МИНИСТЕРСТВО ОБРАЗОВАНИЕ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К. И. Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программной инженерии»

Тұрақ Данияр Арманұлы

На соискателя академической степени магистра

Название диссертации	Разработка программы скрининга резюме с использованием методов машинного обучение
Направление подготовки	7M06101 – Software Engineering

Научный руководитель
PhD, ассоциированный
профессор,
_____Р.Ж. Сатыбалдиева
«__» _____ 2023 г.

Рецензент,
PhD, кандидат физ-мат наук
_____ И.М. Уалиева
«__» _____ 2023 г.

Нормоконтроль
PhD, ассоциированный
профессор
_____А.Т. Ахмедиярова
«__» _____ 2023 г.

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
кандидат физ-мат наук,
ассоциированный профессор
_____ А. Н. Молдагулова
«__»_____ 2023 г.

Алматы 2023

Институт автоматики и информационных технологий

Кафедра Программная инженерия

Специальность: 7M06101 – Software Engineering

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

ЗАДАНИЕ
на выполнение магистерской диссертации

магистранту, обучающемуся Тұрақ Данияр Арманұлы

Тема диссертации: «Разработка программы скрининга резюме с использованием машинного обучение»

Срок сдачи законченной диссертации «___»_____

Исходные данные к магистерской диссертации. Дан анализ моделей и методов скрининга резюме. Поставленные цели и задачи диссертационной работы обуславливают комплексность нахождение подходящих кандидатов, в эпоху больших данных.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации: а) обзор существующих алгоритмов и методов классификации; б) исследование работы с текстом посредством обработки естественных языков; в) нахождение схожести между резюме и описанием работы; г) разработка системы скрининга резюме.

Рекомендуемая основная литература 1. Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, Rocky Bhatia. A Machine Learning approach for automation of Resume Recommendation system. 2. Tejaswini K, Umadevi V, Shashank M Kadiwal, Sanjay Revanna. Design and development of machine learning based resume ranking system

ГРАФИК
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Сбор данных	15.10.2022	Выполнено
Реализация методов машинного обучение	11.02.2023	Выполнено
Реализация полученных результатов исследование	15.12.2023	Выполнено

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант, (уч. степень, звание)	Сроки	Подпись
Стандартизация/ нормоконтроль	А.Т. Ахмедиярова, ассоц. профессор	12.06.2023	
Программная обеспечение/ нормоконтроль	Н.К. Мукажанов, ассоц. профессор	13.06.2023	
Антиплагиат	Б.С. Әубәкіров, ассистент,	12.06.2023	

Дата выдачи задания "____"_____ 2023 г.

Заведующий кафедрой _____ А.Н. Молдагулова

Научный руководитель _____ Р.Ж. Сатыбалдиева

Задание принял к исполнению магистрант _____ Д.А. Тұрақ

Дата "_____"_____ 2023 г.

АҢДАТПА

Магистрлік диссертация зерттеу нәтижелерін ұсынады және бар әдістерді салыстырады, сонымен қатар бар шешімдерді жетілдіреді. Негізгі жұмыс деректер жиынтығын жинау және тазалау және әдістерді, кітапханаларды және модельдерді салыстыру арқылы жасалды. Қазіргі уақытта, оқыту сапасының арту дәуірінде еңбек нарығында көбірек әлеуетті үміткерлер пайда болады. Деректердің мұндай көлемін қолмен қарау мүмкін емес. Осы диссертациялық жұмысты зерделеу барысында біліктілік алгоритмдеріне салыстырмалы талдау, сондай-ақ ұқсас мақалалармен салыстыру жүргізілді. Магистрдің жұмысына: реферат, кіріспе, 4 бөлім, қорытынды, әдебиеттер кіреді. Түйін сөздер: Косинус ұқсастығы, NoSQL, XGBoostClassifier.

АННОТАЦИЯ

В магистерской диссертации изложены результаты исследования и проведено сравнение уже имеющихся методов, а также улучшение уже существующих решений. Основная работа была произведена посредством собирании и очищение датасета и сравнения методов, библиотек и моделей. В настоящее время, в эпоху увеличение качество обучение, все больше и больше потенциальных кандидатов появятся на рынке труда. Вручную просматривать такой объем данных не представляется возможным. В ходе исследование данной диссертационной работы, было произведено сравнительный анализ алгоритмов квалификации, а также сравнение с аналогичными статьями. В магистерской работе представлены: аннотация, введение, 4 раздела, выводы, и литература. Ключевые слова: Cosine Similarity, NoSQL, XGBoostClassifier.

ANNOTATION

The master's thesis presents the results of the study and compares existing methods, as well as improving existing solutions. The main work was done by collecting and cleaning the dataset and comparing methods, libraries and models. Nowadays, in the era of increasing quality of training, more and more potential candidates will appear on the job market. Manually viewing such a volume of data is not possible. During the study of this dissertation work, a comparative analysis of qualification algorithms was made, as well as a comparison with similar articles. The master's work includes: abstract, introduction, 4 sections, conclusions, and literature.
Keywords: Cosine Similarity, NoSQL, XGBoostClassifier.

СОДЕРЖАНИЕ

Введение 8
Нормативные ссылки 11
Обозначение и сокращение 12
1 Анализ предметной области 14
1.1 Аналогичные системы 15
1.2 Сферы применения 15
1.3 Выводы по первому разделу 17
2 Методы и средства машинного обучения 18
2.1 Алгоритмы классификации машинного обучения 18
2.2 Методы работы с текстом посредством NLP 20
2.3 K-ближайший сосед 24
2.4 Дерево решений 24
2.5 Многослойный классификатор 27
2.6 XGBoost 29
2.7 Алгоритм случайного леса 31
2.8 Метод опорных векторов 33
2.9 Коэффициент Жаккара 35
2.10 Косинусное сходство 36
2.11 Выводы по второму разделу 39
3 Разработка системы скрининга 40
3.1 Датасет и его обзор 40
3.2 Предварительная обработка датасета 42
3.3 Построение модели классификации 44
3.4 База данных 47
3.5 Серверная часть приложение 50
3.6 Клиентская часть приложение 52
3.7 Выводы по третьему разделу 53
4 Результаты исследование 54
4.1 Выводы по четвертому разделу 57
Заключение 58
Список к использованных источников 59
Приложение А 62
Приложение Б 67
Приложение С 72

ВВЕДЕНИЕ

В современном мире все больше и больше компаний сталкиваются с огромным объемом резюме, поступающих на открытые вакансии. Ручная обработка и анализ каждого резюме может быть трудоемкой и затратной задачей. Однако развитие методов машинного обучения и искусственного интеллекта открывает новые возможности для автоматизации этого процесса. Разработка программы скрининга резюме с использованием методов машинного обучения представляет собой подход, позволяющий автоматически классифицировать и фильтровать резюме на основе определенных критериев. Это позволяет значительно сократить время и усилия, затрачиваемые на ручной отбор кандидатов [1]. Машинное обучение, как раздел искусственного интеллекта, позволяет компьютерным программам обучаться на основе опыта и данных, чтобы принимать автоматические решения. В случае скрининга резюме, модель машинного обучения может быть обучена на большом наборе резюме, размеченных как подходящие или неподходящие для конкретной вакансии. Это обучение позволяет модели выявлять общие паттерны и признаки, которые указывают на качество и соответствие кандидата требованиям работодателя. В этой диссертационной работе будут рассмотрены основные принципы разработки программы скрининга резюме с использованием методов машинного обучения [1]. Мы рассмотрим процесс сбора и подготовки данных, выбор подходящих алгоритмов машинного обучения, обучение модели, а также оценку и улучшение ее производительности. Также будет рассмотрено влияние этой программы на процесс подбора персонала и возможные преимущества и ограничения ее использования. Разработка программы скрининга резюме с использованием методов машинного обучения представляет собой мощный инструмент для современных компаний, помогающий автоматизировать и ускорить процесс отбора кандидатов [5].

Актуальность темы. В фоне быстрого роста объема информации и количества представленных резюме становится все сложнее и трудоемко для работодателей рассматривать каждое резюме вручную. Использование машинного обучения и автоматического скрининга резюме может значительно упростить и ускорить этот процесс.

Причины, почему эта тема все еще актуальна:

1) Экономия времени: Разработка программы скрининга резюме на основе
2) машинного обучения позволяет автоматизировать процесс оценки резюме. Это помогает сэкономить время, которое можно потратить на более глубокий анализ кандидатов, улучшение интервью и принятие обоснованных решений.
3) Улучшение качества отбора: Машинное обучение может использоваться для создания моделей, которые будут искать ключевые навыки, опыт и образование, соответствующие требованиям вакансии. Это поможет

улучшить качество отбора кандидатов, исключив неподходящие резюме на раннем этапе.

4) Уменьшение субъективности: Автоматический скрининг резюме на основе алгоритмов машинного обучения может уменьшить субъективность, связанную с ручным отбором кандидатов. Машина оценивает кандидатов на основе объективных критериев, установленных работодателем.
5) Анализ больших данных: С помощью машинного обучения можно обрабатывать большие объемы данных, которые содержатся в резюме кандидатов. Автоматический скрининг позволяет эффективно обрабатывать и анализировать сотни и тысячи резюме, выявлять паттерны и тенденции, которые могут быть полезными для будущего найма.
6) Непрерывное развитие технологий машинного обучения: Технологии машинного обучения постоянно развиваются, и появляются новые методы и алгоритмы, которые могут быть применены для улучшения скрининга резюме. Это делает тему все более актуальной, так как всегда есть возможность использовать новые подходы для повышения точности и эффективности программы скрининга.

В целом, разработка программы скрининга резюме с использованием машинного обучения остается актуальной темой, которая может принести значительные выгоды работодателям, улучшить процесс отбора и повысить эффективность найма [3].

При наличии ограничений на резюме и большинстве вводного резюме от компании-клиента этот процесс может быть выполнен. С помощью автоматической классификации резюме и сопоставления процесс отбора кандидатов стал более простым. Когда количество кандидатов больше, проверка резюме вручную становится сложной, поскольку система предназначена для работы с большим количеством резюме. Эта диссертация рассмотрит, как алгоритмы обработки естественного языка (NLP), искусственного интеллекта и машинного обучения помогают в анализе настроений и определении типов личности при проверке резюме. В идеале он будет посвящен тому, как можно использовать обработку естественного языка для определения необходимых типов личности [3].

Целью исследования является исследование разных алгоритмов классификации и методов работы с текстом, а также нахождение схожести между документами. Конечная цель исследования это разработка программы скрининга резюме.

Научная новизна исследования заключается в следующем:

- Исследованы имеющиеся программные системы, методы, алгоритмы и способы обработки и сжатия аэрокосмических изображений, и на основе выявления их преимуществ и недостатков выбрано наиболее актуальное направление;
- Измененная формула Косинусного сходства для нахождение соответствие.

Практическая значимость диссертационной работы состоит в том, что разработанная система может в значительной степени улучшить качество нахождение лучших кандидатов и их классифицирования.

Теоретическая значимость диссертационной работы в том, что велись исследования по уже имеющимся методом для работы с текстом резюме и нахождение схожести между ними. Также проводились исследования по улучшению результатов тестирования для повышения точности модели и увеличению корректности работы алгоритма.

Задачи исследования:

- Анализ предметной области
- Исследование методов обработки естественного языка
- Исследование методов машинного обучения для классификации и нахождение лучших резюме
- Разработка системы
- Тестирование системы и получение результатов

Апробация работы. Диссертационная исследовательская работа опубликована в научном журнале «Студенческий вестник» №19(258) под названием «Разработка программы скрининга резюме с использованием методов машинного обучение».

Предмет исследования: обработка естественного языка, машинное обучение.

Методы исследования: методы классификации данных, модели машинного обучения и др.

Объект исследования: резюме в виде текстов

НОРМАТИВНЫЕ ССЫЛКИ

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

1. Составители: Р.И.Мухамедиев, Ж.М.Алибиева. Методические указания по выполнению и оформлению магистерской диссертации (для специальности: 6М070400 – Вычислительная техника и программное обеспечение). Алматы: КазНИТУ им.К.И.Сатпаева, 2019 г. – 40 с.
2. Инструкция по оформлению диссертации и автореферата. МОН РК, Высший аттестационный комитет. Алматы, 2004
3. ГОСО РК 5.04.033-2008 Послевузовское образование – магистратура. Основные положения. 06.05.2008 г.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Ниже приведены основные понятия которые используются в исследовательской работе.

Информационные системы (ИС) - это организованная совокупность методов, процедур, программного и аппаратного обеспечения, которые собирают, хранят, обрабатывают, передают и используют информацию для достижения определенных целей. ИС включает в себя технические и организационные аспекты, а также людей, которые участвуют в ее функционировании. Основная цель информационной системы состоит в обработке информации для получения полезных результатов, принятии решений, автоматизации бизнес-процессов и обеспечении эффективного управления информацией. ИС могут использоваться в различных областях, включая бизнес, науку, образование, здравоохранение, государственное управление и другие [2].

Машинное обучение (МО) - это подраздел искусственного интеллекта (ИИ), который изучает алгоритмы и статистические модели, позволяющие компьютерам обучаться и делать предсказания или принимать решения на основе данных, без явного программирования. Вместо того чтобы явно программировать компьютер на выполнение конкретной задачи, в машинном обучении используются алгоритмы, которые способны "обучаться" на основе набора данных. Эти алгоритмы анализируют данные, выявляют паттерны и зависимости, и создают модели, которые могут делать прогнозы или принимать решения на новых данных [2].

Набор данных - это совокупность структурированных или неструктурированных данных, которые используются в машинном обучении или других аналитических задачах. Набор данных обычно состоит из множества индивидуальных элементов, называемых образцами, примерами или наблюдениями.Каждый образец в наборе данных представляет собой информацию, относящуюся к определенным атрибутам или признакам. Атрибуты могут быть числовыми, категориальными (например, цвет или тип объекта) или текстовыми. Набор данных может включать один или несколько атрибутов для каждого образца. Набор данных может быть представлен в различных форматах, таких как таблицы (например, в формате CSV - comma-separated values), базы данных, текстовые файлы, изображения, аудио или видеофайлы, а также другие форматы, в зависимости от типа данных и задачи, которую необходимо решить [4].

Хорошо подготовленный набор данных является важным компонентом успешного машинного обучения, поскольку качество данных может существенно влиять на точность и эффективность модели обучения.

Обработка естественного языка (NLP) - это область исследования и применения компьютерных методов и алгоритмов для анализа, понимания и генерации естественного языка, который используется людьми для общения. Цель обработки естественного языка состоит в разработке компьютерных систем, способных взаимодействовать и обрабатывать естественный язык так же, как это делают люди [6].

1 Анализ предметной области

Скрининг резюме с использованием машинного обучения - это процесс автоматизированной оценки и отбора резюме на основе алгоритмов и моделей машинного обучения. Анализ предметной области этого процесса включает несколько ключевых аспектов.

Семантический анализ текста. Для успешного скрининга резюме необходимо анализировать текстовую информацию, содержащуюся в резюме кандидата. Это включает выделение ключевых навыков, квалификации, образования, опыта работы и других релевантных аспектов, которые помогут определить подходящих кандидатов.

Обработка естественного языка (Natural Language Processing, NLP). NLP-технологии играют важную роль в анализе предметной области скрининга резюме. Алгоритмы машинного обучения могут использовать методы NLP, такие как разбор предложений, классификация текста, извлечение сущностей и семантическое моделирование, чтобы понять содержание резюме и извлечь нужные данные.[1]

Индустриальная специфика. Каждая отрасль имеет свои особенности и требования к кандидатам. Поэтому анализ предметной области скрининга резюме также включает понимание конкретных навыков, опыта и квалификации, связанных с данной отраслью. Например, в IT-отрасли могут быть важными навыки программирования, а в финансовой сфере - опыт работы с аналитическими инструментами и финансовыми моделями.

Классификация и ранжирование. Одной из основных задач скрининга резюме является классификация и ранжирование кандидатов на основе их соответствия требованиям вакансии. Алгоритмы машинного обучения могут использовать различные модели классификации, такие как метод опорных векторов (SVM), наивный Байесовский классификатор или глубокие нейронные сети, чтобы оценить и ранжировать резюме.[2]

Обучение на основе данных. Анализ предметной области скрининга резюме также включает сбор и подготовку обучающих данных. Для эффективного обучения моделей машинного обучения необходимо иметь размеченные данные, которые содержат информацию о классификации резюме (подходит/не подходит) и связанные с этими резюме метаданные, такие как навыки, опыт работы и образование кандидатов.

Анализ предметной области скрининга резюме с использованием машинного обучения является сложной и многогранной задачей, требующей комбинации компьютерных наук, лингвистики и понимания требований конкретной отрасли. Однако, при правильной настройке и обучении моделей машинного обучения, такой подход может значительно улучшить эффективность и точность процесса отбора кандидатов.

1.1 Аналогичные системы

Greenhouse: Greenhouse предоставляет платформу для рекрутинга, которая включает функции скрининга резюме с использованием алгоритмов машинного обучения. Она помогает автоматизировать процесс отбора кандидатов и упрощает сотрудничество между различными участниками команды по найму.

Taleo: Taleo - это система управления персоналом, которая также предоставляет функционал для скрининга резюме. Она включает инструменты для автоматического анализа и классификации резюме на основе предварительно заданных критериев и ключевых слов.

Lever: Lever - это платформа для найма, которая также предлагает функционал скрининга резюме с использованием машинного обучения. Она облегчает процесс скрининга, позволяя настраивать правила и фильтры для автоматического отбора наиболее подходящих кандидатов.

SmartRecruiters: SmartRecruiters - это система управления трудоустройством, которая включает инструменты для скрининга и оценки резюме. Она использует алгоритмы машинного обучения для автоматического анализа и сопоставления кандидатов с требованиями вакансии.

1.2 Сферы применения

В прошлом работодателям приходилось вручную просматривать резюме соискателей, что делало поиск подходящих людей для различных спецификаций работы сложной задачей. Организации было трудно найти нужных людей, но это также заняло много времени. Тем не менее, процессы стали более управляемыми благодаря использованию автоматизированных систем ранжирования резюме [2].

ИИ полностью изменил процесс найма. Применение естественного языка меняется. Искусственный интеллект отвечает за проверку резюме. Наваз и Gomez (2019) утверждают, что первым шагом в процессе найма является совокупная проверка и ранжирование. Он включает в себя определение резюме и ранжирование их в соответствии с требованиями работы. По сравнению с традиционными методами использование искусственного интеллекта проще. При ранжировании резюме алгоритмы нейролингвистического программирования используют заранее определенную терминологию.[5]

В процессе набора кандидатов было много изменений. Организации раньше распространяли объявления о вакансиях по телевидению и газетам. Для этих должностей кандидаты могли отправлять свои резюме по почте в соответствующие организации, а резюме сортировались вручную. Если кандидат был включен в шорт-лист, его менеджеры по найму связывались с ним для назначения собеседования (Paschen et al., 2020). Этот процесс потребовал большого количества ресурсов и времени. Впоследствии организации изменили

свои правила найма. В соответствии с последними изменениями кандидаты должны размещать свои резюме на веб-сайтах в определенных форматах. Интеллектуальные алгоритмы были необходимы для преодоления препятствий, возникающих в традиционной системе найма. [3]

Эти алгоритмы должны были анализировать информацию из любого резюме, даже если информация была неправильно структурирована, сортировать и ранжировать ее соответствующим образом (Tambe et al., 2020).[5] Перед тем, как провести анализ и сохранить информацию в базе данных, новая модель алгоритма использует обработку естественного языка для понимания резюме. Модель включает в себя систему ранжирования, которая помогает работодателям ранжировать кандидатов на должности, чтобы выбрать лучших.

Для того, чтобы выбрать одного кандидата на эту должность, достаточно просмотреть резюме 10 000 кандидатов. Поскольку эта система медленная, многие организации ее не поддерживают. Поскольку резюме на рынке могут иметь различные структуры и форматы, идеально, что они не являются стандартной практикой. Предположим, что менеджеры по персоналу должны изучить каждое резюме, чтобы определить подходящих кандидатов. Поскольку есть вероятность упустить нужного человека в этом случае, они могут совершать ошибки (Rouhiainen, 2018).[3] Трудно определить, кто подходит для этой должности. Организации должны надлежащим образом оценивать резюме, чтобы нанять подходящих кандидатов на должности. Проверка резюме — это процесс оценки кандидатов на работу на основе информации, содержащейся в их резюме, такой как образование и опыт.

Программа скрининга резюме с использованием машинного обучения может быть применена в различных сферах и организациях, где проводится процесс подбора персонала. Вот несколько примеров:

1) Компании с большим объемом резюме: Большие компании часто получают огромное количество резюме на каждую вакансию. Программа скрининга резюме может помочь автоматизировать и ускорить процесс отбора, фильтруя резюме и выделяя наиболее подходящих кандидатов.
2) Рекрутинговые агентства: Рекрутинговые агентства могут использовать программу скрининга резюме, чтобы помочь своим клиентам сократить время на поиск и отбор кандидатов. Это может быть особенно полезно, когда агентство работает с несколькими клиентами и получает большой поток резюме.
3) Образовательные учреждения: Университеты, колледжи и другие образовательные учреждения также могут воспользоваться программой скрининга резюме для помощи своим студентам в поиске работы или стажировки. Это может помочь студентам быстрее найти соответствующие возможности и повысить шансы на трудоустройство.
4) Онлайн-платформы по найму персонала: Онлайн-платформы, посредники или рынки труда, которые связывают работодателей с соискателями, могут использовать программу скрининга резюме для автоматического фильтрации и классификации резюме, улучшая точность поиска и соответствия вакансий.

5) Государственные учреждения: Государственные организации, занимающиеся набором персонала, также могут воспользоваться программой скрининга резюме для упрощения и ускорения процесса отбора кандидатов.

В основе любой эффективной стратегии лежит эффективная и эффективная проверка резюме. Цель метода состоит в том, чтобы выбрать лучших кандидатов и отличить их от средних. В системе ручной подачи заявок заявители обычно должны были заполнить всю необходимую информацию, что занимало много времени. Большинство претендентов выразили недовольство результатами процедуры подачи заявок (Semmler & Rose, 2017) [8]. Использование машинного обучения и естественного языкового обучения (NLP) будет практичным, чтобы решить проблемы, связанные с традиционным методом проверки резюме, поскольку оно будет использовать функции, извлеченные из разных кандидатов, а также отображать их профили.

1.3 Выводы по первому разделу

Как мы можем заметить, решение по скринингу резюме существуют с давних пор, имеются уже работающие и эффективные программы по нему. Но, как указывали выше, многие статьи и программы имеют свои недостатки, которые могут пагубно сказаться на корректности скрининга.

2. Методы и средства машинного обучения

2.1 Алгоритмы классификации машинного обучения

Классификация является одним из наиболее важных элементов контролируемого обучения. В этом разделе мы обсудим множество алгоритмов классификации, таких как логистическая регрессия, наивный байесовский алгоритм, деревья решений, оптимизация роя частиц, случайные леса и другие. Мы рассмотрим каждую функцию классификации алгоритма, а также то, как она работает. [3]

Основанный на идеях из компьютерных наук, статистики, когнитивных наук, инженерии, теории оптимизации и многих других областей математики и естественных наук, машинное обучение (МО) — это обширная область, которая охватывает многие дисциплины [1]. Машинное обучение может использоваться для многих целей, но интеллектуальный анализ данных является наиболее важным [2]. Машинное обучение с учителем и без учителя — это две большие группы машинного обучения.

Неконтролируемое машинное обучение используется для получения выводов из наборов данных с входными данными, которые не имеют помеченных ответов [3]. С другой стороны, мы можем сказать, что неконтролируемое машинное обучение не может добиться желаемого результата. Методы контролируемого машинного обучения пытаются выяснить, как входные атрибуты, также известные как независимые переменные, связаны с целевыми атрибутами [4]. Классификация и регрессия — две основные группы методов под наблюдением. В классификации выходная переменная имеет маркировку класса [1]. С другой стороны, в регрессии выходная переменная имеет непрерывные значения.

Чтобы предсказать членство группы в экземплярах данных, классификация — это метод интеллектуального анализа данных, также известный как машинное обучение [1]. Машинное обучение может быть выполнено различными способами, но классификация остается наиболее распространенным [2]. Машинное обучение очень любит классификацию, особенно для планирования и открытия знаний в будущем.

Одной из наиболее важных задач для исследователей машинного обучения и интеллектуального анализа данных является классификация [3]. Рисунок 1 представляет собой общую модель контролируемого обучения, также известную как методы классификации.

Хотя классификация является известным методом машинного обучения, она сталкивается с проблемами, такими как обработка отсутствующих данных. Обучение и классификация могут быть затруднены из-за отсутствия значений в наборе данных. Вот некоторые возможные причины отсутствия данных [3]: Неправильное понимание, признание того, что данные неактуальны на момент ввода, удаление данных из-за отклонения от других документированных данных или неисправности оборудования могут привести к неправильной записи.

Проблема отсутствия данных может быть решена такими методами [10], как; Специалисты по анализу данных могут вручную наблюдать за

пропущенными выборками и вставлять вероятное или возможное значение; игнорировать пропущенные данные; использовать отдельную глобальную константу для замены пропущенных значений целыми значениями; или заменить пропущенные значения средними значениями признаков для конкретного класса. В этой статье мы рассмотрим только некоторые из наиболее популярных методов классификации (рисунок 2.1).

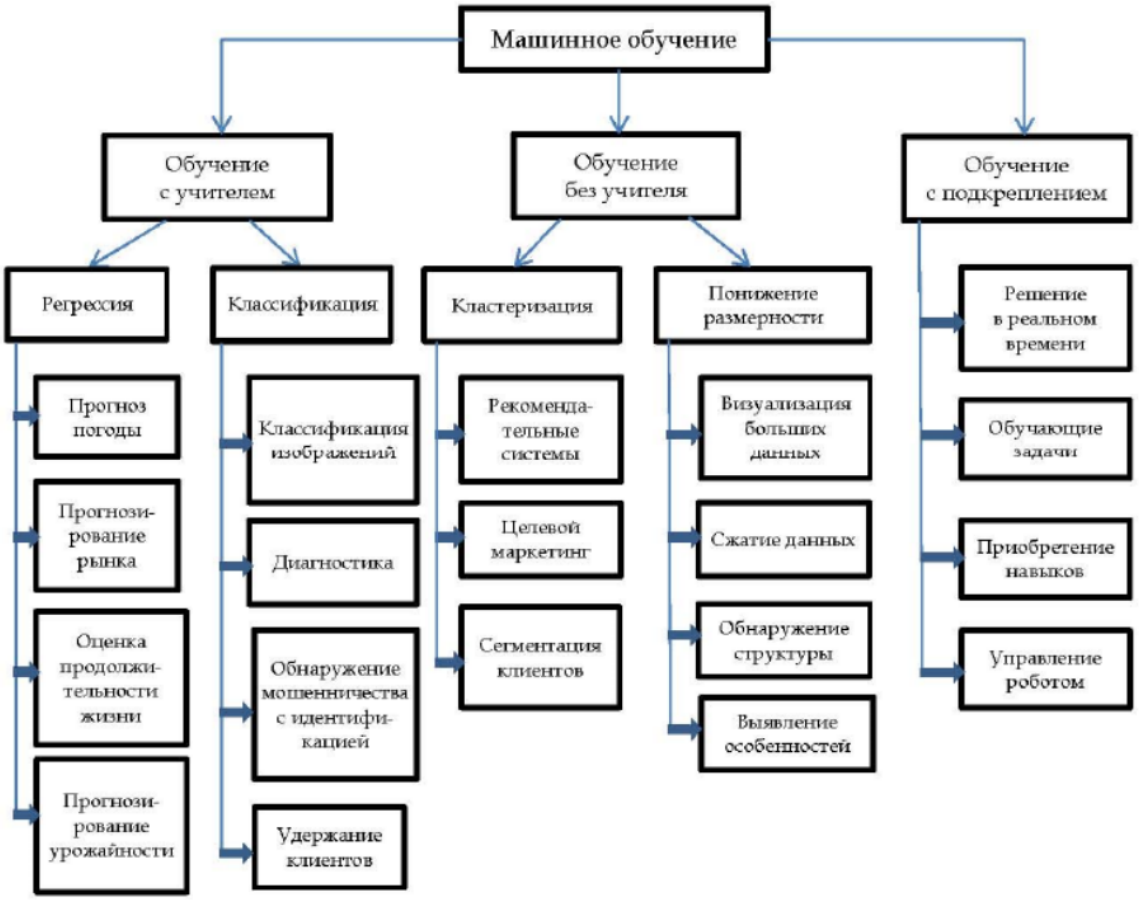

Рисунок 2.1. Классификации МО

Обнаружение потребности в классификации резюме: как люди, все обязаны делать ошибки. Безопасное хранение и совместное использование физических копий очень неудобно и неэффективно. Для преодоления недостатков существенно возрастает потребность в классификации резюме, для чего широко используются инструменты искусственного интеллекта и алгоритмы машинного обучения. Категоризация схожих данных вместе: с развитием компьютерных и информационных технологий большое количество исследовательских работ было опубликовано как в Интернете, так и в автономном режиме, и поскольку новые темы исследований продолжают появляться, пользователям трудно находить и классифицировать свои соответствующие исследования. статьи. Для преодоления ограничений

необходим метод классификации, который может группировать исследовательские статьи в осмысленный класс, в котором публикации, скорее всего, будут иметь схожие темы. Предлагаемые методы в данной диссертации KNeighborsClassifier, MLPClassifier, XGBClassifier и SVM.

2.2 Методы работы с текстом посредством NLP

Предварительная обработка данных, которая является важным шагом в построении модели машинного обучения, зависит от того, насколько хорошо предварительная обработка данных была выполнена, что приводит к заметным результатам.

Предварительная обработка текста — это первый шаг в процессе построения модели в искусственном интеллекте (NLP).

Различные этапы подготовки текста:

1) Токенизация
2) Нижний регистр
3) Удаление стоп-слов
4) Стемминг
5) Лемматизация

Многие люди используют эти различные этапы предварительной обработки текста для уменьшения размерности.

В модели векторного пространства любое слово или термин является осью или мерой. В многомерном пространстве текст или документ изображается в виде вектора (рисунок 2.2).[7]

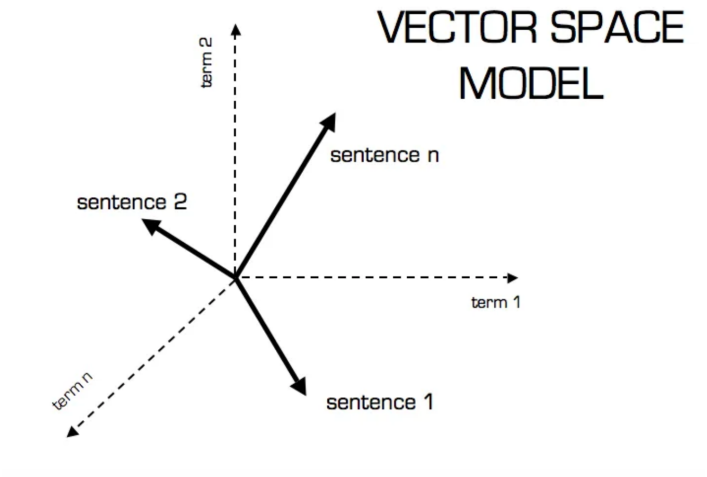

Рисунок 2.2. Векторная модель

Количество измерений отражает количество уникальных слов.

Токенизация представляет собой разделение предложения на отдельные слова.

Нижний регистр — это перевод слова в нижний регистр (Natural Language Processing — nlp). Слова, такие как «книга» и «книга», имеют одно и то же значение, но если их не перевести в нижний регистр, они будут рассматриваться как два разных слова в модели векторного пространства, что дает больше измерений [3].

Удаление стоп-слов: слова, которые часто используются в документах, такие как a, an, the и т. д., называются стоп-словами. Поскольку они не помогают различать два документа, эти слова на самом деле не имеют значения (рисунок 2.3).

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

sentence = "Machine Learning is cool!

stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(sentence)

filtered_sentence = [w for w in word_tokens if not w in stop_words]
print(filtered_sentence)
```

Рисунок 2.3. Удаление стоп-слов

Лемматизация: в отличие от стемминга, лемматизация связывает слова с существующими словами.

Лемматизация или стемминг могут быть использованы. Библиотека nltk и spaCy реализует стеммеры и лемматизаторы. Они основаны на подходе, основанном на правилах (рисунок 2.4).

```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("Machine", pos='n'))
# pos: parts of speech tag, verb
print(lemmatizer.lemmatize("caring", pos='v'))
```

Рисунок 2.4. Лемматизация

Разрешение слова в его лемму требует части речи слова для лемматизации. Это способствует преобразованию слова в его корневую форму. Тем не менее, для этого требуются дополнительные вычислительные

лингвистические навыки, такие как способность тегировать отдельные фрагменты речи [3].

Эти различные этапы предварительной обработки текста широко используются для уменьшения размерности.

TF-IDF (Term Frequency-Inverse Document Frequency) - это широко используемая техника в обработке естественного языка и информационном поиске для числового представления текстовых данных. Она основана на идее, что важность термина в документе пропорциональна частоте его встречаемости в этом документе, но обратно пропорциональна частоте его встречаемости во всех документах набора данных.[8]

Для объяснения TF-IDF математически давайте разберем две составляющие: частота термина (TF) и обратная частота документа (IDF).

Частота термина (TF) - Частота термина измеряет частоту термина в документе. Она количественно оценивает, насколько часто термин появляется в документе относительно общего числа терминов в этом документе.

Обратная частота документа (IDF) - Обратная частота документа измеряет важность термина по всем документам в наборе данных. Она помогает выявить уникальные или редкие термины, которые могут нести более значимый смысл.

Значение IDF рассчитывается путем взятия логарифма от отношения общего числа документов к числу документов, содержащих термин. Используя логарифм, мы ослабляем эффект IDF для часто встречающихся терминов.[8]

TF-IDF является произведением значений TF и IDF. Он объединяет локальную важность термина (TF) с глобальной важностью термина (IDF) для расчета взвешенного значения каждого термина в каждом документе.

$$TF - IDF(t, d) = TF(t, d) * IDF(t), \quad (2.1)$$

Полученное значение TF-IDF представляет важность термина в конкретном документе относительно его важности по всему набору данных.

Используя векторизатор TF-IDF, мы можем преобразовать коллекцию документов в числовое представление, где каждый документ представлен в виде вектора, а каждому термину внутри документа назначается вес на основе его значения TF-IDF. Этот процесс векторизации позволяет эффективно обрабатывать и анализировать текстовые данные различными алгоритмами машинного обучения.

Сглаженная обратная частота документа (Smoothed IDF) представляет собой модификацию обратной частоты документа (IDF) для избегания возможных проблем с делением на ноль или недостаточными данными в некоторых случаях [10].

В стандартной формуле IDF, когда термин не встречается в ни одном документе (т.е., число документов, содержащих термин, равно нулю), мы получаем деление на ноль, что приводит к неопределенности или ошибке. Кроме того, если термин встречается в небольшом числе документов, то его IDF-значение может быть очень высоким и сильно искажать важность термина.

Сглаженная IDF вводит некоторую поправку для решения этих проблем. Один из распространенных подходов к сглаживанию IDF - это добавление единицы к числителю и знаменателю формулы IDF. Таким образом, формула сглаженной IDF выглядит следующим образом [10]:

$$IDF(\text{e}) = log(\frac{N+1}{df+1}) + 1, \quad (2.2)$$

При добавлении единицы в числитель и знаменатель, мы гарантируем, что даже если термин не встречается ни разу в наборе данных, его IDF-значение не будет равно бесконечности, а будет иметь некоторую позитивную величину. Также сглаженная IDF снижает значительность термина, который встречается в небольшом числе документов, чтобы избежать искажения его важности.

Сглаженная IDF обычно используется в комбинации с TF (частотой термина) для вычисления значения TF-IDF, которое учитывает как локальную, так и глобальную важность терминов в документе. Это позволяет получить более сбалансированные и информативные числовые представления текстовых данных.

2.3 K-ближайший сосед

Алгоритм K-ближайших соседей (KNN) является методом машинного обучения, который используется как для классификации, так и для регрессии. Он особенно полезен для задач классификации в промышленности. KNN имеет два важных свойства [12]:

Алгоритм ленивого обучения: KNN относится к алгоритмам ленивого обучения, так как у него нет фазы обучения, и он использует все доступные данные для классификации в процессе работы. Это означает, что модель не строит явную внутреннюю представление данных во время обучения, а только запоминает образцы данных.

Непараметрический алгоритм обучения: KNN также является непараметрическим алгоритмом обучения, поскольку он не делает предположений о распределении данных или о каких-либо параметрах модели. Вместо этого он полагается на близость и сходство между образцами данных для принятия решений о классификации [12].

Классификация является одним из ключевых аспектов контролируемого обучения. Существует множество алгоритмов классификации, таких как логистическая регрессия, наивный байесовский классификатор, деревья решений, случайные леса и другие. Классификация представляет собой процесс прогнозирования принадлежности экземпляров данных к определенным классам.

Машинное обучение (МО) является широкой междисциплинарной областью, объединяющей компьютерные науки, статистику, когнитивные науки,

инженерию, оптимизацию и другие дисциплины [13]. Оно имеет множество применений, включая интеллектуальный анализ данных. Машинное обучение можно разделить на две основные категории: обучение с учителем и обучение без учителя. Обучение с учителем связано с задачами классификации и регрессии, где модель обучается на основе размеченных данных с известными ответами. Обучение без учителя, напротив, используется для обнаружения закономерностей и структуры в неразмеченных данных.

Матрица путаницы (или матрица ошибок) - это инструмент в области машинного обучения, который позволяет оценить производительность алгоритма классификации. Она представляет собой таблицу, где каждая строка соответствует реальному классу данных, а каждый столбец - предсказанному классу. Матрица путаницы позволяет видеть, как часто алгоритм делает ошибки и путает классы. Это полезный инструмент для анализа результатов классификации и получения дополнительной статистики

2.4 Дерево решений

Основной целью дерева решений является создание модели, способной определять необходимое значение переменной с помощью нескольких входных переменных [1]. В большинстве случаев алгоритмы дерева решений строятся в два этапа. Рост дерева — это первый подход. Этот метод разбивает локальные оптимальные условия рекурсивно, пока большая часть записей раздела не будет иметь одинаковую метку класса [14]. Дерево становится легче понять после обрезки, потому что его размер уменьшается [15]. Рассмотрим алгоритмы дерева решений ID3 и C4.5 в этом разделе (рисунок 2.5).

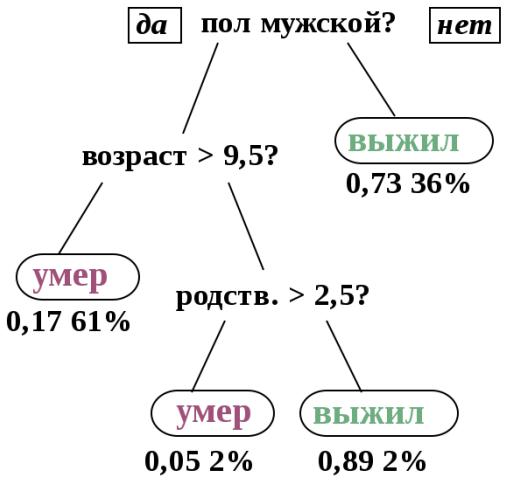

Рисунок 2.5. Пример работы алгоритма Дерево решений

(ii) Алгоритм дерева решений ID3, который также называют итеративный дихотомайзер 3, был разработан в 1986 году [16, 17]. Этот алгоритм широко используется в интеллектуальном анализе данных и машинном обучении благодаря своей простоте и эффективности [16]. Алгоритм ID3 использует данные. Недостатки и преимущества дерева решений ID3 рассматриваются [18]. Сильные стороны включают в себя то, что он включает весь учебный пример в окончательное решение, что делает его простым для понимания. Недостатки включают отсутствие обратного поиска, неспособность обработки отсутствующих значений и оптимизацию по всему миру (рисунок 2.6).

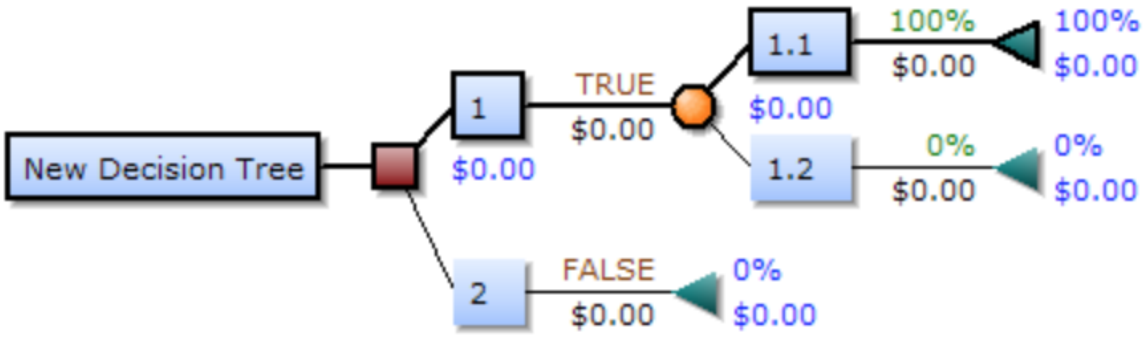

Рисунок 2.6. Узлы решений

C4.5 — это алгоритм построения деревьев решений. Одновременно это снижает недостатки и улучшает алгоритм ID3. Пытались решить проблемные ветви на этапе обрезки C4.5, заменив их конечными узлами, возвращаясь к дереву после его создания [19]. Основные преимущества C4.5 включают работу с дискретными и непрерывными функциями, предоставление возможности предварительной и последующей обрезки и обработку обучающих данных с отсутствующими значениями признаков [18, 20]. Недостатки включают длительное время обработки по сравнению с другими деревьями решений и невозможность работать с небольшим объемом данных [18].

Регрессионное дерево или дерево классификации — это инструмент, который помогает людям принимать решения. Машинное обучение, анализ данных и статистика — все это области, в которых он также используется. Дерево имеет «ветки» и «листья». Признаки, от которых зависит, определяют целевую функцию. В дереве решения «листья» содержат значения целевой функции, а остальные узлы отражают различия случаев. Чтобы классифицировать новый случай, необходимо пройти по дереву до листа и выдать соответствующее значение. В интеллектуальном анализе данных такие деревья решений часто используются. Цель состоит в том, чтобы создать модель, которая, используя несколько переменных на входе

победителей титана CART (RU), может предсказывать значение целевой переменной.Целевая переменная меняется с каждым листом от корня дерева. Каждый внутренний шов имеет входные переменные. Кроме того, можно «изучить» исходные наборы переменных, разделив их на подмножества в зависимости от проверки значений переменных. На каждом из полученных подмножеств эта процедура повторяется. Когда в узле все подмножества имеют те же значения целевой переменной и не добавляют ценностей для предсказаний, рекурсия заканчивается.

Дерево решений следует читать слева направо на вышеуказанном изображении. Дерево решений не может содержать циклические элементы, поэтому сходящиеся пути не существуют, и каждый последующий лист может просто расщепляться. Таким образом, если мы создаем дерево вручную, мы можем столкнуться с проблемой размерности, поэтому, как правило, мы можем получить решения с помощью специализированного программного обеспечения. Чтобы сделать дерево решений проще воспринимать и анализировать, его обычно представляют в виде схемы.

При добыче данных используются две основные группы деревьев решений: Когда предсказываемый результат соответствует классу, к которому принадлежат данные, используется дерево классификации; Когда предсказываемый результат можно представить в виде вещественного числа, например, стоимости дома или времени пребывания пациента в больнице, используется дерево регрессии. Брейман и др. впервые использовали термины, упомянутые выше.[2] Типы рекурсивных алгоритмов построения имеют как общие, так и различные характеристики, такие как критерии выбора разбиения в каждом узле (рисунок 2.7).[2]

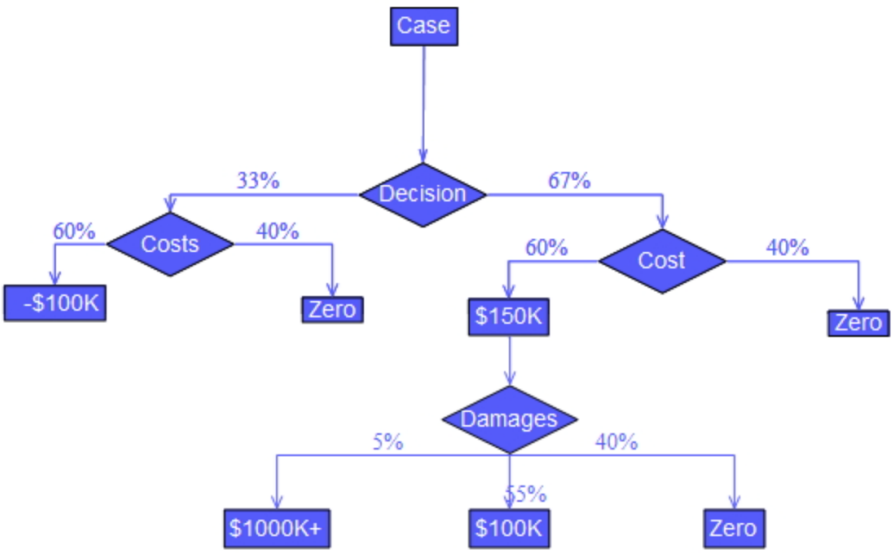

Рисунок 2.7. Дерево для регрессии

Вы можете построить несколько деревьев решений или ансамбли деревьев решений различными способами: Создать деревья решений — наиболее ранний метод. интерполирует данные с заменой, также известной как бутстреп, для создания нескольких деревьев решений, а затем получает результаты голосования деревьев, которые представляют собой их средний прогноз [3]. Классификатор «Случайный лес», основанный на бэггинге, также случайным образом выбирает подмножество признаков в каждом узле, что делает деревья более автономными; для задач классификации и регрессии может быть использован бустинг над деревьями.[4] Победители соревнований по анализу данных часто использовали алгоритм XGBoost, один из вариантов бустинга над деревьями. «Вращение леса» представляет собой процесс рассмотрения каждого дерева решений при первом использовании метода главных компонентов (PCA) на случайные подмножества входных функций.[5]

2.5 Многослойный классификатор

Алгоритм обучения с учителем многослойного классификатор (MLP) изучает функцию обучения на наборе данных, в котором m — количество введенных измерений и o — количество выведенных измерений. Он может анализировать набор функций и цель аппроксиматора нелинейной функции для классификации или регрессии (рисунок 2.8).

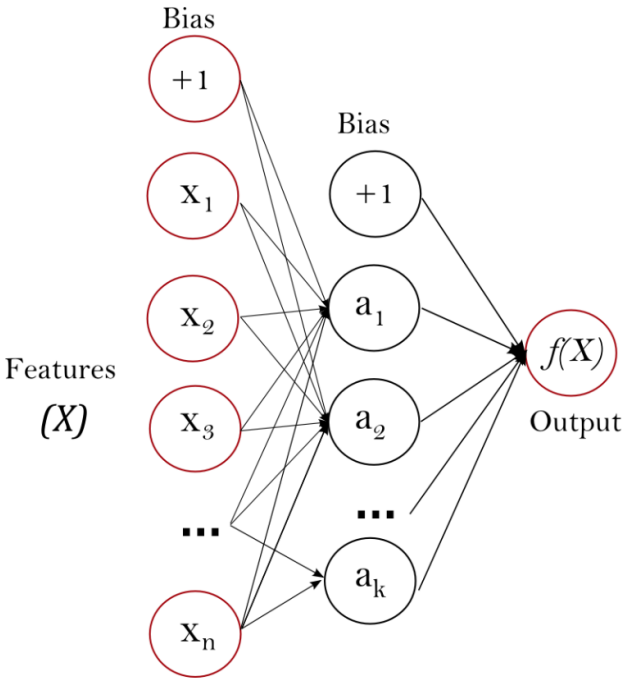

Рисунок 2.8. Алгоритм MLP

Людский мозг состоит из многочисленных биологических нейронных сетей. Мозг получает информацию от органов чувств и обрабатывает ее для создания ощущений, распознавания лиц и выполнения других задач. Нейроны мозга взаимодействуют друг с другом в структуре. Для создания искусственной нейронной сети мы также используем упрощенную архитектуру. Искусственные нейронные сети — это простой пример взаимодействия нейронов [17]. Искусственные нейроны в нейронной сети получают сигналы от нескольких входов, их преобразуют и передают другим искусственным нейронам. Из входных данных искусственный нейрон создает выходной сигнал. Для каждого входа нейрон получает вес, который умножается на соответствующий сигнал. После этого в сумматор поступает общая сумма всех взвешенных сигналов. Сумма, также известная как «нет», затем проходит через функцию активации и превращается в выходной сигнал нейрона [18]. В результате φ(net) является выходом искусственного нейрона. Чтобы добиться желаемых результатов, процесс искусственной нейронной сети копирует процессы, происходящие в биологических мозгах, когда сигналы передаются и обрабатываются между нейронами.

Алгоритм обучения с учителем многослойного персептрона (MLP) обучает на наборе данных, где m — количество измерений для ввода и o — количество измерений для вывода. Algorithm изучает функцию f(): Rm - Ro. Благодаря набору признаков X = x1, x2, x3..., xm и цели y он может изучить аппроксиматор нелинейной функции либо для классификации, либо для регрессии.

Преимущества многослойного перцептрона:

1) Возможность изучать нелинейные модели.
2) Возможность изучения моделей в режиме реального времени (онлайн-обучение) с использованием partial_fit.

К недостаткам многослойного персептрона (MLP) можно отнести:

1) MLP со скрытыми слоями имеют невыпуклую функцию потерь, когда существует более одного локального минимума. Поэтому разные инициализации случайных весов могут привести к разной точности проверки.
2) MLP требует настройки ряда гиперпараметров, таких как количество скрытых нейронов, слоев и итераций.
3) MLP чувствителен к масштабированию функций.

Набор нейронов типа xi (x1, x2,... xm) состоит из входного слоя, также известного как левый слой, и он выполняет входные функции. Каждый нейрон скрытого слоя получает значения из предыдущего слоя с помощью взвешенного линейного суммирования 1x1 + 2x2 +... +mxm. Следующим шагом является нелинейная функция активации g(), являющаяся гиперболической функцией загара. Значения передаются из последнего скрытого слоя в выходной слой.

2.6 XGBoost

Алгоритм XGBoost (Extreme Gradient Boosting) основан на концепции градиентного бустинга и представляет собой оптимизированный метод градиентного бустинга на основе деревьев решений [18]. Градиентный бустинг — это метод машинного обучения, который используется для задач классификации и регрессии и создает модель прогнозирования в виде ансамбля слабых моделей, часто деревьев решений. Обучение ансамбля проводится последовательно, а не параллельно, как в случае с методом бэггинга. На обучающем наборе данных вычисляются отклонения предсказаний уже обученного ансамбля на каждой итерации. Затем новая модель, используя другой алгоритм, подстраивается таким образом, чтобы минимизировать ошибку предыдущего ансамбля.

XGBoost основан на применении градиентного бустинга к деревьям решений. При этом на каждой итерации добавляется новое дерево, которое предсказывает отклонения предыдущих моделей. Это позволяет уменьшить среднеквадратичное отклонение модели, которое является целью оптимизации. Добавление новых деревьев продолжается до тех пор, пока ошибка уменьшается или пока не срабатывает одно из правил "ранней остановки".

Функция потери:

$$L(y_i, p_i) = -\ [y_i log(p_i) + (1 - y_i) log(1 - p_i)], \tag{2.3}$$

где y - наблюдаемая значение,

p - прогнозируемая значение.

Функция минимизации:

$$[\sum_{i=1}^{n} L(y_i, p_i)] + \frac{1}{2}\lambda O^2_{value}, \tag{2.4}$$

где O - выходное значение для листа,

λ - коэффициент регуляризации. Если λ > 0, то мы сокращаем выходное значение.

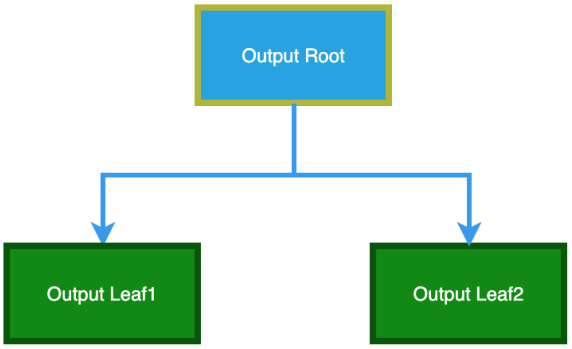

Рисунок 2.9. Дерево решение в XgBoost

Так как мы оптимизируем выходное значение из первого дерево (рисунок 9), то мы можем заменить формула на:

$$[\sum_{i=1}^{n} L(y_i, p_i^o + O_{value})] + \frac{1}{2}\lambda O_{value}^2, \quad (2.5)$$

где p^o - изначальной прогнозируемая значение.

Для построение древо решение будут использована другие формулы.
Формула:

$$Similarity\ Score = \frac{(\sum Residuals_i)^2}{\sum[p_i * (1 - p_i)] + \lambda}, \quad (2.6)$$

Иллюстрация градиентного бустинга показывает поведение модели на конкретной точке в задаче линейной регрессии. На первой итерации модель ансамбля всегда предсказывает среднее значение целевой переменной. Это предсказание может быть грубым, и среднеквадратичное отклонение на данной точке будет велико. Затем мы обучаем модель Δ1, которая корректирует предсказание предыдущего ансамбля F0 [17]. Таким образом, получаем ансамбль F1, предсказание которого состоит из суммы предсказаний моделей F0 и Δ1. Продолжая эту последовательность, мы получаем ансамбль F4, который предсказывает целевую переменную точно, используя предсказания f0, Δ1, Δ2, Δ3 и Δ4 [16].

Таким образом, алгоритм XGBoost применяет градиентный бустинг к деревьям решений, что позволяет построить мощную модель предсказания с уменьшением ошибки на каждой итерации обучения.

Модель XGBoost обладает рядом особенностей, которые позволяют использовать все возможности библиотеки, такие как scikit-learn, и добавлять регуляризацию. Она поддерживает три основные формы градиентного бустинга:

Градиентный бустинг стандартного типа с возможностью настройки скорости обучения.

Стохастический градиентный бустинг с возможностью сэмплирования по строкам и столбцам набора данных.

Регуляризованный градиентный бустинг с поддержкой L1 и L2 регуляризации.

Блочная структура для поддержки параллельного обучения деревьев.

Возможность продолжить обучение, чтобы подготовиться к использованию новых данных.

XGBoost также имеет оптимизации, которые улучшают использование аппаратного обеспечения:

1) Кэширование для оптимизации структуры данных и алгоритма.
2) Различные стратегии обработки пропущенных данных.

3) Блочная структура для поддержки параллельного обучения деревьев.
4) Возможность продолжения обучения для дообучения на новых данных.

Алгоритм XGBoost был разработан с учетом эффективности использования вычислительных ресурсов времени и памяти. Это позволяет наилучшим образом использовать доступные ресурсы для обучения модели.

2.7 Алгоритм случайного леса

Случайный лес, созданный Лео Брейманом и Адель Катлером, представляет собой один из наиболее захватывающих алгоритмов машинного обучения, который до настоящего времени остается универсальным и оригинальным в своей форме. Он проявляет свою универсальность во множестве задач, охватывая около 70% практических применений [13] (за исключением задач обработки изображений) и предлагает решения для классификации, регрессии, кластеризации, поиска аномалий, селекции признаков и других.

Качество RandomForestClassifier возрастает с увеличением числа деревьев, однако время настройки и работы алгоритма также пропорционально увеличивается. Важно отметить, что при увеличении n_estimators часто наблюдается повышение качества на обучающей выборке (даже до 100%), но качество на тестовой выборке приближается к асимптоте. Рекомендуется определить необходимое количество деревьев, учитывая эти факторы (рисунок 2.10).

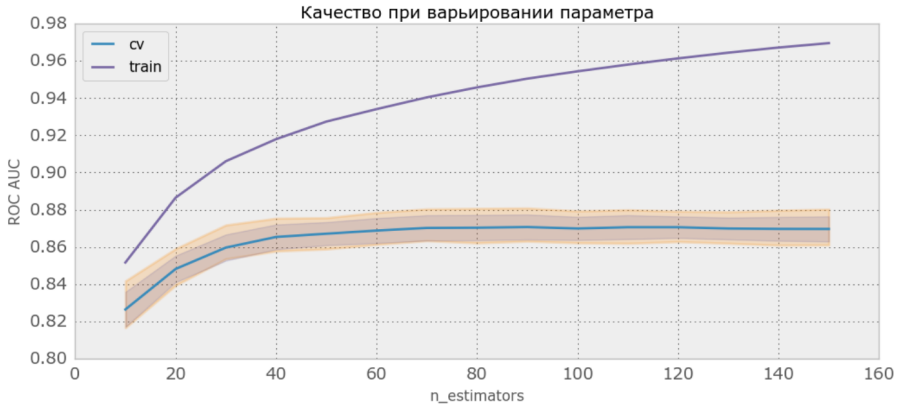

Рисунок 2.10. Качество при варьировании

RandomForestClassifier представляет собой ансамбль решающих деревьев, используемых для классификации [13]. В задачах регрессии, ответы деревьев усредняются, а в задачах классификации решение определяется голосованием большинства. Каждое дерево строится независимо, следуя следующей схеме: случайно выбирается подмножество обучающей выборки размером samplesize

(может быть с повторениями), и на ней строится дерево (каждое дерево имеет свою подвыборку). (рисунок 2.11).

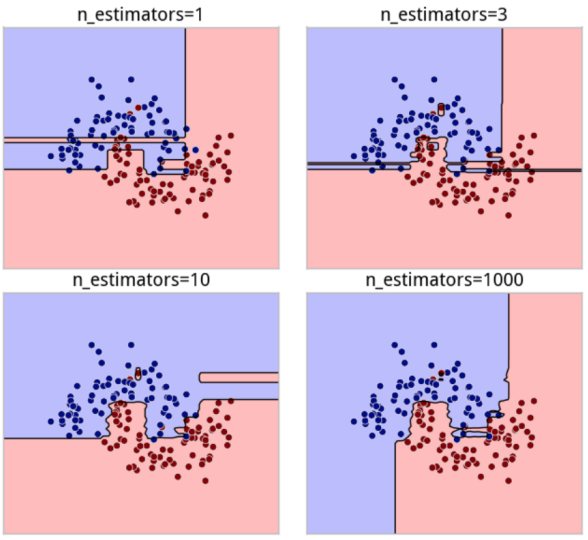

Рисунок 2.11. Деление на группы

В дереве выберите максимальные характеристики случайных признаков для каждого расщепления. Эти характеристики будут принадлежать новым расщеплениям. Лучший признак и расщепление выбираются на основе заранее заданного критерия [14]. Деревья обычно строятся до тех пор, пока в них не останутся представители только одного класса в листьях; однако в современных приложениях есть ограничения на высоту и количество объектов в деревья с листьями и количество объектов, выбранных для разделения. Такая схема построения соответствует основному принципу ансамблирования, где базовые алгоритмы должны быть хорошими и разнообразными (поэтому каждое дерево строится на своей обучающей выборке, а при выборе расщеплений используется случайность).

RandomForestClassifier представляет собой алгоритм машинного обучения, применяемый в задачах классификации. Он относится к категории алгоритмов, называемых "случайный лес" (Random Forest), которые основываются на построении нескольких деревьев решений и объединении их выводов. Принцип работы алгоритма RandomForestClassifier следующий [15]: для каждого дерева в наборе случайно выбирается подмножество признаков, и на основе этих признаков строится отдельное дерево решений. Когда требуется классифицировать новый объект, каждое дерево в лесу предсказывает свою метку, и итоговый результат определяется путем голосования, где класс с наибольшим числом голосов становится окончательным прогнозом.

2.8 Метод опорных векторов

Один из наиболее распространенных методов обучения для решения задач классификации и регрессии — метод опорных векторов, также известный как SVM. Основная концепция метода заключается в создании гиперплоскости, которая разделяет объекты выборки наилучшим образом [15]. Алгоритм предполагает, что средняя ошибка классификатора будет меньше, чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов.

Преимущества SVM:

1) Эффективен на пространствах большой размерности, то есть тогда когда объекты имеют много параметров • остается эффективным в случае когда количество измерений больше количества обучающих примеров • использует подмножество тренировочных данных в функции выработки решений (support vector), что обеспечивает эфф ективное использования памяти.
2) Универсальность - могут использоваться разные ядерные функции (Kernel functions), включая определенные пользователем

Недостатки:

1) Если количество примеров нам ного меньше числа свойств трудно избежать переобучения
2) Не поддерживает напрямую вероятностные оценки принадлежности классу, они рассчитываются с помощью довольно трудоемкой кроссвалидационной функции

Рассмотрим задачу бинарной классификации, в которой мы хотим разделить два класса, помеченных как 0 и 1, используя линейную границу решения.(рисунок 2.12).

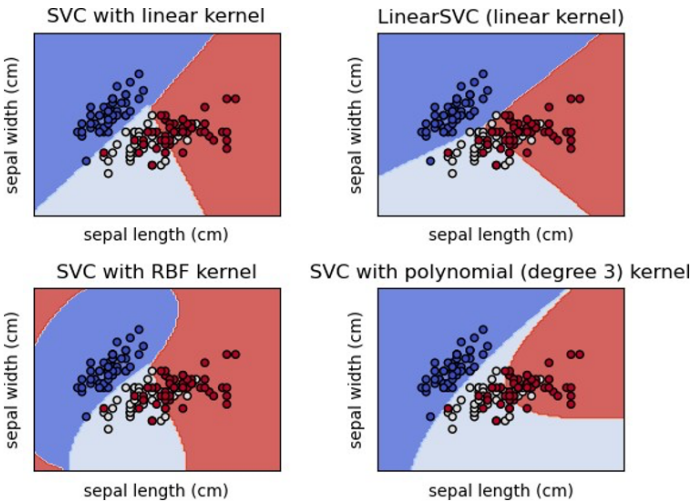

Рисунок 2.12. Виды SVC

Алгоритм SVM направлен на поиск оптимальной гиперплоскости, которая максимально разделяет два класса с максимально возможным запасом. Гиперплоскость определяется линейным уравнением. [16].

Алгоритм SVM направлен на поиск оптимальных w и b путем решения задачи оптимизации, которая включает в себя максимизацию маржи при минимизации ошибок классификации.

Опорные векторы в SVM — это точки данных, которые лежат ближе всего к границе или границе решения. Запас — это расстояние между границей решения и опорными векторами.

Алгоритм SVM выбирает опорные векторы как важные точки данных, которые способствуют определению границы решения. Граница решения определяется опорными векторами, и любые изменения в других точках данных за границей не влияют на границу решения [12].

SVM не ограничены линейными границами решений. Они могут обрабатывать нелинейные границы решений, используя технику, называемую трюком ядра. Хитрость ядра заключается в отображении исходного пространства признаков в многомерное пространство признаков, где классы становятся разделяемыми линейной границей. Примерами часто используемых ядер являются полиномиальное ядро, ядро радиальной базисной функции Гаусса (RBF) и сигмовидное ядро.

Алгоритм SVM находит оптимальную гиперплоскость в преобразованном пространстве признаков, что позволяет ему фиксировать сложные шаблоны и достигать более высокой производительности классификации.

В некоторых случаях классы не могут быть идеально разделены линейной границей. В таких ситуациях алгоритм SVM допускает мягкую маржу.

Мягкая граница SVM допускает некоторые неправильные классификации и позволяет точкам данных находиться на неправильной стороне поля или даже на неправильной стороне границы решения [11]. Эта гибкость контролируется параметром регуляризации, часто обозначаемым как C. Меньшее значение C допускает более широкий диапазон и большее количество неправильных классификаций, в то время как большее значение C приводит к более узкому диапазону с меньшим количеством неправильных классификаций.

SVM могут быть расширены для решения задач классификации нескольких классов. Одним из подходов является стратегия «один против одного» (OvO), при которой обучаются несколько классификаторов SVM, каждый из которых различает два класса.

Другим подходом является стратегия «один против остальных» (OvR), при которой для каждого класса обучается отдельный классификатор SVM по сравнению с остальными классами.

Во время вывода в качестве окончательного прогноза выбирается класс с наибольшей достоверностью от отдельных классификаторов.

2.9 Коэффициент Жаккара

Сходство Жаккара можно использовать для вычисления сходства между двумя асимметричными двоичными переменными. Предположим, бинарная переменная имеет только одно из двух состояний: 0 и 1, где 0 означает, что атрибут отсутствует, а 1 — что он присутствует [5]. Хотя каждое состояние одинаково ценно для симметричных двоичных атрибутов, два состояния не одинаково важны для асимметричных двоичных переменных [6].

$$J(i,j) = sim(i,j) = \frac{a}{a+b+c}, \quad (2.6)$$

где a = количество атрибутов, равных 1 для объектов i и j;

b = количество атрибутов, которые равны 0 для объекта i, но равны 1 для объекта j;

c = количество атрибутов, которые равны 1 для объекта i, но равны 0 для объекта j;

d = количество атрибутов, равных 0 для обоих объектов i и j;

Предположим, что таблица транзакций клиентов содержит 9 элементов и 3 клиентов. Сходство между объектами вычисляется только на основе асимметричных атрибутов (рисунок 2.13).

	item1	item2	item2	item4	item5	item6	item7	item8	item9
C1	0	1	0	0	0	1	0	0	1
C2	0	0	1	0	0	0	0	0	1
C3	1	1	0	0	0	1	0	0	0

Рисунок 2.13. Пример Индекса

Сходство между каждой парой из трех клиентов можно рассчитать с помощью коэффициента Жаккара:

$$J(C1,C2) = \frac{a}{a+b+c} = \frac{1}{1+1+2} = 0.25J(C1,C2), \quad (2.7)$$

Эти измерения показывают, что C1 и C3 имеют одинаковое покупательское поведение, тогда как C2 и C3 не похожи, поскольку они купили совершенно разные товары [7].

Наконец, вместо сходства можно рассчитать несходство или расстояние Жаккара между двумя бинарными атрибутами. Различие на основе этих признаков по коэффициенту Жаккара вычисляется следующим образом:

$$d(i,j) = \frac{b+c}{a+b+c} \Rightarrow 1 - sim(i,j), \quad (2.8)$$

Сходство Жаккара измеряет сходство между двумя наборами данных, чтобы увидеть, какие элементы являются общими, а какие разными. Чтобы определить сходство Жаккара, необходимо делить количество наблюдений в обоих наборах на количество наблюдений в каждом наборе. Иными словами, сходство Жаккара можно определить, разделив размер пересечения на размер объединения двух множеств. Это можно записать в системе обозначений, используя пересечение $(A \cap B)$ и союзы $(A \cup B)$ из двух наборов:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \quad (2.9)$$

где $|A \cap B|$ дает количество членов, разделенных между обоими наборами и

$|A \cup B|$ дает общее количество элементов в обоих наборах (общих и неразделенных).

Сходство Жаккара будет 0 если два набора не имеют общих значений и 1 если эти два набора идентичны. Набор может содержать либо числовые значения, либо строки.

Кроме того, эту функцию можно использовать для нахождения различий между двумя наборами путем вычисления $d(A,B) = 1 - J(A,B)$

Пример: Ниже приведен простой пример для вычисления подобия Жаккара между следующими двумя наборами.

$A = \{0,1,2,5,6\}$

$B = \{0,2,3,4,5,7,9\}$

Сходство Жаккара между двумя наборами вычисляется следующим образом:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\{0,2,5\}}{\{0,1,2,3,4,5,6,7,9\}} = \frac{3}{9} = 0.33, \quad (2.10)$$

2.10 Косинусное сходство

Косинус угла между двумя векторами используется для измерения расстояния между двумя точками на плоскости. Цель меры косинусного подобия заключается в том, что сходство точек данных уменьшается с увеличением расстояния.

$$similarity = cos(\theta) = \frac{A \cdot B}{|A| \cdot |B|}, \quad (2.11)$$

Косинусные сходства обычно используются для символьных типов данных. С точки зрения машинного обучения они также могут использоваться для различных типов данных классификации и помогают нам определять ближайших соседей при использовании их в качестве оценочной метрики в алгоритме KNN. В системе рекомендаций используется тот же принцип косинусоидальных углов: контент, имеющий меньшее сходство, будет считаться наименее рекомендуемым, а контент с более высоким сходством будет считаться лучшим. В текстовых данных также используется косинусная сходимость, чтобы определить, насколько похожи векторизованные тексты из исходного текстового документа [9].

Эта метрика обычно используется в различных областях, включая обработку естественного языка, поиск информации и рекомендательные системы.

Чтобы понять косинусное сходство, давайте рассмотрим два вектора, A и B, в n-мерном пространстве. Каждое измерение представляет функцию или [9] атрибут, а величина каждого измерения представляет значение или силу этой функции. Векторы можно представить в виде:

A = [a_1, a_2, a_3, ..., an]

B = [b_1, b_2, b_3, ..., bn]

Косинусное подобие между A и B вычисляется следующим образом:

$$cosine\_similarity(A, B) = \frac{A \cdot B}{||A|| \cdot ||B||}, \quad (2.12)$$

Здесь (AB) представляет скалярное произведение векторов A и B, которое вычисляется путем умножения соответствующих элементов векторов и их суммирования [9].

||A|| и ||B|| представляют евклидовы нормы (также известные как нормы L2) векторов A и B соответственно [9]. Евклидова норма вектора вычисляется как квадратный корень из суммы квадратов его отдельных компонентов:

$$||A|| = \sqrt{(a_1{}^2 + a_2{}^2 + a_3{}^2 + \ldots + a_n{}^2)}, \quad (2.13)$$

$$||B|| = \sqrt{(a_1{}^2 + a_2{}^2 + a_3{}^2 + \ldots + a_n{}^2)}, \quad (2.14)$$

Результирующее значение косинусного сходства находится в диапазоне от -1 до 1. Значение 1 указывает, что векторы идентичны или указывают в одном направлении, а значение -1 указывает, что они диаметрально противоположны.

Значение, близкое к 0, предполагает незначительное сходство между векторами или его полное отсутствие (рисунок 2.14).

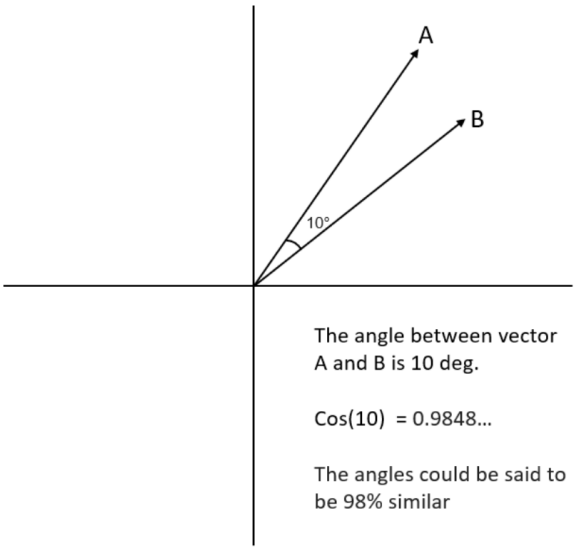

Рисунок 2.14. Косинусное сходство

Косинусное сходство часто используется для сравнения документов, где каждый документ представлен в виде вектора частот терминов или оценок TF-IDF. Он обеспечивает эффективный и действенный анализ подобия, позволяя решать такие задачи, как кластеризация документов, поисковое ранжирование и системы рекомендаций на основе контента [10].

Существуют различные меры расстояния, которые используются в качестве метрики для оценки точек данных. Некоторые из них заключаются в следующем:

- Евклидово расстояние
- Манхэттенское расстояние
- Минковский расстояние
- Расстояние Хэмминга и многое другое.

Среди всех этих популярных метрик для расчета расстояний и при рассмотрении для классификации или текстовых данных вместо косинусного сходства расстояние Хэмминга можно использовать в качестве метрики для KNN, рекомендательных систем и текстовых данных. Но расстояние Хэмминга учитывает только символьный тип данных одинаковой длины, а косинусное сходство позволяет обрабатывать данные переменной длины. При рассмотрении текстовых данных расстояние Хэмминга [10] не будет учитывать часто встречающиеся слова в документе и будет отвечать за получение более низкого индекса подобия из текстового документа, в то время как косинусное сходство учитывает часто встречающиеся слова в текстовом документе и поможет получить более высокое сходство. баллы за текстовые данные.

Для задач классификации косусное сходство в машинном обучении можно использовать; его можно использовать в качестве метрики в алгоритмах классификации KNN для определения наилучшего количества соседей, а также

можно сравнить подходящую модель KNN с различными алгоритмами машинного обучения классификации и Классификатор KNN [15] с косинусной сходством в качестве метрики единственный, который может использоваться для оценки различных параметров производительности, таких как оценка точности, оценка AUC и отчет о классификации. Кроме того, он может быть использован для оценки таких параметров, как точность и полнота.

Таким образом, в машинном обучении можно использовать косинусное сходство в качестве метрики, чтобы определить идеальное количество соседей. В этом случае точки данных с более высоким сходством будут рассматриваться как ближайшие соседи, а точки данных с более низким сходством не будут учитываться. Вот как используется косинусное сходство в машинном обучении.

Косинусные сходства текстовых данных используются для оценки сходства между двумя размещенными текстами или текстовыми документами. На начальном этапе необработанные текстовые данные должны быть токенизированы, чтобы использовать косинусное сходство в текстовых данных. Затем из матрицы подобия можно получить матрицу, которую можно передать в метрики косинусного сходства для оценки сходства между двумя текстовыми документами.

Из-за своей динамической способности адаптироваться к различным типам данных сходство косинусов является наиболее распространенным показателем в различных задачах машинного обучения и обработке текстовых данных. Косинусное сходство, которое полностью зависит от свойств косинусного угла, широко используется в рекомендательных системах, поскольку оно помогает нам рекомендовать контент пользователю в соответствии с его наиболее просматриваемым контентом и характеристиками [16]. В основном это используется для определения сходств между часто встречаемыми терминами в текстовых документах. Благодаря этому сходство косинусов стало популярным показателем для оценки в различных приложениях.

2.11 Выводы по второму разделу

В этом разделе, были рассмотрены разные методы и алгоритмы машинного обучение. Особое внимание было уделено, алгоритмам nlp и классификации, так как в последующем разделе, будут проведено сравнительный анализ методов классификации, чистка датасета и нахождение лучших кандидатов посредством nlp.

3 Разработка системы скрининга

3.1. Датасет и его обзор

Коллекция примеров резюме, взятых с livecareer.com, для классификации данного резюме по любой из меток, определенных в наборе данных.

Датасет содержит более 2400 резюме в виде строки, а также в формате PDF.(рисунок 3.1).

Внутри CSV:

ID: Уникальный идентификатор и имя файла для соответствующего PDF-файла.

Resume_str : Содержит текст резюме только в строковом формате.

Resume_html : содержит данные резюме в формате html, которые были представлены во время очистки веб-страниц.

Категория : Категория работы, для которой использовалось резюме.

Категории которые указанные в датасете: HR, Дизайнер, Информационные технологии, Учитель, Адвокат, Развитие бизнеса, Здравоохранение, Фитнес, Сельское хозяйство, BPO, Продажи, Консультант, Digital-Media, Автомобиль, Шеф-повар, Финансы, Одежда, Инженерное дело, Бухгалтер, Строительство, Связи с общественностью, Банковское дело, Искусство, Авиация.

Ввод [2]:
```
resumeDataSet = pd.read_csv('Resume.csv' ,encoding='utf-8') #UpdatedResumeDataSet
resumeDataSet.head(5)
```

Out[2]:

	ID	Resume_str	Resume_html	Category
0	16852973	HR ADMINISTRATOR/MARKETING ASSOCIATE\...	<div class="fontsize fontface vmargins hmargin...	HR
1	22323967	HR SPECIALIST, US HR OPERATIONS ...	<div class="fontsize fontface vmargins hmargin...	HR
2	33176873	HR DIRECTOR Summary Over 2...	<div class="fontsize fontface vmargins hmargin...	HR
3	27018550	HR SPECIALIST Summary Dedica...	<div class="fontsize fontface vmargins hmargin...	HR
4	17812897	HR MANAGER Skill Highlights ...	<div class="fontsize fontface vmargins hmargin...	HR

Ввод [3]:
```
resumeDataSet.shape
```

Out[3]:
```
(2484, 4)
```

Рисунок 3.1. Датасет

PDF-файл, хранящийся в папке данных, разделен на соответствующие метки в виде папок, причем каждое резюме находится внутри папки в формате PDF с именем файла в качестве идентификатора, определенного в csv.

Вместо количества или частоты мы также можем визуализировать распределение категорий вакансий в процентах, как показано (рисунок 3.2):

```
Ввод [5]: plt.figure(1, figsize=(25,25))
the_grid = GridSpec(2, 2)
plt.subplot(the_grid[0, 1], aspect=1, title='CATEGORY DISTRIBUTION')
source_pie = plt.pie(targetCounts, labels=targetLabels, autopct='%1.1f%%', shadow=True)
```

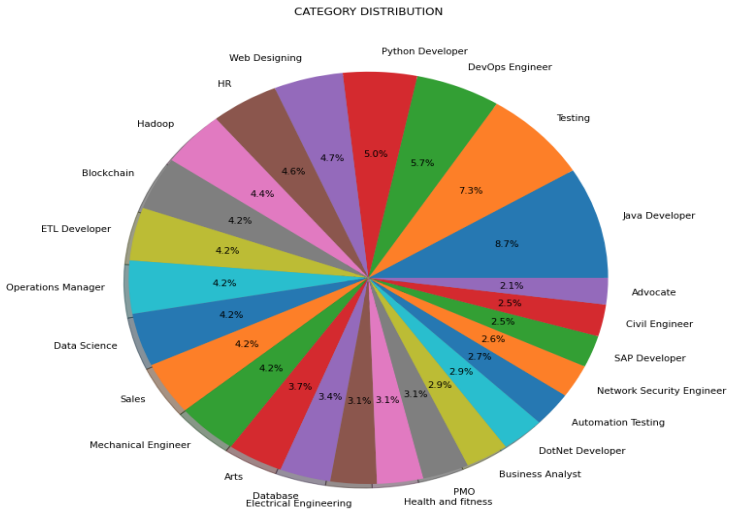

Рисунок 3.2. Pie chart

В данных есть 25 различных категорий. (рисунке 3.3).

```
Ввод [4]: plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)

targetCounts = resumeDataSet['Category'].value_counts().reset_index()['Category']
targetLabels  = resumeDataSet['Category'].value_counts().reset_index()['index']
```

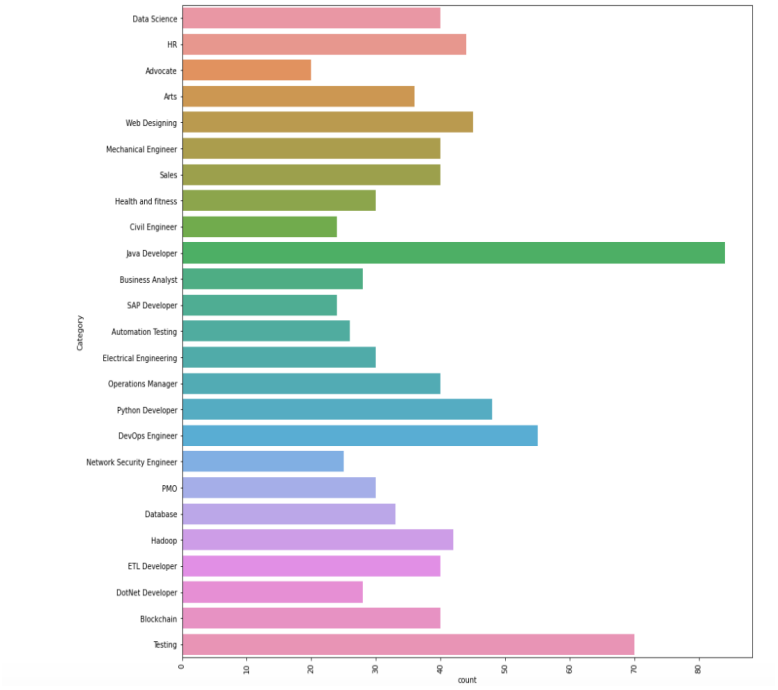

Рисунок 3.3. График категории

Как мы видим, резюме имеет символы, которые будут мешаться в дальнейшем исследование, поэтому датасет должен быть подготовленным.

3.2. Предварительная обработка датасета

Этот процесс очистит входные данные CV от любых ненужных символов. В процессе очистки все специальные символы, цифры и однобуквенные слова удаляются. После выполнения этих процедур у нас был чистый набор данных, лишенный каких-либо особых символов, цифр или однобуквенных слов. С помощью токенов NLTK набор данных разбивается на токены. Кроме того, этапы предварительной обработки, такие как лемматизация, формирование корней и удаление стоп-слов, применяются к токенизированному набору данных. Данные в поле резюме были очищены от цифр и пробелов в дате после импорта необработанного файла резюме. Следующие шаги были предприняты для маскирования данных:

- Фрагменты строки маски, такие как ‘\x’
- Маскируйте фрагменты строки для управляющих последовательностей, таких как \a, \b, \t, \n.
- Маскировать все номера
- Замените все однобуквенные слова пустой строкой.
- Маскировать адреса электронной почты

Чтобы обработать текст, мы использовали регулярные выражение. (рисунке 3.4).

2.1. Clean the ‘Resume’ column

Ввод [6]:

```
def cleanResume(resumeText):
    resumeText = re.sub('httpS+s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@S+', '  ', resumeText)  # remove mentions
    resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[]^_`{|}~"""), ' ', resumeText)
    resumeText = re.sub(r'[^x00-x7f]',r' ', resumeText)
    resumeText = re.sub('s+', ' ', resumeText)  # remove extra whitespace
    return resumeText
resumeDataSet['cleaned_resume'] = resumeDataSet['Resume_str'].apply(lambda x: cleanResume(x))
resumeDataSet.head(5)
```

Out[6]:

	ID	Resume_str	Resume_html	Category	cleaned_resume
0	16852973	HR ADMINISTRATOR/MARKETING ASSOCIATE\...	<div class="fontsize fontface vmargins hmargin...	HR	HR ADMINISTRATOR MARKETING ASSOCIATE ...
1	22323967	HR SPECIALIST, US HR OPERATIONS ...	<div class="fontsize fontface vmargins hmargin...	HR	HR SPECIALIST US HR OPERATIONS ...
2	33176873	HR DIRECTOR Summary Over 2...	<div class="fontsize fontface vmargins hmargin...	HR	HR DIRECTOR Summar Over 2...
3	27018550	HR SPECIALIST Summary Dedica...	<div class="fontsize fontface vmargins hmargin...	HR	HR SPECIALIST Summar Dedica...
4	17812897	HR MANAGER Skill Highlights ...	<div class="fontsize fontface vmargins hmargin...	HR	HR MANAGER Skill Highlight ...

Рисунок 3.4. Обработка колонны Резюме

Также для улучшение оценки модели, нужно кодировать target колонну, в нашем случае это колонна Category.

2.2. Encoding 'Category' ¶

Ввод [7]:
```
var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    resumeDataSet[i] = le.fit_transform(resumeDataSet[i])
resumeDataSet.head(5)
```

Out[7]:

	ID	Resume_str	Resume_html	Category	cleaned_resume
0	16852973	HR ADMINISTRATOR/MARKETING ASSOCIATE\...	<div class="fontsize fontface vmargins hmargin...	19	HR ADMINISTRATOR MARKETING ASSOCIATE ...
1	22323967	HR SPECIALIST, US HR OPERATIONS ...	<div class="fontsize fontface vmargins hmargin...	19	HR SPECIALIST US HR OPERATIONS ...
2	33176873	HR DIRECTOR Summary Over 2...	<div class="fontsize fontface vmargins hmargin...	19	HR DIRECTOR Summar Over 2...
3	27018550	HR SPECIALIST Summary Dedica...	<div class="fontsize fontface vmargins hmargin...	19	HR SPECIALIST Summar Dedica...
4	17812897	HR MANAGER Skill Highlights ...	<div class="fontsize fontface vmargins hmargin...	19	HR MANAGER Skill Highlight ...

Рисунок 3.5. Кодирование колонны Категория

Мы предварительно обработаем и преобразуем столбец «cleaned_resume» в векторы. Есть много способов сделать это, например, «Bag of Words», «Tf-Idf», «Word2Vec» и комбинация этих методов. В данном этапе мы использовали “Tf-Idf”. (рисунок 3.6).

2.3. Preprocessing 'cleaned_resume' column

Here we will preprocess and convert the 'cleaned_resume' column into vectors. There are many ways to do that like 'Bag of Words', 'Tf-Idf', 'Word2Vec' and a combination of these methods.

Ввод [8]:
```
requiredText = resumeDataSet['cleaned_resume'].values
requiredTarget = resumeDataSet['Category'].values
word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    stop_words='english',
    max_features=1500)
word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)
WordFeatures
```

Out[8]:
```
<962x1500 sparse matrix of type '<class 'numpy.float64'>'
        with 124956 stored elements in Compressed Sparse Row format>
```

Рисунок 3.6. Векторизация

Благодаря векторизации, в дальнейшем мы можем построить модель для классификации.(рисунок 3.6).

3.3 Построение модели классификации

В начале рассмотрим результаты KNN.

Результаты алгоритма неудовлетворительны, потому что оценка на тестовой показывает 45% что является плохим результатом.

Чтобы получить более подробную статистику мы должны использовать Confusion matrix. (рисунке 3.7).

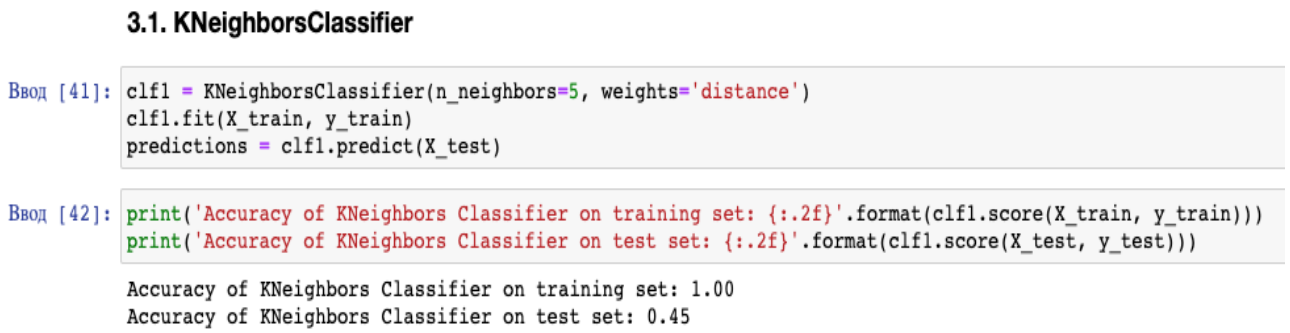

Рисунок 3.7. Алгоритм kNN

Свободный лес состоит из нескольких деревьев выбора. В зависимости от характеристик данных деревья решений разбивают данные на более мелкие группы. Например, длина и цвет лепестков будут признаками в наборе данных о цветах. До тех пор, пока не будет выделено небольшое количество данных под одной меткой (классификация), деревья решений будут продолжать разбивать данные на группы. Выбор места для обеда является конкретным примером. Чтобы найти место для еды, можно использовать Yelp, чтобы узнать, открыто ли место сейчас, насколько популярно оно, каков средний рейтинг и нужно ли мне бронировать столик.

Следующим рассмотрим MLP.(рисунок 3.8).

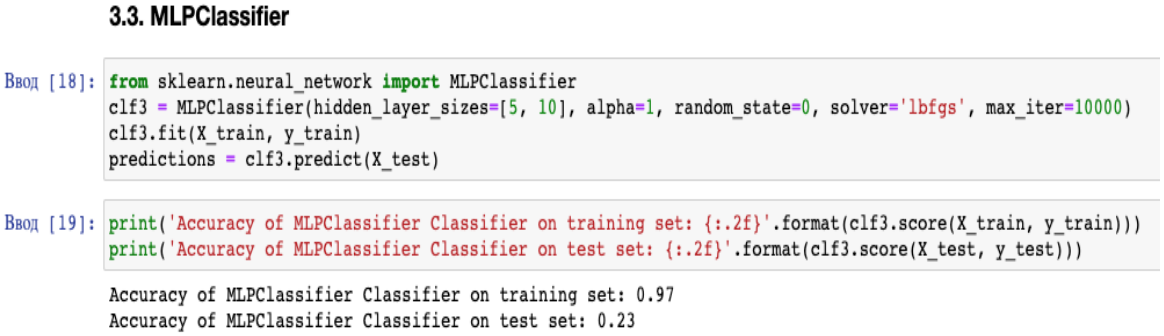

Рисунок 3.8. MLP алгоритм

Мы должны использовать нелинейную функцию активации при работах со скрытыми слоями. Это связано с тем, что классификация рукописных цифр является сложной задачей, которая не может быть выполнена линейно. Наша модель MLP не будет исследовать какие-либо нелинейные отношения в данных, если нет нелинейной функции активации в скрытых слоях. Таким образом, функция активации ReLU используется в обоих скрытых слоях. ReLU — это функция активации, которая не является нелинейной.

Мы используем функцию активации Softmax в выходном слое. Для задачи мультиклассовой классификации это единственный доступный вариант. Выходной слой состоит из десяти узлов, каждый из которых имеет десять меток.

Результаты алгоритма неудовлетворительны, потому что оценка на тестовой показывает 23% что является худшим результатом из всех. (рисунок 3.9).

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
rfc.fit(X_train, y_train)
rfc_score = rfc.score(X_test, y_test)
print('Accuracy of RandomForestClassifier on test set: {:.2f}'.format(rfc_score))
```

```
Accuracy of RandomForestClassifier on test set: 0.65
```

Рисунок 3.9. RandomForestClassifier

Результаты алгоритма удовлетворительны, второй результат из вышеуказанных алгоритмов.

Теперь рассмотрим XGBoost.

Алгоритм XGBoost применяет градиентный бустинг к деревьям решений, что позволяет создавать эффективные модели предсказания с уменьшением ошибки на каждой итерации обучения.

Модель XGBoost имеет несколько функций, которые позволяют добавлять регуляризацию и использовать все возможности библиотеки, такие как scikit-learn. Он поддерживает три основных вида градиентного бустинга: стандартный градиентный бустинг с возможностью настройки скорости обучения. (рисунок 3.10).

XGBoost основан на применении градиентного бустинга к деревьям решений. При этом на каждой итерации добавляется новое дерево, которое предсказывает отклонения предыдущих моделей. Это позволяет уменьшить среднеквадратичное отклонение модели, которое является целью оптимизации. Добавление новых деревьев продолжается до тех пор, пока ошибка уменьшается или пока не срабатывает одно из правил "ранней остановки".

Таким образом, алгоритм XGBoost применяет градиентный бустинг к деревьям решений, что позволяет построить мощную модель предсказания с уменьшением ошибки на каждой итерации обучения.

Модель XGBoost обладает рядом особенностей, которые позволяют использовать все возможности библиотеки, такие как scikit-learn, и добавлять регуляризацию.

```
from sklearn.feature_extraction.text import TfidfVectorizer

X_train,X_test,y_train,y_test = train_test_split(requiredText,requiredTarget,random_state=42, test_size=0.2)

# Initialize the TF-IDF vectorizer with term frequency normalization and smoothed IDF
tfidf_vectorizer = TfidfVectorizer(norm='l2', smooth_idf=True)

# Fit and transform the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initialize the XGBoost classifier
xgb_classifier = XGBClassifier(learning_rate=0.1,
                               n_estimators=100,
                               max_depth=2,
                               min_child_weight=2,
                               gamma=0.2,
                               subsample=0.8,
                               colsample_bytree=0.9,
                               reg_alpha=0.01,
                               reg_lambda=0.01,
                               objective='multi:softmax',
                               num_class=len(set(requiredTarget)))

# Train the classifier
xgb_classifier.fit(X_train_tfidf, y_train)


# Make predictions on the test set
y_pred = xgb_classifier.predict(X_test_tfidf)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
/Users/daniyar/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encod
er in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following:
1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integ
ers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[01:00:47] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the ob
jective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.

0.8390342052313883
```

Рисунок 3.10. XGBoost

Результаты алгоритма удовлетворительны, потому что оценка на тестовой показывает 84% что является лучшим результатом из всех алгоритмов в данной диссертации.

Хотя в дальнейшем для улучшения результата, рекомендуется более качественная подготовка данных. (рисунок 3.11).

3.5 . SVM

Ввод [26]:
```
from sklearn.svm import SVC
clf5 = SVC(kernel = 'linear', C=500)
clf5.fit(X_train, y_train)
predictions = clf5.predict(X_test)
```

Ввод [27]:
```
print('Accuracy of SVC Classifier on training set: {:.2f}'.format(clf5.score(X_train, y_train)))
print('Accuracy of SVC Classifier on test set: {:.2f}'.format(clf5.score(X_test, y_test)))
```

```
Accuracy of SVC Classifier on training set: 1.00
Accuracy of SVC Classifier on test set: 0.63
```

Рисунок 3.11. SVM

Результаты алгоритма удовлетворительны, потому что оценка на тестовой показывает 63% что является вторым лучшим результатом из всех алгоритмов в данной диссертаций.

3.4. База данных.

БД основан на парах «ключ-значение». В отличие от других типов баз данных, базы данных с парами «ключ-значение» обеспечивают беспрецедентное горизонтальное масштабирование и поддерживают высокую разделяемость. Игровые, рекламные и приложения IoT являются хорошими примерами использования баз данных типа «ключ-значение». При любом масштабе Amazon DynamoDB обеспечивает стабильную работу БД [19] с задержкой нескольких миллисекунд. Поскольку функция Snapchat Stories связана с самым большим объемом записей в хранилище, основной причиной переноса Snapchat Stories в сервис DynamoDB была устойчивая производительность.

Документ: из-за того, что JSON является эффективной и простой моделью данных для разработчиков, данные часто представлены в коде приложения как объект или документ в формате JSON. С помощью той же документной модели, которую используют в коде приложения, разработчики могут использовать документные базы данных для хранения и запросов данных в базе данных. Гибкость, полуструктурированность и иерархичность документов и баз данных позволяют им развиваться в соответствии с потребностями различных приложений. В системах управления контентом, каталогах и пользовательских профилях, где каждый документ уникален и изменяется со временем, документная модель работает хорошо. Amazon DocumentDB (совместимая с MongoDB) и MongoDB — распространенные документные базы данных, которые предлагают гибкие API, которые просты в использовании.

БД с графиками. С помощью графических баз данных разработка и запуск приложений, работающих с наборами сложных данных, становится проще. Социальные сети, сервисы рекомендаций, системы выявления мошенничества и графы знаний — это типичные примеры использования графовых баз данных. Amazon Neptune — это полностью управляемый графический базовый сервер. Благодаря графовым API TinkerPop и RDF/SPARQL Neptune поддерживает модель Property Graph и Resource Description Framework (RDF). Giraph и Neo4j — распространенные графовые БД. БД [19] в памяти Игровые и рекламные приложения часто используют таблицы лидеров, хранение сессий и аналитику в режиме реального времени. Такие возможности требуют ответа всего в несколько микросекунд, но поток может резко увеличиться в любое время. Надежный сервис базы данных в памяти Amazon MemoryDB для Redis уменьшает задержку чтения до миллисекунд и обеспечивает надежность в нескольких зонах доступности. MemoryDB обладает исключительной надежностью и производительностью, что позволяет ей использоваться как

основной базой данных для современных приложений, основанных на микросервисах. Amazon ElastiCache — это полностью управляемый сервис кэширования в памяти, который работает с Redis и Memcached для обслуживания рабочих нагрузок с высокой пропускной способностью и низкой задержкой. Вместо того, чтобы хранить данные на диске, пользователи, такие как Tinder, требуют, чтобы их приложения отвечали в режиме реального времени, используют системы хранения данных в памяти. Amazon DynamoDB Accelerator (DAX) — это еще один пример специально разработанного хранилища данных. DAX [17] ускоряет считывание данных для DynamoDB.

БД поиска. Многие приложения создают журналы, чтобы помочь разработчикам найти и исправить неполадки. Специально разработанный сервис Amazon OpenSearch позволяет визуализировать и анализировать автоматически генерируемые потоки данных в режиме реального времени с помощью индексирования, агрегации частично структурированных журналов и метрик и поиска по ним. Кроме того, Amazon OpenSearch — это мощный и эффективный инструмент для полнотекстового поиска. Для различных особо важных целей, от операционного мониторинга и устранения неисправностей до отслеживания стека распределенных приложений и оптимизации затрат, Expedia использует более 150 доменов сервиса Amazon OpenSearch, а также 30 миллиардов документов.

Несмотря на то, что в исследовании использовался датасет, в будущем для улучшения проекта необходимо подключить базу данных. Поскольку он является NoSql-базой данных, рекомендуется использовать MongoDb для этого проекта.

Самый интересный аспект этого термина заключается в том, что хотя он был использован впервые в конце 90-х годов, его настоящий смысл в том виде, в котором он используется сегодня, появился только в середине 2009 года. Изначально это была опенсорсная база данных, разработанная Карлом Строззи. Он хранил все данные в виде ASCII файлов и позволял получить доступ к данным с помощью шелловских скриптов, а не SQL. Она не имела ничего общего с «NoSQL» в его нынешнем виде. В июне 2009 года Йохан Оскарссон организовал встречу в Сан-Франциско [17], чтобы обсудить новые тенденции на ИТ-рынке хранения и обработки данных. Новые опенсорсные продукты, такие как Dynamo и BigTable, стали основным мотиватором встречи. Для создания яркой вывески для встречи необходимо было придумать краткий и емкий термин, который хорошо подходит для хэштега в Твиттере. Эрик Эванс из RackSpace предложил термин «NoSQL». Хотя термин был разработан только на одной конференции и не имел значительной смысловой нагрузки, он быстро распространился по всей сети, как вирусная реклама, и в конечном итоге стал названием целого направления в ИТ-индустрии. К слову, на конференции выступали Voldemort (копия Amazon Dynamo), Cassandra, Hbase (копия Google BigTable), Hypertable, CouchDB и MongoDB. Важно отметить, что слово «NoSQL» является полностью стихийным и не имеет официально признанного определения или научной основы. Название скорее отражает направление развития ИТ в направлении реляционных баз данных. Хотя есть сторонники и

прямое определение «Нет только SQL», это расшифровывается как «Не только SQL». Прамод Садаладж и Мартин Фаулер попытались объединить и систематизировать информацию о мире NoSQL в своей недавней книге «NoSQL Distilled».

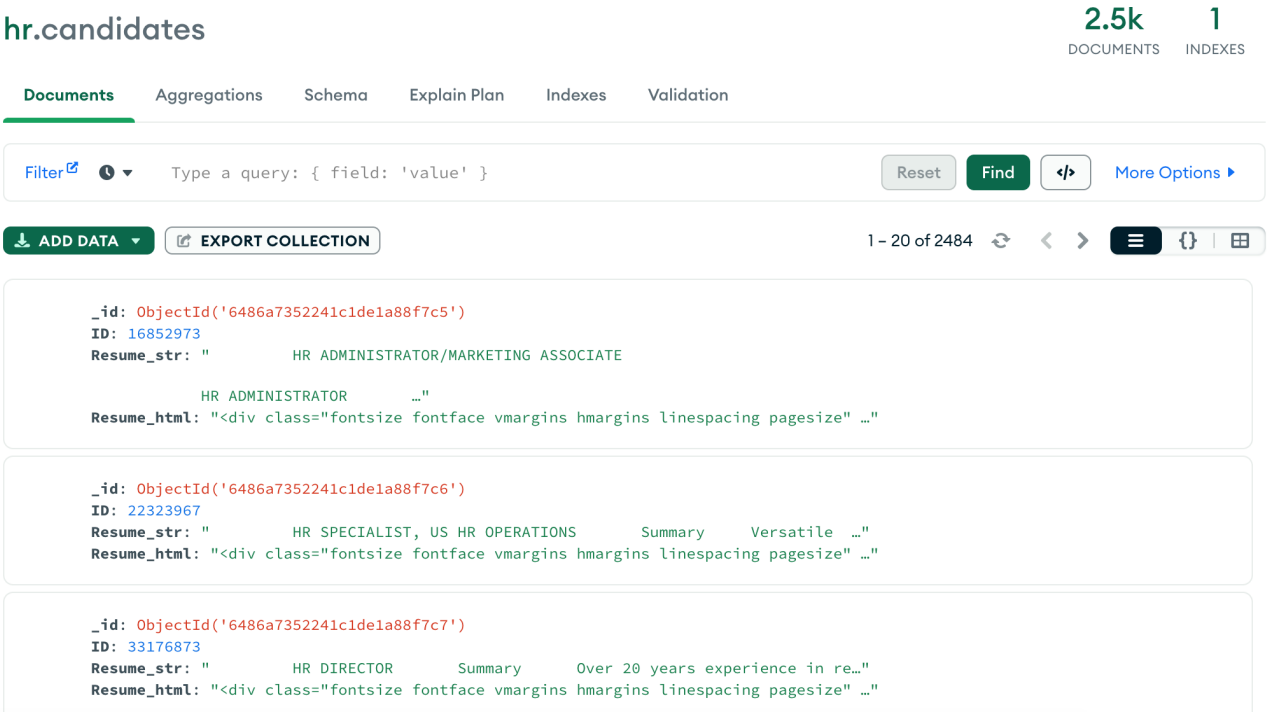

Рисунок 3.12. База Данных программы

Еще одним приятным следствием отсутствия схемы является эффективность работы с разреженными данными. В случае, когда поле date_published присутствует в одном документе, а в другом — нет, никакого пустого поля date_published не создано [20]. Семейство таблиц-баз данных NoSQL использует знакомые концепции колонок и таблиц, что делает их менее очевидным примером. Тем не менее, поскольку нет схемы, колонки не объявляются декларативно и могут быть добавлены или изменены во время пользовательской сессии работы с базой. Это позволяет в частности реализовать списки с динамическими колонками. Неструктурированная схема имеет несколько недостатков. К ним относятся накладные расходы в коде приложения при смене модели данных, а также отсутствие различных ограничений со стороны базы. Кроме того, отсутствие словарей на стороне базы затрудняет понимание и контроль структуры данных, когда база используется в нескольких проектах одновременно. Но гибкость все-таки является преимуществом в современном мире, который быстро меняется. В качестве примера можно привести Твиттер, который лет пять назад хранил вместе с твиттом лишь небольшое количество дополнительной информации, состоящей из времени, управления Twitter и нескольких килобайт метаданных. Тем не менее, в настоящее время в базе хранятся дополнительные несколько килобайт метаданных, помимо самого сообщения [20].

Ответы на запросы. Сопоставление (==), сравнение () и регулярные выражения — это некоторые из стандартных типов запросов, которые могут быть использованы.

Размещение данных: вы можете хранить любые структурированные, частично структурированные или даже полиморфные данные.

Масштабируемость: добавление дополнительных машин в кластер серверов позволяет обрабатывать больше запросов.

Гибкость и маневренность: позволяет быстро создавать приложения.

Структура документа. Вы можете объединить все данные о модели данных в один документ.

Программируемая схема Вы можете изменять схему базы данных «на лету», что сокращает время, необходимое для добавления новых функций или устранения существующих проблем.

Функции реляционных баз данных: вы можете выполнять действия, которые обычно выполняются реляционными базами данных, такие как индексирование.

Когда дело доходит до операционной стороны, MongoDB предлагает ряд инструментов и возможностей, которые недоступны другим системам баз данных:

Масштабируемость: MongoDB можно масштабировать до любого размера, который вам нужно, независимо от того, нужен ли вам автономный сервер или целые кластеры независимых серверов.

Поддержка балансировки нагрузки: MongoDB автоматически перемещает данные между отдельными частями.

Поддержка автоматического аварийного переключения: новый основной сервер автоматически запускается и работает в случае выхода из строя вашего старого основного сервера.

Инструменты управления: облачная служба управления MongoDB (MMS) позволяет вам отслеживать свои машины.

Способность использовать память. В большинстве случаев MongoDB работает лучше, чем реляционные базы данных, потому что ее файлы отображаются в память.

3.5 Серверная часть приложение

Django - это мощный фреймворк для разработки веб-приложений, написанный на языке Python. Он предоставляет разработчикам удобные инструменты и структуру для быстрой и эффективной разработки веб-приложений. Вот подробное описание Django:

1) Архитектура MTV:

Django использует архитектуру MTV (Model-Template-View), которая разделяет логику приложения на три компонента:

- Модели (Models): Определяют структуру данных и взаимодействие с базой данных.
- Шаблоны (Templates): Определяют внешний вид страницы, используя шаблонный язык Django.
- Представления (Views): Обрабатывают запросы, взаимодействуют с моделями и передают данные в шаблоны.

2) ORM (Object-Relational Mapping):

Django предоставляет ORM, который позволяет работать с базами данных с использованием объектно-ориентированного подхода. Он позволяет определять модели, которые отображаются на таблицы в базе данных, и выполнять операции CRUD (Create, Read, Update, Delete) с помощью Python-кода.

3) Автоматическое администрирование:

Django предлагает встроенное административное приложение, которое автоматически создает интерфейс администратора для управления данными моделей. Это облегчает добавление, изменение и удаление данных в базе данных без необходимости написания пользовательского интерфейса.

4) URL-маршрутизация:

Django использует URL-маршрутизацию для связывания URL-адресов с представлениями. Вы можете определить шаблоны URL-адресов и указать, какие представления обрабатывают соответствующие запросы. Это позволяет легко создавать иерархические структуры URL-адресов и управлять навигацией в приложении.

5) Шаблонный язык:

Django предлагает мощный шаблонный язык для создания динамических HTML-страниц. Он позволяет вставлять переменные, выполнить циклы и условные операторы, обрабатывать формы и многое другое. Шаблонный язык Django помогает разработчикам создавать гибкие и переиспользуемые шаблоны.

6) Обработка форм:

Django предоставляет инструменты для создания и обработки веб-форм. Он автоматически обрабатывает валидацию данных, предоставляет защиту от атак CSRF (межсайтовой подделки запроса) и упрощает взаимодействие с базой данных при сохранении данных формы.

7) Международная поддержка:

Django имеет встроенную поддержку для локализации и интернационализации. Вы можете легко создавать многоязычные приложения, переводить текст и форматировать даты и числа с учетом национальных стандартов.

8) Безопасность:

Django обеспечивает набор инструментов и механизмов для обеспечения безопасности веб-приложений. Это включает защиту от уязвимостей, таких как инъекции SQL и XSS (межсайтовые сценарии).

9) Расширяемость:

Django предлагает различные пакеты и расширения, которые позволяют расширять его функциональность. Вы можете использовать сторонние пакеты для добавления новых возможностей, таких как авторизация через социальные сети, работа с геоданными и многое другое.

Django - это мощный фреймворк, который облегчает разработку веб-приложений с помощью простоты использования, масштабируемости и обширной функциональности. Он предоставляет широкий набор инструментов для работы с базами данных, обработки форм, шаблонов и многое другое, что делает его одним из наиболее популярных фреймворков для разработки веб-приложений на языке Python.

3.6 Клиентская часть приложение

Angular - это популярный фреймворк для разработки веб-приложений, созданный компанией Google. Он предоставляет мощные инструменты и структуру для создания масштабируемых и динамических приложений на основе JavaScript. Вот подробное описание Angular:

1) Компонентная архитектура:

Angular основан на компонентной архитектуре, где приложение разбивается на небольшие и независимые компоненты. Каждый компонент объединяет HTML-шаблон, CSS-стили и JavaScript-код вместе, что делает их легко управляемыми и переиспользуемыми. Компоненты взаимодействуют друг с другом, образуя древовидную структуру.

2) TypeScript:

Angular использует TypeScript - сильно типизированный язык программирования, основанный на JavaScript. TypeScript предоставляет возможности статической типизации, интеллектуальную подсказку кода и другие возможности, которые помогают улучшить производительность, надежность и поддерживаемость кода.

3) Управление состоянием:

Angular предоставляет механизмы для управления состоянием приложения. Один из таких механизмов - это сервисы и инжекторы зависимостей. Сервисы предоставляют функциональность, которая может быть разделена и использована в разных компонентах, а инжекторы зависимостей обеспечивают внедрение зависимостей в компоненты, обеспечивая легкую тестируемость и модульность.

4) Роутинг:

Angular имеет встроенный механизм маршрутизации, который позволяет управлять навигацией между различными компонентами на основе URL-адресов. Вы можете определить маршруты и настроить соответствующие компоненты для отображения при определенных URL-адресах. Маршрутизация

в Angular позволяет создавать многостраничные приложения с глубокой навигацией.

5) Двустороннее связывание данных:

Angular предлагает двустороннее связывание данных, что означает, что изменение данных в модели автоматически обновляет представление и наоборот. Это упрощает разработку интерактивных пользовательских интерфейсов и уменьшает необходимость вручную обновлять представление при изменении данных.

6) Расширяемость:

Angular предоставляет инструменты и фреймворк для тестирования приложений. Вы можете написать модульные и интеграционные тесты для проверки функциональности компонентов, сервисов, маршрутизации и других частей приложения. Это помогает обеспечить надежность и качество вашего приложения.

7) Тестирование:

Angular можно расширять с помощью сторонних библиотек, пакетов и модулей. Он имеет огромное сообщество разработчиков и экосистему, где можно найти множество готовых решений и компонентов для ускорения разработки.

8) Поддержка Material Design:

Angular имеет встроенную поддержку Material Design - дизайн-языка, разработанного Google. Вы можете использовать предварительно созданные компоненты и стили Material Design для создания современных и красивых пользовательских интерфейсов.

9) Документация и сообщество:

Angular имеет обширную документацию и активное сообщество разработчиков. Вы можете найти официальную документацию Angular, руководства, примеры кода, блоги и форумы для получения помощи и поддержки при работе с фреймворком.

Angular - это мощный фреймворк для разработки веб-приложений, который предоставляет разработчикам широкий набор инструментов и функциональности. Он обладает компонентной архитектурой, обеспечивает управление состоянием, имеет встроенную маршрутизацию, поддержку Material Design и многое другое. Angular помогает создавать высокопроизводительные, масштабируемые и переиспользуемые приложения на основе JavaScript.

3.7. Выводы по третьему разделу

В этом разделе, проведено изучение и обработка датасета, использование алгоритмов классификации и нахождение схожести между векторами описание работы и резюме. Также, изучены клиентская и программная часть приложение.

4 Результаты исследование

В данной диссертационной работе было использована несколько алгоритмов, таких, как XGBoost, SVM, KNN, MLP, RandomForest.
В ходе исследования, мы можем заключить в наших исследованиях, XGBoost алгоритм показал себя лучше всех с результатом в 84% точности. В таблице 1 приведена результаты исследование методов классификации.

Таблица 4.1

Результаты исследование

Имя	Score	Precision	Recall	F-score
XGBClassifier	84%	75%	69%	68%
SVM	63%	58%	68%	60%
KNeighborsClassifier	45%	48%	41%	42%
MLPClassifier	23%	22%	32%	29%
RandomForestClassifier	65%	62%	69%	61%

Recall, Precision и F-score — это оценочные показатели, обычно используемые в машинном обучении и поиске информации для измерения производительности модели или системы бинарной классификации. Они дают представление о различных аспектах предсказательной способности модели.

Recall измеряет долю фактических положительных примеров, которые модель правильно идентифицирует. Он количественно определяет способность модели находить все положительные экземпляры в наборе данных. Высокий отзыв указывает на то, что модель может эффективно идентифицировать положительные экземпляры, сводя к минимуму ложноотрицательные результаты.

$$Recall = \frac{TP}{TP+FN}, \quad (4.1)$$

Precision измеряет долю предсказанных положительных случаев, которые на самом деле являются положительными. Он количественно определяет точность положительных прогнозов, сделанных моделью. Высокая точность указывает на то, что модель имеет низкий уровень ложных срабатываний.

$$Precision = \frac{TP}{TP+FP}, \quad (4.2)$$

F-score (или оценка F1):

F-score является гармоническим средним значением точности и полноты. Он объединяет обе метрики в одно значение, которое уравновешивает компромисс между точностью и полнотой. F-score обеспечивает

исчерпывающую оценку производительности модели, особенно в сценариях, где важны как точность, так и полнота.

$$F_{score} = \frac{2*(Precision * Recall)}{Precision+Recall}, \quad (4.3)$$

F-score варьируется от 0 до 1, где 1 указывает на идеальную точность и полноту, а 0 — на худшую производительность. Он обычно используется, когда данные несбалансированы или когда точность и полнота имеют решающее значение.

Эти метрики широко используются в различных приложениях, таких как поиск информации, классификация текстов, анализ настроений и медицинская диагностика. Они помогают оценить эффективность и надежность прогнозов модели, позволяя принимать обоснованные решения о производительности и оптимизации модели.

В области машинного обучения и, в частности, проблем статистической классификации, матрица путаницы, также известная как матрица ошибок, представляет собой особый макет таблицы, который позволяет продемонстрировать производительность алгоритма, обычно контролируемого обучения; при неконтролируемом обучении это обычно называется матрицей согласования. В матрице каждая строка представляет экземпляры реального класса, а каждый столбец представляет экземпляры предсказанного класса, или наоборот — оба варианта встречаются в литературе, и система часто путает два класса. (рисунок 4.1).

	precision	recall	f1-score	support
ACCOUNTANT	1.00	0.97	0.98	29
ADVOCATE	0.96	0.87	0.91	30
AGRICULTURE	0.56	0.62	0.59	8
APPAREL	0.78	0.70	0.74	20
ARTS	0.50	0.56	0.53	18
AUTOMOBILE	1.00	0.17	0.29	6
AVIATION	0.83	0.90	0.86	21
BANKING	0.73	0.70	0.71	23
BPO	1.00	0.50	0.67	2
BUSINESS-DEVELOPMENT	0.96	0.96	0.96	27
CHEF	0.95	0.83	0.89	24
CONSTRUCTION	0.92	0.97	0.94	34
CONSULTANT	0.89	0.85	0.87	20
DESIGNER	0.95	1.00	0.97	19
DIGITAL-MEDIA	0.95	0.84	0.89	25
ENGINEERING	0.86	0.90	0.88	21
FINANCE	0.67	0.84	0.74	19
FITNESS	1.00	0.53	0.69	19
HEALTHCARE	0.76	0.80	0.78	20
HR	0.75	0.83	0.79	18
INFORMATION-TECHNOLOGY	0.84	1.00	0.91	26
PUBLIC-RELATIONS	0.87	0.76	0.81	17
SALES	0.84	0.90	0.87	29
TEACHER	0.69	0.91	0.78	22
accuracy			0.84	497
macro avg	0.84	0.79	0.79	497
weighted avg	0.85	0.84	0.84	497

Рисунок 4.1. Матрица путанницы

В результате исследование схожести описание работы и резюме, мы выбрали косинусное сходство, из-за разности размерности векторов. Также, из-за того, что некоторые навыки кандидатов могут иметь более значимый

результат при выборе кандидата, было решено изменить стандартную формулу нахождение косинусного сходство. В новый формулы:

$$New\ Cosine\ Similarity\ = \frac{A \cdot B}{||A|| \cdot ||B||} + important\_skills\_score\ /\ 100 \quad (19)$$

где, important skills score - это нахождение навыков в резюме, и их оценка по степени важности.(рисунке 4.2).

В конечным итоге наша программа в бэкенд части выглядит как показано на (рисунке 4.4).

```
def fetch_resume_data(request):
    db = get_mongodb_connection()
    # Replace with your collection name
    collection = db['candidates']

    # Fetch the resume data from MongoDB
    resume_data = collection.find()

    # Prepare the data for processing
    body_unicode = request.body.decode('utf-8')
    body_data = json.loads(body_unicode)

    job_description = body_data.get('job_description')
   # job_description = "JavaScript"
    # request.POST.get('job_description')
```

PROBLEMS 5 OUTPUT TERMINAL DEBUG CONSOLE

```
Starting development server at http://127.0.0.1:9000/
Quit the server with CONTROL-C.

[13/Jun/2023 21:04:25] "POST /playground/api/resumes/ HTTP/1.1" 200 72064
[13/Jun/2023 21:07:14] "POST /playground/api/resumes/ HTTP/1.1" 200 79800
[13/Jun/2023 21:07:35,807] - Broken pipe from ('127.0.0.1', 58408)
[13/Jun/2023 21:08:19] "POST /playground/api/resumes/ HTTP/1.1" 200 79507
```

Рисунок 4.4. Серверная часть приложение

Когда как Клиентская часть приложение показана на (рисунке 4.5).

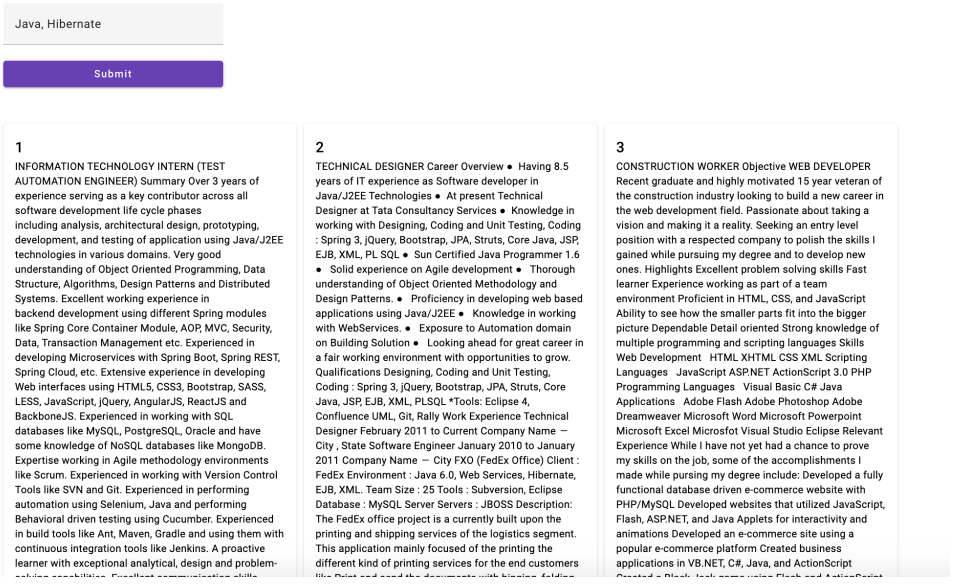

Рисунок 4.5. Клиентская часть приложение

4.1 Выводы по четвертому разделу

В этом разделы, было проведено сравнительный анализ алгоритмов классификации и показана решение по нахождению лучших кандидатов.

ЗАКЛЮЧЕНИЕ

В заключении этой диссертационной работы, были рассмотрены алгоритмы классификации, методы nlp, методы нахождение схожести документов. Основные научные результаты диссертации, практические выводы и рекомендации, полученные при выполнении исследований, заключаются в следующем:

1. Произведен анализ работ, связанные с скринингом текстом и резюме. Были выявлены стороны требующие улучшение из существующих решении.
2. Исследованы датасеты подходящие под требование, было выбрано подходящий датасет.
3. Были использованы разные методы nlp, в которым мы смогли повысить результаты оценки модели благодаря использованию сглаженной версии векторизации вместо стандартной.
4. Были рассмотрены разные модели классификации и произведено их сравнительный анализ, в которым XGBoostClassifier показал, лучшие результаты.
5. Был выбран и модифицирован алгоритм косинусного сходство для нахождение лучших кандидатов подходящие под тербование.
6. Было разработана программа скрининга резюме с помощью фреймворков Django и Angular.

После исследование, мы обнаружили что классификатор имеет 83.95% точности на тестовых данных, что является приемлемым результатом.

Огромное количество заявок, поступающих в организацию на каждую вакансию. Поиск соответствующего заявления кандидата из пула резюме в настоящее время является утомительной задачей для любой организации. Процесс классификации резюме кандидата является ручным, отнимает много времени и является пустой тратой ресурсов. Чтобы преодолеть эту проблему, мы предложили модель, основанную на автоматизированном машинном обучении, которая рекомендует подходящее резюме кандидата для отдела кадров на основе данного описания работы. Предлагаемая модель работала в два этапа: во-первых, классифицировать резюме по разным категориям. во-вторых, рекомендует резюме на основе индекса сходства с заданным описанием работы. Предлагаемый подход эффективно фиксирует информацию о резюме, их семантику и дает точность 83,75% с помощью XGBoostClassifier. Если отрасль предоставляет большое количество резюме, то можно разработать отраслевую модель с использованием предложенного подхода. Привлечение экспертов в предметной области, таких как HR-специалист, поможет построить более точную модель, а обратная связь от HR-специалиста поможет итеративно улучшить модель.

Рекруитинговая система было создана и протестирована в ходе работы этой магистерской работы.

Диссертационная исследовательская работа опубликована в научном журнале «Студенческий вестник» №20(259) под названием «Разработка программы скрининга резюме с использованием методов машинного обучение».

Все поставленные задачи в ходе этой диссертации были выполнены.

Автор выражает благодарность Сатыбалдиевой Рысхан, как моему наставнику.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Сони Дж., Ансари У., Шарма Д. и Сони С. 2011 г. Прогностический анализ данных для медицинской диагностики: обзор прогнозирования сердечных заболеваний. Международный журнал компьютерных приложений, 17 (8), 43–8.

2 Дангаре К.С. и Апте С.С. 2012 г. Улучшенное исследование системы прогнозирования сердечных заболеваний с использованием методов классификации интеллектуального анализа данных. Международный журнал компьютерных приложений, 47(10), 44-8.

3 Башир С., Камар У. и Джавед М.Ю. 2014 г. Система поддержки принятия решений на основе ансамбля для интеллектуальной диагностики заболеваний сердца. В Международной конференции по информационному обществу. 2014 г. (стр. 259-64). IEEE.

4 Джи Ш., Чан И, О Ди Джей, О Би Х, Ли Ш, Пак С В и Юн И Д. 2014 г. Модель прогнозирования ишемической болезни сердца: Корейско исследование сердца. BMJ открытый, 4(5), e005025.

5 Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, Rocky Bhatia, A Machine Learning approach for automation of Resume Recommendation system, Procedia Comp. Sci. 167. 2020 г. 2318–2327.

6 Jyothis Joseph, Jaimy Sunny, R Raveena, BlessyElzaByju, KC Laya, Resume Anal- yser: Automated Resume Ranking Software, Int. J. Res. Appl. Sci. Eng. Tech. (IJRASET) 8 (7) 896–899.

7 Aseel B. Kmail, Mohammed Maree, Mohammed Belkhatir, Saadat M. Alhashmi, An automatic online recruitment system based on exploiting multiple semantic re- sources and concept-relatedness measures, in: 2015 IEEE 27th International Confer- ence on Tools with Artificial Intelligence (ICTAI), IEEE, 2015, pp. 620–627.

8 Ramjeet Singh Yadav, A.K. Soni, Saurabh Pal, A study of academic performance eval- uation using Fuzzy Logic techniques, in: 2014 International Conference on Comput- ing for Sustainable Global Development (INDIACom), IEEE, 2014, pp. 48–53.

9 Evanthia Faliagka, Kostas Ramantas, Athanasios Tsakalidis, Giannis Tzimas, Appli- cation of machine learning algorithms to an online recruitment system, in: Proc. International Conference on Internet and Web Applications and Services, 2012, pp. 215–220.

10 Siham Jabri, Azzeddine Dahbi, Taoufiq Gadi, Abdelhak Bassir, Ranking of text doc- uments using TF-IDF weighting and association rules mining, in: 2018 4th interna- tional conference on optimization and applications (ICOA), IEEE, 2018, pp. 1–6.

11 Evanthia Faliagka, Kostas Ramantas, Athanasios Tsakalidis, Giannis Tzimas, Appli- cation of machine learning algorithms to an online recruitment

system, in: Proc. International Conference on Internet and Web Applications and Services, 2012, pp. 215–220.

12 Xingsheng Guo, Houssem Jerbi, Michael P. O'Mahony, An analysis framework for content-based job recommendation, 22nd International Conference on Case-Based Reasoning (ICCBR), 2014 29 September-01 October.

13 Pradeep Kumar Roy, Jyoti Prakash Singh, Amitava Nag, Finding active expert users for question routing in community question answering sites, in: International Confer- ence on Machine Learning and Data Mining in Pattern Recognition, Cham, Springer, 2018, pp. 440–451.

14 Kangning Wei, Jinghua Huang, Shaohong Fu, A survey of e-commerce recommender systems, in: 2007 international conference on service systems and service manage- ment, IEEE, 2007, pp. 1–5.

15 Shaha T. Al-Otaibi, Mourad Ykhlef, A survey of job recommender systems, Int. J. Phys. Sci. 7 (29). 2012, 5127–5142.

16 Jochen Malinowski, Tobias Keim, Oliver Wendt, Tim Weitzel, Matching people and jobs: A bilateral recommendation approach, in: Proceedings of the 39th An- nual Hawaii International Conference on System Sciences (HICSS'06), 6, IEEE, 2006 137c-137c.

17 Херон, М. Смерти: основные причины в 2017 г. . Национальные статистические отчеты о естественном движении населения;68(6). По состоянию на 19 ноября 2019 г.

18 Adem Golec, Esra Kahya, A fuzzy model for competency-based employee evaluation and selection, Comput. Ind. Eng. 52 (1), 2013 143–161.

19 Grigori Sidorov, Alexander Gelbukh, Helena Gomez-Adorno, David Pinto, SoftSimi- larity and Soft Cosine Measure: Similarity of Features in Vector Space Model, Com- putacion y Sistemas 18 (3) 2014, 491–504.

20 Paul Resnick, Hal R. Varian, Recommender systems, Commun. ACM 40 (3), 1997, 56–58. 375

```
import pandas as pd
import numpy as np
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)

targetCounts = resumeDataSet['Category'].value_counts().reset_index()['Category']
targetLabels  = resumeDataSet['Category'].value_counts().reset_index()['index']

plt.figure(1, figsize=(25,25))
the_grid = GridSpec(2, 2)
plt.subplot(the_grid[0, 1], aspect=1, title='CATEGORY DISTRIBUTION')
source_pie = plt.pie(targetCounts, labels=targetLabels, autopct='%1.1f%%', shadow=True)


def cleanResume(resumeText):
    resumeText = re.sub('httpS+s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@S+', '  ', resumeText)  # remove mentions
     resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[]^_`{|}~"""), '
', resumeText)
    resumeText = re.sub(r'[^x00-x7f]',r' ', resumeText)
    resumeText = re.sub('s+', ' ', resumeText)  # remove extra whitespace
    return resumeText
resumeDataSet['cleaned_resume'] = resumeDataSet['Resume_str'].apply(lambda x:
cleanResume(x))
resumeDataSet.head(5)

requiredText = resumeDataSet['cleaned_resume'].values
requiredTarget = resumeDataSet['Category'].values
word_vectorizer = TfidfVectorizer(
    norm='l2',
    smooth_idf=True,
    sublinear_tf=True,
```

```
    stop_words='english',
    max_features=1500)
word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)
WordFeatures

clf1 = KNeighborsClassifier(n_neighbors=5, weights='distance')
clf1.fit(X_train, y_train)
predictions = clf1.predict(X_test)
print('Accuracy of KNeighbors Classifier on training set: {:.2f}'.format(clf1.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set: {:.2f}'.format(clf1.score(X_test, y_test)))
from sklearn.linear_model import LogisticRegression
clf2 = LogisticRegression(C=1e1, solver='lbfgs',max_iter=10000)
clf2.fit(X_train, y_train)
predictions = clf2.predict(X_test)
print('Accuracy of LogisticRegression Classifier on training set: {:.2f}'.format(clf2.score(X_train, y_train)))
print('Accuracy of LogisticRegression Classifier on test set: {:.2f}'.format(clf2.score(X_test, y_test)))

from sklearn.neural_network import MLPClassifier
clf3 = MLPClassifier(hidden_layer_sizes=[5, 10], alpha=1, random_state=0, solver='lbfgs', max_iter=10000)
clf3.fit(X_train, y_train)
predictions = clf3.predict(X_test)
print('Accuracy of MLPClassifier Classifier on training set: {:.2f}'.format(clf3.score(X_train, y_train)))
print('Accuracy of MLPClassifier Classifier on test set: {:.2f}'.format(clf3.score(X_test, y_test)))

from sklearn.svm import SVC
clf5 = SVC(kernel = 'linear', C=500)
clf5.fit(X_train, y_train)
predictions = clf5.predict(X_test)
print('Accuracy of SVC Classifier on training set: {:.2f}'.format(clf5.score(X_train, y_train)))
print('Accuracy of SVC Classifier on test set: {:.2f}'.format(clf5.score(X_test, y_test)))


xgb_classifier = XGBClassifier(learning_rate=0.1,
                    n_estimators=100,
                    max_depth=2,
```

```
                    min_child_weight=2,
                    gamma=0.2,
                    subsample=0.8,
                    colsample_bytree=0.9,
                    reg_alpha=0.01,
                    reg_lambda=0.01,
                    objective='multi:softmax',
                    num_class=len(set(requiredTarget)))

xgb_classifier.fit(X_train, y_train)

y_pred = xgb_classifier.predict(X_test)

from sklearn.feature_extraction.text import TfidfVectorizer

X_train,X_test,y_train,y_test                                                    =
train_test_split(requiredText,requiredTarget,random_state=42, test_size=0.2)

tfidf_vectorizer = TfidfVectorizer(norm='l2', smooth_idf=True)

X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

X_test_tfidf = tfidf_vectorizer.transform(X_test)

xgb_classifier = XGBClassifier(learning_rate=0.1,
                    n_estimators=100,
                    max_depth=2,
                    min_child_weight=2,
                    gamma=0.2,
                    subsample=0.8,
                    colsample_bytree=0.9,
                    reg_alpha=0.01,
                    reg_lambda=0.01,
                    objective='multi:softmax',
                    num_class=len(set(requiredTarget)))


xgb_classifier.fit(X_train_tfidf, y_train)

y_pred = xgb_classifier.predict(X_test_tfidf)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

job_description={
   'INFORMATION-TECHNOLOGY': 'between 1 to 3 years of experience. Has skills in Java, OOPs,Singletone, Design Patterns, Interface, Serialization,Maven, Gradle, Hibernate. Finished bachelor'
}

results = {}
category = 'INFORMATION-TECHNOLOGY'  # Specify the desired category
stoplist = stopwords.words('english')
resumes = resumeDataSet[resumeDataSet['Category'] == category]['cleaned_resume'].values

vectorizer = TfidfVectorizer(stop_words=stoplist)

resume_tfidf = vectorizer.fit_transform(resumes)

for jd_category in job_description.keys():
   jd_string = job_description[jd_category]

   jd_tfidf = vectorizer.transform([jd_string])

   sims = cosine_similarity(jd_tfidf, resume_tfidf)

  top_scores = sorted(enumerate(sims[0]), key=lambda x: x[1], reverse=True)[:10]

   for i in range(len(top_scores)):
      score = top_scores[i][1]
      resume_index = top_scores[i][0]
      resume = resumes[resume_index]
      if category not in results:
         results[category] = []
      results[category].append((resume, score))

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
X_train,X_test,y_train,y_test =
train_test_split(requiredText,requiredTarget,random_state=42, test_size=0.2)

# Initialize the TF-IDF vectorizer with term frequency normalization and smoothed
IDF
tfidf_vectorizer = TfidfVectorizer(norm='l2', smooth_idf=True)

# Fit and transform the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initialize the XGBoost classifier
xgb_classifier = XGBClassifier(learning_rate=0.1,
                        n_estimators=100,
                        max_depth=2,
                        min_child_weight=1,
                        gamma=0,
                        subsample=0.8,
                        colsample_bytree=0.8,
                        reg_alpha=0.01,
                        reg_lambda=0.01,
                        objective='multi:softmax',
                        num_class=len(set(requiredTarget)))

# Train the classifier
xgb_classifier.fit(X_train_tfidf, y_train)


# Make predictions on the test set
y_pred = xgb_classifier.predict(X_test_tfidf)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
from django.http import JsonResponse
from .mongodb import get_mongodb_connection
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import json


def fetch_resume_data(request):
    db = get_mongodb_connection()
    # Replace with your collection name
    collection = db['candidates']

    # Fetch the resume data from MongoDB
    resume_data = collection.find()

    # Prepare the data for processing
    body_unicode = request.body.decode('utf-8')
    body_data = json.loads(body_unicode)

    job_description = body_data.get('job_description')
   # job_description = "JavaScript"
    # request.POST.get('job_description')

    important_skills = [
        {
            'skill': 'Beginner',
            'weight': 1.1
        },
        {
            'skill': 'Intermediate',
            'weight': 1.3
        },
        {
            'skill': 'Advanced',
            'weight': 1.7
        },
    ]

    results = []
    stoplist = stopwords.words('english')
```

```
    resumes = [resume['Resume_str'] for resume in resume_data]

    vectorizer = TfidfVectorizer(stop_words=stoplist)
    resume_tfidf = vectorizer.fit_transform(resumes)

    # Calculate the score_skills for each resume
    score_skills = []
    for resume in resumes:
        score = 0
        for skill in important_skills:
            if skill['skill'].lower() in resume.lower():
                score += skill['weight']
        score_skills.append(score)

    jd_tfidf = vectorizer.transform([job_description])
    sims = cosine_similarity(jd_tfidf, resume_tfidf)
    similarity_scores = sims[0] + np.array(score_skills) / 100

    top_scores = sorted(enumerate(similarity_scores),
                        key=lambda x: x[1], reverse=True)[:10]

    for i in range(len(top_scores)):
        score = top_scores[i][1]
        resume_index = top_scores[i][0]
        resume = resumes[resume_index]
        results.append((resume, score))

    return JsonResponse({'results': results})

from pymongo import MongoClient


def get_mongodb_connection():
    # Replace with your MongoDB connection URL
    client = MongoClient(
        "mongodb+srv://daniyarturak:qwerty123@cluster0.mkvp2ex.mongodb.net")
    db = client['hr']  # Replace with your database name
    return db

from django.urls import path
from .views import fetch_resume_data

urlpatterns = [
    path('api/resumes/', fetch_resume_data, name='fetch_resume_data'),
]
```

```
"""
Django settings for storefront project.

Generated by 'django-admin startproject' using Django 4.2.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY                                                              =
"django-insecure--n-s%e%f45ij9rl9psi3li-6l-&mi63f3*f6$*@)c71-#-jynz"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    'corsheaders',
]

MIDDLEWARE = [
```

```
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    # "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "corsheaders.middleware.CorsMiddleware",
    'django.middleware.common.CommonMiddleware',
]

CORS_ORIGIN_WHITELIST = [
    'http://localhost:4200',
]

ROOT_URLCONF = "storefront.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "storefront.wsgi.application"


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}
```

```
# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
                                                                    "NAME":
"django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
     {"NAME":  "django.contrib.auth.password_validation.MinimumLengthValidator",
},
                                                                  {"NAME":
"django.contrib.auth.password_validation.CommonPasswordValidator", },
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
},
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = "static/"

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
```

```
<div class="filter">
  <mat-form-field class="example-full-width">
    <input
      class="input"
      matInput
      [(ngModel)]="jobDescription"
      placeholder="Enter job description"
    />
  </mat-form-field>
  <button mat-raised-button color="primary" (click)="onSubmit()">Submit</button>
</div>

<div class="list-items">
  <mat-card class="example-card" *ngFor="let resume of resumes; let i = index">
    <mat-card-header>
      <mat-card-title>{{ i + 1 }}</mat-card-title>
    </mat-card-header>
    <mat-card-content>
      <p>{{ resume[0] }}</p>
    </mat-card-content>
    <mat-card-actions>
      <button mat-button>LIKE</button>
      <button mat-button>SHARE</button>
    </mat-card-actions>
  </mat-card>
</div>

import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CandidatesService } from './services/candidates.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
})
export class AppComponent {
  jobDescription = '';
  resumes: any[] = [];

  constructor(private resumeService: CandidatesService) {}

  onSubmit() {
```

```
    console.log(this.jobDescription);
    this.resumeService.getResumes(this.jobDescription).subscribe((data) => {
      this.resumes = data.results;
    });
  }
}

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class CandidatesService {
    private apiUrl = 'http://127.0.0.1:9000/playground/api/resumes/'; // Replace with
your Django server URL

  constructor(private http: HttpClient) {}

  getResumes(data: any): Observable<any> {
    return this.http.post<any>(
      'http://127.0.0.1:9000/playground/api/resumes/',
      {
        job_description: data,
      }
    );
  }
}

.filter {
  max-width: 300px;
  display: flex;
  flex-direction: column;
  padding-left: 50px;
  padding-top: 50px;

  .input {
    height: 100px;
  }
}

.list-items {
  padding: 50px;
  display: flex;
```

```
  flex-flow: row wrap;
  row-gap: 30px;
  column-gap: 10px;

  .example-card {
    max-width: 400px;
  }
}
```