МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 «Прикладная математика и информатика»
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Исследование и реализация алгоритма градиентного бустинга для прогнозирования данных

Обучающийся Н.А. Лобырев
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.ф.н., доцент, Н.В. Ященко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

## Аннотация

Тема выпускной квалификационной работы – «Исследование и реализация алгоритма градиентного бустинга для прогнозирования данных».

Ключевые слова: исследование алгоритма, градиентный бустинг, прогнозирование данных.

Объектом исследования бакалаврской работы является процесс прогнозирования данных.

Предметом исследования бакалаврской работы является алгоритм градиентного бустинга.

Цель бакалаврской работы – исследование и программная реализация алгоритма градиентного бустинга для прогнозирования данных.

Методы исследования – методы и алгоритмы прогнозирования данных, методы и технологии разработки программного обеспечения.

Первая глава работы посвящена постановке задачи исследования алгоритма градиентного бустинга для прогнозирования данных.

Вторая глава работы посвящена обзору и анализу алгоритмов градиентного бустинга для прогнозирования данных.

В третьей главе рассматривается процесс разработки и тестирования программы, реализующей градиентного бустинга для прогнозирования данных.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Выпускная квалификационная работа состоит из 43 страниц текста, 17 рисунков, 3 таблиц и 25 источников.

## Abstract

The title of the graduation work is Research and implementation of a gradient boosting algorithm for data forecasting.

Keywords: algorithm research, gradient boosting, data forecasting.

The research and implementation of efficient gradient boosting algorithms are important tasks with both scientific and practical relevance.

The object of the graduation work is a data forecasting process.

The subject of the graduation work is a gradient boosting algorithm.

The aim of the work is to study and implement a gradient boosting algorithm for data forecasting.

Research methods include data forecasting techniques and software development methods and technologies.

The work consists of an introduction, three chapters, a conclusion, and a list of references and sources used.

The research and implementation of gradient boosting algorithms contribute to the advancement of data analysis and machine learning theory.

The practical application of such algorithms enables solving real-world problems across various industries.

The graduation work consists of 43 pages of text, 17 figures, 3 tables and 25 sources.

## Оглавление

# Введение

Прогнозирование – это метод, который использует исторические данные для принятия обоснованных решений о будущих событиях или условиях. Это не просто догадки. Являясь инструментом для предприятий и инвесторов, прогнозирование использует экспертный анализ и применяет сложные модели для распределения портфелей и бюджетов. Однако, как показывает практика, зачастую прогнозы не всегда надежны.

Тем не менее, прогнозирование данных занимает центральное место в современных инвестициях и деловых практиках. Компании нанимают и расширяются на основе прогнозирования показателей продаж, рыночного спроса или экономических показателей. Инвесторы торгуют акциями, вкладывают средства в фонды или поспешно выходят с рынка на основе прогнозов относительно цен на акции, процентных ставок или более широких движений рынка. Модели потребительских расходов, тенденции рынка труда и даже геополитические события – все это попадает в компетенцию прогнозистов.

Для повышения надежности прогнозов используются различные алгоритмы в том числе алгоритмы градиентного бустинга. Модели градиентного бустинга приобрели популярность в сообществе машинного обучения благодаря своей способности достигать превосходных результатов в широком диапазоне вариантов использования, включая как регрессию, так и классификацию. Хотя эти модели традиционно были менее распространены в прогнозировании, они могут быть весьма эффективными в этой области.

Таким образом, исследование и программная реализация алгоритмов градиентного бустинга представляет научный и практический интерес.

Объектом исследования бакалаврской работы является процесс прогнозирования данных.

Предметом исследования бакалаврской работы является алгоритм градиентного бустинга.

Цель бакалаврской работы – исследование и программная реализация алгоритма градиентного бустинга для прогнозирования данных.

Для достижения данной цели необходимо выполнить следующие задачи:

- выполнить постановку задачи исследования алгоритма градиентного бустинга для прогнозирования данных;
- проанализировать алгоритмы алгоритма градиентного бустинга для прогнозирования данных;
- выполнить программную реализацию и протестировать алгоритм градиентного бустинга для прогнозирования данных.

Методы исследования – методы и алгоритмы прогнозирования данных, методы и технологии разработки программного обеспечения.

Практическая значимость бакалаврской работы заключается в разработке программы, позволяющий формировать прогнозы с помощью алгоритма градиентного бустинга.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы и используемых источников.

Выпускная квалификационная работа состоит из 43 страниц текста, 17 рисунков, 3 таблиц и 25 источников.

# Глава 1 Постановка задачи исследования алгоритма градиентного бустинга для прогнозирования данных

## 1.1 Математическое описание задачи прогнозирования данных

Прогнозирование – это метод предсказания будущего события или состояния путем анализа закономерностей и выявления тенденций в предыдущих и текущих данных. Он использует математические подходы и применяет статистические модели для создания прогнозов [9].

«Задачи диагностики и прогнозирования некоторой величины $Y$ по доступным значениям переменных $X_1, \dots, X_n$ возникают в различных областях человеческой деятельности:

- постановка медицинского диагноза или результатов лечения по совокупности клинических и лабораторных показателей;
- прогноз свойств ещё не синтезированного химического соединения по его молекулярной формуле;
- диагностика хода технологического процесса; диагностика состояния технического оборудования;
- прогноз финансовых индикаторов и другие» [6].

Понимание математических моделей в прогнозировании является важнейшим аспектом анализа и прогнозирования данных.

Рассмотрим различные аспекты, связанные с математическими моделями и их применением в прогнозировании [13].

Математические модели служат мощными инструментами для прогнозирования, фиксируя сложные взаимосвязи и закономерности в данных. Эти модели позволяют аналитикам делать обоснованные прогнозы и принимать решения на основе исторических тенденций и статистического анализа.

Типы математических моделей:

- модели регрессии. Регрессионный анализ обычно используется в

прогнозировании для установления связей между зависимыми и независимыми переменными. Подгоняя линию регрессии к данным, аналитики могут оценить будущие значения на основе выявленных закономерностей;

- модели временных рядов фокусируются на анализе точек данных, собранных с течением времени. Такие методы, как скользящие средние, экспоненциальное сглаживание и модели авторегрессионного интегрированного скользящего среднего, используются для прогнозирования будущих значений на основе исторических моделей;
- нейронные сети, вдохновленные человеческим мозгом, все чаще используются в прогнозировании. Эти модели могут улавливать сложные нелинейные взаимосвязи и адаптироваться к изменяющимся рыночным условиям, что делает их ценными инструментами для точных прогнозов.

Рассмотрим формализованную постановку задачи прогнозирования на примере классической линейной парной модели регрессии (КЛПМР) [2].

Пусть имеется КЛПМР вида (1):

$$y_i = \beta_1 + \beta_2 x_i + \varepsilon_i, \quad (1)$$

где $y_i$ – значения зависимой переменной;

$x_i$ – значения независимой переменной (регрессора);

$\varepsilon_i$ – случайные ошибки;

$\beta_1$, $\beta_2$ – истинные значения параметров модели, которые на практике никогда не известны исследователю.

Предположим, что мы получили с помощью метода наименьших квадратов (МНК), линию регрессии, оцененную на основе $n$ наблюдений (2):

$$\widehat{y_i} = \widehat{\beta_1} + \widehat{\beta_2} x_i, \quad (2)$$

где $\widehat{\beta_1}$ и $\widehat{\beta_2}$ оценки параметров, полученные при помощи МНК, на основе случайной выборки.

Пусть у нас есть известное ($n$+1)-е наблюдение регрессора $x_{n+1}$, тогда прогнозное значение переменной $y_{n+1}$ можно определить следующим образом (3):

$$\widehat{y_{n+1}} = \widehat{\beta_1} + \widehat{\beta_2} x_{n+1}, \quad (3)$$

Стандартная ошибка прогноза определяется по формуле (4):

$$\delta = \sqrt{s^2 * \left(1 + \frac{1}{n} + \frac{(x_{n+1} - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right)}, \quad (4)$$

где $s^2$ – несмещенная оценка, используемая для расчета стандартных ошибок коэффициентов.

Следует отметить, что при выборе математической модели для прогнозирования следует учитывать несколько факторов:

- доступность и качество исторических данных играют решающую роль в определении пригодности конкретной модели. Для установления надежных закономерностей и взаимосвязей необходимо достаточное количество точек данных.
- сложность модели должна быть сбалансирована с ее интерпретируемостью. Хотя более сложные модели могут давать лучшую точность, их может быть сложно понять и объяснить;
- каждая математическая модель имеет свои предположения и ограничения.

Важно понимать эти ограничения и оценивать их влияние на точность прогнозов.

## 1.2 Обзор и анализ методов прогнозирования данных

В последнее время для прогнозирования данных широко применяются ансамблевые методы прогнозирования.

Ансамблевое прогнозирование – это метод численного прогнозирования, который включает в себя создание репрезентативной выборки потенциальных будущих состояний модельной системы с использованием нескольких моделей или вариаций входных данных, параметров или конфигураций [12].

Рассмотрим основные методы ансамблевого прогнозирования: метод взвешенного среднего, стекинг, бутстрэп, бэггинг и бустинг [16].

Метод взвешенного среднего (Weighted Average) широко используется для объединения результатов модели и является самым ранним методом, использованным в основополагающей работе по ансамблевым моделям.

Выходным данным каждой базовой модели присваивается вес, который зависит от некоторых критериев эффективности базовых моделей; и веса всех базовых моделей в сумме дают единицу. Обоснованием правила ансамбля средневзвешенного значения является предоставление лучшим моделям более высокого веса в выходных данных ансамблевой модели. Веса, назначаемые различным выходным данным модели, могут быть основаны на различных критериях. Общие критерии включают дисперсию ошибки, вклад моделей в снижение дисперсии ошибки и процент правильной классификации в моделях классификации.

Прогноз в данном методе рассчитывается по формуле (5):

$$\overline{Y_w} = \frac{\sum_{k=1}^{n} x_k w_k}{\sum_{k=1}^{n} w_k}, \tag{5}$$

где $x_1, x_2, ..., x_n$ – значения временного ряда:

$w_1, w_2, ..., w_n$ – веса.

Метод стекинга (Stacking) – это ансамблевый метод, который объединяет несколько моделей для уменьшения их смещений и улучшения предсказательной эффективности. Более конкретно, прогнозы каждой модели (базовые модели) стекируются и используются в качестве входных данных для окончательной модели (метамодели) для вычисления прогноза.

Стекинговая регрессия – это метод интеграционного обучения, который объединяет несколько регрессионных моделей с помощью мета-регрессора.

Помимо этого, каждая базовая регрессионная модель обучается с помощью всего обучающего набора, а выходные данные каждой базовой регрессионной модели используются в качестве входных данных для мета-регрессора в качестве мета-признаков в процессе интеграционного обучения. Затем мета-регрессор объединяет несколько моделей, подгоняя мета-признаки к каждой базовой регрессионной модели.

На рисунке 1 показана концептуальная модель стекинга.

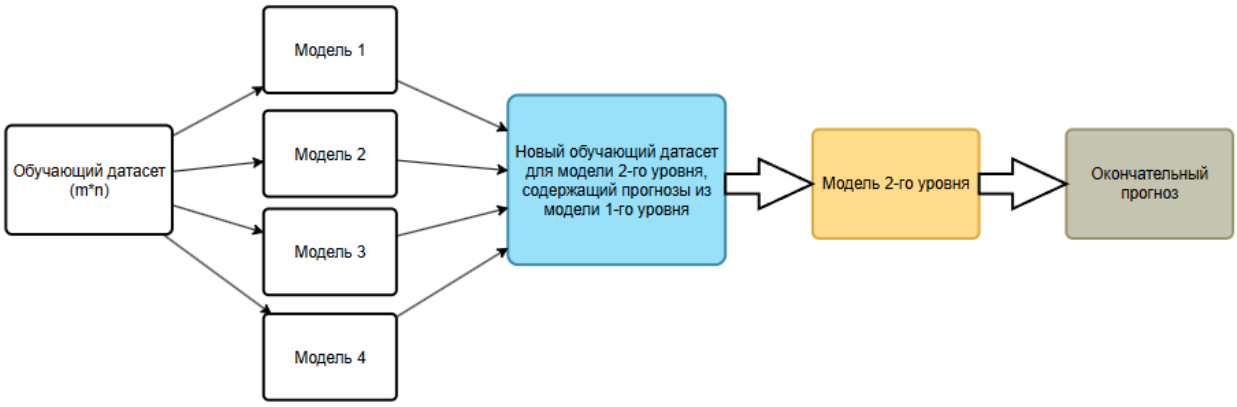


Рисунок 1 – Концептуальная модель стекинга

Стекинг эффективен, поскольку он использует сильные стороны различных алгоритмов и пытается смягчить их индивидуальные слабости. Объединяя несколько моделей, он может улавливать сложные закономерности в данных и повышать точность прогноза [18].

Методы бутстрэпа и бэггинга (Bootstrapping and bagging) относятся к

практике многократного взятия случайных небольших выборок, а также подмножеств объясняющих переменных из больших обучающих данных с заменой (бутстрэп-выборки), затем подгонки моделей к каждой выборке и объединения этих моделей (бэггинг). Идея методов бутстрап и бэггинга заключается в том, что, согласно Перроне и Куперу, существует вероятность того, что некоторые другие модели могут работать лучше, чем одна модель на некоторых невидимых данных, поэтому средняя производительность этих моделей имеет большую надежность, чем при использовании одной модели [5].

Псевдокод алгоритма классификации методом бутстрэпа показан на рисунке 2.

```
для i = 1 до m {
D' = выборка бутстрапа из D (выборка с заменой)
Ci = I(D')
}
C*(x) = argmax #{i:Ci(x)=y} (чаще всего прогнозируемая метка y)
y∈Y
```

Рисунок 2 – Псевдокод алгоритма классификации методом бутстрэпа

Для классификации в качестве входных данных используется обучающий датасет $D$, индуктор $I$ и набор бутстрэп-выборок $m$.

Выходом является сгенерированный классификатор $C^*$.

Алгоритм состоит из следующих шагов:

Шаг 1. Создать $m$ новых обучающих датасетов $D_i$ из обучающего датасета $D$ с заменой.

Шаг 2. Классификатор $C_i$ строится из каждого датасета $D_i$ с помощью $I$ для определения классификации датасета $D_i$.

Шаг 3. Наконец, классификатор $C^*$ генерируется с использованием ранее созданного набора классификаторов $C_i$ на исходном датасете $D$. При

этом классификация, предсказываемая чаще всего подклассификаторами $C_i$, является окончательной классификацией.

Следует отметить, что хорошо известный алгоритм Random Forest представляет собой ансамблевую модель, основанную на объединении нескольких деревьев классификации [20].

Бустинг (boosting) относится к классу ансамблевого моделирования, который превращает несколько слабых алгоритмов обучения в модель с высокой точностью. Сначала обучается слабая модель (регрессор или классификатор), что едва ли лучше случайного угадывания; последующие модели обучаются предсказывать оставшуюся ошибку предыдущей модели итеративным образом.

Существует много вариантов методов бустинга. Эти варианты имеют одну и ту же цель, но имеют небольшие различия в реализации.

Рассмотрим метод градиентного бустинга.

«Градиентный бустинг (Gradient Boosting) – это метод машинного обучения, который в основном используется для задач регрессии и классификации. Он направлен на создание предсказательной модели в виде ансамбля слабых предсказательных моделей, обычно деревьев решений» [14].

Возникший из идеи усиления слабых учеников, градиентный бустинг итеративно добавляет модели в ансамбль, фокусируясь на областях, где существующие модели работают плохо.

Рассмотрим классический алгоритм градиентного бустинга.

«Шаг 1. Предположим, что $X$ и $Y$ – это входные данные и цель, имеющие $N$ образцов. Наша цель – исследовать функцию $f(x)$, которая сопоставляет входные признаки $X$ с целевыми переменными $y$. Это расширенные деревья, т. е. сумма деревьев.

Функция потерь представляет собой разницу между фактическими и прогнозируемыми переменными (6):

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i). \quad (6)$$

Шаг 2. Мы хотим минимизировать функцию потерь L(f) относительно f (7):

$$\widehat{f_0}(x) = argmin_f\, L(f) = argmin_f \sum_{i=1}^{N} L(y_i, f(x_i). \quad (7)$$

Если наш алгоритм градиентного бустинга состоит из *M* этапов, то для улучшения алгоритма можно добавить новую оценку $h_m$, имеющую $1 \leq m \leq M$ (8):

$$\hat{y}_i = F_{m+1}(x_i) + h_m(x_i). \quad (8)$$

Шаг 3. Самый крутой спуск.

Для градиентного бустинга на этапе *M* метод наискорейшего спуска находит, $h_m = -\rho_m g_m$, где $\rho_m$ – константа, известная как длина шага, а $g_m$ – градиент функции потерь *L(f)* (9)» [14]:

$$g_{im} = -\left[\frac{\partial L(y_i, f(x_i)}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}. \quad (9)$$

Шаг 4. Решение.

Градиент аналогично для *M* деревьев (10):

$$f_m(x) = f_{m-1}(x) + argmin_{h_m \in H}\left(\sum_{i=1}^{N} L\big(y_i, f_{m-1}(x_i), h_m(x_i)\big)\right)(x). \quad (10)$$

Текущее решение будет (11):

$$f_m = f_{m-1} - \rho_m g_m. \quad (11)$$

Для сравнения характеристик рассмотренных методов ансамблевого прогнозирования построена таблица 1.

Таблица 1 – Характеристики методов ансамблевого прогнозирования

| Метод | Преимущества | Недостатки |
|---|---|---|
| Взвешенное среднее | Быстрее реагирует на последние изменения в данных, что делает его полезным для определения тенденций и сдвигов спроса.<br>Вес можно настроить, чтобы отразить важность различных точек данных, что позволяет делать индивидуальные прогнозы на основе конкретных бизнес-потребностей.<br>По сравнению с простыми скользящими средними, взвешенные средние уменьшают эффект отставания, обеспечивая более точное отражение текущих условий.<br>Может привести к более точным прогнозам, особенно на нестабильных рынках. | Расчет сложнее, чем простых средних, для чего могут потребоваться более сложные инструменты или программное обеспечение.<br>Выбор весов может быть субъективным, а неправильное взвешивание может привести к вводящим в заблуждение прогнозам.<br>Чувствительность к выбросам. |

Продолжение таблицы 1

| Метод | Преимущества | Недостатки |
|---|---|---|
| Стекинг | Объединяя прогнозы из нескольких моделей, стекирование может использовать сильные стороны каждой модели, что часто приводит к лучшей точности по сравнению с отдельными моделями.<br>Может обеспечить более надежные прогнозы, поскольку снижает риск переобучения для определенного набора данных.<br>Позволяет прогнозисту использовать различные методы моделирования и их уникальные сильные стороны.<br>Метамодель может научиться лучше обобщать разные наборы данных, что может быть особенно полезно в динамических средах. | Усложняет процесс моделирования.<br>Обучение нескольких моделей и метамодели может быть ресурсоемким, требуя больше вычислительной мощности и времени, особенно при больших наборах данных.<br>Есть риск переобучения.<br>ложность стекированных моделей может сделать их менее интерпретируемыми, чем более простые модели, |
| Бутстрэп и бэггинг | Качественная оценка распределения выборки.<br>Можно применять к широкому спектру статистических задач и моделей.<br>Может обеспечить более надежные оценки эффективности модели.<br>Позволяет строить доверительные интервалы для прогнозов.<br>Не требует предположения о нормальности, что может быть выгодно при работе с ненормально распределенными данными. | Вычислительно интенсивный.<br>Есть риск переобучения:<br>Ограничен размером исходной выборки. |

Продолжение таблицы 1

| Метод | Преимущества | Недостатки |
|---|---|---|
| Градиентный бустинг | Обеспечивает высокую точность прогнозирования и эффективен при выявлении сложных закономерностей в данных.<br>Можно использовать как для задач регрессии, так и для задач классификации.<br>Может естественным образом обрабатывать пропущенные значения в наборе данных.<br>Дает представление о важности признаков.<br>Некоторые реализации оптимизированы для скорости и использования памяти. | Существует риск переобучения.<br>Обучение может быть вычислительно дорогим и трудоемким, особенно с большими.<br>Сложность.<br>Чувствительность к гиперпараметрам.<br>Требует больше данных.<br>Не подходит для всех типов данных. |

Как следует из таблицы, метод градиентного бустинга обладает рядом преимуществ, что делает его популярным выбором для решения задач прогнозирования данных.

Это подтверждает актуальность решения задачи исследования и реализации алгоритма градиентного бустинга для прогнозирования данных.

Выводы по главе 1

Методы ансамблевого прогнозирования сильно разрознены в разных дисциплинах и часто служат разным целям. Полный потенциал ансамблевого прогнозирования и многие из его преимуществ н еще не реализованы.

Метод градиентного бустинга обладает рядом преимуществ, что подтверждает актуальность решения задачи исследования и реализации алгоритма градиентного бустинга для прогнозирования данных.

## Глава 2 Обзор и анализ алгоритмов градиентного бустинга

Рассмотрим свойства популярных алгоритмов градиентного бустинга XGboost, Catboost и LightGBM.

### 2.1 Алгоритм XGBoost

XGBoost (Extreme Gradient Boosting) – это оптимизированная распределенная библиотека градиентного бустинга, разработанная для обеспечения высокой эффективности, гибкости и переносимости.

«Это ансамблевый метод обучения, который объединяет предсказания нескольких слабых моделей для получения более сильного предсказания.

XGBoost стал одним из самых популярных и широко используемых алгоритмов машинного обучения благодаря своей способности обрабатывать большие наборы данных и достижению передовой производительности во многих задачах машинного обучения, таких как классификация и регрессия» [22].

Схема алгоритма XGBoost показана на рисунке 3.

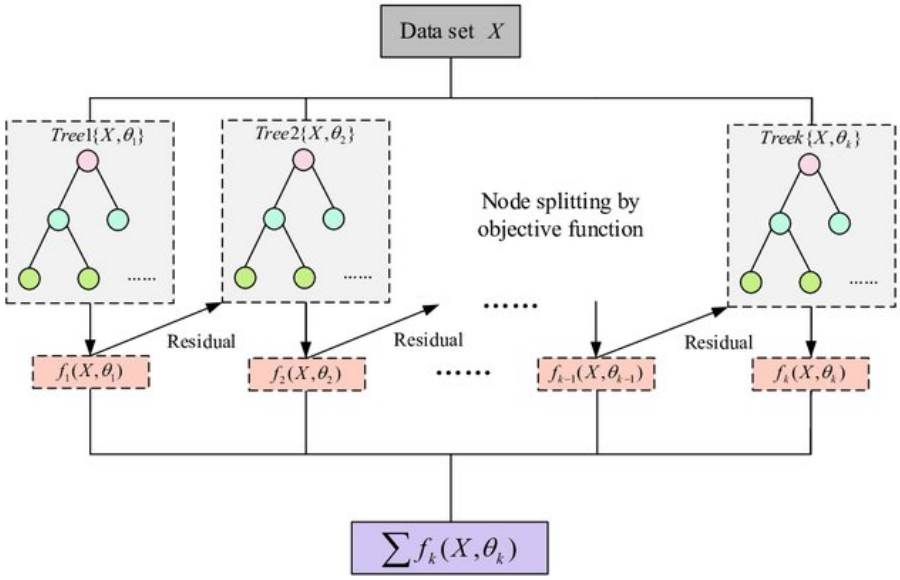


Рисунок 3 – Схема работы алгоритма XGBoost

«Рассмотрим для примера задачу регрессии и введем следующие обозначения:

– $\{x_i, y_i\}_{i=1}^{N}$ – *обучающая выборка;*
– $K$ – количество деревьев в ансамбле;
– $f_k: x \to y$ – $k$-е дерево ансамбля как функция;
– $F: x \to y$ – весь ансамбль как функция;
– $loss: y_{true} \times y_{pred} \to R$ – выбранная пользователем функция потерь;
– $T_k$ – количество листьев в k-м дереве ансамбля;
– $w_k$ – вектор, составленный из выходных значений на всех листьях $k$-го дерева» [11].

В XGBoost ответы суммируются по всем деревьям ансамбля (12):

$$F(x) = \sum_{k=1}^{K} f_k(x) \tag{12}$$

Суммарная функция потерь в XGBoost выглядит следующим образом (13):

$$total_{loss} = \sum_{i=1}^{N} loss\big(y_i, F(x_i)\big) + \gamma \sum_{k=1}^{K} T_k + \frac{1}{2}\lambda \sum_{k=1}^{K} \|w_k\|^2, \tag{13}$$

где γ, λ – гиперпараметры.

«Первое слагаемое – это основная функция потерь, второе слагаемое штрафует деревья за слишком большое количество листьев, третье слагаемое за слишком большие предсказания. Третье слагаемое является нетипичным в машинном обучении. Оно обеспечивает то, что каждое дерево вносит минимальный вклад в результат. Функция потерь используется при построении каждого следующего дерева, то есть функция потерь оптимизируется по параметрам лишь последнего дерева, не затрагивая предыдущие.

Для минимизации total_loss используется метод второго порядка, то есть рассчитываются не только производные, но и вторые производные функции потерь по предсказаниям предыдущих деревьев.

Одной из ключевых особенностей XGBoost является эффективная обработка пропущенных значений, что позволяет обрабатывать реальные данные с пропущенными значениями без необходимости значительной предварительной обработки. Кроме того, XGBoost имеет встроенную поддержку параллельной обработки, что позволяет обучать модели на больших наборах данных за разумное время» [22].

XGBoost можно использовать в различных приложениях, включая соревнования Kaggle, системы рекомендаций и прогнозирование рейтинга кликов, среди прочего. Он также обладает широкими возможностями настройки и позволяет выполнять тонкую настройку различных параметров модели для оптимизации производительности.

## 2.2 Алгоритм CatBoost

CatBoost (Categorical Boosting) – это алгоритм, который предназначен для эффективной работы в задачах классификации и регрессии [17].

Способность CatBoost обрабатывать категориальные переменные без необходимости ручного кодирования является одним из его основных преимуществ. Он использует метод, известный как Ordered Boosting, для решения проблем, с которыми сталкиваются категориальные признаки, такие как большая кардинальность. Это позволяет CatBoost автоматически обрабатывать категориальные данные, экономя время и усилия пользователя.

Основная идея CatBoost заключается в его способности эффективно и действенно обрабатывать категориальные признаки.

Рассмотрим математическое описание алгоритма CatBoost.

Пусть имеется обучающий датасеи с $N$ образцами и $M$ признаками, где каждый образец обозначается как $(x_i, y_i)$, где $x_i$ – это вектор из $M$ признаков, а

$y_i$ – соответствующая целевая переменная.

CatBoost стремится обучить функцию $F(x)$, которая предсказывает целевую переменную вида (14):

$$F(x) = F_0(x) + \sum_{m=1}^{M} \sum_{n=1}^{N} f_m(x_i), \tag{14}$$

где $F(x)$ представляет собой общую функцию прогнозирования, которую CatBoost стремится изучить. Она берет входной вектор $x$ и прогнозирует соответствующую целевую переменную $y$;

$F_0(x)$ – базовый прогноз;

$\sum m,M$ – суммирование по ансамблю деревьев. $M$ – общее количество деревьев в ансамбле;

$\sum n,N$ – суммирование по обучающим образцам. $N$ – общее количество обучающих образцов;

$f_m(x_i)$ – прогноз $m$-го дерева для i-го обучающего образца.

Псевдокод алгоритма CatBoost показана на рисунке 4.

**input :** $\{(x_i, y_i)\}_{i=1}^{n}, I, \alpha, L, s$

$\sigma_r \leftarrow$ random permutation of $[1, n]$ for $r = 0 \dots s$;
$M_0(i) \leftarrow 0$ for $i = 1 \dots n$;
**for** $j \leftarrow 1$ **to** $\lceil \log_2 n \rceil$ **do**
  $M_{r,j}(i) \leftarrow 0$ for $r = 1 \dots s,\ i = 1 \dots 2^{j+1}$;
**for** $t \leftarrow 1$ **to** $I$ **do**
  $T_t, \{M_r\}_{r=1}^{s} \leftarrow BuildTree(\{M_r\}_{r=1}^{s}, \{(x_i, y_i)\}_{i=1}^{n}, \alpha, L, \{\sigma_i\}_{i=1}^{s})$;
  $leaf_0(i) \leftarrow GetLeaf(x_i, T_t, \sigma_0)$ for $i = 1 \dots n$;
  $grad_0 \leftarrow CalcGradient(L, M_0, y)$;
  **foreach** *leaf j* **in** $T_t$ **do**
    $b_j^t \leftarrow -\text{avg}(grad_0(i)$ for $i : leaf_0(i) = j)$;
  **for** $i = 1 \dots n$ **do**
    $M_0(i) \leftarrow M_0(i) + \alpha b^t_{leaf_0(i)}$
**return** $F(x) = \sum_{t=1}^{I} \sum_j \alpha b_j^t \mathbb{1}_{GetLeaf(x,T_t,ApplyMode)=j}$;

Рисунок 4 – Псевдокод алгоритма CatBoost

Алгоритм CatBoost реализует реализует новую технику Ordered Boosting, которая генерирует числовое представление путем перестановки категориальных переменных. Этот метод сохраняет информацию о категории, позволяя модели использовать мощный метод градиентного усиления.

### 2.3 Алгоритм LightGBM

«Алгоритм LightGBM (Light Gradient Boosting Machine) – это реализация градиентного бустинга с открытым исходным кодом, разработанная для того, чтобы быть эффективной и даже, возможно, более эффективной, чем другие реализации» [15].

Основные важные особенности LightGBM:

- односторонняя выборка на основе градиента (Gradient-based One-Side Sampling - GOSS): уделяет больше внимания недостаточно обученным точкам данных;
- exclusive Feature Bundling (EFB): эффективное представление разреженных функций, таких как функции (признаков) полученных с помощью one-hot-encoded features (для преобразования категориальных признаков в числовые);
- идея: при построении деревьев мы можем уделять больше внимания (важности) недостаточно обученным точкам данных;
- мы можем использовать градиенты как некую меру важности;
- небольшой градиент: означает небольшую ошибку, точка данных хорошо изучена. (не важный);
- большой градиент: означает большую ошибку, точка данных плохо изучена. (важный).

Рассмотрим описание алгоритма (с параметрами a и b):

- отсортируйте данные по абсолютному значению градиента;
- сохраняйте верхние a × 100% выборок данных (большие градиенты);
- произвольно выберите b × 100% от остальных данных (небольшие

градиенты);

– усильте небольшие градиенты путем умножения $(1-a)/b$ при вычислении информационного выигрыша.

Схема работы алгоритма LightGBM показана на рисунке 5 [4].

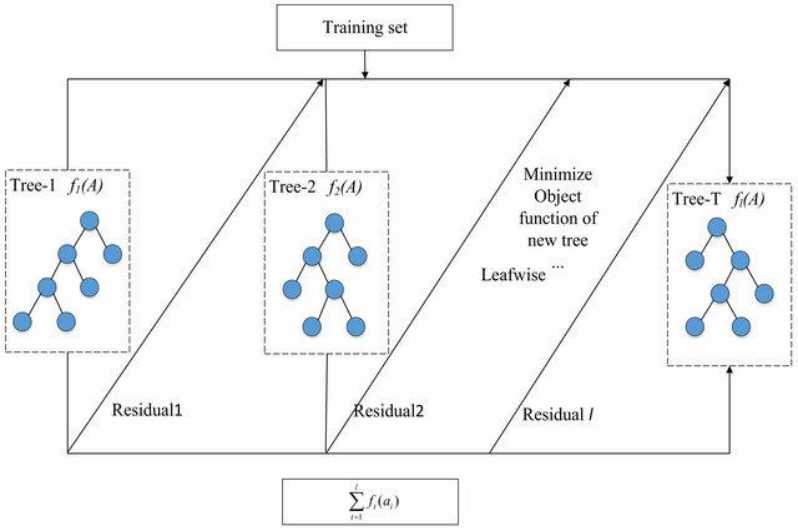


Рисунок 5 – Схема работы алгоритма LightGBM

Для сравнения характеристик алгоритмов градиентного бустинга построена таблица 2.

Таблица 2 – Характеристики алгоритмов градиентного бустинга

| Алгоритм | Преимущества | Недостатки |
|---|---|---|
| XGBoost | Высокая производительность.<br>Скорость и масштабируемость.<br>Гибкость.<br>Регуляризация.<br>Предоставляет информацию о важности признаков, помогая в интерпретации модели и выборе признаков.<br>Параллельная обработка: поддерживает параллельные вычисления, значительно сокращая время обучения. | Сложность.<br>Ресурсоемкость.<br>Настройка гиперпараметров.<br>Риск переобучения. |

Продолжение таблицы 2

| | | |
|---|---|---|
| CatBoost | Обработка категориальных признаков: изначально поддерживает категориальные признаки без необходимости в обширной предварительной обработке (например, кодирование one-hot), что делает его удобным для использования с наборами данных со многими категориальными переменными.<br>Устойчивость к переобучению: Включает методы для уменьшения переобучения, такие как упорядоченное усиление, что помогает улучшить обобщение на невидимых данных.<br>Высокая производительность: часто достигает конкурентоспособной точности и производительности, аналогичной или лучшей, чем другие алгоритмы усиления, такие как XGBoost и LightGBM.<br>Простота использования: библиотека разработана так, чтобы быть удобной для пользователя, с простым API, который позволяет пользователям быстро реализовывать модели без глубоких знаний в настройке.<br>Автоматический выбор признаков: может автоматически выбирать и преобразовывать признаки, что упрощает работу со сложными наборами данных.<br>Быстрое обучение: в целом эффективен и может эффективно обрабатывать большие наборы данных.<br>Интерпретируемость: предоставляет инструменты для интерпретации модели, включая метрики важности признаков, которые могут помочь пользователям понять решения модели. | Потребление памяти: может потреблять значительный объем памяти, особенно с большими наборами данных, что может быть проблемой в средах с ограниченными ресурсами.<br>Более длительное время обучения: может занять больше времени по сравнению с другими алгоритмами повышения, особенно если он не оптимизирован должным образом.<br>Ограниченная поддержка сообщества: несмотря на рост, сообщество и экосистема вокруг CatBoost могут быть не такими обширными, как у более известных библиотек, таких как XGBoost или LightGBM.<br>Сложность настройки гиперпараметров: имеет несколько гиперпараметров, которые требуют настройки для оптимальной производительности, что может быть сложным для новичков.<br>Меньше гибкости с пользовательскими функциями потерь. |

Продолжение таблицы 2

| | | |
|---|---|---|
| LightGBM | Скорость и эффективность: разработан так, чтобы быть быстрее других фреймворков градиентного бустинга.<br>Масштабируемость: может эффективно обрабатывать большие наборы данных с миллионами экземпляров и признаков, что делает его подходящим для приложений с большими данными.<br>Поддержка категориальных признаков: имеет встроенную поддержку категориальных признаков, что позволяет автоматически обрабатывать их без необходимости обширной предварительной обработки.<br>Эффективность памяти: использует более эффективный подход к памяти по сравнению с некоторыми другими алгоритмами бустинга, что делает его подходящим для сред с ограниченными ресурсами памяти.<br>Параллельное и распределенное обучение.<br>Гибкость: предоставляет множество гиперпараметров для настройки, что позволяет пользователям оптимизировать модель для конкретных задач.<br>Важность признаков: предлагает встроенные инструменты для оценки важности признаков, которые могут помочь в интерпретации модели и выборе признаков. | Сложность настройки гиперпараметров.<br>Риск переобучения: существует риск переобучения, особенно на небольших наборах данных или если модель не регуляризована должным образом.<br>Чувствительность к зашумленным данным.<br>Меньшая интерпретируемость.<br>Ограниченная поддержка пользовательских функций потерь.<br>По сравнению с некоторыми другими библиотеками, обладает меньшей гибкостью при реализации пользовательских функций потерь, что может ограничить ее применимость в определенных специализированных сценариях. |

Выбор алгоритма градиентного бустинга зависит от конкретной задачи, объема данных и требований к производительности.

Оба алгоритма имеют свои сильные и слабые стороны, и их использование может варьироваться в зависимости от контекста задачи.

Выводы к главе 2

Градиентный бустинг – это мощный метод машинного обучения, который включает в себя несколько популярных реализаций, таких как XGBoost, CatBoost и LightGBM. Эти алгоритмы отличаются по скорости, точности и способам обработки данных.

XGBoost известен своей высокой производительностью и эффективностью, но у него есть свои ограничения, особенно при работе с категориальными признаками.

CatBoost также обеспечивает значительный потенциал производительности, поскольку он работает замечательно хорошо с параметрами по умолчанию, значительно улучшая производительность при настройке.

LightGBM разработан так, чтобы быть быстрее других фреймворков градиентного бустинга.

Выбор алгоритма градиентного бустинга зависит от конкретной задачи, объема данных и требований к производительности. Все алгоритмы имеют свои сильные и слабые стороны, и их использование может варьироваться в зависимости от контекста задачи.

# Глава 3 Реализация и тестирование алгоритма градиентного бустинга для прогнозирования данных

## 3.1 Выбор средства разработки программы

«Для реализации алгоритма поиска ассоциативных правил в большом наборе данных используем язык Python.

Для выбора среды разработки сравним характеристики двух сред разработки: Jupyter Notebook и Replit.com» [7].

Рассмотрим возможности среды Jupyter Notebook.

«Jupyter Notebook – это веб-приложение с открытым исходным кодом, которое позволяет пользователям создавать и обмениваться документами, содержащими живой код, уравнения, визуализации и повествовательный текст. Оно широко используется в науке о данных, машинном обучении, научных вычислениях и академических исследованиях благодаря своей способности сочетать выполнение кода с расширенным форматированием текста (рисунок 6)» [19].


Рисунок 6 – Главная страница среды Jupyter Notebook

Вот некоторые ключевые особенности и аспекты Jupyter Notebook:

- интерактивное кодирование: пользователи могут писать и выполнять код в реальном времени, что особенно полезно для тестирования и отладки;
- «поддержка нескольких языков: хотя Jupyter изначально поддерживал Python, теперь он поддерживает множество языков программирования с помощью «ядер». К ним относятся R, Julia, Scala и другие» [19];
- поддержка форматированного текста: пользователи могут включать форматированный текст, изображения, ссылки, уравнения LaTeX и многое другое, что позволяет создавать исчерпывающую документацию вместе с кодом;
- визуализация данных: Jupyter Notebook может отображать визуализации непосредственно в документе с помощью таких библиотек, как Matplotlib, Seaborn и Plotly;
- совместное использование блокнотов: блокнотами можно легко делиться с другими в различных форматах, включая HTML, PDF и Markdown, или их можно делиться через такие платформы, как GitHub или JupyterHub;
- расширения и настройка: пользователи могут улучшить функциональность Jupyter с помощью различных расширений и плагинов, что позволяет создавать настраиваемые рабочие процессы и функции;
- интеграция с библиотеками науки о данных: Jupyter обычно используется с популярными библиотеками науки о данных, такими как NumPy, Pandas и TensorFlow, что делает его основным продуктом в сообществе науки о данных.

Рассмотрим возможности среды Replit.com (рисунок 7) [21].

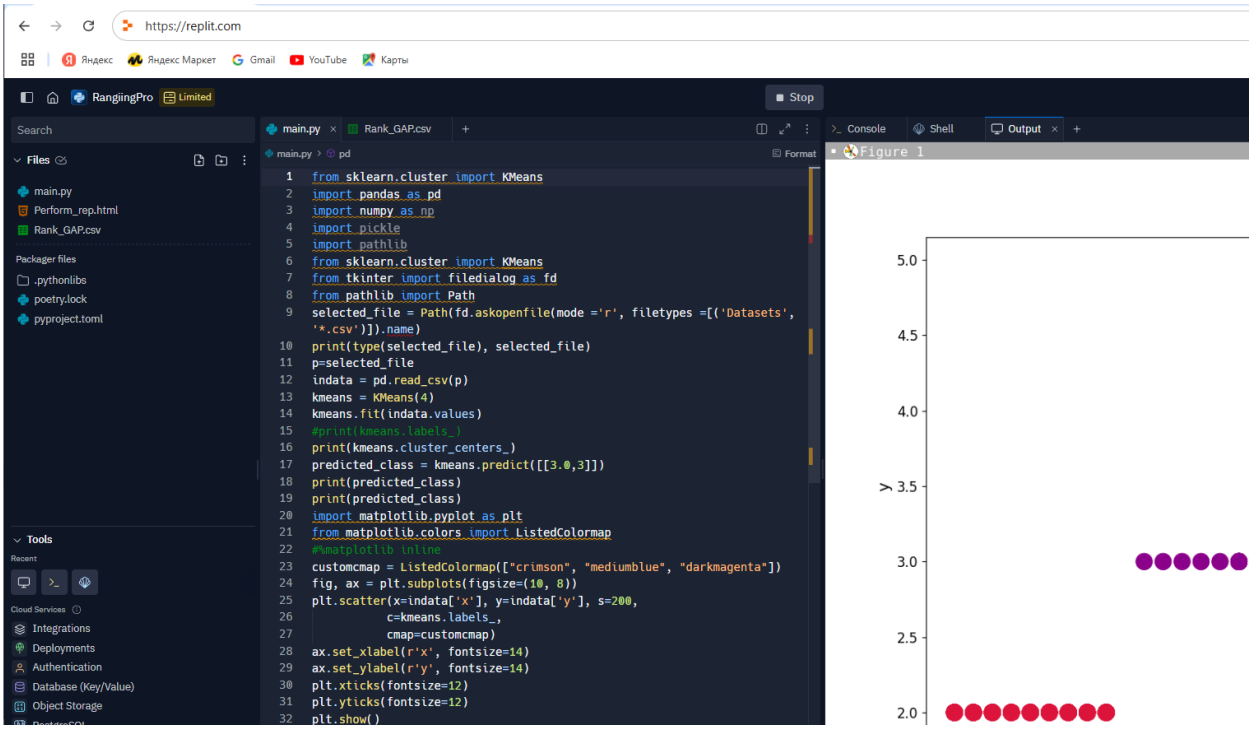


Рисунок 7 – Главная страница среды Replit.com

Replit.com – это интегрированная среда разработки (IDE), которая позволяет пользователям писать, запускать и совместно работать над кодом на более чем 50 языках программирования. Она предоставляет удобную облачную платформу для создания проектов, что делает ее доступной как для новичков, так и для опытных разработчиков.

Replit поддерживает совместную работу в реальном времени, что позволяет нескольким пользователям работать над одним проектом одновременно.

Основные особенности Replit:

– поддержка нескольких языков: Replit поддерживает широкий спектр языков программирования, включая Python, JavaScript, Java, HTML и CSS и многие другие. Эта универсальность делает его подходящим для различных проектов по кодированию;
– совместная работа в реальном времени: пользователи могут приглашать других людей для редактирования и совместной работы

над своими проектами в реальном времени, что улучшает возможности командной работы и обучения;

- интегрированная среда разработки: платформа включает в себя мощную IDE с такими функциями, как подсветка синтаксиса, инструменты отладки и контроль версий, что позволяет пользователям эффективно писать и управлять своим кодом;
- мгновенное выполнение кода: пользователи могут мгновенно запускать свой код и видеть результаты в окне предварительного просмотра, что облегчает быструю разработку и тестирование.

Replit имеет активное сообщество, где пользователи могут делиться своими проектами, искать помощь и учиться друг у друга. Он также предлагает шаблоны и руководства, которые помогут новичкам начать работу.

Для сравнения рассмотренных сред программирования используем таблицу 3.

Таблица 3 – Сравнение характеристик сред Jupyter Notebook и Replit.com

| Характеристика | Jupyter Notebook | Replit.com |
|---|---|---|
| Интерфейс | Ориентирован на научные исследования, с возможностью добавления текста в формате Markdown, визуализаций и интерактивных графиков. Удобен для анализа данных и создания отчетов | «Предоставляет более традиционную среду разработки с редактором кода, терминалом и возможностью запуска приложений. Более ориентирован на разработку программного обеспечения и совместную работу» [21] |
| Установка и доступ | Обычно устанавливается локально на компьютере с использованием Anaconda или pip. Также доступен в облачных вариантах, таких как Google Colab. | Полностью облачная платформа, не требует установки. Доступен через веб-браузер, что позволяет легко начинать работу без необходимости настройки окружения |

Продолжение таблицы 3

| Характеристика | Jupyter Notebook | Replit.com |
|---|---|---|
| Совместная работа | Поддерживает совместную работу, но в основном через экспорт и обмен файлами (например, .ipynb). Для реального времени требуется использование сторонних решений, таких как JupyterHub или Google Colab | Предоставляет встроенные функции совместной работы, позволяя нескольким пользователям редактировать код одновременно в реальном времени. Это делает его идеальным для учебных целей и командных проектов |
| Цели использования | Идеален для анализа данных, машинного обучения, визуализации данных и научных исследований. Часто используется в образовательных целях для демонстрации концепций программирования и анализа | Более универсален и подходит для создания веб-приложений, игр, скриптов и других проектов. Хорошо подходит для обучения программированию и совместной разработки |
| Визуализация и графика | Поддерживает сложные визуализации с использованием библиотек, таких как Matplotlib, Seaborn и Plotly. Позволяет интегрировать графики непосредственно в документ | Основной фокус на коде и приложениях, хотя можно использовать библиотеки для графики, но возможности визуализации не так развиты, как в Jupyter |

По результатам сравнения целей использования и с учетом предпочтений разработчика выбираем среду разработки Jupyter Notebook.

### 3.2 Разработка и тестирование программы

«Для представления программной архитектуры используем диаграмму классов UML» [23].

«Для построения диаграмм UML использован онлайн-сервис Visual Paradigm» [24].

На рисунке 8 показана диаграмма классов градиентного бустинга на

основе алгоритма XGBoost [25].

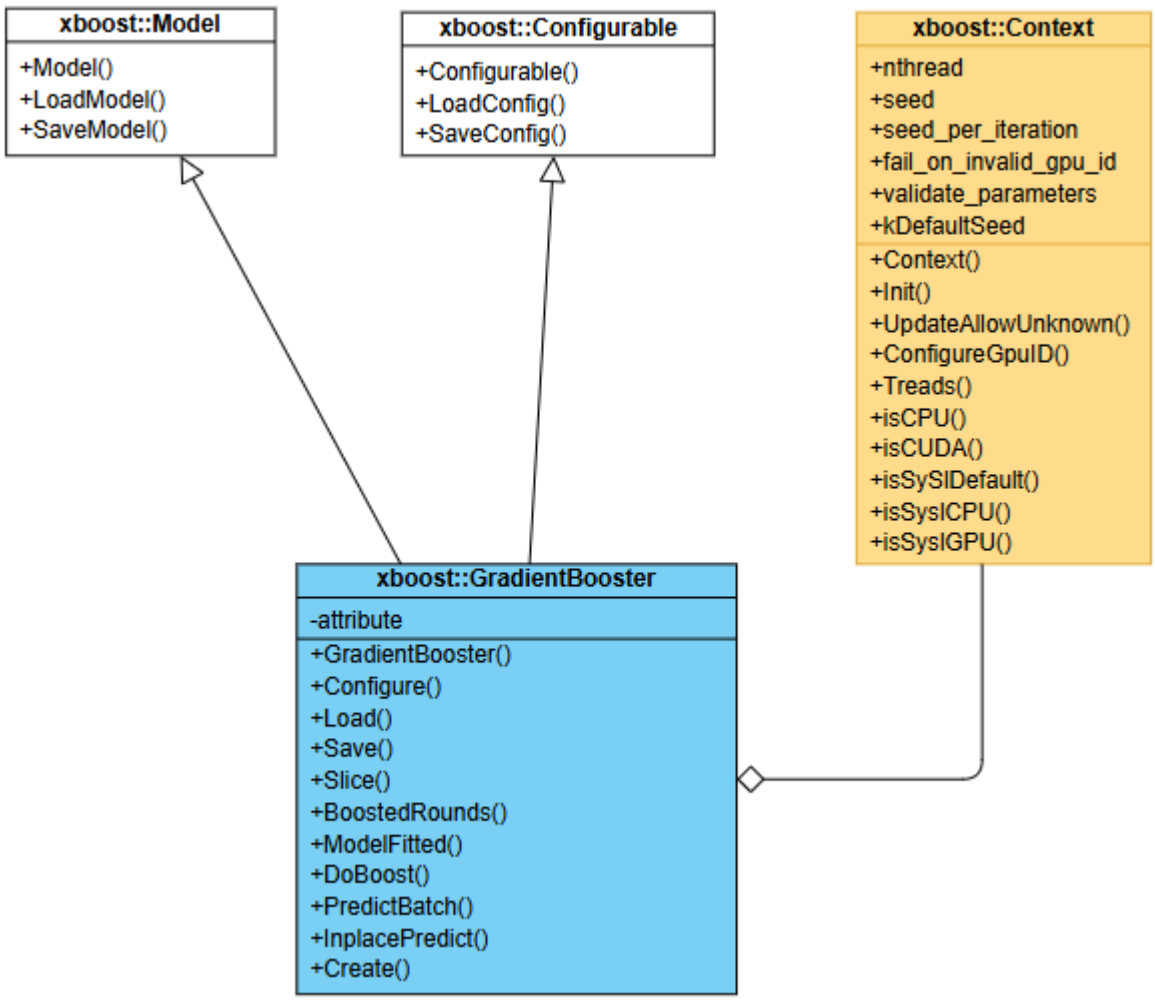


Рисунок 8 – Диаграмма классов градиентного бустинга на основе алгоритма XGBoost

Как следует из диаграммы класс GradientBooster является наследником классов Model и Configurable.

Класс Model – модель обучения для XGBoost.

Класс Configurable – конфигурация для XGBoost.

Этот класс также связан агрегацией с классом Context.

Класс Context – контекст выполнения для XGBoost. Содержит информацию, такую как потоки и устройство.

Операция PredictBatch() генерирует прогнозы для заданной матрицы признаков.

Разработана диаграмма компонентов программы градиентного бустинга, показанная на рисунке 9.

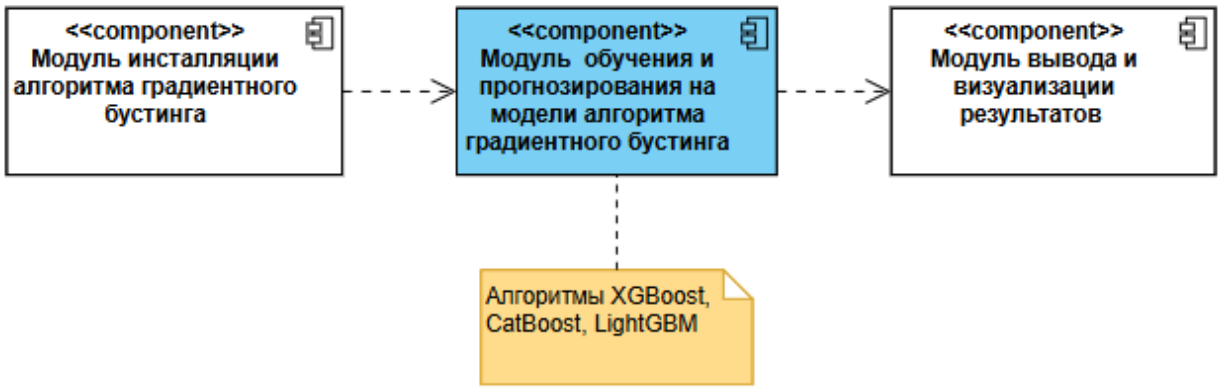


Рисунок 9 – Диаграмма компонентов программы градиентного бустинга

Рассмотрим работу градиентного бустинга на примере алгоритма XGBoost.

В качестве входного набора данных использован свободно распространяемый датасет Cancer Patients Data в формате CSV [10].

В этом наборе данных вы найдете информацию о сотнях онкологических больных и об их образе жизни.

Набор данных содержит уникальный идентификатор для каждого пациента, характеристики его образа жизни и уровень риска (Level) заболевания раком.

При разработке программы использована библиотека scikit-learn

«Scikit-learn (Sklearn) – самая полезная и надежная библиотека для машинного обучения на Python. Она предоставляет набор эффективных инструментов для машинного обучения и статистического моделирования, включая классификацию, регрессию, кластеризацию и уменьшение размерности через интерфейс согласованности на Python. Эта библиотека, которая в основном написана на Python, построена на NumPy, SciPy и Matplotlib» [1].

На рисунке 10 показаны код и результаты инсталляции алгоритма XGBoost и загрузки входного датасета.

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\anaconda3\lib\site-packages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\anaconda3\lib\site-packages (from xgboost) (1.11.1)
Note: you may need to restart the kernel to use updated packages.
```

```
import numpy as np
import pandas as pd
import xgboost
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_log_error
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
```

```
dataset=pd.read_csv('cancer patient data sets.csv')
dataset.head()
```

| | index | Patient Id | Age | Gender | Air Pollution | Alcohol use | Dust Allergy | OccuPational Hazards | Genetic Risk | chronic Lung Disease | ... | Fatigue | Weight Loss | Shortness of Breath | Wheezing | Swallowing Difficulty | Clubbing of Finger Nails |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | P1 | 33 | 1 | 2 | 4 | 5 | 4 | 3 | 2 | ... | 3 | 4 | 2 | 2 | 3 | 1 |
| **1** | 1 | P10 | 17 | 1 | 3 | 1 | 5 | 3 | 4 | 2 | ... | 1 | 3 | 7 | 8 | 6 | 2 |
| **2** | 2 | P100 | 35 | 1 | 4 | 5 | 6 | 5 | 5 | 4 | ... | 8 | 7 | 9 | 2 | 1 | 4 |
| **3** | 3 | P1000 | 37 | 1 | 7 | 7 | 7 | 7 | 6 | 7 | ... | 4 | 2 | 3 | 1 | 4 | 5 |
| **4** | 4 | P101 | 46 | 1 | 6 | 8 | 7 | 7 | 7 | 6 | ... | 3 | 2 | 4 | 1 | 4 | 2 |

5 rows × 26 columns

Рисунок 10 – Код и результаты инсталляции алгоритма XGBoost и загрузки входного датасета

Далее вычисляем количество уникальных значений признака Level во входном датасете (рисунок 11).

```
1  dataset['Level'].value_counts()
```

```
Level
High      365
Medium    332
Low       303
Name: count, dtype: int64
```

Рисунок 11 – Вычисление количества уникальных значений признака Level во входном датасете

Для упрощения вычислений удаляем из датасета ненужные признаки и снова вычисляем количества уникальных значений признака Level (рисунок 12).

```
1 #удаление ненужных признаков
2 col_drop=['index','Patient Id','Air Pollution','Alcohol use','Dust Allergy','OccuPational Hazards','chronic Lung Disease','F
3 s=dataset.drop(col_drop,axis=1)
4 s['Level'].value_counts()
```

```
Level
High      365
Medium    332
Low       303
Name: count, dtype: int64
```

Рисунок 12 – Вычисление количества уникальных значений признака Level во входном датасете после удаления ненужных признаков

Следует напомнить, что алгоритмы машинного обучения, как мы знаем, не умеют работать с категориальными данными, выраженными с помощью строковых значений.

Для этого строки необходимо закодировать (encode) числами (рисунок 13) [8].

```
enc = OrdinalEncoder()
le = LabelEncoder()
ob=s.select_dtypes(include=['object'])

for colsn in ob:
    s[colsn] = le.fit_transform(s[colsn].astype(str))

```

```
s['Level'].value_counts()
```

```
Level
0    365
2    332
1    303
Name: count, dtype: int64
```

```
cc= s.select_dtypes(include=['object']).columns
cc
```

```
Index([], dtype='object')
```

```
s.head()
```

| | Age | Gender | Genetic Risk | Balanced Diet | Obesity | Smoking | Passive Smoker | Chest Pain | Coughing of Blood | Weight Loss | Shortness of Breath | Wheezing | Frequent Cold | Dry Cough | Level |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 33 | 1 | 3 | 2 | 4 | 3 | 2 | 2 | 4 | 4 | 2 | 2 | 2 | 3 | 1 |
| **1** | 17 | 1 | 4 | 2 | 2 | 2 | 4 | 2 | 3 | 3 | 7 | 8 | 1 | 7 | 2 |
| **2** | 35 | 1 | 5 | 6 | 7 | 2 | 3 | 4 | 8 | 7 | 9 | 2 | 6 | 7 | 0 |
| **3** | 37 | 1 | 6 | 7 | 7 | 7 | 7 | 7 | 8 | 2 | 3 | 1 | 6 | 7 | 0 |
| **4** | 46 | 1 | 7 | 7 | 7 | 8 | 7 | 7 | 9 | 2 | 4 | 1 | 4 | 2 | 0 |

Рисунок 13 – Код и результат числового кодирования признаков датасета

Далее производим выделение целевого признака Level и разделение данных на обучающую и тестовую выборку (рисунок 14).

```
#разделение данных X и y - выделение целевого признака
X = s.drop('Level',axis=1)
Y = s['Level']
```

```
# разделение данных- обучение/тест
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)
X_train
```

| | Age | Gender | Genetic Risk | Balanced Diet | Obesity | Smoking | Passive Smoker | Chest Pain | Coughing of Blood | Weight Loss | Shortness of Breath | Wheezing | Frequent Cold | Dry Cough |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **508** | 38 | 2 | 7 | 2 | 4 | 1 | 2 | 4 | 3 | 7 | 6 | 5 | 3 | 4 |
| **435** | 49 | 1 | 5 | 6 | 7 | 2 | 3 | 4 | 8 | 7 | 9 | 2 | 6 | 7 |
| **132** | 53 | 2 | 5 | 6 | 7 | 2 | 3 | 4 | 8 | 7 | 9 | 2 | 6 | 7 |
| **556** | 64 | 1 | 7 | 7 | 7 | 7 | 8 | 7 | 7 | 6 | 5 | 7 | 3 | 1 |
| **252** | 35 | 2 | 6 | 6 | 4 | 6 | 8 | 7 | 6 | 5 | 4 | 6 | 6 | 5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **579** | 33 | 1 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 5 | 6 | 5 | 6 | 5 |
| **502** | 44 | 1 | 3 | 1 | 2 | 7 | 6 | 2 | 2 | 2 | 3 | 2 | 3 | 2 |
| **537** | 46 | 1 | 7 | 7 | 7 | 8 | 7 | 7 | 9 | 2 | 4 | 1 | 4 | 2 |
| **196** | 38 | 2 | 2 | 3 | 3 | 3 | 2 | 3 | 4 | 1 | 2 | 4 | 4 | 2 |
| **175** | 38 | 2 | 7 | 2 | 4 | 1 | 2 | 4 | 3 | 7 | 6 | 5 | 3 | 4 |

670 rows × 14 columns

Рисунок 14 – Разделение данных на обучающую и тестовую выборку

Выполняем обучение и прогнозирование на модели XGBoost (рисунок

15).

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)
model = xgboost.XGBClassifier()
model.fit(X_train, y_train)
X_train
```

| | Age | Gender | Genetic Risk | Balanced Diet | Obesity | Smoking | Passive Smoker | Chest Pain | Coughing of Blood | Weight Loss | Shortness of Breath | Wheezing | Frequent Cold | Dry Cough |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **508** | 38 | 2 | 7 | 2 | 4 | 1 | 2 | 4 | 3 | 7 | 6 | 5 | 3 | 4 |
| **435** | 49 | 1 | 5 | 6 | 7 | 2 | 3 | 4 | 8 | 7 | 9 | 2 | 6 | 7 |
| **132** | 53 | 2 | 5 | 6 | 7 | 2 | 3 | 4 | 8 | 7 | 9 | 2 | 6 | 7 |
| **556** | 64 | 1 | 7 | 7 | 7 | 7 | 8 | 7 | 7 | 6 | 5 | 7 | 3 | 1 |
| **252** | 35 | 2 | 6 | 6 | 4 | 6 | 8 | 7 | 6 | 5 | 4 | 6 | 6 | 5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **579** | 33 | 1 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 5 | 6 | 5 | 6 | 5 |
| **502** | 44 | 1 | 3 | 1 | 2 | 7 | 6 | 2 | 2 | 2 | 3 | 2 | 3 | 2 |
| **537** | 46 | 1 | 7 | 7 | 7 | 8 | 7 | 7 | 9 | 2 | 4 | 1 | 4 | 2 |
| **196** | 38 | 2 | 2 | 3 | 3 | 3 | 2 | 3 | 4 | 1 | 2 | 4 | 4 | 2 |
| **175** | 38 | 2 | 7 | 2 | 4 | 1 | 2 | 4 | 3 | 7 | 6 | 5 | 3 | 4 |

670 rows × 14 columns

Рисунок 15 – Обучение и прогнозирование на модели XGBoost

Создаем и оцениваем прогнозы по тестовым данным (рисунок 16).

```
# делаем прогнозы по тестовым данным
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

```
# оцениваем прогнозы
accuracy_1 = accuracy_score(y_test, predictions)
print("Accuracy XGB: %.2f%%" % (accuracy_1 * 100.0))
```

```
Accuracy XGB: 100.00%
```

```
features_xg =[33,1,3,3,3,2,2,3,3,5,6,5,6,5]
pred_y = model.predict([features_xg])
print('Predicted Class: %d' % pred_y[0])
```

```
Predicted Class: 2
```

Рисунок 16 – Создание и оценка прогнозов по тестовым данным

Понимание важности характеристик имеет решающее значение при

работе с моделями XGBoost. Это помогает определить, какие характеристики оказывают наиболее существенное влияние на прогнозы модели.

XGBoost предоставляет встроенную функцию plot_importance(), которая позволяет легко визуализировать важность функций  (рисунок 17) [3].

```
# строим график признаков
xgboost.plot_importance(model)
```

```
<Axes: title={'center': 'Feature importance'}, xlabel='F score', ylabel='Features'>
```

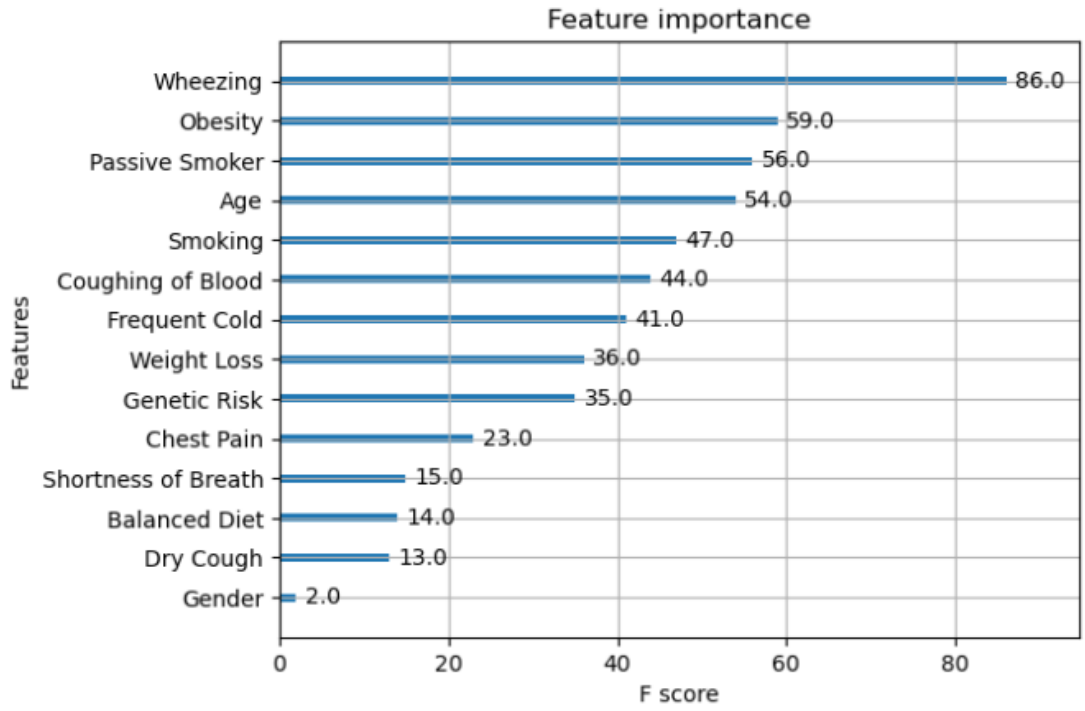


Рисунок 17 – График важности признаков

Таким образом, алгоритм XGBoost обеспечивает эффективное решение задачи прогнозирования данных на основе метода градиентного бустинга.

Выводы по главе 3

Алгоритм XGBoost обеспечивает эффективное решение задачи прогнозирования данных на основе метода градиентного бустинга.

## Заключение

Выполненные в рамках бакалаврской работы задачи представлены следующими основными результатами:

- произведена постановка задачи исследования алгоритма градиентного бустинга для прогнозирования данных. Как показал анализ, Методы ансамблевого прогнозирования сильно разрознены в разных дисциплинах и часто служат разным целям. Полный потенциал ансамблевого прогнозирования и многие из его преимуществ н еще не реализованы. Метод градиентного бустинга обладает рядом преимуществ, что подтверждает актуальность решения задачи исследования и реализации алгоритма градиентного бустинга для прогнозирования данных;
- дан обзор и произведен анализ алгоритмов градиентного бустинга для прогнозирования данных. Как показал анализ, градиентный бустинг – это мощный метод машинного обучения, который включает в себя несколько популярных реализаций, таких как XGBoost, CatBoost и LightGBM. Эти алгоритмы отличаются по скорости, точности и способам обработки данных. Выбор алгоритма градиентного бустинга зависит от конкретной задачи, объема данных и требований к производительности. Оба алгоритма имеют свои сильные и слабые стороны, и их использование может варьироваться в зависимости от контекста задачи;
- выполнены реализация и тестирование алгоритма градиентного бустинга для прогнозирования данных. Для реализации программ выбраны язык Python и среда Jupyter Notebook. Представлена работа градиентного бустинга на примере алгоритма XGBoost. XGBoost стал одним из самых популярных и широко используемых алгоритмов машинного обучения благодаря своей способности обрабатывать большие наборы данных и достижению передовой

производительности во многих задачах машинного обучения, таких как классификация и регрессия. Понимание важности характеристик имеет решающее значение при работе с моделями XGBoost. Это помогает определить, какие характеристики оказывают наиболее существенное влияние на прогнозы модели В качестве входного набора данных использован свободно распространяемый датасет Cancer Patients Data в формате CSV. При разработке программы использована библиотека scikit-learn.

Тестирование подтвердило, что алгоритм XGBoost обеспечивает эффективное решение задачи прогнозирования данных на основе метода градиентного бустинга.

Результаты бакалаврской работы могут представлять интерес для разработчиков и специалистов, занимающимся исследованием и реализацией алгоритмов градиентного бустинга для прогнозирования данных.

## Список используемой литературы и используемых источников


1. Библиотека Scikit-learn: как создать свой первый ML-проект [Электронный ресурс]. URL: https://skillbox.ru/media/code/biblioteka-scikitlearn-kak-sozdat-svoy-pervyy-mlproekt/ (дата обращения: 20.11.2024).

2. Доугерти К. Введение в эконометрику: Учебник. 3-е изд. / Пер. с англ. М.: ИНФРА-М, 2009. 445 с. (дата обращения: 20.11.2024).

3. Как использовать xgboost.plot_importance() [Электронный ресурс]. URL: https://xgboosting.com/how-to-use-xgboost.plot_importance/ дата обращения: 20.11.2024).

4. Как разработать ансамбль Light Gradient Boosted Machine (LightGBM) [Электронный ресурс]. URL: https://habr.com/ru/companies/skillfactory/articles/530594/ (дата обращения: 20.11.2024).

5. Методы сбора ансамблей алгоритмов машинного обучения: стекинг, бэггинг, бустинг [Электронный ресурс]. URL: https://habr.com/ru/articles/561732/ (дата обращения: 20.11.2024).

6. Сенько О.В. Задачи прогнозирования [Электронный ресурс]. URL: http://www.machinelearning.ru/wiki/images/e/ed/MOTP14_1.pdf (дата обращения: 20.11.2024).

7. Сузи Р. А. Язык программирования Python : учебное пособие. Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 350 с. URL: https://www.iprbookshop.ru/97589.html (дата обращения: 12.11.2024).

8. Числовое кодирование категориальных переменных [Электронный ресурс]. URL: https://teletype.in/@dt_analytic/YLOUyQcqSAL (дата обращения: 20.11.2024).

9. Caballar R., Stryker C. What is forecasting? [Электронный ресурс]. URL: https://www.ibm.com/think/topics/forecasting (дата обращения: 20.11.2024).

10. Cancer Patients Data [Электронный ресурс]. URL: https://www.kaggle.com/datasets/rishidamarla/cancer-patients-data?resource=download (дата обращения: 20.11.2024).

11. CatBoost, XGBoost и выразительная способность решающих деревьев [Электронный ресурс]. URL: https://habr.com/ru/companies/ods/articles/645887/#4 (дата обращения: 20.11.2024).

12. Ensemble Forecasting [Электронный ресурс]. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/ensemble-forecasting (дата обращения: 20.11.2024).

13. Forecasting mathematics: How to apply and solve the mathematical problems and equations of financial forecasting [Электронный ресурс]. URL: https://fastercapital.com/content/Forecasting-mathematics--How-to-apply-and-solve-the-mathematical-problems-and-equations-of-financial-forecasting.html#Introduction-to-Financial-Forecasting (дата обращения: 20.11.2024).

14. Gradient Boosting in ML [Электронный ресурс]. URL: https://www.geeksforgeeks.org/ml-gradient-boosting/ (дата обращения: 20.11.2024).

15. Guolin Ke et.al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

16. Hao Wu & David Levinson, 2022. "The Ensemble Approach to Forecasting: A Review and Synthesis," Working Papers 2021-10, University of Minnesota: Nexus Research Group.

17. How CatBoost algorithm works [Электронный ресурс]. URL: https://www.geeksforgeeks.org/catboost-algorithms/ (дата обращения: 20.11.2024).

18. Joaquín Amat Rodrigo, Javier Escobar Ortiz “Stacking ensemble of machine learning models to improve forecasting” [Электронный ресурс]. URL:

https://cienciadedatos.net/documentos/py52-stacking-ensemble-models-forecasting.html (дата обращения: 20.11.2024).

19. Jupyter Notebook: The Classic Notebook Interface [Электронный ресурс]. URL: https://jupyter.org/ (12.11.2024).

20. Random Forest Algorithm in Machine Learning [Электронный ресурс]. URL: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/ (дата обращения: 20.11.2024).

21. Replit.com [Электронный ресурс]. URL: https://replit.com/~(20.11.2024).

22. Rui Guo at el. Degradation state recognition of piston pump based on ICEEMDAN and XGBoost, ppl. Sci. 2020, 10, 6593.

23. Unified Modeling Language (UML) Diagrams [Электронный ресурс]. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата обращения: 20.11.2024).

24. Visual Paradigm: Online Productivity Suite [Электронный ресурс]. URL: https://online.visual-paradigm.com/ (дата обращения: 20.11.2024).

25. Xgboost [Электронный ресурс]. URL: https://xgboost.readthedocs.io/en/latest/dev/classxgboost_1_1GradientBooster.html (дата обращения: 20.11.2024).