

**СОДЕРЖДАНИЕ**

[**Введение** 4](#_Toc185530977)

[**1.ОБЩАЯ ИНФОРМАЦИЯ** 5](#_Toc185530978)

[1.2. Google Colab 5](#_Toc185530979)

[1.2. Ансамблевые модели 5](#_Toc185530980)

[**2. ПРАКТИЧЕСКАЯ ЧАСТЬ** 7](#_Toc185530981)

[2.1. Выбор датасета 7](#_Toc185530982)

[2.2. Исследование данных 7](#_Toc185530983)

[2.3. Визуальны просмотр данных 9](#_Toc185530984)

[2.4. Обработка данных 9](#_Toc185530985)

[2.5. Выбор моделей 10](#_Toc185530986)

[**3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ** 11](#_Toc185530987)

[3.1. Создание одиночных моделей 11](#_Toc185530988)

[3.2. Создание ансамблевых моделей 13](#_Toc185530989)

[3.3. Сравнение точности всех моделей 15](#_Toc185530990)

[**ЗАКЛЮЧЕНИЕ** 16](#_Toc185530991)

[**СПИСОК ЛИТЕРАТУРЫ** 17](#_Toc185530992)

# **Введение**

В нынешней финансовой индустрии кредитный скоринг играет важную роль в принятие решения о выдаче кредитов и оценке кредитоспособности заёмщиков. Очень важна точность и надежность прогнозирования чтобы избежать дефолта для минимизации рисков и обеспечения финансовой стабильности.

В основном для кредитного скоринга используются методы основанные на статических моделях и линейных регрессия, но при сложных и нелинейных зависимостях сталкиваются с ограничениями обработки в данных. В последнее время в машинном обучении ансамблевые методы показали себя как инструмент для повышения надёжности и точности прогнозирования в задачах кредитного скоринга.

Целью курсовой работы является анализ датасета и построения ансамблевой модели для решения поставленной задачи.

Для этого нам предстоит ознакомиться с платформой, на которой будет выполняться поставленная задача, исследовать и провести обработку данных преобразовав категориальные значения в числовые, создание и оценка моделей для более высокой точности, обучение моделей и сравнение результатов.

# **1.ОБЩАЯ ИНФОРМАЦИЯ**

## 1.2. Google Colab

Google Colab – бесплатная платформа для работы на языке Python прямо в браузере для повышения навыков и исследований в машинном обучении. Его не нужно настраивать, нужно лишь подключение к интернету, все работы сохраняются в облаке Google, можно так же работать совместно с людьми и делиться результатами.

Код можно прописывать в отдельных ячейках, это дает возможность выполнять код по порядку и просматривать каждый блок на наличие ошибок или результатов, а его интерфейс очень схож с Jupyter Notebook. Colab очень прост в использовании, у него понятный интерфейс, что делает его доступным для новый пользователей. В нем можно использовать огромное количество разных библиотек языка Python, такие как NumPy, pandas, Matpotlib, sklearn и другие.

Для своей работы я решил выбрать именно эту платформу, так как в ней очень удобно строить графики, диаграммы, изучать и работать с набором данных.

Google Colab — это мощный и удобный инструмент для работы с Python в облаке. Он подходит для тех, кто хочет экспериментировать, изучать и разрабатывать с анализом данных и машинным обучением в целом, особенно с бесплатным доступом ко всем вычислительным ресурсам.

## 1.2. Ансамблевые модели

Ансамблевые модели – одна из техник машинного обучения, которые комбинируют несколько моделей для более точного прогноза, потому что несколько моделей превосходит в точности любую другую одну модель. Особенно хорошо они себя показывают, когда базовые модели разнообразны, иными словами допускают разные ошибки. Как итог базовые модели объединяются в итоговый прогноз при помощи разных типов (Bagging, Boosting, Voting и тд). В своей работе я использовал Bagging и Voting, немного о них.

Bagging – техника основанная на создание множества моделей с использованием случайных подвыборок данных. Основная идея в том, что объединяя их предсказания, увеличивается точность и снижается разброс.  
В своей работе я применил Bagging при помощи RandomForest.

Voting – метод который объединяет прогнозы сразу нескольких моделей и выбирает ту, которая получила большее количество голосов.

Преимущества ансамблевых методов: У ансамблей более высокая точность, они более устойчивы к переобучению, так как ошибки моделей (отдельных) компенсируют друг друга, они универсальны, иными словами могут использоваться с различными типами моделей, а так же они хорошо справляются с данными в которых есть нелинейности.

Недостатки ансамблевых методов: Качество ансамбля влияет на качество базовых моделей, иными словами если базовые модели плохие, то ансамбль будет не сильно лучше.

# **2. ПРАКТИЧЕСКАЯ ЧАСТЬ**

## 2.1. Выбор датасета

Первым делом, нужно выбрать подходящий датасет с нужным объемом данных. Я выбрал bank.csv, он состосит из 16 колонок и 4502 строк.

Ссылка на датасет: <https://www.kaggle.com/datasets/kapturovalexander/bank-credit-scoring>

## 2.2. Исследование данных

Данные обязательно нужно обработать, так как в них могут содержаться ошибки, пропуски или аномалии. Данные могут быть разного формата, поэтому нужно привести данные в один формат, это простит работу и обучении модели.

В моем датасете нету NaN значений, что упрощает обработку данных.

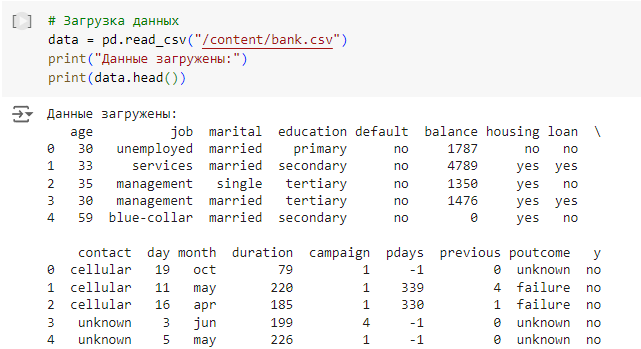


Рисунок 1. Исходные данные

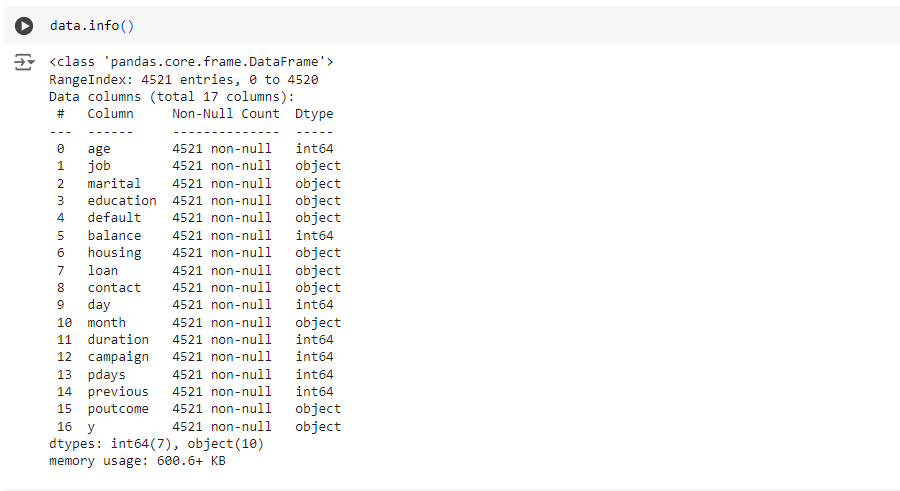


Рисунок 2. Формат данных

В таблице мы наблюдаем два формата данных int64 и object. Данные типа int64 представляют из себя целое число, оно может быть отрицательным и положительным. Данные типа object это формат для хранения строковых данных. Что бы обучить модель, предстоит изменить все строчные данные, на числовые.

## 2.3. Визуальны просмотр данных

Построив корреляционную матрицу (рисунок 3), видно представление корреляций между переменными, что показывает какие переменные связаны друг с другом.

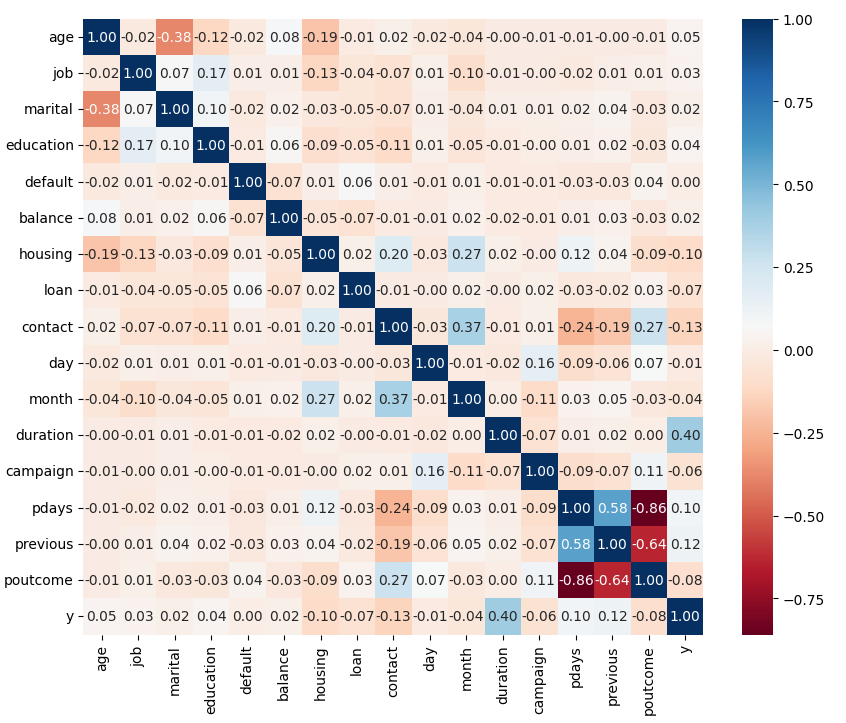


Рисунок 3. Корреляционная матрица

Практическая чать

## 2.4. Обработка данных

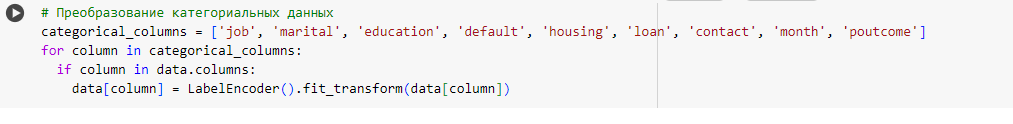


Рисунок 4. Категориальные данные

Список categorical\_columns содержащий названия столбцов, в этих столбцах могут содержаться текстовые значения, цикл for перебирает каждый столбец в categorical\_columns, а внутри самого цикла выполняется проверка о том, существует ли тот или иной столбец. LebelEncoder преобразует уникальные категории в целое число, fit находит уникальные категории и присваивает им числовые метки, transform заменяет оригинальные значения на числовые метки и результат присваивается обратно в data[column], заменяя тем самым текстовые значения на числовые.

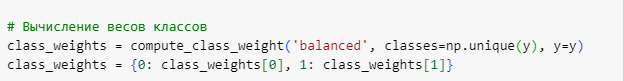


Рисунок 5. Балансировка классов

Функция возвращает массив весов классов, где каждый вес соответствует классу и создаем словарь.

## 2.5. Выбор моделей

В своей работе я решил создать три модели: RandomForest, GradientBoosting и XGBoost.

RandomForest – ансамблевый метод, который для решений использует множество деревьев и улучшает устойчивость и точность модели, обучая каждое дерево по случайной подвыборке данных.

Преимущества: Устойчивость к переобучению, обрабатывает категориальные и числовые данные, хорошо работает на больших наборах данных.

Недостатки: Может быть медленным при предсказании, если количество деревьев большое.

Gradient Boosting – ансамблевый метод, который добавляет деревья последовательно, каждое новое дерево добавляется для исправления ошибок которые допустили предыдущие деревья, так же использует градиентный спуск чтобы минимизировать функции потерь.

Преимущества: Высокая точность, работает с пропущенными значениями и гибкость в выборе функций потерь.

Недостатки: Сложен в настройке и требует больше времени на обучение.

XGBooost – улучшенная версия градиентного бустинга, он добавляет дополнительные методы оптимизации и регуляризации, что делает его более быстрым и эффективным.

# **3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ**

## 3.1. Создание одиночных моделей



Рисунок 6. Создание моделей и графиков важности признаков

На данном фрагменте кода показано что мы последовательно обучаем несколько моделей, можем оценить их точность и визуализировать важность признаков этих моделей. Это дает большее понимание какие признаки важны для предсказания. Пример одной модели можно посмотреть на рисунке 7.

На рисунке 7 видно, что модель правильно предсказала 89% случаев из общего числа предсказания, это высокая достаточно точность.

Recall(плотнота) – показывает насколько хорошо модель находит положительные примеры для каждого класса. В классе «0» 95%, для него она справляется хорошо, а вот для класса «1» плотность достаточно низкая, всего 40%, это говорит о том что модель пропускает положительные случаи класса «1».

F1-Score – среднее между точностью и плотностью, тут так же для класса «0» производительность высокая 94%, а в классе «1» всего 45%, что указывает на проблемы с предсказанием.

Support – поддержка показывает дисбаланс классов, для класса «0» 807, а для класса «1» 98, это может негативно влиять на производительность модели.

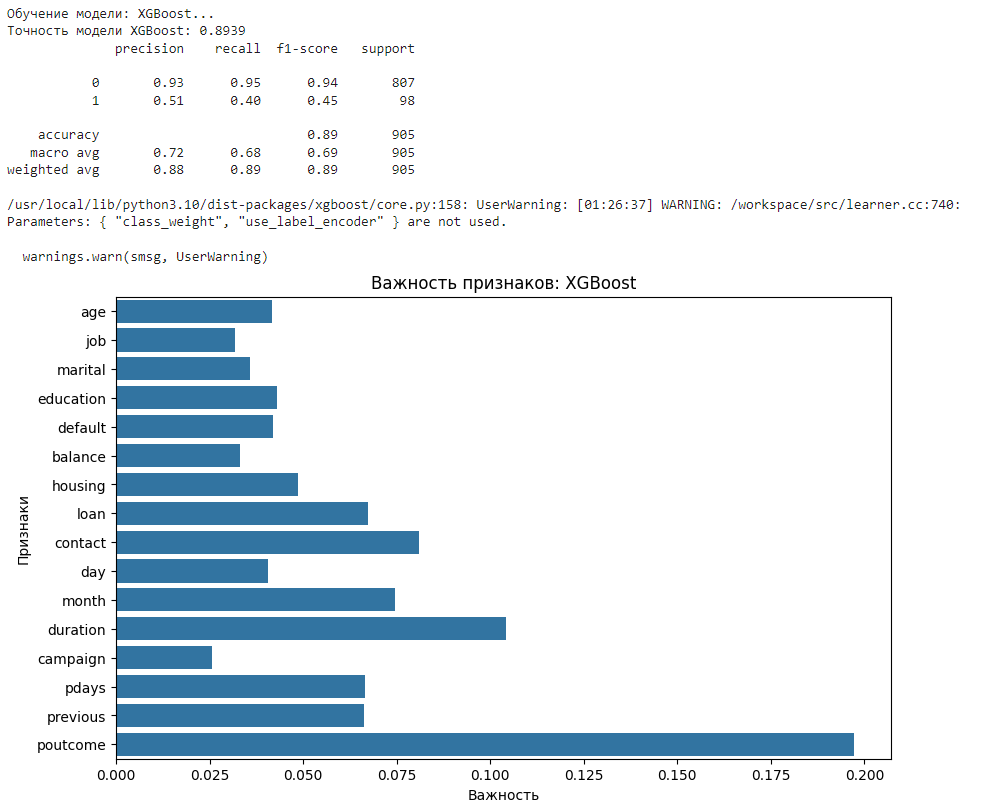


Рисунок 7. Точность и важность признаков XGB

## 3.2. Создание ансамблевых моделей

В своей работе я создал два метода ансамблевой модели, а именно Bagging и Voting. Оба метода могут заметно увеличить производительность по сравнению использования одной модели, особенно если отдельные модели имеют сильные и слабые стороны. Разберем одну из них, Рисунок 8.

На данном экземпляре кода показано, что сначала бы добавляем классификатор XGBoost, устанавливаем метрику для оценки модели (используем логарифмическую потерю) и балансируем классы. Создаем ансамблевую модель и добавляем список кортежей, где содержится сама модель и ее название (Rf\_model и XGB\_model) и указываем на мягкое голосование, где учитывается не только классы, а вероятность предсказаний.

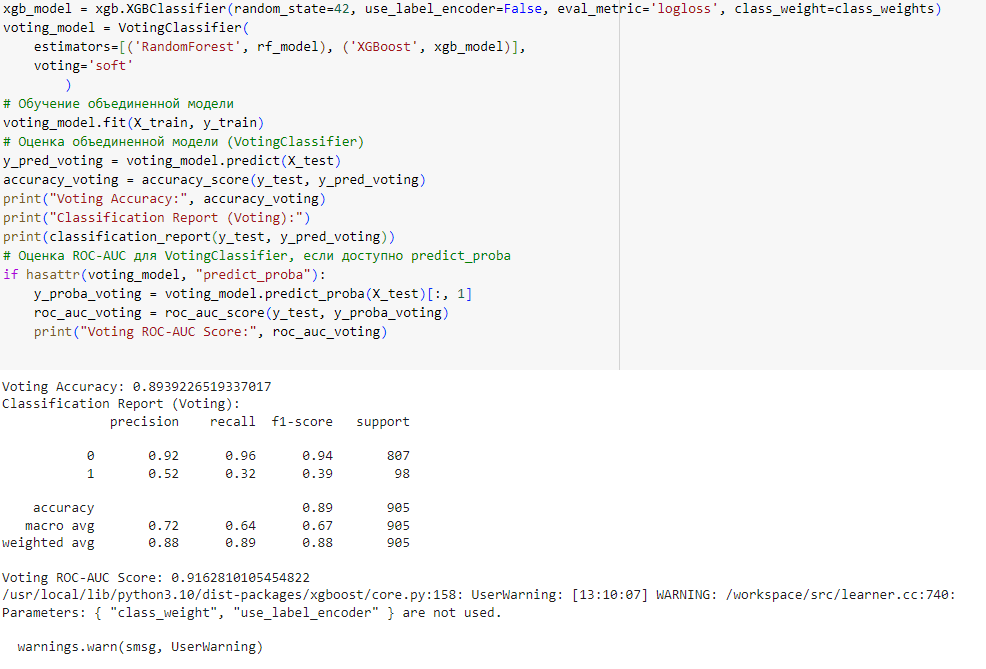


Рисунок 8. Модель Voting

В данном фрагменте кода используется ROC-AUC, это метрика, которую используют для оценки классификаторов(бинарных). Она основана на анализе ROC кривой, где важно показать соотношение между истинными и ложными положительными результатами. Результаты можно посмотреть на рисунке 8.

True Positive Rate – это доля истинно положительных предсказаний из имеющихся положительных примеров (recall).

False Positive Rate – это ложноположительные предсказания из всех отрицательных примеров.

ROC-кривая учитывает все пороги классификации, на ней легко сравнить все модели и их производительность, а так же она менее чувствительна к несбалансированным классам.

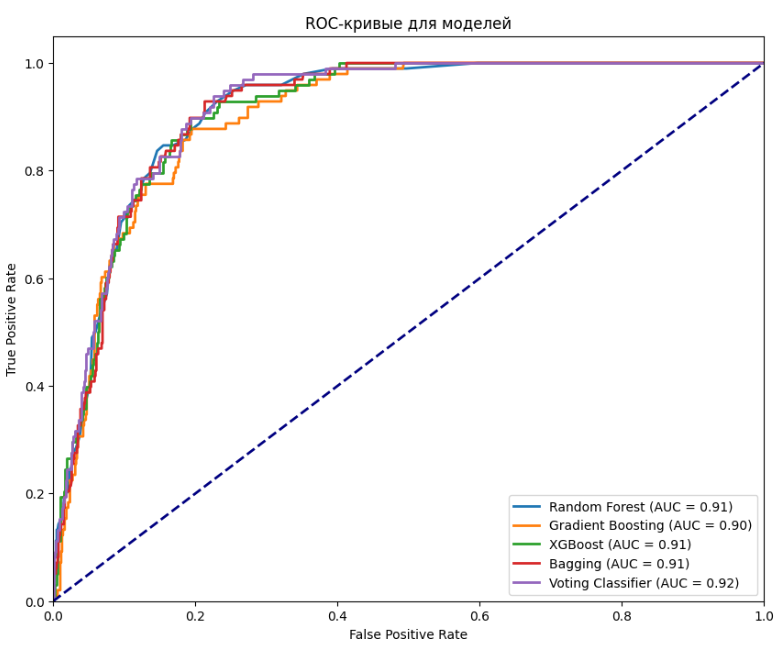


Рисунок 8. ROC-кривая

## 3.3. Сравнение точности всех моделей

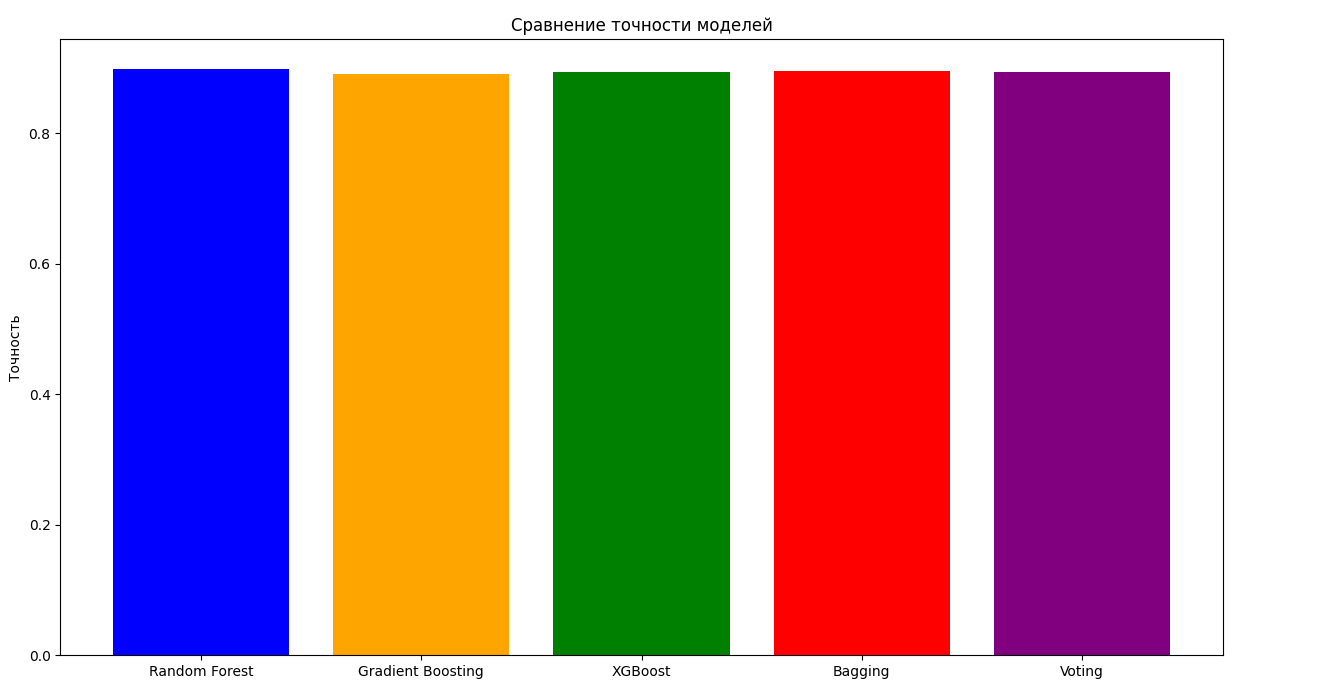
****

Рисунок 9. Точность всех моделей

В ходе курсовой работы были протестированы пять моделей для решения задачи и можно сделать вывод про эти модели.

Лучшая по точности модель это Bagging, ее точность составляет (0.8961), но модель Voting превосходит все другие модели по ROC-AUC, что делает ее более предпочтительной в контексте различных классов. Так же имеются проблемы с «1» классом из-за несбалансированности классов.

# **ЗАКЛЮЧЕНИЕ**

В данной курсовой работе были решены все поставленные задачи и построены модели.

Ознакомились с платформой для работы с данными, исследовали и провели обработку данных преобразовав категориальные значения в числовые, а также провели балансировку классов. Построив модели провели анализ результатов и сделали выводы какая модель лучше всех показала себя при бинарной классификации. Также были построены графики для визуального просмотра результатов точности модели.

В ходе данной работы я получил новый опыт в сфере машинного обучения и буду использовать полученные навыки и знания в своих будущих проектах.

Ссылка на репозиторий: <https://github.com/RomaTruba/KursavayaScoring>

# **СПИСОК ЛИТЕРАТУРЫ**

1. “Основы Data Science и машинного обучения” автор Гусев Е. (2019)
2. “Практическое машинное обучение с помощью Scikit-Learn, Keras и TensorFlow” (2-е издание) Орельен Жерон (2019)
3. **Методы и алгоритмы анализа данных»** Айвазян С.А., Мхитарян В.С. (2020)
4. “Машинное обучение: методы, модели, применение” Воронцов К.В. (2019)
5. Справочник по Python для обработки данных: основные инструменты для работы с данными» Джейка ВандерПласа (2016)
6. “Разработка функций для машинного обучения” Элис Чжэн и Аманды Казари (2018)
7. “Интеллектуальный анализ данных” Дюк В. (2021)
8. “Распознавание образов и машинное обучение” Кристофера М. Бишопа (2006)
9. Kaggle [Электронный ресурс] // Веб-сайт. 2023. URL: <https://www.kaggle.com>