# АННОТАЦИЯ

Данная тема посвящена программным средствам решения прикладных задач в области искусственного интеллекта. В работе рассматриваются основные задачи, которые решаются с помощью общих алгоритмов и прикладных программных средств. Описывается категория программных средств, включающая методы и модели машинного обучения, нейронные сети, алгоритмы генетического программирования и другие технологии. В работе приводятся результаты исследований и практических применений различных программных средств, а также оценивается их эффективность и перспективы для развития технологий искусственного интеллекта.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 1](#_Toc169202909)

[ВВЕДЕНИЕ 3](#_Toc169202910)

[1 ПОСТАНОВКА ПРИКЛАДНОЙ ЗАДАЧИ 4](#_Toc169202911)

[2 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 8](#_Toc169202912)

[2.1 СТРУКТУРА BERT 11](#_Toc169202913)

[2.2 АНАЛОГИ BERT И ОБОСНОВАНИЕ ВЫБОРА ДАННОЙ МОДЕЛИ 14](#_Toc169202914)

[2.3 ВОЗМОЖНЫЕ ПОДХОДЫ К ПОСТРОЕНИЮ ПРЕДСКАЗАТЕЛЬНОЙ МОДЕЛИ 17](#_Toc169202915)

[2.4 ОБОСНОВАНИЕ ВЫБОРА МЕТОДА РЕШЕНИЯ ЗАДАЧИ 22](#_Toc169202916)

[2.5. ОБЪЯСНЕНИЕ ПРИНЦИПОВ РАБОТЫ ГРАДИЕНТНОГО БУСТИНГА 24](#_Toc169202917)

[2.6. МЕТРИКИ ДЛЯ ОЦЕНКИ МОДЕЛИ 26](#_Toc169202918)

[3 РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 29](#_Toc169202919)

[ЗАКЛЮЧЕНИЕ 34](#_Toc169202920)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc169202921)

[ПРИЛОЖЕНИЕ А 36](#_Toc169202922)

# ВВЕДЕНИЕ

Современные программные средства для решения прикладных задач искусственного интеллекта становятся все более востребованными и необходимыми в различных отраслях. Ныне мы можем наблюдать стремительное развитие технологий ИИ, что открывает перед компаниями и организациями новые горизонты для повышения эффективности и решения разнообразных задач. Алгоритмы и программы на базе искусственного интеллекта находят широкое применение в самых разных сферах — от медицинских исследований и диагностики до автоматизации производственных процессов и управления бизнес-процессами.

Одной из ключевых и актуальных задач в современном мире является эффективное сопоставление вакансий и резюме. Эта задача требует высокой точности и адаптивности в обработке текстовых данных, что становится возможным благодаря использованию передовых технологий ИИ. В данной выпускной квалификационной работе исследуется создание интеллектуальной системы сопоставления вакансий и резюме, основанной на адаптивном анализе текста.

Разработка и внедрение таких интеллектуальных систем способны значительно повысить эффективность процесса подбора персонала, делая его более точным, быстрым и адаптивным к изменениям на рынке труда. В рамках данной работы будет подробно рассмотрен процесс создания интеллектуальной системы, включая методы и технологии, используемые для анализа текста, а также будут представлены результаты проведенных экспериментов и их анализ.

Таким образом, данная работа направлена на изучение и разработку инновационных решений в области подбора персонала с использованием технологий искусственного интеллекта, что является важным шагом на пути к улучшению и оптимизации HR-процессов в различных организациях и компаниях.

# ПОСТАНОВКА ПРИКЛАДНОЙ ЗАДАЧИ

Анализ данных о вакансиях и резюме является важной задачей в сфере управления человеческими ресурсами. Он позволяет HR-специалистам и рекрутерам эффективно подбирать кандидатов на открытые позиции, анализировать рынок труда и улучшать процессы найма. В современных условиях, когда объемы данных стремительно растут, возникает необходимость автоматизации анализа и предсказания результатов найма с использованием методов машинного обучения и обработки естественного языка.

**Описание задачи**

Целью данной работы является создание модели, способной предсказывать успешность кандидатов на основе их резюме и характеристик вакансий. Для этого необходимо выполнить несколько ключевых этапов:

1. Преобразование текстовых данных в числовой формат для возможности их использования в алгоритмах машинного обучения.

2. Обучение модели на полученных данных для предсказания успешности кандидатов.

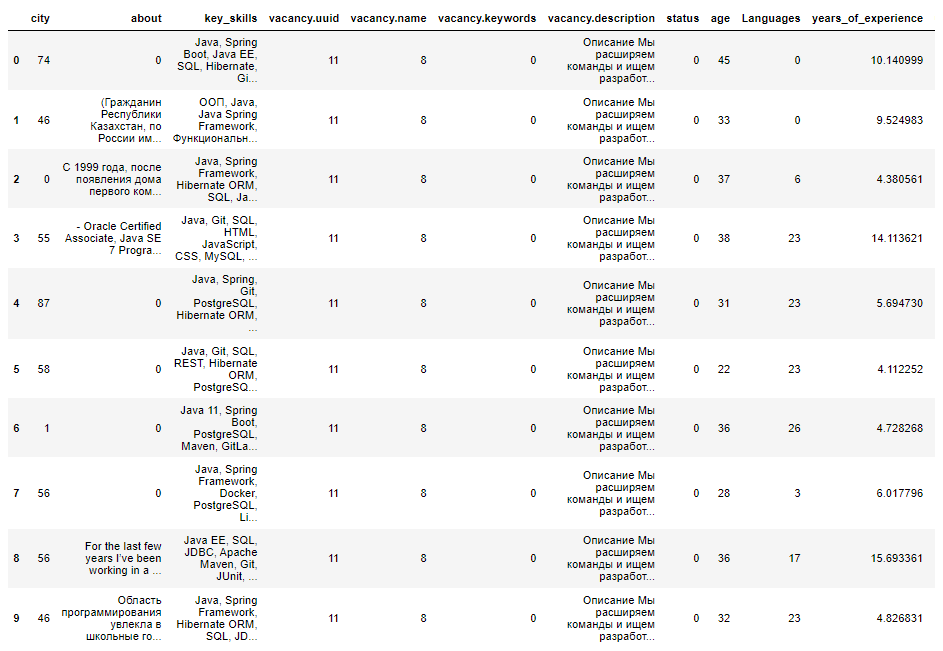
3. Оценка точности и эффективности разработанной модели.

**Предварительное описание данных**

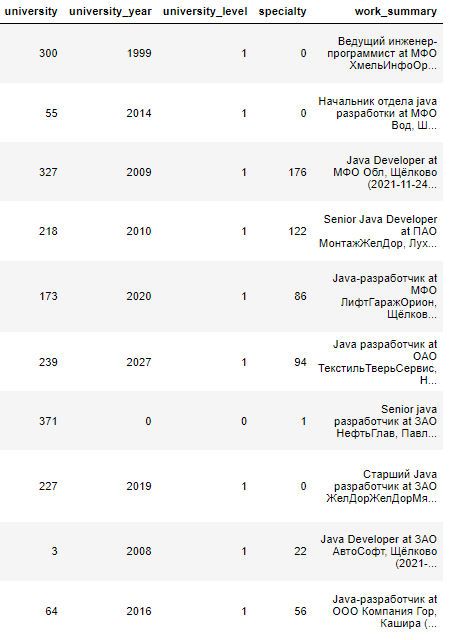
Перед тем, как перейти к более подробному описанию задачи, необходимо дать краткое описание данным, на основе которых и будет проводиться обучение модели. В рамках работы используется датасет, содержащий информацию о вакансиях и откликнувшихся на них кандидатах. Таблица состоит из 656 строк и 16 столбцов, каждый из которых представляет собой различные характеристики резюме и вакансий. Набор данных состоит из таких колонок, как:

1. city: Город, в котором находится соискатель или вакансия.
2. about: Дополнительная информация о соискателе.
3. key\_skills: Ключевые навыки, указанные в резюме.
4. vacancy.uuid: Уникальный идентификатор вакансии.
5. vacancy.name: Название вакансии.
6. vacancy.keywords: Ключевые слова, связанные с вакансией.
7. vacancy.description: Описание вакансии.
8. status: Статус резюме или вакансии.
9. age: Возраст соискателя.
10. Languages: Владение языками
11. years\_of\_experience: Количество лет опыта работы.
12. university: Университет, который окончил соискатель.
13. university\_year: Год окончания университета.
14. university\_level: Уровень образования (например, бакалавр, магистр).
15. specialty: Специальность соискателя.
16. work\_summary: Краткое описание опыта работы соискателя.

На следующих изображениях представлены фрагменты датасета, благодаря которым можно лучше понять контекст данной задачи:



**Рисунок 1 — Датасет**

****

**Рисунок 2 — Датасет**

Следующим важным шагом является преобразование данных. Для преобразования текстовых данных в числовой формат будет использована модель BERT (Bidirectional Encoder Representations from Transformers), которая позволяет получать контекстно-зависимые векторные представления слов.

BERT (Bidirectional Encoder Representations from Transformers) — это модель для обработки естественного языка, разработанная Google. Она использует двухсторонний подход к анализу текста, учитывая контекст слов как слева, так и справа. Модель предварительно обучена на большом корпусе текстов и может быть дообучена для конкретных задач. Благодаря этому BERT показывает высокую точность в задачах NLP, таких как классификация текста и распознавание именованных сущностей.

Основные этапы преобразования включают:

1. Токенизация текста: Разбиение текста на токены и (при необходимости) добавление специальных токенов.

2. Получение эмбеддингов: Отправление токенизированного текста в модель BERT для получения векторных представлений.

**Обучение модели**

После векторизации данных будет обучена модель градиентного бустинга для предсказания успешности кандидатов. Градиентный бустинг выбран благодаря своей высокой точности и способности обрабатывать большие объемы данных. Основные этапы обучения включают:

1. Разделение данных на обучающую и тестовую выборки.

2. Обучение модели на обучающих данных.

3. Оценка точности модели на тестовых данных.

**Ожидаемые результаты**

В результате выполнения данной работы будет создана модель, способная предсказывать успешность кандидатов на основе их резюме и характеристик вакансий. Это позволит улучшить процессы найма и сократить время на подбор подходящих кандидатов. Оценка точности модели покажет ее эффективность и возможность использования в реальных условиях.

# АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Для решения поставленной задачи предсказания успешности кандидатов с использованием BERT и градиентного бустинга необходимо следовать нескольким важным этапам:

1. Определение задачи и постановка целей:

* В первую очередь, необходимо четко определить задачу, которую требуется автоматизировать: предсказание успешности кандидатов на основе их резюме и характеристик вакансий.
* Поставить конкретные цели, такие как улучшение точности подбора персонала, снижение времени на анализ резюме, повышение качества найма.

2. Определение метода решения:

* На основе определенной задачи и целей выбирается подходящий метод решения. В данном случае это комбинация методов обработки естественного языка (NLP) и машинного обучения:
* Использование модели BERT для векторизации текстовых данных.
* Применение алгоритма градиентного бустинга для построения предсказательной модели.

3. Сбор и подготовка данных:

Сбор данных, которые будут использоваться для обучения модели. В данном случае это данные о вакансиях и резюме кандидатов, содержащие информацию о навыках, опыте работы, образовании и т.д.

Учитывая тот факт, что набор данных уже был обработан ранее, этот аспект не нуждается во внимании. Достаточно будет лишь извлечь данные из набора и приступить к работе с ними, а затем провести их нормализацию (приведение данных к единому масштабу).

4. Векторизация текстовых данных:

* Преобразование текстовых столбцов (таких как `vacancy.description`, `about`, `key\_skills`, `work\_summary`) в числовой формат с использованием модели BERT:
* Токенизация текста: разбиение текста на токены и добавление специальных токенов.
* Обработка токенизированного текста через модель BERT для получения векторных представлений.
* Сохранение полученных эмбеддингов для дальнейшего использования в моделях машинного обучения.

5. Обучение модели:

* После предобработки и векторизации данных следует этап обучения модели. На этом этапе выбранная модель градиентного бустинга обучается на подготовленных данных:
* Разделение данных на обучающую и тестовую выборки.
* Обучение модели на обучающих данных с использованием алгоритма градиентного бустинга.
* Настройка гиперпараметров модели для достижения наилучших результатов.

6. Тестирование и оценка результатов:

* После обучения модель должна быть протестирована на новых данных, которые ранее не использовались для обучения. Это поможет оценить качество модели и ее способность к предсказанию решений в реальных условиях:
* Оценка точности модели на тестовых данных с использованием различных метрик (точность, полнота, F-мера и т.д.).
* Анализ результатов и выявление возможных областей для улучшения.

Примечание: следующий пункт не будет освещён в рамках данной работы, но важен для понимания полного цикла работы модели ИИ.

7. Развитие и поддержка модели:

Выпущенная модель требует постоянной поддержки и развития. Для этого необходимо:

* Постоянно собирать новые данные и периодически переобучать модель на этих данных, чтобы она могла адаптироваться к изменениям и сохранять высокую точность предсказаний.
* Дорабатывать алгоритмы и функционал модели для повышения ее эффективности и точности.
* Мониторинг производительности модели в реальных условиях и своевременное реагирование на изменения.

Этот алгоритм решения задачи позволяет эффективно использовать мощные инструменты обработки естественного языка и машинного обучения для автоматизации анализа и предсказания успешности кандидатов, что существенно улучшает процессы найма и управления человеческими ресурсами.

## Структура Bert

BERT (Bidirectional Encoder Representations from Transformers) — это модель глубокого обучения, предназначенная для обработки естественного языка. BERT строится на основе архитектуры Transformer, которая использует механизмы самовнимания для моделирования контекстуальных отношений между словами в тексте.

Энкодер Transformer в BERT: Центральным элементом BERT является энкодер Transformer, который содержит множество слоёв самовнимания. Каждый слой имеет две основные подструктуры: механизм многоголового самовнимания и позиционно полносвязный слой. В механизме самовнимания выходы с предыдущего слоя обрабатываются таким образом, что модель может сфокусироваться на различных словах в предложении для лучшего понимания контекста.

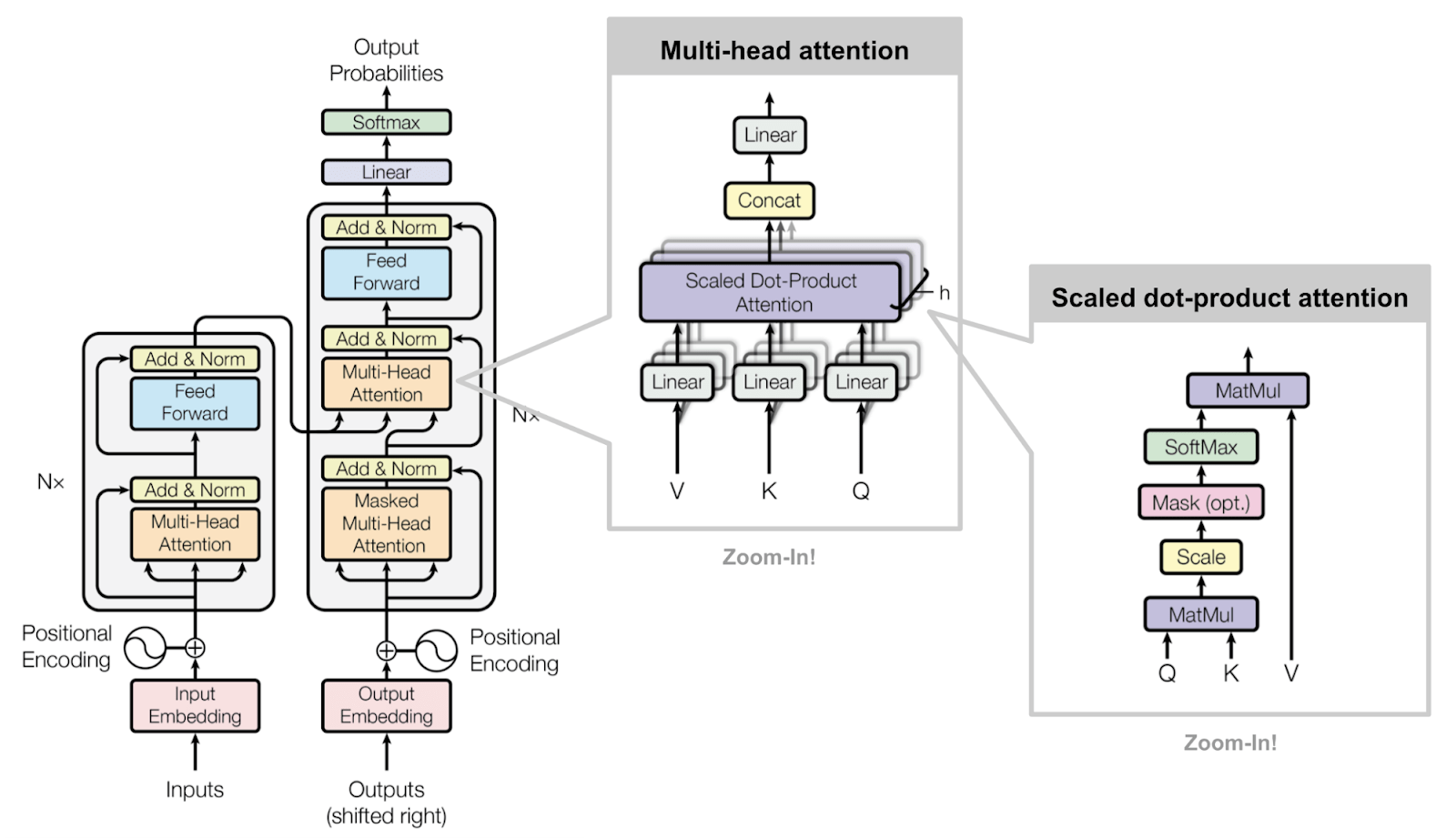
Механизм многоголового самовнимания: Этот механизм позволяет модели одновременно обрабатывать информацию из разных представлений на разных позициях входных данных. В каждой "голове" выполняются взвешенные операции внимания, которые интегрируют информацию по всему входному предложению, учитывая текущее слово.

Позиционно полносвязный слой: После механизма самовнимания следует слой, который состоит из двух линейных преобразований с активационной функцией между ними. Эти преобразования универсальны и одинаковы для каждой позиции в предложении, что позволяет модели обучаться на выявление сложных зависимостей в данных.

Бидирекциональная обработка: В отличие от предыдущих моделей, которые смотрели либо слева направо, либо справа налево, BERT обрабатывает текст в обоих направлениях одновременно. Это достигается за счёт специальной подготовки входных данных и использования маскированного языкового моделирования, что позволяет модели лучше понимать контекст и тонкости языка.

Проблемы с исчезающим и взрывающимся градиентом: Как и в случае с другими глубокими архитектурами, BERT использует нормализацию по слоям и оптимизированные функции активации для борьбы с исчезающими и взрывающимися градиентами, что позволяет модели обучаться эффективно и стабильно на больших объемах данных.

BERT стал основой для множества последующих моделей в области NLP и продолжает влиять на развитие технологий обработки естественного языка за счет своей способности эффективно моделировать и обрабатывать языковые данные на глубоком уровне.

На следующем рисунке изображена архитектура Bert

**Рисунок 3 — Архитектура BERT**

Выбор BERT для преобразования текстовых столбцов в векторный вид в задаче предсказания отбора резюме на вакансию является хорошим решением по ряду причин:

1. Понимание контекста: BERT способен эффективно анализировать и понимать контекст текста, что критически важно при работе с резюме. Резюме содержат множество специфических для контекста слов и фраз, значение которых может меняться в зависимости от соседних слов. BERT учитывает оба направления контекста (слева направо и справа налево), что позволяет ему точнее понимать значение каждого слова в предложении.

2. Превосходная обработка естественного языка: За счет обучения на больших корпусах данных и способности к обобщению, BERT может эффективно обрабатывать различные стили и форматы текста, что делает его идеальным для анализа текстов из резюме разных кандидатов.

3. Извлечение значимых признаков: BERT способен извлекать значимые признаки из текста, которые могут быть использованы для классификации или кластеризации резюме. Эти признаки включают не только ключевые навыки и опыт, но и более тонкие аспекты, такие как стиль коммуникации и способности к решению проблем.

4. Адаптивность и гибкость: BERT легко адаптируется к специфическим задачам и доменам, что делает его подходящим для использования в различных отраслях и профессиях. Его можно дообучить на конкретном наборе данных (например, текстах резюме для определенной индустрии), чтобы улучшить точность и релевантность выдаваемых результатов.

5. Улучшение качества подбора персонала: Использование BERT может значительно улучшить качество подбора персонала, обеспечивая более точное сопоставление кандидатов с требованиями вакансии. Это приводит к более эффективному отбору резюме, снижению времени на рассмотрение кандидатов и улучшению общей эффективности процесса найма.

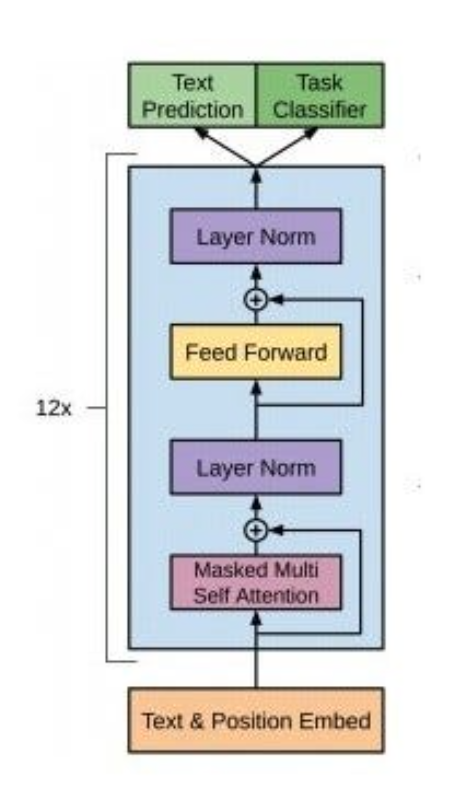
6. Сокращение человеческих предвзятостей: Автоматизация процесса отбора резюме с помощью BERT может помочь снизить человеческие предвзятости, так как модель обучается обрабатывать информацию объективно, исходя из представленных данных, а не субъективных мнений.

В итоге, использование BERT для анализа и векторизации текстов в резюме дает возможность более глубоко и точно оценить потенциал кандидатов, что делает процесс подбора персонала более эффективным и справедливым.

## Аналоги BERT и обоснование выбора данной модели

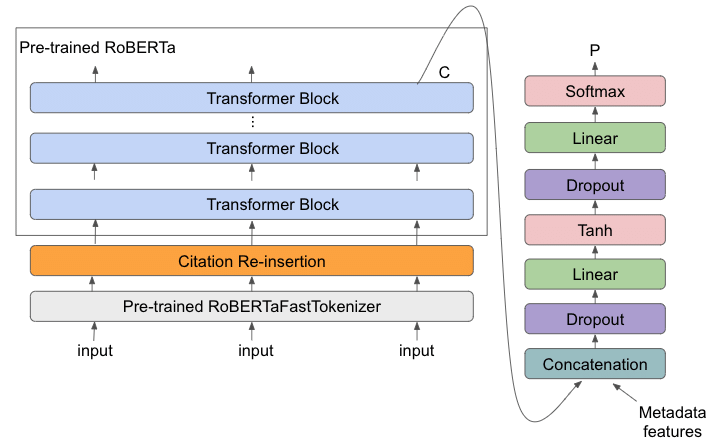
BERT, хотя и является одной из наиболее мощных моделей обработки естественного языка, не единственная в своем роде. Существуют другие аналоги, такие как GPT (Generative Pre-trained Transformer), RoBERTa (Robustly optimized BERT approach), и DistilBERT, каждый из которых имеет свои особенности и преимущества.

1. GPT — это модель, основанная также на трансформерах, но ориентированная на генерацию текста. В отличие от BERT, GPT обучается предсказывать следующее слово в предложении, что делает ее идеальной для задач, связанных с созданием текстов. Однако для задачи анализа текста, где важно учитывать контекст слов как слева, так и справа, BERT показывает себя более эффективно.



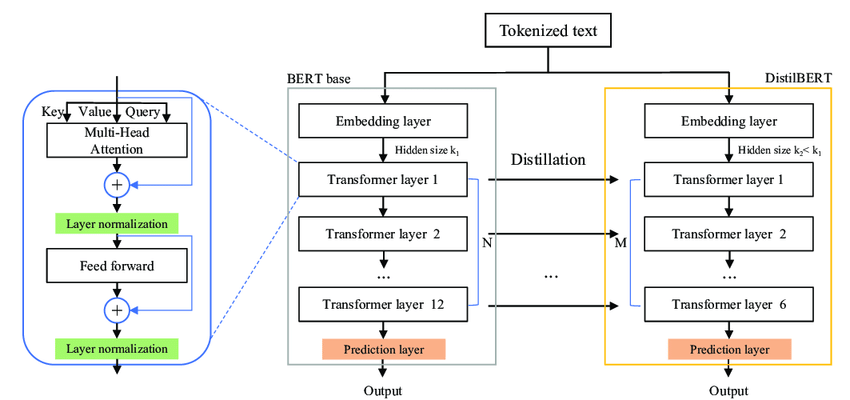
**Рисунок 4 — Архитектура GPT**

2. RoBERTa — это версия BERT, которая была оптимизирована с использованием более крупных датасетов и дольшего времени обучения. RoBERTa показывает улучшенные результаты на многих задачах NLP за счет более глубокой настройки параметров и исключения некоторых BERT’овских ограничений, таких как Next Sentence Prediction. Тем не менее, для задачи анализа текста, RoBERTa может быть излишне сложной и ресурсоемкой.



**Рисунок 5 — Архитектура ROBERTa**

3. DistilBERT — упрощенная и более легкая версия BERT, предназначенная для ситуаций, где ресурсы ограничены. DistilBERT сохраняет около 95% точности BERT, потребляя при этом меньше ресурсов. Однако в задачах, требующих максимальной точности и глубокого понимания контекста, полноценный BERT может быть предпочтительнее.



**Рисунок 6 — Архитектура DistillBERTa**

Обоснование выбора BERT:

1. Бидирекциональное понимание контекста

BERT уникален тем, что обрабатывает тексты, учитывая контекст с обеих сторон — как справа, так и слева от текущего слова. Это отличает его от традиционных однонаправленных моделей, таких как GPT, где контекст слова рассматривается лишь с одной стороны. В сфере HR это особенно важно, так как многие ключевые навыки и опыт работы могут быть правильно интерпретированы только с учетом всего контекста предложения.

2. Маскированное языковое моделирование

Техника маскированного языкового моделирования (MLM) в BERT позволяет модели лучше понимать связи между словами. BERT обучается на предсказании маскированных слов, что требует понимания их контекста, а не просто последовательности слов. Это помогает модели выявлять более глубокие смысловые связи в тексте, что критично для анализа профессиональных текстов, где одно и то же слово может иметь различные значения в зависимости от контекста.

3. Передача и применение знаний

BERT предварительно обучен на огромном корпусе текста из Интернета, что обеспечивает широкое понимание естественного языка. Это позволяет модели лучше адаптироваться к специфике профессиональной лексики, такой как терминология в различных отраслях. Важно, что модель может быть дообучена на специфических данных (например, корпусе профессиональных резюме и описаний вакансий), что повышает точность и релевантность результатов анализа.

4. Обработка больших и сложных данных

BERT эффективно работает с большими объемами данных благодаря своей архитектуре, основанной на трансформерах. Это позволяет модели обрабатывать большие и сложные наборы текстов без значительной потери производительности, что важно при работе с обширными базами данных вакансий и резюме.

5. Возможности тонкой настройки

Настройка BERT под конкретные задачи (fine-tuning) позволяет добиться высокой точности в специфических приложениях, таких как классификация резюме, оценка соответствия квалификации и даже анализ эмоциональной окраски текста. Это обеспечивает значительную гибкость и мощь в руках разработчиков систем по подбору персонала.

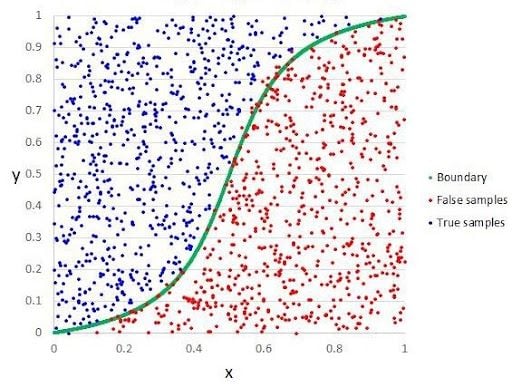
В итоге, выбор BERT для задачи анализа резюме и вакансий не только обоснован высокой эффективностью этой модели в обработке естественного языка, но и ее способностью к адаптации, масштабированию и точной настройке под нужды конкретной задачи, делая процесс отбора персонала более точным и объективным.

## Возможные подходы к построению предсказательной модели

Для построения эффективной предсказательной модели, способной оценивать успешность кандидатов по их резюме и характеристикам вакансий, можно использовать различные методы машинного обучения. Эти методы могут включать как классические алгоритмы, так и более современные подходы, использующие глубокое обучение. В этой главе мы рассмотрим несколько потенциальных подходов, которые могут быть применены в рамках нашего исследования.

**Логистическая регрессия**

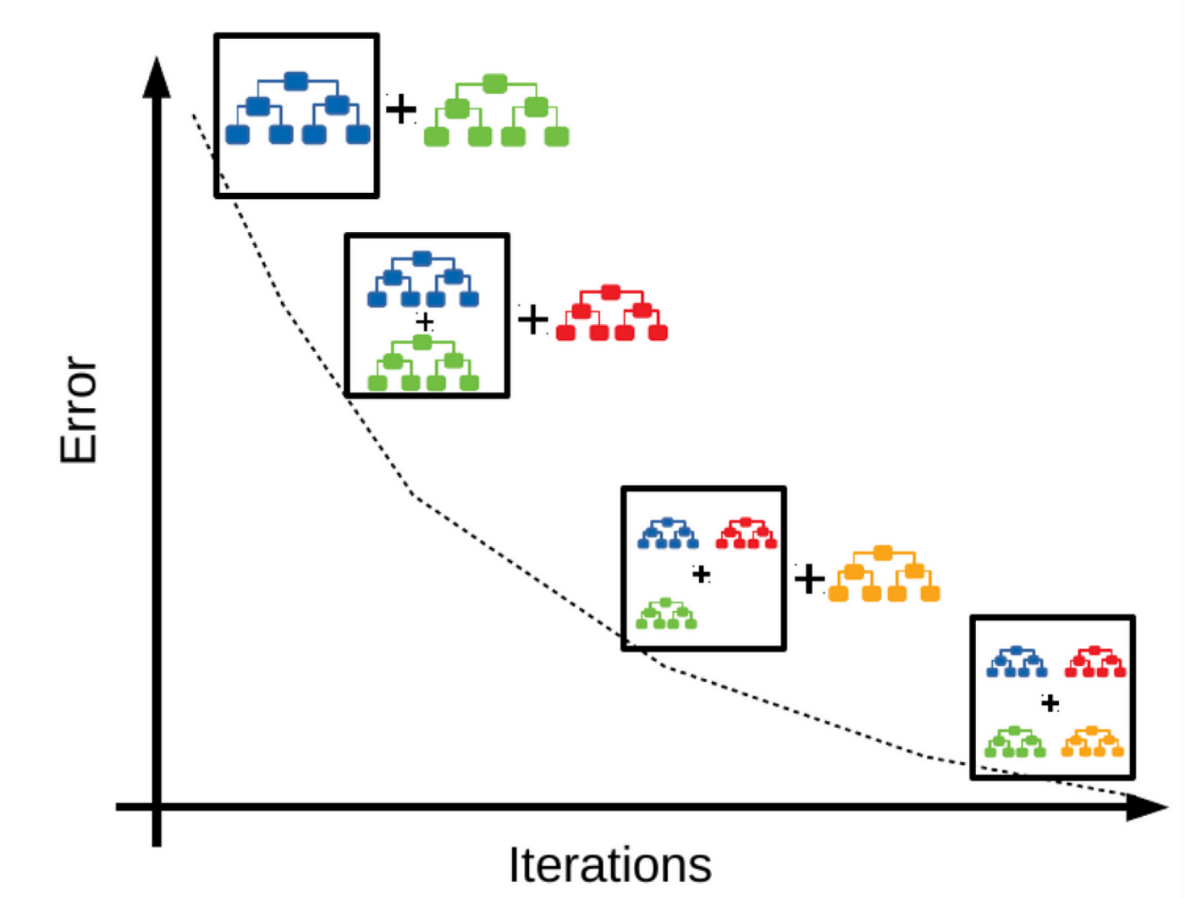
Логистическая регрессия — это статистический метод анализа набора данных, в котором одна или несколько независимых переменных определяют исход. Этот метод хорошо подходит для задач бинарной классификации и может быть использован для предсказания того, подходит ли кандидат на вакансию. Плюсы логистической регрессии включают простоту интерпретации результатов и меньшую вычислительную сложность по сравнению с более сложными моделями.



**Рисунок 7 — Логистическая регрессия**

**Градиентный бустинг**

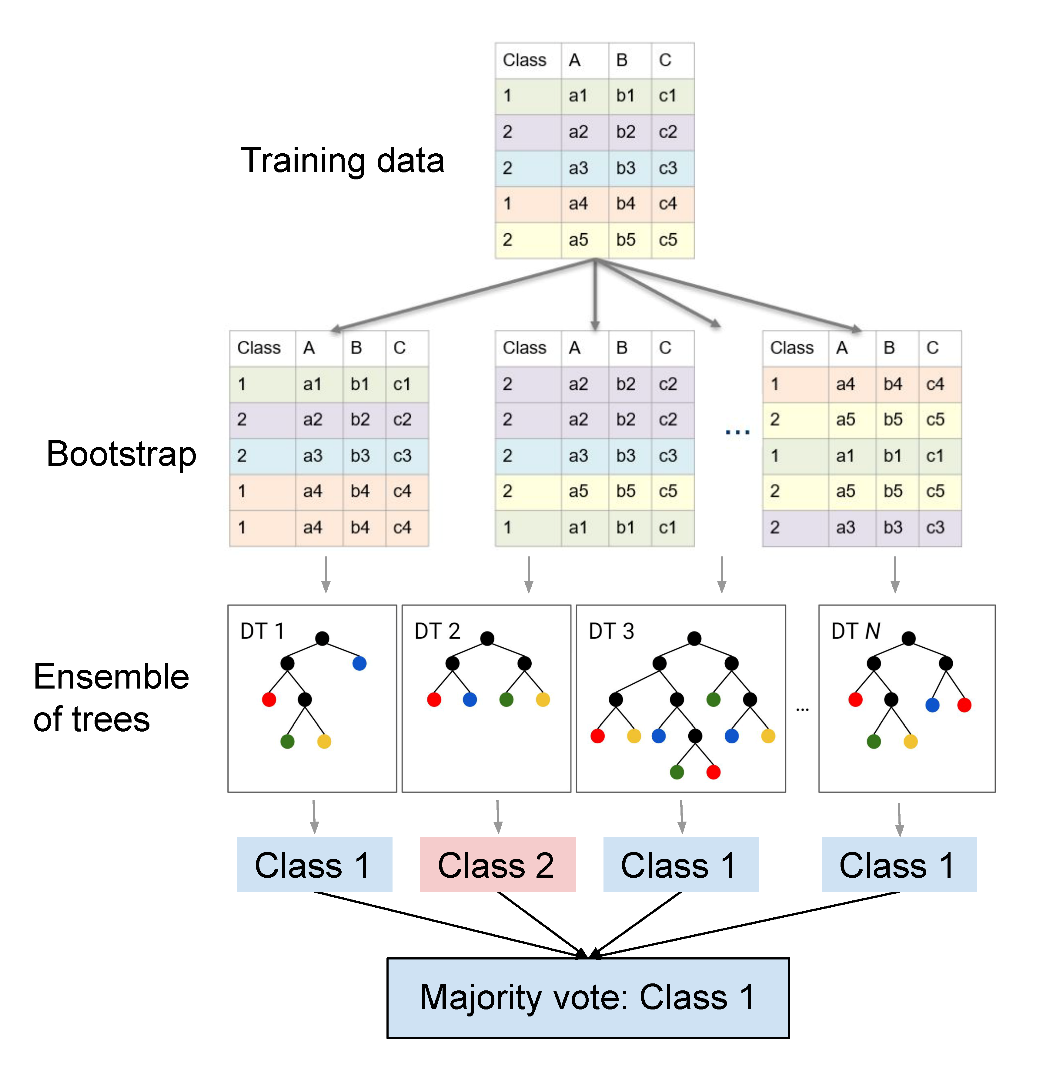
Градиентный бустинг — это техника машинного обучения для задач регрессии и классификации, которая строит модель в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Этот метод особенно эффективен благодаря своей способности минимизировать ошибку предсказания и адаптивности к сложным нелинейным закономерностям в данных. Градиентный бустинг хорошо работает даже с небольшими объемами данных и может обеспечить высокую точность предсказания.



**Рисунок 8 — Градиентный бустинг**

**Случайный лес**

Случайный лес — это метод ансамблевого обучения, базирующийся на агрегировании результатов множества деревьев решений. Этот метод хорошо подходит для классификации и регрессии и известен своей способностью к обработке больших данных с множеством входных переменных. Случайный лес помогает уменьшить переобучение, которое часто встречается в деревьях решений, и увеличивает точность модели благодаря механизму голосования или усреднения.



**Рисунок 9 — Случайный лес**

**Нейронные сети**

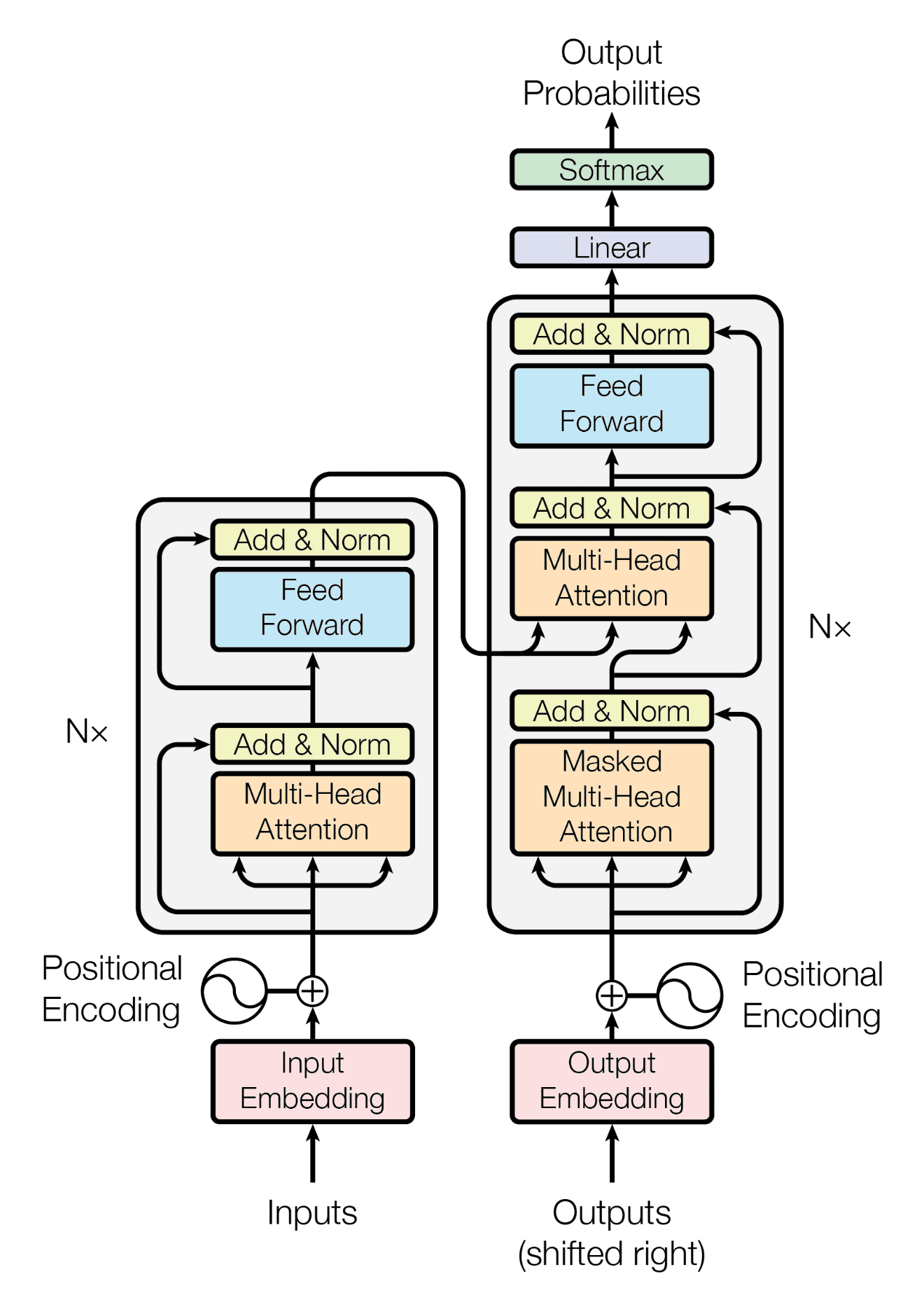
Глубокое обучение с использованием нейронных сетей может предложить значительные преимущества для обработки естественного языка, особенно при работе с большими и сложными наборами данных. Нейронные сети могут обучаться на задачах, связанных с текстом, и эффективно обрабатывать сложные зависимости и нелинейности. Использование сверточных нейронных сетей (CNN) или рекуррентных нейронных сетей (RNN) может быть особенно полезно для извлечения значимых признаков из текста резюме и вакансий.



**Рисунок 10 — Нейронная сети**

**Трансформеры**

Исходя из успеха архитектуры трансформеров, как в случае с BERT, мы можем разработать специализированную модель, которая адаптирована специально для наших нужд в предсказании успешности кандидатов. Модели, основанные на трансформерах, могут эффективно учитывать контекст в тексте и выделять значимые признаки, которые другие модели могут пропустить.



**Рисунок 11 — Трансформер**

**Анализ и выбор подхода**

Выбор подхода к построению модели зависит от множества факторов, включая объем доступных данных, требования к интерпретируемости модели, вычислительные ресурсы и специфику задачи. В данном контексте, мною была выбрана модель градиентного бустинга вместе с подбором гиперпараметров при помощи GridSearch. Обоснование этого метода описано в следующем разделе.

## **Обоснование выбора метода решения задачи**

Выбор подхода к построению модели зависит от множества факторов, включая объем доступных данных, требования к интерпретируемости модели, вычислительные ресурсы и специфику задачи. В контексте нашей задачи, градиентный бустинг в сочетании с техникой GridSearch для оптимизации гиперпараметров представляет собой привлекательный выбор по нескольким причинам.

1. Высокая точность и адаптивность

Градиентный бустинг известен своей высокой точностью и способностью адаптироваться к сложным закономерностям в данных. Метод строит ансамбль из простых моделей (чаще всего деревьев решений), последовательно улучшая предсказания на основе предыдущих ошибок. Это делает его особенно эффективным для нашей задачи, где необходимо учитывать множество переменных и их взаимосвязи в резюме и описаниях вакансий.

2. Управление переобучением

Переобучение — распространенная проблема в машинном обучении, особенно при работе с текстовыми данными, где возможно множество интерпретаций. Градиентный бустинг позволяет контролировать переобучение через различные параметры, такие как глубина дерева, количество деревьев и скорость обучения, что обеспечивает баланс между сложностью модели и ее обобщающей способностью.

3. Обработка различных типов данных

В нашем наборе данных присутствуют как числовые, так и категориальные переменные. Градиентный бустинг эффективно работает с такими смешанными данными, автоматически обрабатывая категориальные переменные и интегрируя их в процесс обучения без необходимости предварительной обработки.

**Применение GridSearch для оптимизации градиентного бустинга**

1. Автоматический подбор гиперпараметров

GridSearch — это метод систематического перебора заданных значений гиперпараметров для поиска тех, которые дают наилучший результат. Это особенно важно для градиентного бустинга, где правильная настройка параметров, таких как количество деревьев, глубина деревьев и скорость обучения, может значительно повлиять на производительность модели.

2. Улучшение производительности модели

Использование GridSearch позволяет автоматизированно находить оптимальное сочетание параметров, максимизируя точность и эффективность предсказаний. Это обеспечивает более высокую производительность модели по сравнению с произвольно выбранными или стандартными параметрами.

3. Валидация результатов

GridSearch также включает кросс-валидацию, которая используется для оценки эффективности модели на различных подмножествах данных. Это гарантирует, что модель будет обобщать, а не просто запоминать тренировочные данные, что особенно важно в HR-задачах, где новые данные могут значительно отличаться от тех, на которых производилось обучение.

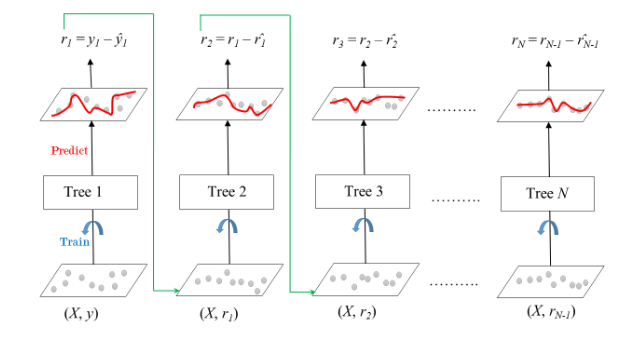
Выбор градиентного бустинга с применением GridSearch для нашей задачи предсказания успешности кандидатов основан на потребности в создании мощной, точной и адаптивной модели. Этот подход позволяет максимально использовать доступные данные и обеспечивает гибкость для адаптации к изменяющимся условиям рынка труда.

## Объяснение принципов работы градиентного бустинга

Градиентный бустинг — это мощная техника машинного обучения, которая используется для построения предсказательных моделей, основанных на ансамблях слабых моделей предсказания, обычно деревьев решений. Основная идея заключается в последовательном улучшении предсказаний за счет коррекции ошибок предыдущих моделей. В этой главе рассмотрены принципы работы градиентного бустинга.

* Структура модели

Градиентный бустинг строит модель путем последовательного добавления новых моделей, так что каждая последующая модель пытается исправить ошибки, сделанные предыдущими. Процесс начинается с создания базовой модели (обычно это дерево решений), которая делает первичные предсказания. Затем вычисляются ошибки этих предсказаний относительно истинных значений.



**Рисунок 12 — Градиентный бустинг**

* Вычисление ошибок

Ошибки предсказаний рассматриваются как градиенты функции потерь, которая оценивает точность модели. На каждом шаге строится новое дерево, цель которого — минимизировать эти ошибки, то есть фактически улучшить модель в тех областях, где она предсказывает неверно.

* Минимизация функции потерь

Функция потерь — это метрика, которая измеряет разницу между предсказанными значениями и истинными значениями. Градиентный бустинг использует градиент этой функции для определения направления в пространстве параметров, которое приведет к наибольшему уменьшению ошибок. С каждым шагом добавляется новое дерево, которое направлено на минимизацию этих градиентов.

* Уменьшение шага обучения

Один из ключевых параметров в градиентном бустинге — это темп обучения или скорость обучения. Этот параметр контролирует, насколько сильно каждое новое дерево влияет на общую модель. Маленький темп обучения может привести к тому, что для достижения высокой точности потребуется больше деревьев, но это также может помочь избежать переобучения.

* Предотвращение переобучения

Градиентный бустинг включает механизмы для предотвращения переобучения, что особенно важно при работе с комплексными данными. К таким механизмам относятся:

- Ограничение глубины деревьев.

- Установка минимального количества образцов, необходимых для разделения узла дерева.

- Введение случайности через технику стохастического градиентного бустинга, при котором для построения каждого дерева используется случайная подвыборка данных.

* Адаптивность и гибкость

Одно из главных преимуществ градиентного бустинга заключается в его адаптивности: метод способен эффективно обрабатывать различные типы данных и находить сложные нелинейные зависимости в них. Эта гибкость делает градиентный бустинг особенно подходящим для задач, где данные содержат множество переменных разных типов, таких как текстовые данные в HR-аналитике.

* Заключение

Градиентный бустинг — это высокоэффективный и гибкий метод машинного обучения, идеально подходящий для сложных задач предсказания, таких как анализ резюме и вакансий. Его способность минимизировать ошибки и адаптироваться к разнообразным данным делает его одним из лучших выборов для исследования в области HR-технологий.

## Метрики для оценки модели

В процессе разработки и оценки модели сопоставления вакансий и резюме, использующей методы машинного обучения, критически важно выбрать правильные метрики для измерения эффективности модели. Эти метрики помогут определить, насколько хорошо модель справляется с задачей и какие аспекты нуждаются в доработке. Вот несколько ключевых метрик, которые могут быть использованы в данном контексте:

1. Точность (Accuracy)

Точность — это одна из наиболее очевидных метрик для оценки производительности классификационных моделей. Она измеряет долю правильных предсказаний из всех предсказаний и является полезным показателем, когда классы сбалансированы. В контексте сопоставления вакансий и резюме, точность может показать, насколько часто модель правильно сопоставляет кандидатов с подходящими вакансиями.

2. Точность (Precision) и Полнота (Recall)

Точность и полнота — это две метрики, которые часто рассматриваются вместе, чтобы оценить качество предсказаний модели, особенно в условиях несбалансированных классов:

- Точность оценивает, какой процент предсказанных сопоставлений действительно правильный. Это критически важно, когда неверное сопоставление может стоить времени и ресурсов.

- Полнота измеряет, какой процент реальных сопоставлений был правильно идентифицирован моделью. Это особенно важно для обеспечения того, чтобы квалифицированные кандидаты не были упущены.

3. F-мера (F1 Score)

F-мера (или F1 Score) — это гармоническое среднее точности и полноты. Эта метрика полезна, когда нужно сбалансировать точность и полноту, особенно если более важно одновременно минимизировать ложные положительные и ложные отрицательные результаты. F-мера является одной из лучших метрик для оценки эффективности моделей в задачах, где обе метрики (точность и полнота) критичны.

4. AUC-ROC

Кривая рабочих характеристик приемника (ROC) и площадь под кривой (AUC) — это метрики, которые используются для оценки качества моделей в условиях, когда классы несбалансированы. Эти метрики измеряют способность модели правильно классифицировать кандидатов как подходящих или неподходящих по всем порогам классификации.

5. Матрица ошибок (Confusion Matrix)

Матрица ошибок предоставляет более детальное представление о производительности модели, показывая количество истинных положительных, истинных отрицательных, ложных положительных и ложных отрицательных результатов. Это позволяет более глубоко анализировать специфику ошибок модели.

6. Средняя абсолютная ошибка (MAE) и Среднеквадратичная ошибка (MSE)

В задачах регрессии, например, при предсказании степени соответствия кандидата вакансии, можно использовать метрики такие как средняя абсолютная ошибка (MAE) и среднеквадратичная ошибка (MSE). Они измеряют среднее отклонение предсказанных значений от фактических, помогая оценить точность количественных предсказаний модели.

Эти метрики позволят оценить, насколько эффективно модель справляется с задачей сопоставления вакансий и резюме, а также помогут идентифицировать области для дальнейших улучшений.

# РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

В данном исследовании была применена модель BERT, а затем градиентный бустинг с подбором гиперпараметров. В приведенном коде осуществляется процесс подготовки данных, их анализ, обучение модели машинного обучения и оценка ее качества. Код использует библиотеки Python для работы с данными, машинного обучения и обработки естественного языка. Ниже представлено подробное описание каждого этапа этого процесса.

|  |
| --- |
| ### Импорт библиотек  import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split, GridSearchCV  from sklearn.preprocessing import LabelEncoder, StandardScaler  from sklearn.ensemble import GradientBoostingClassifier  from sklearn.metrics import classification\_report, roc\_auc\_score  from sklearn.utils import resample  from transformers import BertTokenizer, BertModel  import torch |

Листинг 1

Здесь импортируются необходимые библиотеки и модули:

- `pandas` и `numpy` для работы с данными.

- `train\_test\_split` и `GridSearchCV` из `sklearn` для разделения данных на обучающую и тестовую выборки и оптимизации гиперпараметров модели.

- `LabelEncoder` и `StandardScaler` для предварительной обработки данных.

- `GradientBoostingClassifier` для использования алгоритма градиентного бустинга.

- `classification\_report` и `roc\_auc\_score` для оценки производительности модели.

- `resample` для балансировки классов.

- `BertTokenizer` и `BertModel` для работы с предобученной моделью BERT.

- `torch` для работы с тензорами и возможности использования GPU.

Определение того, будет ли использоваться GPU (если доступно) или CPU для обработки данных в PyTorch.

|  |
| --- |
| ### Подготовка к использованию GPU  device = torch.device("cuda" if torch.cuda.is\_available() else "cpu") |

Листинг 2

Данные загружаются из файла CSV в DataFrame `pandas`.

|  |
| --- |
| ### Загрузка данных  df = pd.read\_csv("data.csv") |

Листинг 3

Загружается мультиязычная версия модели BERT и соответствующий токенизатор, модель перемещается на выбранное устройство (GPU или CPU).

|  |
| --- |
| Загрузка модели BERT и токенизатора  tokenizer = BertTokenizer.from\_pretrained('bert-base-multilingual-cased')  model = BertModel.from\_pretrained('bert-base-multilingual-cased').to(device) |

Листинг 4

Функция принимает тексты, токенизирует их, передает на вход модели BERT и возвращает среднее векторных представлений последнего скрытого слоя модели. Это представление используется как фичи для машинного обучения.

|  |
| --- |
| Функция для получения эмбеддингов BERT  def get\_bert\_embeddings\_batch(texts):  inputs = tokenizer(texts, return\_tensors='pt', truncation=True, padding=True, max\_length=128).to(device)  with torch.no\_grad():  outputs = model(\*\*inputs)  return outputs.last\_hidden\_state.mean(dim=1).cpu().numpy() |

Листинг 5

Функция принимает тексты, токенизирует их, передает на вход модели BERT и возвращает среднее векторных представлений последнего скрытого слоя модели. Это представление используется как фичи для машинного обучения.

|  |
| --- |
| ### Обработка текстовых данных  text\_columns = ['about', 'key\_skills', 'vacancy.description', 'work\_summary']  batch\_size = 32  for col in text\_columns:  embeddings = []  texts = df[col].astype(str).fillna('').tolist()  for i in range(0, len(texts), batch\_size):  batch\_texts = texts[i:i + batch\_size]  batch\_embeddings = get\_bert\_embeddings\_batch(batch\_texts)  embeddings.append(batch\_embeddings)  embeddings = np.vstack(embeddings)  df\_bert = pd.DataFrame(embeddings, columns=[f"{col}\_bert\_{i}" for i in range(embeddings.shape[1])])  df.reset\_index(drop=True, inplace=True)  df\_bert.reset\_index(drop=True, inplace=True)  df = pd.concat([df, df\_bert], axis=1) |

Листинг 6

Для каждого текстового столбца данные преобразуются в векторные представления с помощью BERT. Полученные данные объединяются с основным набором данных.

Классы данных балансируются путем увеличения числа объектов меньшего класса.

|  |
| --- |
| ### Балансировка классов  df\_majority = df[df.status == 0]  df\_minority = df[df.status == 1]  df\_minority\_upsampled = resample(df\_minority, replace=True, n\_samples=len(df\_majority), random\_state=123)  df\_balanced = pd.concat([df\_majority, df\_minority\_upsampled]) |

Листинг 7

Данные разделяются на признаки и целевую переменную, далее на обучающую и тестовую выборку. Признаки масштабируются для улучшения производительности модели.

|  |
| --- |
| Подготовка данных к обучению  X = df\_balanced.drop(columns=['status'])  y = df\_balanced['status']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test) |

Листинг 8

Используется `GridSearchCV` для нахождения оптимальных гиперпараметров градиентного бустинга. Модель обучается на обучающем наборе данных.

|  |
| --- |
| ### Оптимизация гиперпараметров и обучение модели  param\_grid = {'n\_estimators': [100, 200], 'learning\_rate': [0.01, 0.1], 'max\_depth': [3, 5, 7]}  grid\_search = GridSearchCV(estimator=GradientBoostingClassifier(random\_state=42), param\_grid=param\_grid, cv=5, scoring='roc\_auc')  grid\_search.fit(X\_train, y\_train)  best\_model = grid\_search.best\_estimator\_ |

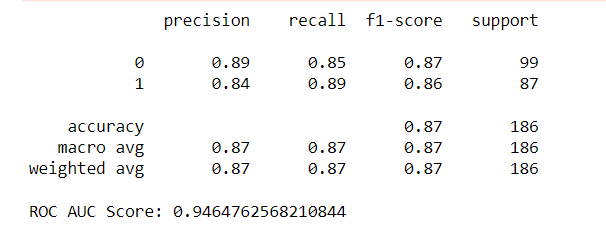
Листинг 9

На тестовой выборке выполняется предсказание. Рассчитывается отчет о классификации и ROC AUC, что позволяет оценить качество модели.

|  |
| --- |
| ### Предсказание и оценка модели  y\_pred = best\_model.predict(X\_test)  y\_pred\_proba = best\_model.predict\_proba(X\_test)[:, 1]  print(classification\_report(y\_test, y\_pred))  print(f"ROC AUC Score: {roc\_auc\_score(y\_test, y\_pred\_proba)}") |

Листинг 10

После выполнения кода и обучения модели получаем основные метрики качества модели. Исходя из полученных значений, предсказания получаются довольно точными, что говорит об успешном выполнении поставленной задачи.



**Рисунок 13 — Результаты выполнения кода**

# ЗАКЛЮЧЕНИЕ

В заключение стоит отметить, что использование программных средств, основанных на искусственном интеллекте, в решении прикладных задач оказывает огромное влияние на современные процессы обработки данных и управления информацией. Применение таких технологий, как градиентный бустинг и модели BERT для анализа текстовой информации, позволяет не только существенно повысить точность анализа данных, но и открывает новые возможности для автоматизации и оптимизации рабочих процессов в самых разных сферах. В частности, в области управления человеческими ресурсами это может привести к более эффективному подбору персонала и оптимизации внутренних процедур. Программы на основе искусственного интеллекта, способные обрабатывать и анализировать большие объемы данных, становятся ключевыми инструментами в руках специалистов, позволяя им делать обоснованные и точные решения быстрее. Таким образом, интеграция ИИ в программные решения прикладных задач продолжит оставаться одним из приоритетных направлений в развитии IT-индустрии, способствуя трансформации и модернизации традиционных подходов к управлению и анализу данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы градиентного бустинга - https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

2. Документация BERT от Hugging Face - https://huggingface.co/transformers/model\_doc/bert.html

3. Использование PyTorch для работы с нейронными сетями - https://pytorch.org/docs/stable/index.html

4. Примеры использования GridSearchCV - https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html

5. Методы масштабирования данных в машинном обучении - https://scikit-learn.org/stable/modules/preprocessing.html

6. Обучение по подбору персонала с использованием ИИ - https://towardsdatascience.com/machine-learning-for-recruitment-fc12aee2b6a6

7. Руководство по Python Pandas для обработки данных - https://pandas.pydata.org/pandas-docs/stable/user\_guide/index.html

8. Работа с большими наборами данных в Python - https://realpython.com/pandas-python-explore-dataset/

9. Принципы оценки моделей машинного обучения - https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

10. Техники балансировки классов в наборах данных - https://elitedatascience.com/imbalanced-classes

# Приложение А

*Листинг А — Код приложения*

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split, GridSearchCV  from sklearn.preprocessing import LabelEncoder, StandardScaler  from sklearn.ensemble import GradientBoostingClassifier  from sklearn.metrics import classification\_report, roc\_auc\_score  from sklearn.utils import resample  from transformers import BertTokenizer, BertModel  import torch  # Проверка доступности GPU  device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")  df = pd.read\_csv("data.csv")  # Загрузка токенизатора и модели BERT  tokenizer = BertTokenizer.from\_pretrained('bert-base-multilingual-cased')  model = BertModel.from\_pretrained('bert-base-multilingual-cased').to(device)  def get\_bert\_embeddings\_batch(texts):  inputs = tokenizer(texts, return\_tensors='pt', truncation=True, padding=True, max\_length=128).to(device)  with torch.no\_grad():  outputs = model(\*\*inputs)  return outputs.last\_hidden\_state.mean(dim=1).cpu().numpy()  # Применение BERT для текстовых столбцов партиями  text\_columns = ['about', 'key\_skills', 'vacancy.description', 'work\_summary']  batch\_size = 32  for col in text\_columns:  embeddings = []  texts = df[col].astype(str).fillna('').tolist()  for i in range(0, len(texts), batch\_size):  batch\_texts = texts[i:i + batch\_size]  batch\_embeddings = get\_bert\_embeddings\_batch(batch\_texts)  embeddings.append(batch\_embeddings)  embeddings = np.vstack(embeddings)  df\_bert = pd.DataFrame(embeddings, columns=[f"{col}\_bert\_{i}" for i in range(embeddings.shape[1])])  # Сброс индексов перед конкатенацией  df.reset\_index(drop=True, inplace=True)  df\_bert.reset\_index(drop=True, inplace=True)  df = pd.concat([df, df\_bert], axis=1)  # Удаление ненужных столбцов  drop\_columns = ['uuid', 'first\_name', 'last\_name', 'birth\_date', 'about', 'key\_skills', 'experienceItem',  'educationItem', 'languageItems', 'vacancy.description', 'vacancy.keywords', 'vacancy.comment',  'languageItem']  df.drop(columns=drop\_columns, inplace=True)  # Балансировка классов  df\_majority = df[df.status == 0]  df\_minority = df[df.status == 1]  df\_minority\_upsampled = resample(df\_minority,  replace=True, # Дублирование меньшинства  n\_samples=len(df\_majority), # Доведение до размера большинства  random\_state=123) # Случайный сид  df\_balanced = pd.concat([df\_majority, df\_minority\_upsampled])  # Разделение данных на признаки и целевую переменную  X = df\_balanced.drop(columns=['status'])  y = df\_balanced['status']  # Разделение на обучающую и тестовую выборки  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Масштабирование признаков  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test)  # Оптимизация гиперпараметров с помощью GridSearchCV  param\_grid = {  'n\_estimators': [100, 200],  'learning\_rate': [0.01, 0.1],  'max\_depth': [3, 5, 7]  }  grid\_search = GridSearchCV(estimator=GradientBoostingClassifier(random\_state=42), param\_grid=param\_grid, cv=5,  scoring='roc\_auc')  grid\_search.fit(X\_train, y\_train)  best\_model = grid\_search.best\_estimator\_  # Предсказание и оценка модели  y\_pred = best\_model.predict(X\_test)  y\_pred\_proba = best\_model.predict\_proba(X\_test)[:, 1]  print(classification\_report(y\_test, y\_pred))  print(f"ROC AUC Score: {roc\_auc\_score(y\_test, y\_pred\_proba)}") |