать и правильно обработать исходные данные;
- Созданный программный код может улучшить существующие модели. Необходимо провести тестирование полученного алгоритма на больших наборах данных для его усовершенствования, а также максимально автоматизировать его модель для уменьшения количества предобработки исходных данных.
- Были получены переменные, оказывающие наиболее сильное влияние на целевую переменную «выживаемости», а также

получены варианты комбинаций этих переменных для повышения/понижения выживаемости.

Список использованной литературы

1. Adner R. Ecosystem as Structure: An Actionable Construct for Strategy Journal of Management, 43 (1) (2017), стр. 39-58
2. Hannah D., Eisenhardt K.M. How firms navigate cooperation and competition in nascent ecosystems Strategic Management Journal, 39 (12) (2018), стр. 3163-3192
3. Jacobides M.G., Cennamo C., A. Towards Gawer. A theory of ecosystems Strategic Management Journal, 39 (8) (2018), стр. 2255-2276
4. Talmar M., Walrave B., Podoynitsyna K.S., Holmström J., Romme G.L. Mapping analyzing and designing innovation ecosystems: The Ecosystem Pie Model Long Range Planning, 53 (4) (2020)
5. Формирование стратегии компании. // [Автор статьи Баканов Г.Б.]. – [2014]. – URL: http://bizlog.ru/lib/b8/9_2_2.htm. – (дата обращения: 11 март 2022)
6. Тезисное описание алгоритма Random Forest. // [Автор статьи Дмитриевский Максим]. - [12 апрель 2018]. - URL: https://www.mql5.com/ru/articles/3856. - (дата обращения: 12 май 2022)
7. Что такое бизнес-экосистемы и зачем они нужны. // [Автор статьи Макарова Юлия]. - [10 сентябрь 2021]. - URL: https://trends.rbc.ru/trends/innovation/6087e5899a7947ed35fdbbf3. - (дата обращения: 24 март 2022)
8. Что такое N-граммы и с чем их едят? // [Автор статьи Скоринкин Данил]. - [13 июль 2018]. - URL: https://sysblok.ru/knowhow/chto-takoe-n-grammy-i-s-chem-ih-edjat/. - (дата обращения: 12 апрель 2022)
9. Стоит ли создавать бизнес-экосистему: рассмотрим преимущества и недостатки. // [Автор статьи Heads and Hands]. - [17 апрель 2020]. - URL: https://vc.ru/services/121003-stoit-li-sozdavat-biznes-

ekosistemu-rassmotrim-preimushchestva-i-nedostatki. - (дата обращения: 24 март 2022)

10. Multiple regression as a machine learning algorithm. // [Автор статьи Alam Mahbubul]. - [13 ноябрь 2020]. - URL: https://towardsdatascience.com/multiple-regression-as-a-machine-learning-algorithm-a98a6b9f307b. - (дата обращения: 29 апрель 2022)
11. An Optimum Approach Towards the Bag of Words with Code Illustration in Python. // [Автор статьи Balodi Tanesh]. - [22 август 2019]. - URL: https://www.analyticssteps.com/blogs/an-optimum-approach-towards-the-bag-of-words-with-code-illustration-in-python. - (дата обращения: 2 апрель 2022)
12. Correlation and Machine Learning. // [Автор статьи Barai Abhishek]. - [18 ноябрь 2020]. - URL: https://medium.com/analytics-vidhya/correlation-and-machine-learning-fee0ffc5faac. - (дата обращения: 29 апрель 2022)
13. Machine Learning (ML) for Natural Language Processing (NLP). // [Автор статьи Barba Paul]. - [29 сентябрь 2020]. - URL: https://www.lexalytics.com/lexablog/machine-learning-natural-language-processing. - (дата обращения: 24 март 2022)
14. A Gentle Introduction to the Bag-of-Words Model. // [Автор статьи Brownlee Jason]. - [9 октябрь 2017]. - URL: https://machinelearningmastery.com/gentle-introduction-bag-words-model/. - (дата обращения: 12 апрель 2022)
15. KPI of the Day – Accounting: $ Working capital per employee // [Автор статьи Costea Andrei]. - [31 октябрь 2018]. - URL: https://www.performancemagazine.org/kpi-accounting-working-capital-employee/#:~:text=It%20is%20defined%20as%20the,assets%20and%20employee%20related%20costs. - (дата обращения: 28 май 2022)

16. Overfitting in machine learning. // [Автор статьи Ellis Christina]. - URL: https://crunchingthedata.com/overfitting-in-machine-learning/. - (дата обращения: 12 май 2022)
17. Random forest overfitting. // [Автор статьи Ellis Christina]. - URL: https://crunchingthedata.com/random-forest-overfitting/. - (дата обращения: 19 май 2022)
18. Current Ratio. // [Автор статьи Fernando Jason]. - [21 октябрь 2021]. - URL: https://www.investopedia.com/terms/c/currentratio.asp. - (дата обращения: 28 май 2022)
19. Working Capital. // [Автор статьи Fernando Jason]. - [27 октябрь 2021]. - URL: https://www.investopedia.com/terms/w/workingcapital.asp. - (дата обращения: 28 май 2022)
20. Is there an open business model right for your company? // [Авторы статьи Karolin Frankenberger, Tobias Weiblen, Oliver Gassmann]. - [август 2014]. - URL: https://www.ideasforleaders.com/ideas/is-there-an-open-business-model-right-for-your-company. - (дата обращения: 13 март 2022)
21. Understanding Boxplots // [Автор статьи Galarnyk Michael]. - [12 сентябрь 2018]. - URL: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51. - (дата обращения: 18 апрель 2022)
22. Operating Cash Flow Ratio // [Автор статьи Hargrave Marshall]. - [12 апрель 2021]. - URL: https://www.investopedia.com/terms/o/ocfratio.asp. - (дата обращения: 28 май 2022)
23. What Is Return on Assets (ROA)? // [Автор статьи Hargrave Marshall]. - [5 март 2022]. - URL: https://www.investopedia.com/terms/r/returnonassets.asp#:~:text=ROA%20is%20calculated%20by%20dividing%20a%20firm's%20net%20inc

ome%20by,found%20on%20its%20balance%20sheet. - (дата обращения: 28 май 2022)

24. Stemming? Lemmatization? What? // [Автор статьи Heidenreich Hunter]. - [21 декабрь 2018]. - URL: https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8. - (дата обращения: 2 апрель 2022)
25. Machine Learning. // [Автор статьи IBM Cloud Education]. - [15 июль 2020]. - URL: https://www.ibm.com/cloud/learn/machine-learning. - (дата обращения: 24 март 2022)
26. Handle Data Outlier in Machine Learning. // [Автор статьи Kanani Bhavika]. - [2 сентябрь 2019]. - URL: https://studymachinelearning.com/handle-outlier-in-python/. - (дата обращения: 18 апрель 2022)
27. Cash Ratio. // [Автор статьи Kenton Will]. - [20 май 2022]. - URL: https://www.investopedia.com/terms/c/cash-ratio.asp. - (дата обращения: 28 май 2022)
28. Fixed Asset. // [Автор статьи Kenton Will]. - [11 март 2022]. - URL: https://www.investopedia.com/terms/f/fixedasset.asp. - (дата обращения: 28 май 2022)
29. Interaction effect in multiple regression. // [Автор статьи Khot Sufyan]. - [4 март 2020]. - URL: https://towardsdatascience.com/interaction-effect-in-multiple-regression-3091a5d0fadd. - (дата обращения: 29 апрель 2022)
30. Ensemble Methods in Machine Learning: What are They and Why Use Them? // [Автор статьи Lutins Evan]. - [2 август 2017]. - URL: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f. - (дата обращения: 3 май 2022)
31. Introduction to Random Forest in Machine Learning. // [Автор статьи Mbaabu Onesmus]. - [11 декабрь 2020]. - URL:

https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning// - (дата обращения: 12 май 2022)

32. Hierarchical Operations and Supply Chain Planning // [Автор статьи Miller Tan]. - [январь 2002]. - URL: https://www.researchgate.net/publication/321517014_Hierarchical_Operations_and_Supply_Chain_Planning. - (дата обращения: 13 март 2022)
33. Identify Outliers With Pandas, Statsmodels, and Seaborn. // [Автор статьи Moreno Amanda Iglesias]. - [31 июль 2020]. - URL: https://medium.com/swlh/identify-outliers-with-pandas-statsmodels-and-seaborn-2766103bf67c. - (дата обращения: 18 апрель 2022)
34. Breaking Down Supply Chain Management // [Автор статьи Morrow Mary Kate]. - [12 февраля 2021]. - URL: https://altametrics.com/supply-chain-management.html. - (дата обращения: 19 март 2022)
35. How Do You “Design” a Business Ecosystem? // [Авторы статьи Pidun Ulrich, Reeves Martin, Schüssler Maximilian]. - [20 февраля 2020]. - URL: https://www.bcg.com/publications/2020/how-do-you-design-a-business-ecosystem. - (дата обращения: 24 март 2022)
36. Dealing with Interaction Effects in Regression. // [Автор статьи Rekha M]. - [28 ноябрь 2019]. - URL: https://blog.clairvoyantsoft.com/dealing-with-interaction-effects-in-regression-db647ed38691?gi=1c79e107f537. - (дата обращения: 3 май 2022)
37. What Is Cost of Goods Sold and How Do You Calculate It? // [Автор статьи Rosenberg Eric]. - [16 июль 2021]. - URL: https://squareup.com/us/en/townsquare/what-is-cost-of-goods-sold#:~:text=Cost%20of%20goods%20sold%20is%20the%20total%20amount%20your%20business,producing%20or%20selling%20the%20good. - (дата обращения: 28 май 2022)

38. Understanding the 3 most common loss functions for Machine Learning Regression. // [Автор статьи Seif George]. - [21 май 2019]. - URL: https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3. - (дата обращения: 3 май 2022)
39. Simple Guide to Regular Expressions. // [Автор статьи Sharma Chirag]. - [20 май 2020]. - URL: https://towardsdatascience.com/simplest-guide-for-regular-expressions-nlp-made-easy-bbb33cd694be. - (дата обращения: 2 апрель 2022)
40. Когда нужна и не нужна вертикальная интеграция. // [Авторы статьи Stuckey John, White David]. – [1993]. – URL: http://vestnikmckinsey.ru/strategic-planning/kogda-nuzhna-i-ne-nuzhna-vertikal-naya-integraciya. – (дата обращения: 11 март 2022)
41. Knowing all about Outliers in Machine Learning. // [Автор статьи Tripathi Mayank]. - [16 июнь 2020]. - URL: https://datascience.foundation/sciencewhitepaper/knowing-all-about-outliers-in-machine-learning. - (дата обращения: 12 апрель 2022)
42. What Is Correlation in Machine Learning? // [Автор статьи Upadhyay Amit]. - [5 август 2020]. - URL: https://medium.com/analytics-vidhya/what-is-correlation-4fe0c6fbed47. - (дата обращения: 29 апрель 2022)
43. Standardization in machine learning. // [Автор статьи Vinay Sachin]. - [26 январь 2021]. - URL: https://www.linkedin.com/pulse/standardization-machine-learning-sachin-vinay?trk=public_profile_article_view. - (дата обращения:17 март 2022)
44. Вертикальная интеграция. // – URL: https://studref.com/662755/finansy/vertikalnaya_integratsiya. – (дата обращения: 11 март 2022)

45. Горизонтальная и вертикальная интеграция: в чем разница? // - [08 мая 2021]. - URL: https://nesrakonk.ru/what-difference-between-horizontal-integration-and-vertical-integration/. - (дата обращения: 13 март 2022)
46. Интеграция. // – URL: https://studref.com/662754/finansy/integratsiya. – (дата обращения: 11 март 2022)
47. Что такое вертикальная интеграция: основы. // [12 февраля 2021]. – URL: https://sendpulse.com/ru/support/glossary/vertical-integration. – (дата обращения: 11 март 2022)
48. EBIT Margin. // - URL: https://businessnovice.net/definition/ebit-margin/. - (дата обращения: 28 май 2022)
49. Provisions. // - URL: https://corporatefinanceinstitute.com/resources/knowledge/finance/provisions/#:~:text=Provisions%20represent%20funds%20put%20aside,on%20a%20company's%20balance%20sheet. - (дата обращения: 28 май 2022)
50. Python-NLTK и анализ текстовых данных (базовая обработка естественного языка). // - URL: https://russianblogs.com/article/9089276476/. - (дата обращения: 19 май 2022)
51. Total assets definition. // - [04 март 2022]. - URL: https://www.accountingtools.com/articles/total-assets#:~:text=Total%20assets%20refers%20to%20the,balance%20sheet%20of%20the%20business. - (дата обращения: 28 май 2022)
52. Z-score. // - [17 февраль 2022]. - URL: https://ru.wikipedia.org/wiki/Z-%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B0. - (дата обращения: 18 апрель 2022)

Приложение 1. Код программы МОСЛ

condition.py

```
import math
from settings import max_value, min_value, parsimony_rate,
expressiveness_rate

class Condition(object):
    def __init__(self, feature, comparison, threshold):
        self._comparison = comparison
        self._threshold = threshold
        self._feature = feature

    def __str__(self):
        category = ' ' if self._comparison == '>' else ' '
        value = '(' + self._comparison + " " + str(round(self._threshold, 2))
+ ')'
        return self._feature + " is " + category + value

    def apply(self, sample):
        value = sample[self._feature]
        return value > self._threshold if self._comparison == '>' else value
<= self._threshold

    def residual_points(self):
        return max_value - self._threshold if self._comparison == '>' else
self._threshold - min_value

    def expressiveness(self):
        return 1 - math.log10(self.residual_points() * parsimony_rate *
expressiveness_rate + 1)
```

main.py

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from itertools import groupby
import math
import random
import shutil
import glob, os
from condition import Condition
from manual_cases import manual_cases
from settings import max_value, min_value, y_threshold,
intersection_threshold, min_conditions, results_count, expressiveness_rate
import pygal
from pygal.style import Style
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly
from collections import Counter
import warnings


warnings.filterwarnings("ignore")
```

```
def get_survey_data():
    if os.path.exists("df_variables.csv"):
        print("--df_variables.csv found")
        return pd.read_csv("df_variables.csv", sep=',')
    return df

def visualize_tree(tree, tree_id, feature_names):
    """Create tree png using graphviz.

    Args
    ----
    tree -- scikit-learn DecsisionTree.
    feature_names -- list of feature names.
    """
    #print("visualizing tree " + str(tree_id))
    class_names = [str(class_name) for class_name in tree.classes_]
    with open("images/dt" + str(tree_id) + ".dot", 'w') as f:
        export_graphviz(tree, out_file=f,
                        feature_names=feature_names, class_names=class_names)

    command = ["dot", "-Tpng", "images/dt" + str(tree_id) + ".dot", "-o",
"images/dt" + str(tree_id) + ".png"]
    #try:
    #    subprocess.check_call(command)
    #except:
    #    exit("Could not run dot, ie graphviz, to "
    #         "produce visualization")

def visualize_data(data, condition_x, condition_y, feature_category,
filename):
    style = Style(colors=('#00cc00', 'red', 'blue', 'blue'))
    xy_chart = pygal.XY(stroke=False, style=style)
    high_points = [(row[condition_x._feature], row[condition_y._feature]) for
i, row in data.iterrows() if row[feature_category] == True]
    low_points = [(row[condition_x._feature], row[condition_y._feature]) for
i, row in data.iterrows() if row[feature_category] == False]
    xy_chart.x_title = str(condition_x)
    xy_chart.y_title = str(condition_y)
    high_points_counter = Counter(high_points)
    low_points_counter = Counter(low_points)

    xy_chart.add('higher than', high_points,
                 formatter=lambda x: '[%.2f,%.2f] (%s)' % (x[0], x[1],
high_points_counter.get(tuple(x))))
    xy_chart.add('lower than', low_points, dots_size=1,
                 formatter=lambda x: '[%.2f,%.2f] (%s)' % (x[0], x[1],
low_points_counter.get(tuple(x))))
    xy_chart.add(condition_x._feature + ' = ' + str(condition_x._threshold),
                 [(condition_x._threshold, min_value),
(condition_x._threshold, max_value)],
                 stroke=True)
    xy_chart.add(condition_y._feature + ' = ' + str(condition_y._threshold),
                 [(min_value, condition_y._threshold), (max_value,
condition_y._threshold)],
                 stroke=True)
    xy_chart.render_to_file('images/' + filename + '.svg')

def rate(coverage, precision, parsimony):
    return coverage * precision * pow(parsimony, 1 / expressiveness_rate)

def tree_to_cases(tree, tree_id, feature_names, coverage):
    """Produce decision tree as table with features as rows and resulting
cases as columns
```

```
    Args
    ----
    tree -- scikit-leant DescisionTree.
    feature_names -- list of feature names.
    target_names -- list of target (class) names.

    Notes
    -----
    based on http://stackoverflow.com/a/30104792.
    """
    left      = tree.tree_.children_left
    right     = tree.tree_.children_right
    threshold = tree.tree_.threshold
    n_node_samples = tree.tree_.n_node_samples
    features  = [feature_names[i] for i in tree.tree_.feature]
    value = tree.tree_.value
    impurity = tree.tree_.impurity
    classes = tree.classes_

    def recurse(node, conditions):
        node_precision = 1 - impurity[node]
        node_parsimony = np.prod([condition.expressiveness() for condition in
conditions])

        if (threshold[node] != -2):
            left_result = recurse(left[node], conditions +
[Condition(features[node], "<", threshold[node])])
            left_case = left_result[0]
            right_result = recurse(right[node], conditions +
[Condition(features[node], ">", threshold[node])])
            right_case = right_result[0]

            samples = [left_case['samples'][0] + right_case['samples'][0],
left_case['samples'][1] + right_case['samples'][1]]
            category = samples[1] > samples[0]
            samples_total = samples[0] + samples[1]

            if left_case['category'] == category:
                exception = left_case['exception']
            else:
                exception = left_case

            if right_case['category'] == category:
                exception = right_case['exception'] if
right_case['exception']['own_rate'] > exception['own_rate'] else exception
            else:
                exception = right_case if right_case['own_rate'] >
exception['own_rate'] else exception

            own_samples = samples[category]
            own_coverage = coverage(own_samples)
            own_rate = rate(own_coverage, node_precision, node_parsimony)

            if (exception['own_rate'] >= 0):
                exception_samples = exception['samples'][not category]
                exception_parsimony = exception['own_parsimony']
                exception_precision = exception['own_precision']

                pair_coverage = coverage(own_samples + exception_samples)
                pair_precision = (exception_precision + node_precision) / 2
                pair_parsimony = exception_parsimony * node_parsimony
```

```
                pair_rate = rate(pair_coverage, pair_precision,
pair_parsimony)
            else:
                pair_rate = own_rate

            case = {
                'label': ('High' if category else 'Low'),
                'category': category,
                'count': samples_total,
                'conditions': conditions,
                'tree_id': tree_id,
                'samples': samples,
                'own_rate': own_rate,
                'own_parsimony': node_parsimony,
                'own_precision': node_precision,
                'pair_rate': pair_rate,
                'exception': exception
            }
            return [case] + left_result + right_result
        else:
            target = value[node][0]
            best_index = np.argmax(target)
            best_category = classes[best_index]
            node_samples = n_node_samples[node]
            node_coverage = coverage(node_samples)
            case = {
                'label': ('High' if best_category else 'Low'),
                'category': best_category,
                'count': node_samples,
                'conditions': conditions,
                'tree_id': tree_id,
                'samples': [0.0, node_samples] if best_category else
[node_samples, 0.0],
                'own_rate': rate(node_coverage, node_precision,
node_parsimony),
                'own_parsimony': node_parsimony,
                'own_precision': node_precision,
                'pair_rate': rate(node_coverage, node_precision,
node_parsimony),
                'exception': {
                    'own_rate': -1.0,
                    'pair_rate': -1.0,
                    'samples': [0.0, 0.0]
                }
            }
            return [case]

    return recurse(0, [])


def cases_to_table(cases, feature_names):
    def cell(feature, case):
        feature_conditions = [str(condition) for condition in
case['conditions'] if condition._feature == feature]
        return ' && '.join(feature_conditions) if len(feature_conditions) > 0
else '-'

    features_used = [feature for feature in feature_names if
any(cell(feature, case) != '-' for case in cases)]

    lines = ['feature;' + ';'.join(('T' + str(case['tree_id']) + ':' +
case['label'] + ' (' + str(case['count']) + ')' for case in cases))]
```

```
    for feature in features_used:
        lines.append(feature + ';' + ';'.join(cell(feature, case) for case in
cases))

    return lines

def case_to_text(case):
    exception = case['exception']
    conditions = case['significant_conditions'] if 'significant_conditions'
in case else case['conditions']
    text = 'From tree ' + str(case['tree_id']) + \
           ': If ' + ' and '.join(str(condition) for condition in conditions)
+ ' then value is ' + case['label']

    if exception['own_rate'] >= 0:
        text += '\nexcept for cases ' + \
                ' and '.join(str(condition) for condition in
exception['conditions'][len(case['conditions']):]) + \
                ' where value is ' + exception['label']
    return text

def filter_conditions(conditions):
    significant = []
    for condition in conditions:
        if any([other._feature == condition._feature \
                and other._comparison == condition._comparison \
                and other.residual_points() < condition.residual_points() for
other in conditions]):
            continue
        significant.append(condition)

    return significant


def feature_intersection(conditions_a, conditions_b, feature):
    lower_bound = min_value
    upper_bound = max_value
    for condition in (conditions_a + conditions_b):
        if condition._feature == feature:
            if (condition._comparison == '>'):
                lower_bound = max(lower_bound, condition._threshold)
            else:
                upper_bound = min(upper_bound, condition._threshold)

    return max(upper_bound - lower_bound, 0)

def volume_intersection(conditions_a, conditions_b, features):
    return np.prod([feature_intersection(conditions_a, conditions_b, feature)
for feature in features])

def volume_rate(conditions_a, conditions_b, features):
    volume_a = volume_intersection(conditions_a, [], features)
    volume_b = volume_intersection(conditions_b, [], features)
    intersection = 1.0 * volume_intersection(conditions_a, conditions_b,
features)
    return max(intersection / volume_a, intersection / volume_b)

def filter_cases(cases, features, max_count):
    for case in cases:
        if case['own_rate'] > case['pair_rate']:
            case['pair_rate'] = case['own_rate']
            case['exception'] = {
                'own_rate': -1.0,
```

```
                'pair_rate': -1.0,
                'samples': [0.0, 0.0]
            }

    cases.sort(key=lambda case: -case['pair_rate'])

    result = []

    for case in cases:
        if len(result) >= max_count:
            break
        case['conditions'] = filter_conditions(case['conditions'])
        if case['exception']['own_rate'] >= 0:
            case['exception']['conditions'] =
filter_conditions(case['exception']['conditions'])

        valid_length = len(case['conditions']) >= min_conditions
        max_intersect = max([volume_rate(existing['conditions'],
case['conditions'], features)
                             for existing in result]) if len(result) > 0 else
0
        unique = max_intersect < intersection_threshold
        if valid_length and unique:
            result.append(case)

    return result


def predict_by_cases(cases, sample, default = lambda: random.choice([True,
False])):
    def check_case(case):
        conditions = case['significant_conditions'] if
'significant_conditions' in case else case['conditions']
        return all(condition.apply(sample) for condition in conditions)

    for case in cases:
        exception = case['exception']
        if exception['pair_rate'] >= 0 and check_case(exception):
            return exception['category']
        if check_case(case):
            return case['category']

    return default()


def evaluate_case(case, X, Y):
    cases_predictions = [predict_by_cases([case], row, lambda: None) for
index, row in X.iterrows()]
    results = [Y[index] == predicted_item for index, predicted_item in
enumerate(cases_predictions) if predicted_item is not None]
    matching_results = [result for result in results if result]
    coverage = round(1.0 *  len(results) / len(X), 2)
    precision = round(1.0 * len(matching_results) / len(results), 2) if
len(results) > 0 else 0.0
    return coverage, precision

def coverage(samples):
    return 100.0 * samples * math.log(samples) / (train_count *
math.log(train_count))

shutil.rmtree('images/', True)
os.makedirs('images/')
```

```
df = get_survey_data()


print("* df.head()", df.head(), sep="\n", end="\n\n")
print("* df.tail()", df.tail(), sep="\n", end="\n\n")


features = list(df.columns[:4])
features.extend(df.columns[5:8])
print("* features:", features, sep="\n")

y_feature = "SURVIVAL"

for x in range(2):
    train, test = train_test_split(df, test_size=0.2, random_state=x)

    Y_train = train[y_feature] > y_threshold
    X_train = train[features]
    Y_test = test[y_feature] > y_threshold
    X_test = test[features]

    train_count = len(train)
    train = pd.concat([X_train, Y_train], axis=1)

    forest = RandomForestClassifier(n_estimators=100, min_samples_split=20,
criterion='entropy',random_state=x)
    forest.fit(X_train, Y_train)

    Y_train = Y_train.values.tolist()
    Y_test = Y_test.values.tolist()

    cases = [case for tree_id, dt in enumerate(forest.estimators_) for case
in tree_to_cases(dt, tree_id, features, coverage)]
    cases = filter_cases(cases, features, results_count)
    trees_to_visualize = {case['tree_id'] for case in cases}
    for tree_id, dt in enumerate(forest.estimators_):
         if tree_id in trees_to_visualize:
             visualize_tree(dt, tree_id, features)

    #print('\n')

    for case in cases:
        text = case_to_text(case)

        if (len(case['conditions']) == 2):
                visualize_data(train, case['conditions'][0],
case['conditions'][1], y_feature, text.split('\n')[0])

        train_matching, train_precision = evaluate_case(case, X_train,
Y_train)
        test_matching, test_precision = evaluate_case(case, X_test, Y_test)
        text += '\n'
        text += '      | {0: ^16} | {1: ^16} | {2:
^16}\n'.format('precision', 'coverage', 'rate')
        text += 'Train | {0: ^16} | {1: ^16} | {2:
^16}\n'.format(train_precision, train_matching, round(case['pair_rate'], 2))
        text += 'Test  | {0: ^16} | {1: ^16} |\n'.format(test_precision,
test_matching)
        print(text + '\n')
```

settings.py

```
max_value = 7                                   #minimum feature value in train
data
min_value = 1                                   #maximum feature value in train
data
y_threshold = 3                                 #threshold separating High and
Low target values
intersection_threshold = 0.5                    #threshold to classifty a case as
different
min_conditions = 4                               #minimum conditions count in a
case
results_count = 10                              #count of cases to select
parsimony_rate = 0.4                            #amount of parsimonity in rate
expressiveness_rate = 0.8                       #amount of expressiveness in rate
```