Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка образовательных материалов и проектов в сфере Data Science»

по дисциплине «Проектный практикум»

Заказчик: Ильинский А. Д. _____

Куратор: Ильинский А. Д.

Ассистент кафедры ИИТ _____

Студенты команды Capybara Tech Artworks

Тарануха А. Н. _____

Махлонов Д. А. _____

Чернов И. М. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный мир стремительно движется в сторону цифровизации и автоматизации процессов, что приводит к росту спроса на специалистов в области Data Science. Машинное обучение, анализ данных и искусственный интеллект находят применение в самых разных сферах — от медицины и финансов до маркетинга и развлечений. Однако доступ к качественным образовательным материалам, которые позволяют освоить ключевые навыки Data Science на практике, остается ограниченным.

Целью данного проекта является разработка образовательных материалов и проектов в сфере Data Science, направленных на формирование у обучающихся практических навыков работы с данными, машинным обучением и аналитикой.

Основные задачи проекта:

1) Создание интерактивных блокнотов (Jupyter Notebook) с пошаговым решением реальных задач Data Science.

2) Демонстрация применения различных методов машинного обучения и анализа данных на реальных наборах данных.

Актуальность проекта обусловлена растущим спросом на специалистов в области Data Science. Решение классических задач, таких как прогнозирование выживаемости пассажиров "Титаника" или предсказание популярности музыкальных треков, помогает закрепить ключевые концепции: предобработку данных, feature engineering, выбор моделей и их оценку.

Ожидаемые результаты:

– Готовые блокноты с решением задач (например, "Титаник" и "Предсказание популярности треков Spotify"), включающие полный цикл анализа данных.

– Пошаговые объяснения, визуализации и выводы, помогающие понять логику работы алгоритмов.

По завершении проекта планируется достичь следующих результатов:

– Создание двух полноценных учебных кейсов, демонстрирующих применение методов Data Science.

– Повышение уровня практической подготовки пользователей, работающих с данными.

– Формирование базы для дальнейшего расширения коллекции образовательных материалов.

Данный проект не только способствует развитию навыков в области анализа данных, но и помогает сократить разрыв между теоретическим обучением и реальными задачами, с которыми сталкиваются специалисты.

1 Требования к проекту

Для нашего проекта были выделены и назначены следующие требования:

1) EDA и исследовательский анализ:

– Визуализация данных (гистограммы, box-plot, scatter-plot и др.).

– Комментарии и выводы к графикам.

– Анализ корреляции признаков с целевой переменной.

2) Feature Engineering:

– Создание новых признаков, проверка их значимости.

– Использование feature_importances_ для оценки важности фичей.

– Построение простой модели для проверки качества новых признаков.

3) Эксперименты с моделями:

– Тестирование минимум по одной модели из каждого семейства:

– Линейные (логистическая регрессия, Ridge, SGD).

– Деревья (решающее дерево, случайный лес).

– Градиентный бустинг (XGBoost, CatBoost, LightGBM).

– Нейросети (простой MLP).

– Кросс-валидация (≥5 фолдов) и сравнение по F1 и ROC-AUC.

– Выбор лучшей модели и обоснование выбора.

4) Формат сдачи:

– Готовый проект на GitHub с четкой структурой и комментариями.

2 Архитектура программного продукта

Выбор платформы:

– Google Colab — облачная среда для совместной работы, бесплатный доступ к GPU/TPU.

– GitHub — хостинг кода и контроль версий.

Используемые библиотеки:

– Анализ данных: pandas, numpy, seaborn, matplotlib.

– Машинное обучение: scikit-learn, XGBoost, CatBoost, LightGBM, TensorFlow/Keras.

– Валидация: cross_val_score, GridSearchCV.

Структура проекта:

– Data Preprocessing – загрузка, очистка, кодирование.

– EDA – визуализация, статистика.

– Feature Engineering – создание и отбор признаков.

– Model Training – обучение и валидация моделей.

– Results & Conclusions – сравнение метрик, выводы.

3 Методология разработки и процесс тестирования

Методология:

– Итеративный подход – поэтапное улучшение модели (от простой к сложной).

– Кросс-валидация – оценка стабильности моделей.

Тестирование и ошибки:

– Проблема: Первые модели (логистическая регрессия) давали низкий F1 из-за несбалансированных данных и нелинейных зависимостей метрик.

Решение: Использование более сложных и адаптивных моделей.

– Проблема: Переобучение деревьев.

Решение: Регуляризация (ограничение глубины, min_samples_split).

– Проблема: Нейросеть сходилась медленно.

Решение: Добавление Batch Normalization и изменение архитектуры.

4 Планирование и распределение задач

Начальный этап:

– Обсуждение сильных сторон участников.

– Распределение ролей (EDA → Feature Engineering → Models → Testing).

Контроль выполнения:

– Тимлид (Тарануха А.Н.) следил за соблюдением требований и дедлайнов.

– Еженедельно проводились обсуждения прогресса. Члены команды помогали друг другу при возникновении сложностей.

5 Организация работы команды и вклад участников

Каждый участник команды отвечал за определённый этап разработки, что позволило эффективно распределить задачи и обеспечить высокое качество итоговых материалов.

Тарануха А.Н.

– Занимался предобработкой данных (EDA — Exploratory Data Analysis) и исследованием датасетов.

– Проводил анализ распределений, проверку на пропуски и выбросы, статистическую оценку данных.

– Визуализировал зависимости между признаками и целевой переменной.

– Формулировал выводы после каждого этапа анализа.

Махлонов А.Д.

– Отвечал за Feature Engineering — создание новых признаков для улучшения классификации модели.

– Анализировал корреляцию новых признаков с целевой переменной.

– Строил простые базовые модели (логистическая регрессия, линейные методы) для проверки значимости фичей.

– Разрабатывал сложные модели машинного обучения (деревья, градиентный бустинг, нейросети).

Чернов И.М.

– Проводил тестирование моделей, подбирал гипер-параметры с помощью кросс-валидации (Приложение А) (≥5 фолдов).

– Сравнивал эффективность алгоритмов по метрикам (F1, ROC-AUC).

6 Разработка блокнотов и сравнение результатов

6.1 Результаты по задаче "Титаник" (классификация)

Перед исследованием данные были очищены от ячеек с большим количеством пропусков, а категориальные параметры были переведены в числа.

6.1.1 EDA

Ящик с усами показал, что мужчины в среднем старше и имеют больше выбросов по возрасту (Рисунок 1)

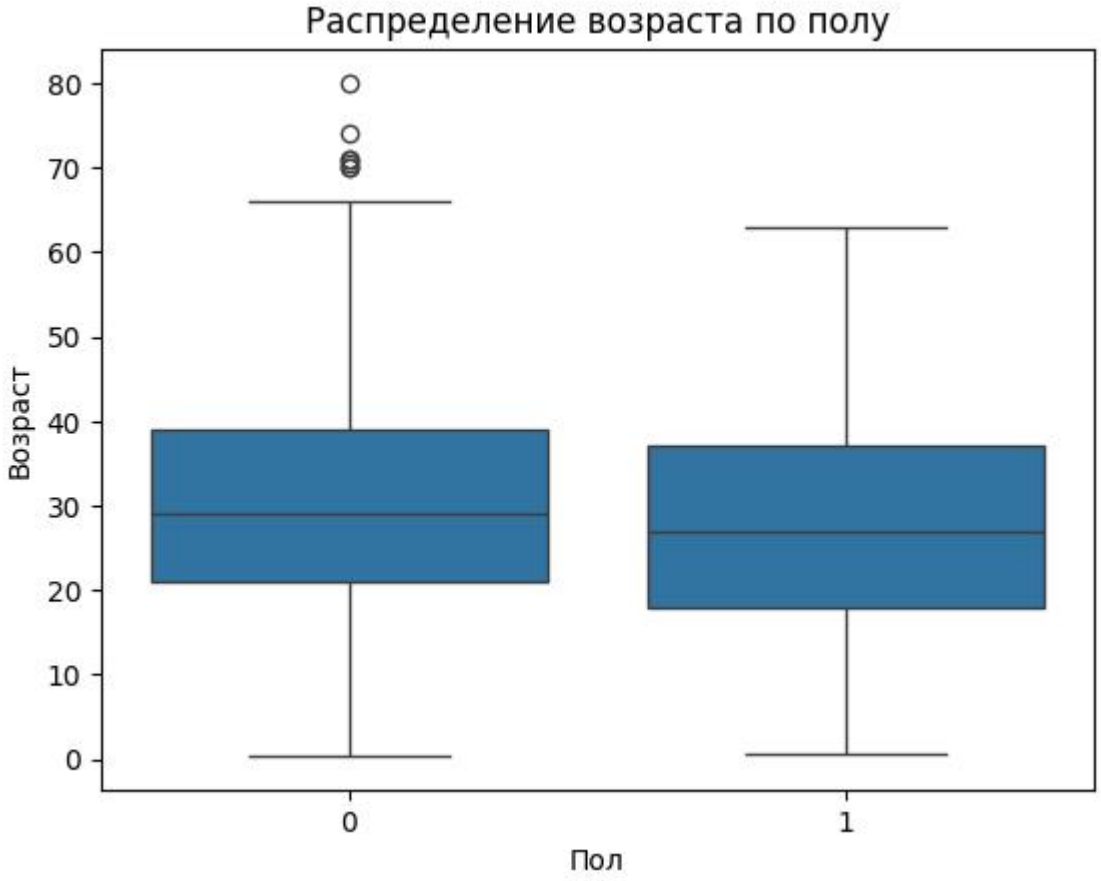

Рисунок 1 – График “Распределение возраста по полу”

Мы объединили семантически близкие Parch и SibSp в единый параметр с целью определения размера семьи. Пассажиры с 1-3 родственниками выживали чаще, чем одиночки или большие семьи (Рисунок 2).

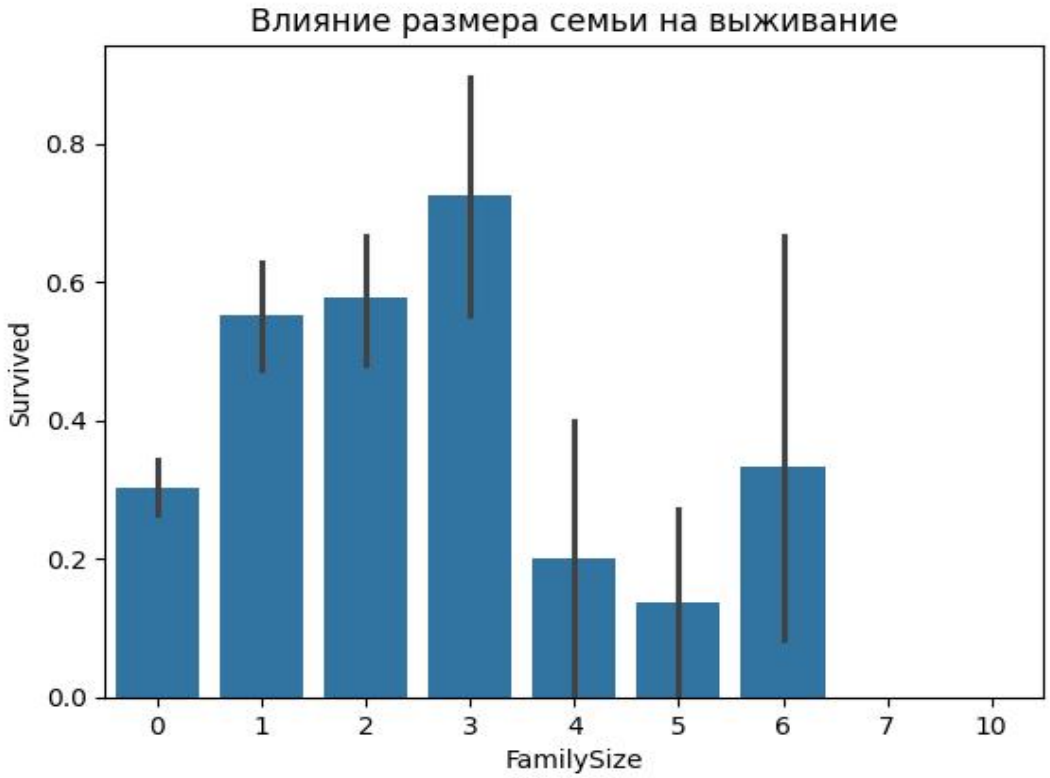

Рисунок 2 – График “Влияние семьи на выживание”

На графике выживаемости к классу каюты видим очень высокую корреляцию. Пассажиры из более "дорогих" классов кают выживают значительно вероятнее (Рисунок 3).

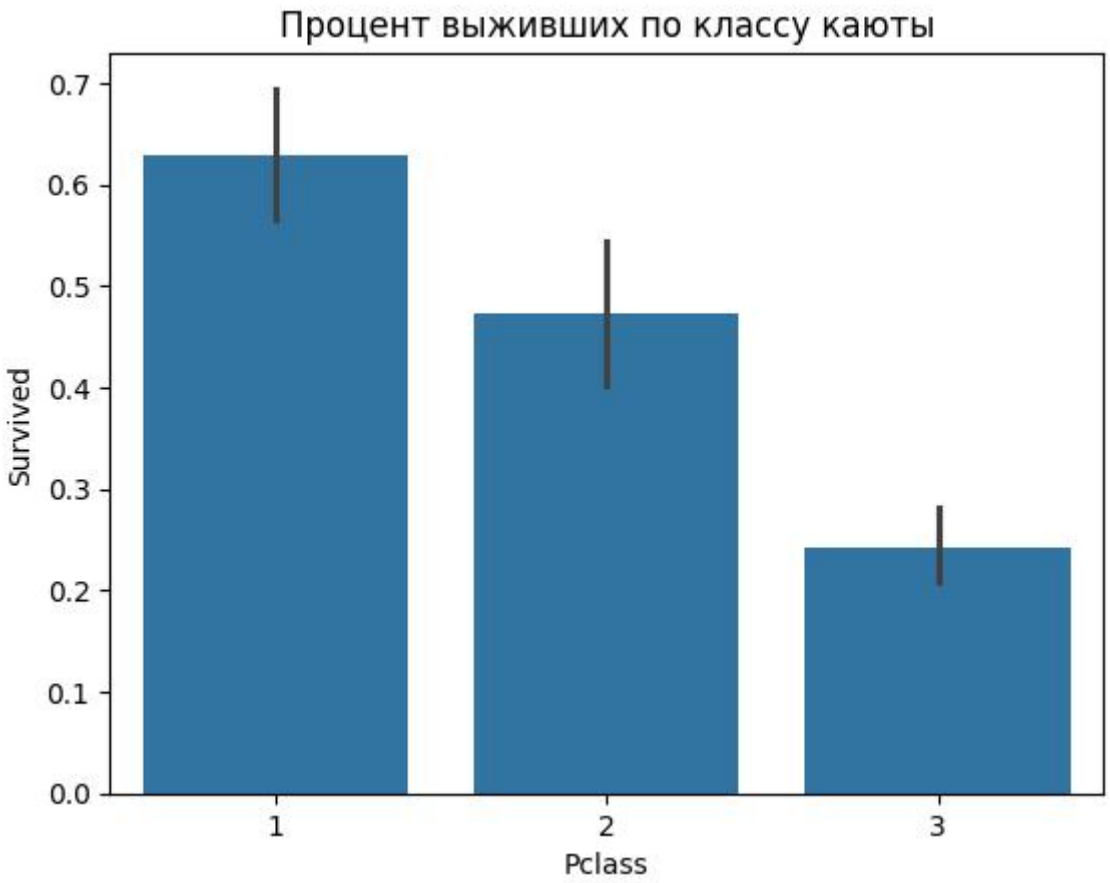

Рисунок 3 – График “Процент выживших по классу каюты”

Ящик с усами показывает, что в среднем пассажиры с более высокой ценой билета выживали чаще (Рисунок 4). Необходимо проверить корреляцию с Pclass.

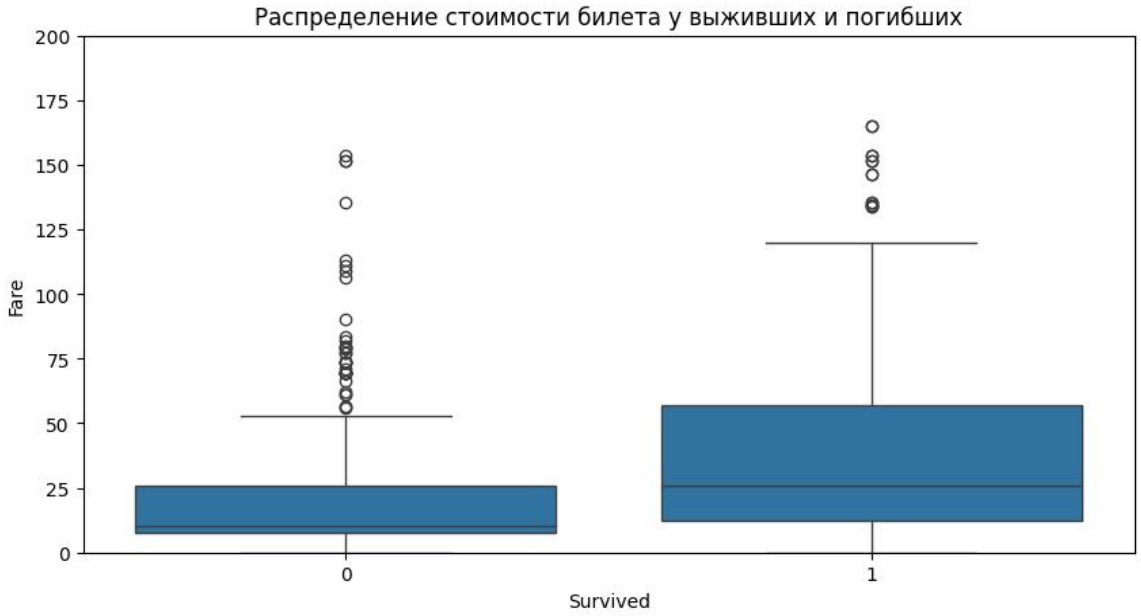

Рисунок 4 – График “Распределение стоимости билета у вижившх и погибших”

На точечном графике видим, что разброс цены для первого класса был очень большим, поэтому важно сохранить оба параметра, для отличия дорогого второго класса и дешевого первого, к примеру (Рисунок 5).

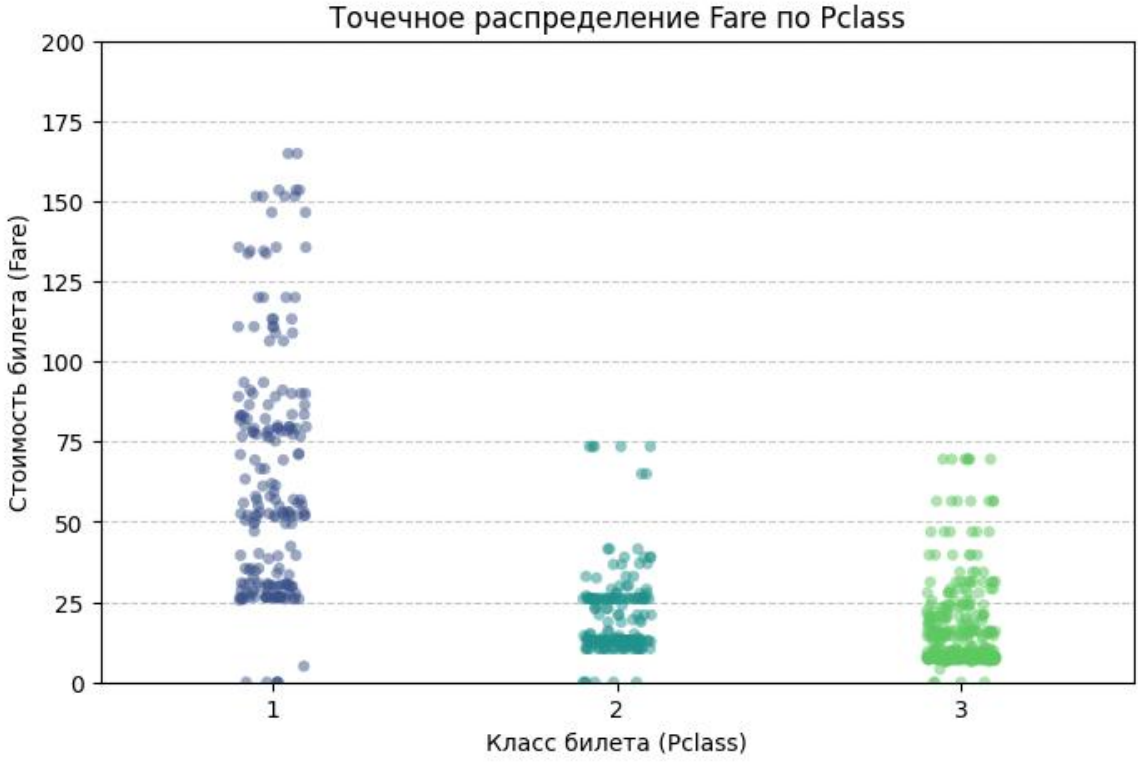

Рисунок 5 – График “Точечное распределение Fare по Pclass”

Также на основе модели К-ближайших соседей были заполнены недостающие ячейки Age.

Также в датасете был обнаружен значительный дисбаланс классов (Рисунок 6), поэтому основной метрикой сравнения была выбрана F-мера.

Survived	count
0	549
1	342

Рисунок 6 – Дисбаланс классов

Был построен график корреляции с таргетом, по которому было определено, что параметр Family Size нужно удалить, чтобы не загрязнять данные (Рисунок 7).

	Survived	Pclass	Sex	Age	Fare	FamilySize
Survived	1.000000	-0.338481	0.543351	-0.111891	0.257307	0.016639
Pclass	-0.338481	1.000000	-0.131900	-0.357595	-0.549500	0.065997
Sex	0.543351	-0.131900	1.000000	-0.132805	0.182333	0.200988
Age	-0.111891	-0.357595	-0.132805	1.000000	0.091416	-0.227301
Fare	0.257307	-0.549500	0.182333	0.091416	1.000000	0.217138
FamilySize	0.016639	0.065997	0.200988	-0.227301	0.217138	1.000000

Рисунок 7 – Корреляция с таргетом

6.1.2 Feature engineering

Для улучшения модели были созданы дополнительные метрики.

Первой была создана метрика Age Group, основанная на Age. Она позволяет убрать числовой шум и давать модели возраст в понятной группе. Дети (Child) имеют самый высокий уровень выживаемости (≈58%), тогда как пожилые (Senior) - самый низкий (≈23%) (Рисунок 8).

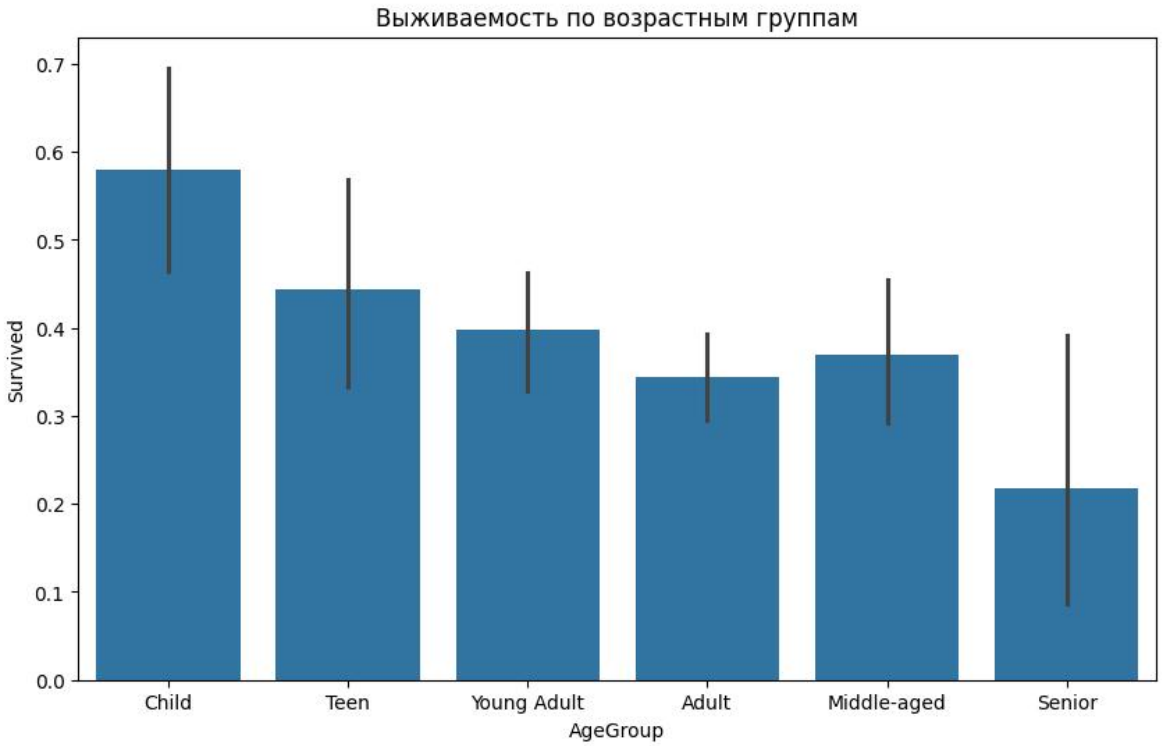

Рисунок 8 – График “Выживаемость по возрастным группам”

Стоимость билетов тоже была разделена на группы. Пассажиры с билетами категории "Very High" выживали в ≈58% случаев, тогда как в категории "Low" — только ≈20%. Стоимость билета сильно коррелирует с выживаемостью (Рисунок 9).

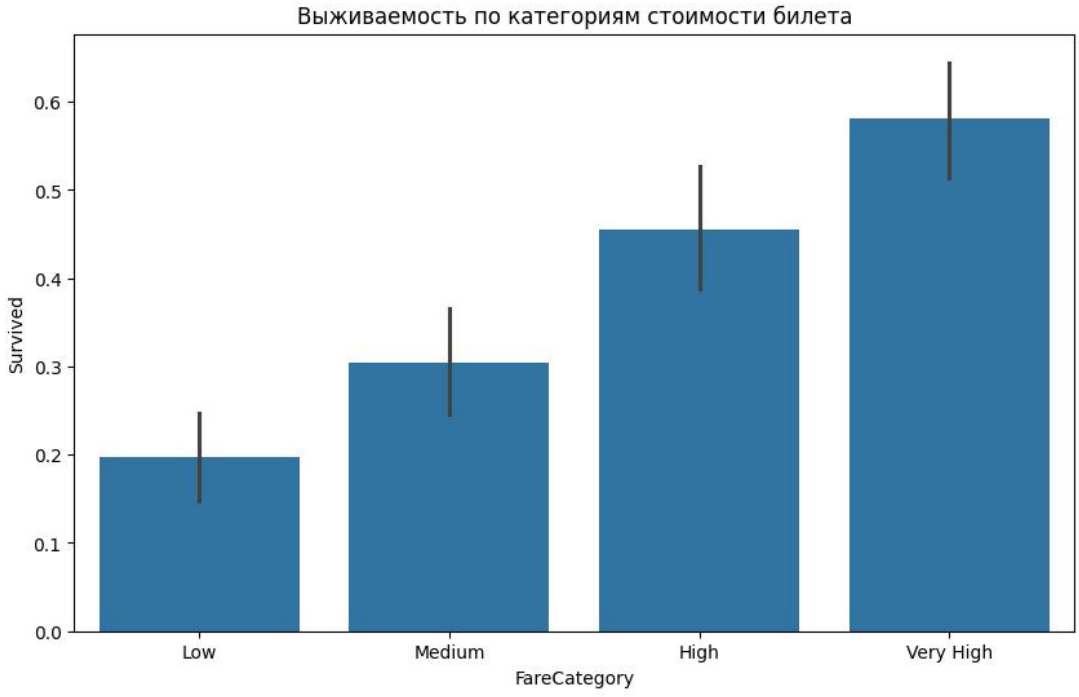

Рисунок 9 – График “Выживаемость по категориям стоимости билета”

В таблице корреляции видим высокую корреляцию новых параметров с таргетом (Рисунок 10). Базовые Age и Fare были удалены, чтобы не нагружать модель семантически близкими данными.

	Survived	Pclass	Sex	Age	Fare	AgeGroup	FareCategory
Survived	1.000000	-0.338481	0.543351	-0.111891	0.257307	-0.120479	0.299357
Pclass	-0.338481	1.000000	-0.131900	-0.357595	-0.549500	-0.294534	-0.634271
Sex	0.543351	-0.131900	1.000000	-0.132805	0.182333	-0.148983	0.243613
Age	-0.111891	-0.357595	-0.132805	1.000000	0.091416	0.936966	0.092766
Fare	0.257307	-0.549500	0.182333	0.091416	1.000000	0.055567	0.579345
AgeGroup	-0.120479	-0.294534	-0.148983	0.936966	0.055567	1.000000	0.014588
FareCategory	0.299357	-0.634271	0.243613	0.092766	0.579345	0.014588	1.000000

Рисунок 10 – Корреляция новых параметров с таргетом

6.1.3 Построение моделей и снятие метрик

Были протестированы модели различных семейств с оценкой по F1-мере и ROC-AUC, результаты представлены в таблице 1.

Таблица 1 – Результаты тестирования моделей

Модель	F1-мера	ROC-AUC
Логистическая регрессия	0.792	-
Ridge	0.755	-
SGD	0.513	-
Нейросеть (128-512-1)	0.776	0.882
Дерево решений	0.857	0.917
Случайный лес	0.857	0.905
Градиентный бустинг (XGBoost)	0.857	0.913
Градиентный бустинг (CatBoost)	0.857	0.912
Градиентный бустинг (LightGBM)	0.857	0.917

Выводы:

– Все нелинейные модели (деревья, бустинг) показали одинаковый F1-скор (~0.857), что указывает на потолок предсказательной способности при текущем наборе признаков.

– Нейросеть уступила деревьям и бустингу, что может быть связано с небольшим объемом данных.

– Линейные модели (кроме логистической регрессии) показали худшие результаты, подтверждая нелинейную природу данных.

6.2 Результаты по задаче "Spotify" (регрессия)

Перед исследованием из датасета были убраны колонки с метриками, не совпадающими с объективной реальностью описиваемых музыкальных произведений, а также семантически спорные колонки. Также были убраны колонки, которые не могут иметь корреляции с популярностью в контексте нашего исследования. Дубликаты треков были заменены на один с самой высокой популярностью.

6.2.1 EDA

По графику корреляции можно заметить интересные свойства (Рисунок 11). Например высокую позитивную корреляцию с (energy-loudness) и (danceability-valence), а также высокую отрицательную корреляцию между (energy-acousticness), (loudness-acousticness), (instrumentalness-loudness). Таким образом можно убрать параметр energy из-за его близости с loudness. Также следует убрать параметр valence из-за его близкой корреляции с danceability, energy, loudness и instrumentalness. Целевой параметр popularity значительно коррелирует с danceability, loudness, instrumentalness и track_genre.

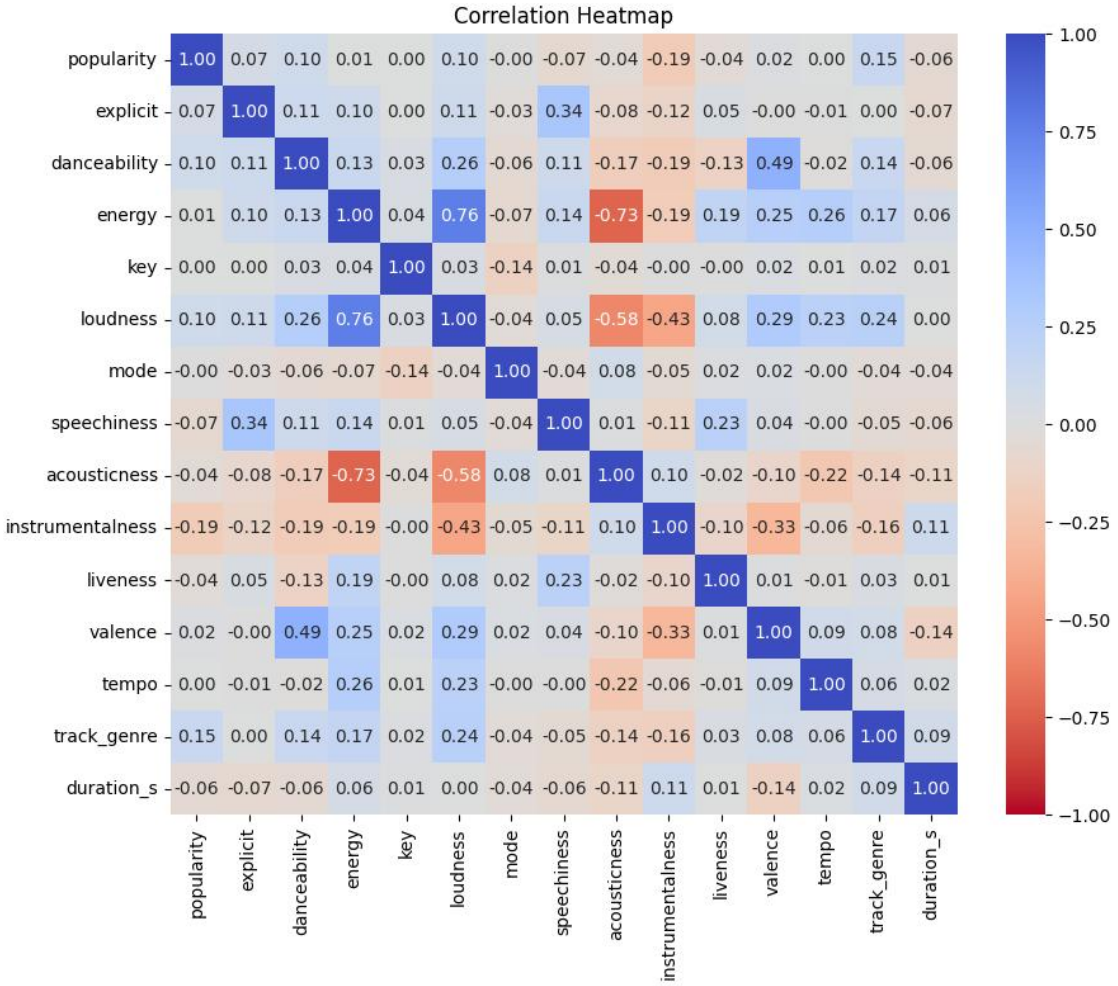

Рисунок 11 – График корреляции

В описании датасета написано, что треки с параметром speechiness >= 0.66 вероятнее всего являются полностью речью. Такие данные могут помешать нам точно предсказывать параметр, потому что направленность датасета на музыку. Поэтому они были удалены из выборки.

Также в описании датасета говорится о том, что значение -1 в колонке genre является неопределенным. Но в датасете таких значений обнаружено не было, поэтому параметр стоит оставить (Рисунок 12).

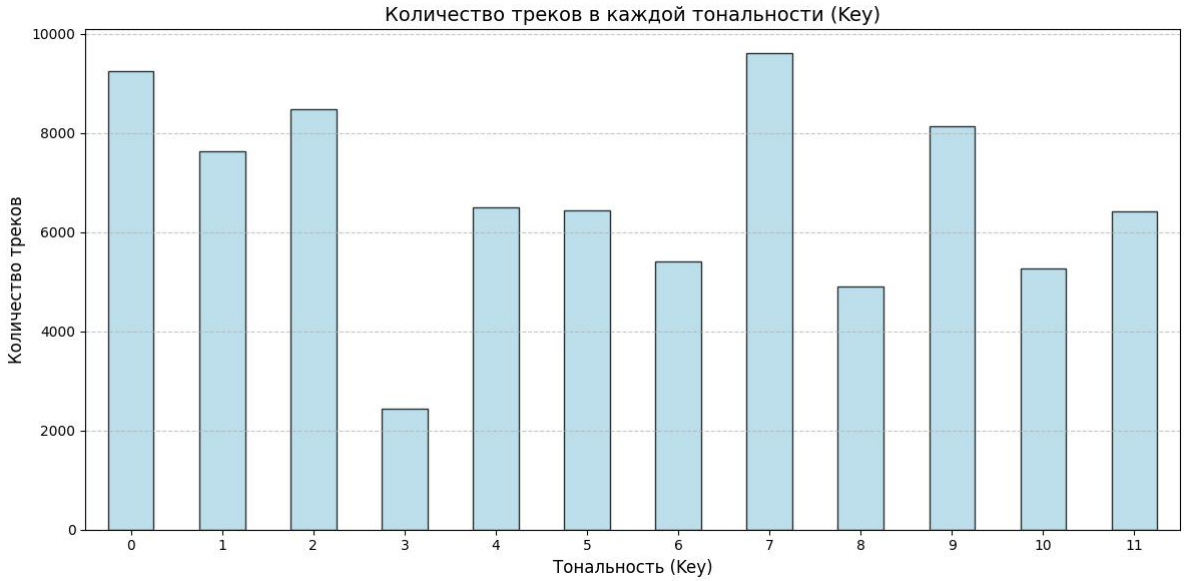

Рисунок 12 – График “Количество треков в каждой тональности”

График распределения популярности треков (Рисунок 13) большие выбросы в районе 0-5 и 20-25, а также очень низкое кол-во треков с популярностью 80+.

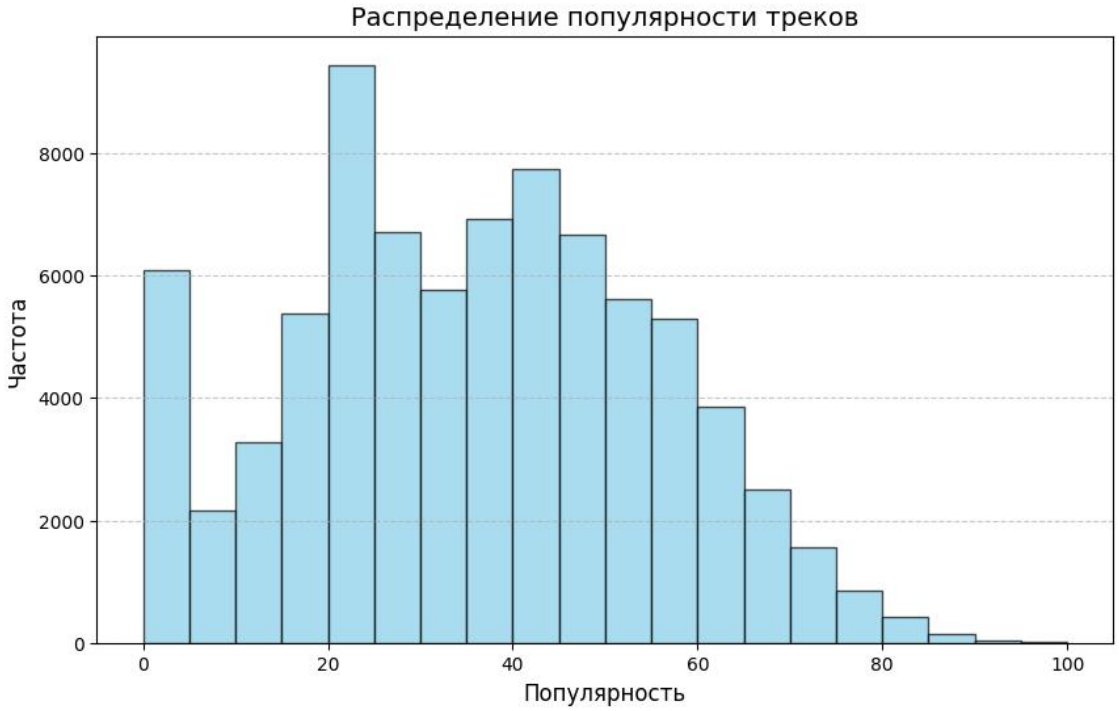

Рисунок 13 – График “Распределение популярности треков”

Данный график показал сильную корреляцию между различными жанрами и степенью их популярности (Рисунок 14).

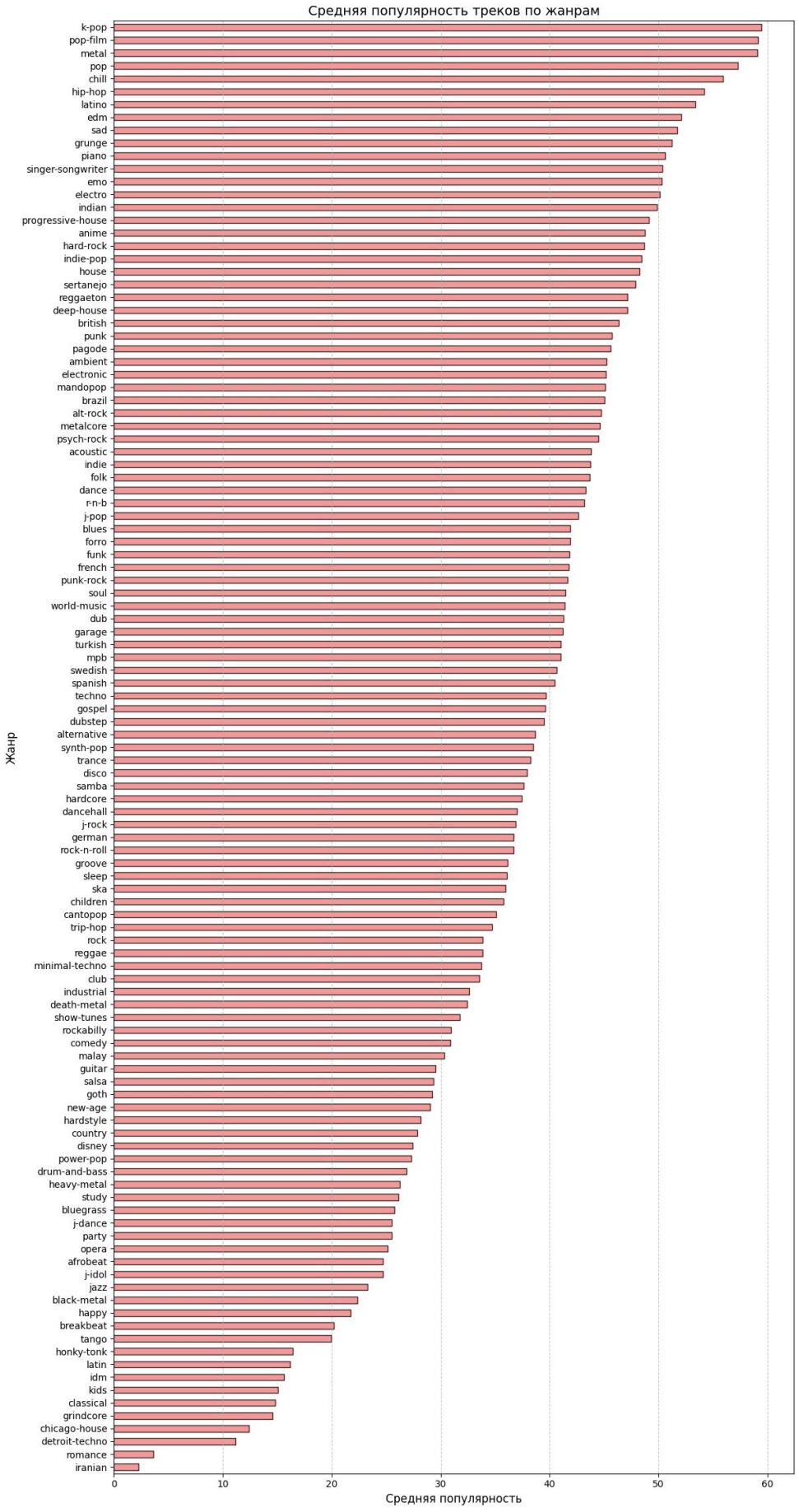

Рисунок 14 – График “Средняя популярность треков по жанрам”

График поля explicit показал высокий дисбаланс (Рисунок 15).

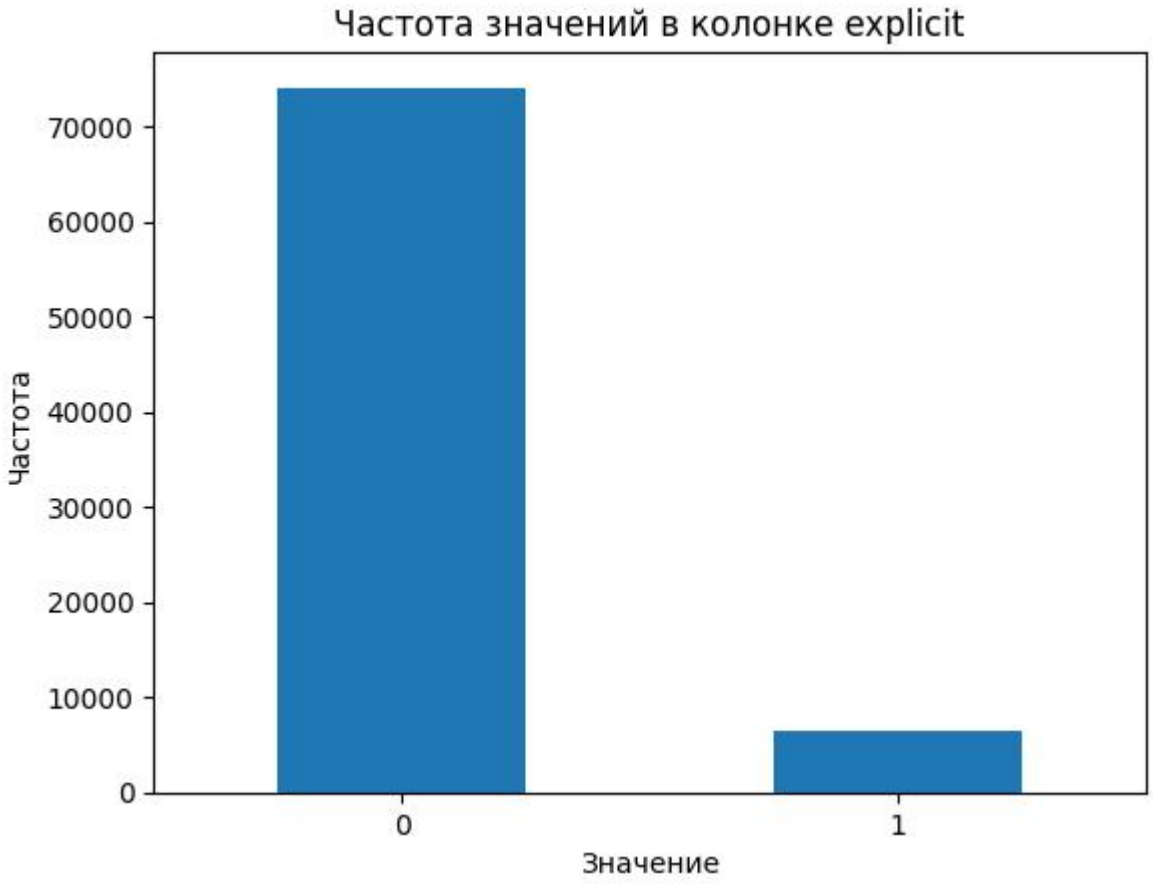

Рисунок 15 – График “Частота значений в колонке explicit”

6.2.2 Feature engineering

Построим дополнительные метрики для улучшения точности модели.

Параметр raveness является комбинацией из нормализованных danceability, tempo и loudness. Признак показал неплохую положительную корреляцию с популярностью, хотя и заметно меньшую, чем genre или instrumentalness (Рисунок 16).

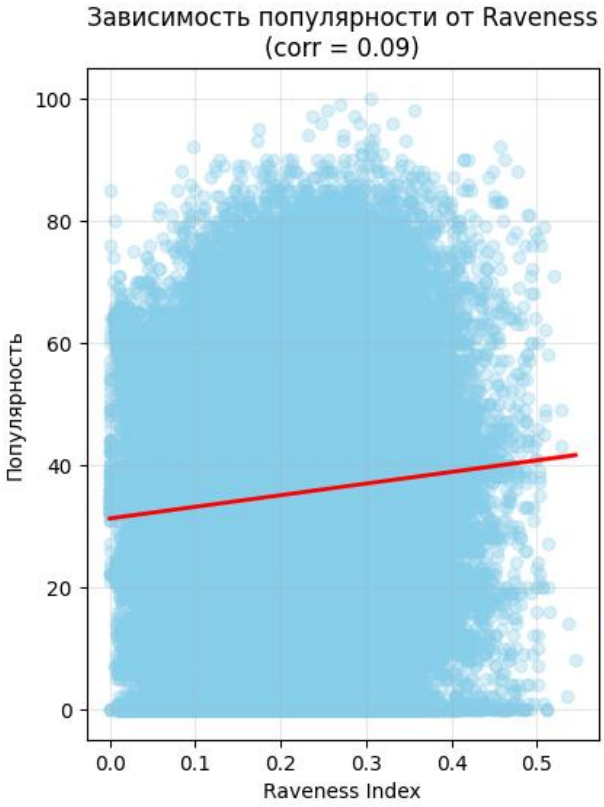

Рисунок 16 – График зависимости популярности от Raveness

Также была посчитана метрика instrumental_proportion. Акустичность и инструментальность трека вместе будут давать более точную характеристику того, насколько много в треке живых инструментов. Фича демонстрирует небольшую негативную корреляцию (Рисунок 17). В контексте нашего датасета такая метрика кореляции является приемлимой.

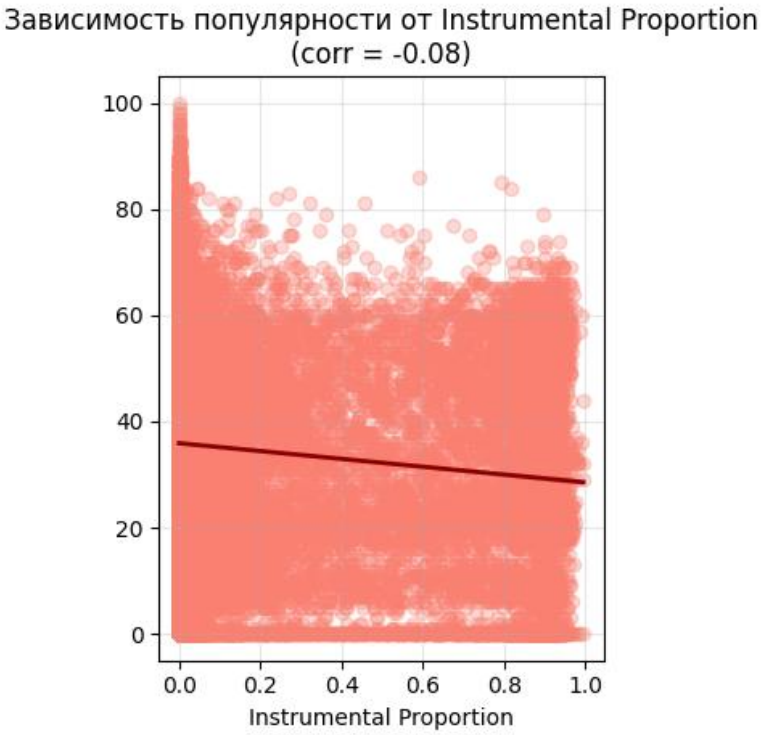

Рисунок 17 – График зависимости популярности от Instrumental Proportion

Также на основе модели Random Forest построен график Feature Importance (Рисунок 18), показавший сильную зависимость от жанра, но также и большое количество небольших корреляций с другими фичами.

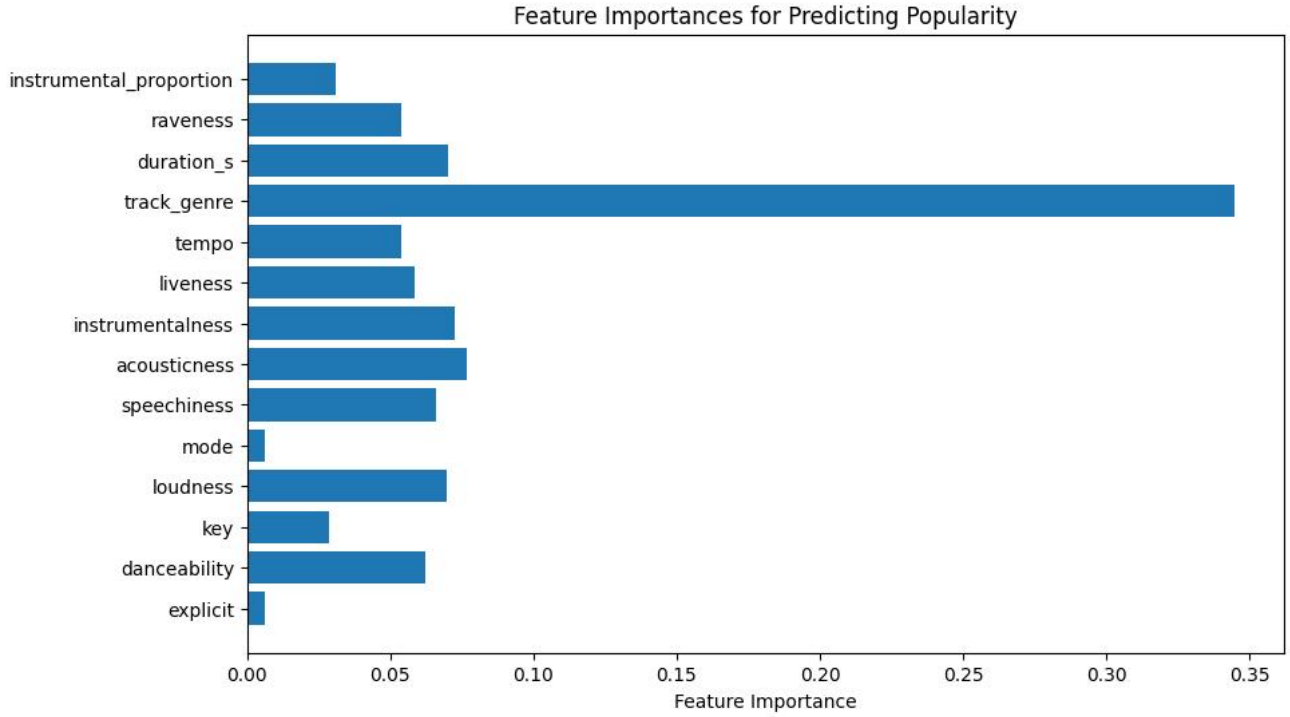

Рисунок 18 – График “Feature Importances для Predicting Popularity”

6.2.3 Построение моделей и снятие метрик

Модели с оценкой по MSE, использовавшиеся для предсказания популярности треков, представлены в таблице 2.

Таблица 2 – Модели использованные для предсказания популярности треков

Модель	MSE
Линейная регрессия	354.72
Случайный лес (Random Forest)	223.82
XGBoost	224.72
Нейросеть (TensorFlow/Keras)	312.81

Результаты кросс-валидации (Приложение А) на пяти фолдах модели Random Forest представлены в таблице 3.

Таблица 3 – Результаты кросс-валидации

№ Фолда	MSE
1	223.87
2	224.69
3	226.47
4	224.15
5	222.05

Итоговые метрики:

– Среднее MSE: 224.25 ± 1.42

– MSE на тестовом наборе: 219.45

6.3 Сравнительный анализ

Эффективность моделей:

– Random Forest стал лучшей моделью.

– XGBoost/CatBoost/LightGBM показали сопоставимое качество, но требовали больше настройки.

– Нейросети не оправдали ожиданий из-за недостатка данных.

Промежуточные выводы:

– Для малых данных предпочтительны Random Forest/Gradient Boosting.

– Нейросети стоит применять только при большом объеме тренировочных данных.

6.4 Выводы по разделу

Random Forest стал лучшим для обеих задач:

– Классификация: F1 = 0.857 (наравне с бустингом).

– Регрессия: MSE = 219.45 (опередил XGBoost на 0.9).

Универсальность Random Forest подтверждена: лидер в регрессии и классификации.

Градиентный бустинг требует тонкой настройки, но почти не уступает.

Линейные модели проигрывают на сложных данных, но полезны для базовых benchmark-тестов.

Особенности данных:

Данные результаты согласуются с современными практиками Data Science, где ансамблевые методы часто превосходят классические подходы на структурированных данных.

ЗАКЛЮЧЕНИЕ

Разработанные проекты полностью соответствуют поставленным заказчиком требованиям. Для задачи "Титаник" проведен полный цикл анализа: от EDA с визуализациями и выводами до тестирования 9 моделей с кросс-валидацией. Лучшие результаты (F1 = 0.857) достигнуты на ансамблевых методах (Random Forest, XGBoost). Для задачи "Spotify" реализована регрессия с сравнением 4 моделей. Random Forest показал наименьшую MSE (219.45), что подтверждено кросс-валидацией (среднее MSE = 224.25 ± 1.42). Jupyter-блокноты содержат пошаговые комментарии, графики с интерпретацией, что соответствует требованиям к образовательным материалам.

Критические достижения. Выполнен сравнительный анализ моделей из разных семейств (линейные, деревья, бустинг, нейросети). Подтверждена гипотеза о применимости Random Forest для небольших датасетов. Стандартизирован подход к EDA и Feature Engineering, что упрощает воспроизводимость для обучающихся.

1. Оценка качества и выявленные проблемы

Сильные стороны:

1) Стабильность моделей: Кросс-валидация показала низкий разброс метрик (например, ±1.42 MSE для Random Forest).

2) Интерпретируемость: Визуализации и выводы помогают понять логику решений.

Недостатки и их влияние:

1) Низкая эффективность нейросетей (MSE = 312.81 для Spotify, F1 = 0.776 для "Титаника"):

– Причина: Недостаток данных для глубокого обучения.

– Влияние: Ограничивает применение DL в учебных примерах.

2) "Потолок" метрик для "Титаника" (F1 ≈ 0.857):

– Причина: Исчерпан потенциал признаков.

– Влияние: Без доп. Feature Engineering улучшение невозможно.

3) Чувствительность линейных моделей к данным:

Пример: SGD (F1 = 0.513) и Linear Regression (MSE = 354.72) показали худшие результаты.

Вывод: Требуют строгой предобработки (скейлинг, борьба с выбросами).

2. Предложения по улучшению и развитию

Для текущего проекта:

1) Добавить Feature Engineering. А именно - для "Титаника" — генерация новых признаков на основе продвинутого анализа параметров(например, титулы из имен), для Spotify — анализ текстов песен (NLP) или временных особенностей треков.

2) Улучшить нейросети - использовать transfer learning (например, предобученные embeddings для Spotify), gприменить синтетическое увеличение данных (SMOTE для классификации).

Для будущих проектов:

1) Расширить набор задач: добавить unsupervised learning (кластеризация жанров Spotify), реализовать pipeline для автоматического подбора моделей (AutoML).

2) Оптимизировать производительность: внедрить DVC (Data Version Control) для управления экспериментами.

Полученные выводы:

Для задач с малыми данными фокусироваться на ансамблевых методах (Random Forest, LightGBM). Всегда сравнивать несколько семейств моделей для объективной оценки. Документировать гипотезы на каждом этапе (например, "Почему MSE выросло после добавления признака X?").

Проект успешно решает образовательные задачи, демонстрируя полный цикл работы в Data Science — от анализа данных до выбора модели. Ключевые уроки: ансамблевые методы (Random Forest, бустинг) — "золотой стандарт" для структурированных данных, нейросети требуют осторожности

при малых выборках, качество продукта можно повысить через улучшение признаков и дополнение датасета.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рашка, С. Python и машинное обучение. – Москва : Издательство не указано, без года. – 450 с.

2. Учебник по машинному обучению [Электронный ресурс] / Школа анализа данных Яндекса (ШАД). – URL: https://academy.yandex.ru/handbook/ml (дата обращения: 8.04.2025).

3. Николенко, С. И. Глубокое обучение. Погружение в мир нейронных сетей. – Санкт-Петербург : Питер, 2018. – 480 с.

4. Статьи на Хабре [Электронный ресурс]. – Пример: «Машинное обучение для начинающих: основы нейронных сетей» / Хабр. – URL: https://habr.com/ru/companies/skillfactory/articles/525214/ (дата обращения: 12.04.2025).

5. Google [Электронный ресурс] : поисковая система. – URL: https://www.google.com (дата обращения: 7.04.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

Кросс-валидация моделей

1. Кросс-валидация модели LightGBM в задании “Титаник”:

```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score, classification_report, confusion_matrix, roc_auc_score
import numpy as np

kFold = StratifiedKFold(n_splits=6, shuffle=True, random_state=52)

f1_scores = []
roc_auc_scores = []
reports = []
matrices = []

for fold_idx, (train_index, test_index) in enumerate(kFold.split(X, y)):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    lgbm_grid = GridSearchCV(LGBMClassifier(random_state=52, verbosity=-1), lgbm_params, cv=5)
    lgbm_grid.fit(X_train, y_train)
    model = lgbm_grid.best_estimator_

    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:, 1]

    roc = roc_auc_score(y_test, y_pred_proba)
    f1 = f1_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    matrix = confusion_matrix(y_test, y_pred)
```

```
    roc_auc_scores.append(roc)
    f1_scores.append(f1)
    reports.append(report)
    matrices.append(matrix)

    print(f'\nФолд {fold_idx + 1}:')
    print(f'F1-мера: {f1:.4f}')
    print(f'ROC-AUC: {roc:.4f}')
    print('Classification Report:')
    print(report)
    print('Confusion Matrix:')
    print(matrix)

print('\nИтоговые средние метрики:')
print(f'Средняя      F1-мера:      {np.mean(f1_scores):.4f}      ±
{np.std(f1_scores):.4f}')
print(f'Среднее    ROC-AUC:    {np.mean(roc_auc_scores):.4f}    ±
{np.std(roc_auc_scores):.4f}')
```

Output:

```
Фолд 1:
F1-мера: 0.7723
ROC-AUC: 0.8986
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.95      0.88        92
           1       0.89      0.68      0.77        57

    accuracy                           0.85       149
   macro avg       0.86      0.81      0.83       149
weighted avg       0.85      0.85      0.84       149

Confusion Matrix:
[[87  5]
```

```
 [18 39]]

Фолд 2:
F1-мера: 0.7500
ROC-AUC: 0.8494
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.91      0.87        92
           1       0.83      0.68      0.75        57

    accuracy                           0.83       149
   macro avg       0.83      0.80      0.81       149
weighted avg       0.83      0.83      0.82       149

Confusion Matrix:
[[84  8]
 [18 39]]

Фолд 3:
F1-мера: 0.7767
ROC-AUC: 0.8534
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        92
           1       0.87      0.70      0.78        57

    accuracy                           0.85       149
   macro avg       0.85      0.82      0.83       149
weighted avg       0.85      0.85      0.84       149

Confusion Matrix:
[[86  6]
 [17 40]]
```

```
Фолд 4:
F1-мера: 0.7767
ROC-AUC: 0.8793
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        91
           1       0.87      0.70      0.78        57

    accuracy                           0.84       148
   macro avg       0.85      0.82      0.83       148
weighted avg       0.85      0.84      0.84       148

Confusion Matrix:
[[85  6]
 [17 40]]

Фолд 5:
F1-мера: 0.7423
ROC-AUC: 0.8715
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.96      0.87        91
           1       0.90      0.63      0.74        57

    accuracy                           0.83       148
   macro avg       0.85      0.79      0.81       148
weighted avg       0.84      0.83      0.82       148

Confusion Matrix:
[[87  4]
 [21 36]]
```

```
Фолд 6:
F1-мера: 0.7925
ROC-AUC: 0.8761
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        91
           1       0.86      0.74      0.79        57

    accuracy                           0.85       148
   macro avg       0.85      0.83      0.84       148
weighted avg       0.85      0.85      0.85       148

Confusion Matrix:
[[84  7]
 [15 42]]

Итоговые средние метрики:
Средняя F1-мера: 0.7684 ± 0.0171
Среднее ROC-AUC: 0.8714 ± 0.0165
```

2. Поиск лучшей модели и её кросс-валидация в задании “Спотифай”:

```
from sklearn.model_selection import KFold

X = data_temp.drop(columns=['popularity'])
y = data_temp['popularity']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 1. Линейная регрессия
lr = LinearRegression()
lr_score = -cross_val_score(lr, X_train, y_train, cv=5,
scoring='neg_mean_squared_error').mean()

# 2. Случайный лес
```

```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf_score = -cross_val_score(rf, X_train, y_train, cv=5,
scoring='neg_mean_squared_error').mean()

# 3. XGBoost
xgb = XGBRegressor(n_estimators=100, random_state=42)
xgb_score = -cross_val_score(xgb, X_train, y_train, cv=5,
scoring='neg_mean_squared_error').mean()

# 4. Нейронная сеть на TensorFlow/Keras
def create_nn_model(input_dim):
    m = Sequential()
    m.add(Dense(128, input_dim=input_dim, activation='relu'))
    m.add(Dense(256, activation='relu'))
    m.add(Dense(1))
    m.compile(optimizer='adam', loss='mean_squared_error')
    return m

model = create_nn_model(X_train.shape[1])


def evaluate_model(model, X_train, y_train):
    kfold = 5
    mse_scores = []
    for fold in range(kfold):
        model.fit(X_train, y_train, epochs=10, batch_size=32,
verbose=0)
        predictions = model.predict(X_train)
        mse = mean_squared_error(y_train, predictions)
        mse_scores.append(mse)
    return np.mean(mse_scores)

nn_score = evaluate_model(model, X_train, y_train)

print(f"Linear Regression MSE: {lr_score:.2f}")
```

```
print(f"Random Forest MSE: {rf_score:.2f}")
print(f"XGBoost MSE: {xgb_score:.2f}")
print(f"Neural Network (TensorFlow/Keras) MSE: {nn_score:.2f}")

best_model_name = min(
    [('Linear Regression', lr_score),
     ('Random Forest', rf_score),
     ('XGBoost', xgb_score),
     ('Neural Network', nn_score)],
    key=lambda x: x[1])[0]

print(f"\nBest model: {best_model_name}")

kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []

print(f"\nКросс-валидация для лучшей модели
({best_model_name}):")
for fold_idx, (train_index, val_index) in
enumerate(kf.split(X_train, y_train)):
    if best_model_name == 'Linear Regression':
        model = LinearRegression()
    elif best_model_name == 'Random Forest':
        model = RandomForestRegressor(n_estimators=100,
random_state=42)
    elif best_model_name == 'XGBoost':
        model = XGBRegressor(n_estimators=100, random_state=42)
    elif best_model_name == 'Neural Network':
        model = create_nn_model(X_train.shape[1])

    if best_model_name == 'Neural Network':
        model.fit(X_train.iloc[train_index],
y_train.iloc[train_index],
                  epochs=10, batch_size=32, verbose=0)
    else:
```

```
        model.fit(X_train.iloc[train_index],
y_train.iloc[train_index])

    y_pred = model.predict(X_train.iloc[val_index])
    mse = mean_squared_error(y_train.iloc[val_index], y_pred)
    mse_scores.append(mse)

    print(f'Фолд {fold_idx + 1}: MSE = {mse:.2f}')

print('\nИтоговые метрики:')
print(f'Среднее      MSE:      {np.mean(mse_scores):.2f}      ±
{np.std(mse_scores):.2f}')

if best_model_name == 'Neural Network':
    final_model = create_nn_model(X_train.shape[1])
    final_model.fit(X_train,  y_train,  epochs=10,  batch_size=32,
verbose=0)
else:
    final_model = model.fit(X_train, y_train)

test_mse              =              mean_squared_error(y_test,
final_model.predict(X_test))
print(f'\nMSE на тестовом наборе: {test_mse:.2f}')
```

Output:

```
Linear Regression MSE: 354.72
Random Forest MSE: 223.82
XGBoost MSE: 224.72
Neural Network (TensorFlow/Keras) MSE: 312.81

Best model: Random Forest

Кросс-валидация для лучшей модели (Random Forest):
Фолд 1: MSE = 223.87
Фолд 2: MSE = 224.69
```

```
Фолд 3: MSE = 226.47
Фолд 4: MSE = 224.15
Фолд 5: MSE = 222.05

Итоговые метрики:
Среднее MSE: 224.25 ± 1.42

MSE на тестовом наборе: 219.45
```