**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_Toc802147807)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc150286719)

[Анализ проблемы исследования 5](#_Toc348866334)

[Исходные данные 5](#_Toc2105919114)

[Реализация проекта 6](#_Toc1828448465)

[Этап 1. Подготовка данных к анализу 7](#_Toc2073685650)

[Этап 2. Предварительный анализ данных 9](#_Toc686554110)

[Этап 3. Корреляционный анализ данных 14](#_Toc559830221)

[Этап 4. Моделирование 16](#_Toc1016037789)

[Этап 5. Прогнозирование 19](#_Toc1741551975)

[Заключение 19](#_Toc1564262114)

[Список использованных источников и литературы 20](#_Toc846465145)

[Приложения 21](#_Toc1095734795)

# **ПАСПОРТ ПРОЕКТА**

# СОДЕРЖАНИЕ ПРОЕКТА

## Анализ проблемы исследования

## Исходные данные

## Реализация проекта

### Этап 1. Подготовка данных к анализу

Подключим все необходимые библиотеки:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import scipy.stats as stats

import statsmodels.api.

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

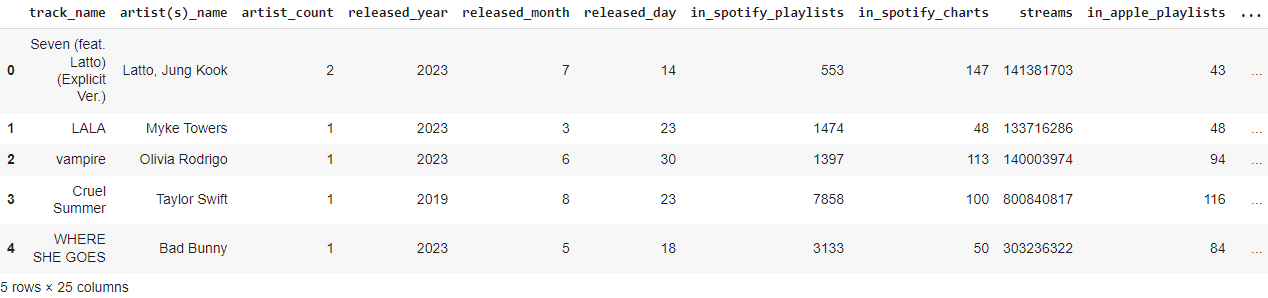
from sklearn.metrics import r2\_score, root\_mean\_squared\_error

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score

Загрузим данные из файла по ссылке и выведем первые 5 строк:

df = pd.read\_csv("https://drive.google.com/uc?id=1ACZWi\_ZwPqKYUpQ0Zt0N9j3-CV6eGD86", sep =',')

df.head()

Рисунок 1. Исходный датафрейм

Сразу удалим нечисловые столбцы, которые не понадобятся для дальнейшего анализа:

df = df.drop(columns=['track\_name', 'artist(s)\_name', 'key', 'mode', 'cover\_url'])

Проверим имеющиеся столбцы на типы данных и пропуски значений. Если присутствуют количественные столбцы с нечисловым типом данных, выполним преобразование типа для них:

df.info()

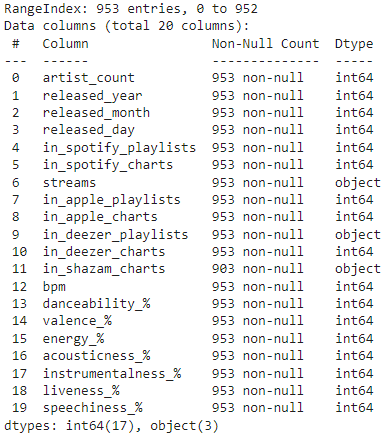


Рисунок 2. Типы данных колонок

Видим, что необходимо преобразовать тип столбцов *streams*, *in\_deezer\_playlists* и *in\_shazam\_charts*, ведь по смыслу они числовые. Так как данные содержат некоторое количество некорректных значений, при преобразовании они заменяются на пустые значения. Таких пропусков получается немного, поэтому удалим строки с ними и преобразуем их уже в целочисленные значения:

# Преобразование столбцов в числовой формат, некорректные значения заменяются на NaN

df['streams'] = pd.to\_numeric(df['streams'], errors='coerce')

df['in\_deezer\_playlists'] = pd.to\_numeric(df['in\_deezer\_playlists'], errors='coerce')

df['in\_shazam\_charts'] = pd.to\_numeric(df['in\_shazam\_charts'], errors='coerce')

# Удаление строк, где столбцы содержат NaN

df = df.dropna(subset=['streams'])

df = df.dropna(subset=['in\_deezer\_playlists'])

df = df.dropna(subset=['in\_shazam\_charts'])

# Преобразование оставшихся значений обратно в целые числа

df['streams'] = df['streams'].astype(int)

df['in\_deezer\_playlists'] = df['in\_deezer\_playlists'].astype(int)

df['in\_shazam\_charts'] = df['in\_shazam\_charts'].astype(int)

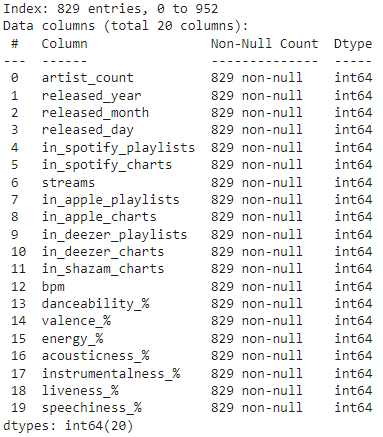


Рисунок 3. Данные о столбцах после обработки

Результатом первого этапа является набор данных в виде датафрейма, готовый для анализа.

### **Этап 2. Предварительный анализ данных**

Вычислим описательные статистики по колонкам (среднее, моду, медиану, стандартное отклонение).

Среднее арифметическое выборки — это число, равное отношению суммы всех чисел выборки к их количеству.

Мода выборки — это значение, которое встречается в выборке чаще остальных.

Медиана выборки — это центральное значение упорядоченного ряда данных. Если расположить числа по возрастанию, медиана будет находиться посередине: половина значений окажется меньше, а другая половина — больше.

Стандартное отклонение — это ключевой показатель, характеризующий степень разброса значений относительно их среднего. Оно вычисляется как квадратный корень из дисперсии.

df.describe()

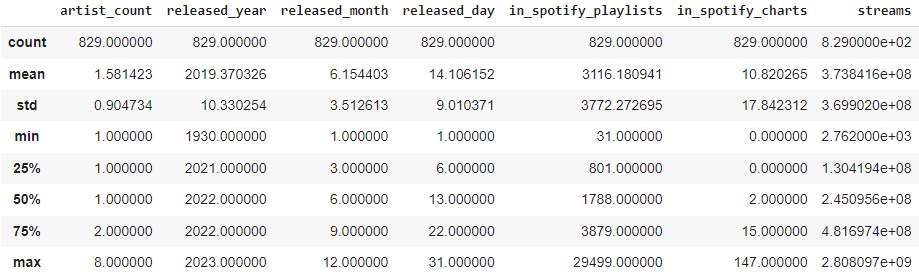


Рисунок 4. Описательные статистики по колонкам

Аномальные наблюдения (выбросы) — это значения, которые сильно отличаются от большинства других значений в наборе данных. Они могут возникнуть из-за ошибок ввода, манипуляций с данными или аномальных процессов.

Для определения выбросов можно использовать графический метод. Например, график «ящик с усами» (boxplot) показывает медиану, нижний и верхний квартили, а также выбросы в данных. Выбросы определяются как значения, находящиеся за пределами «усов», которые представляют 1,5 межквартильного размаха.

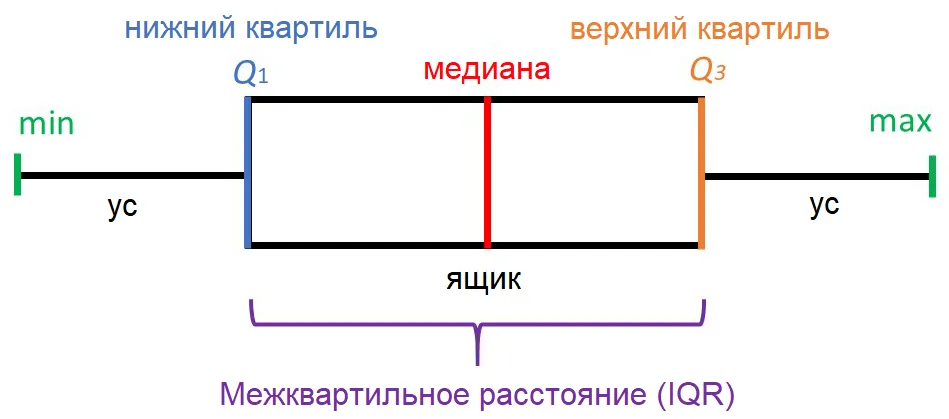


Рисунок 5. “Ящик с усами”

Построим “Ящик с усами” для каждого числового признака на отдельном графике в матрице:

fig, axes = plt.subplots(4, 5, figsize=(20, 15))

# Перебираем все столбцы и создаем boxplot на каждой оси

for i, col in enumerate(df.columns):

ax = axes[i // 5, i % 5] # Определяем расположение на сетке

ax.boxplot(df[col])

ax.set\_title(col) # Добавляем название столбца

plt.tight\_layout() # Делаем оформление компактным

plt.show()

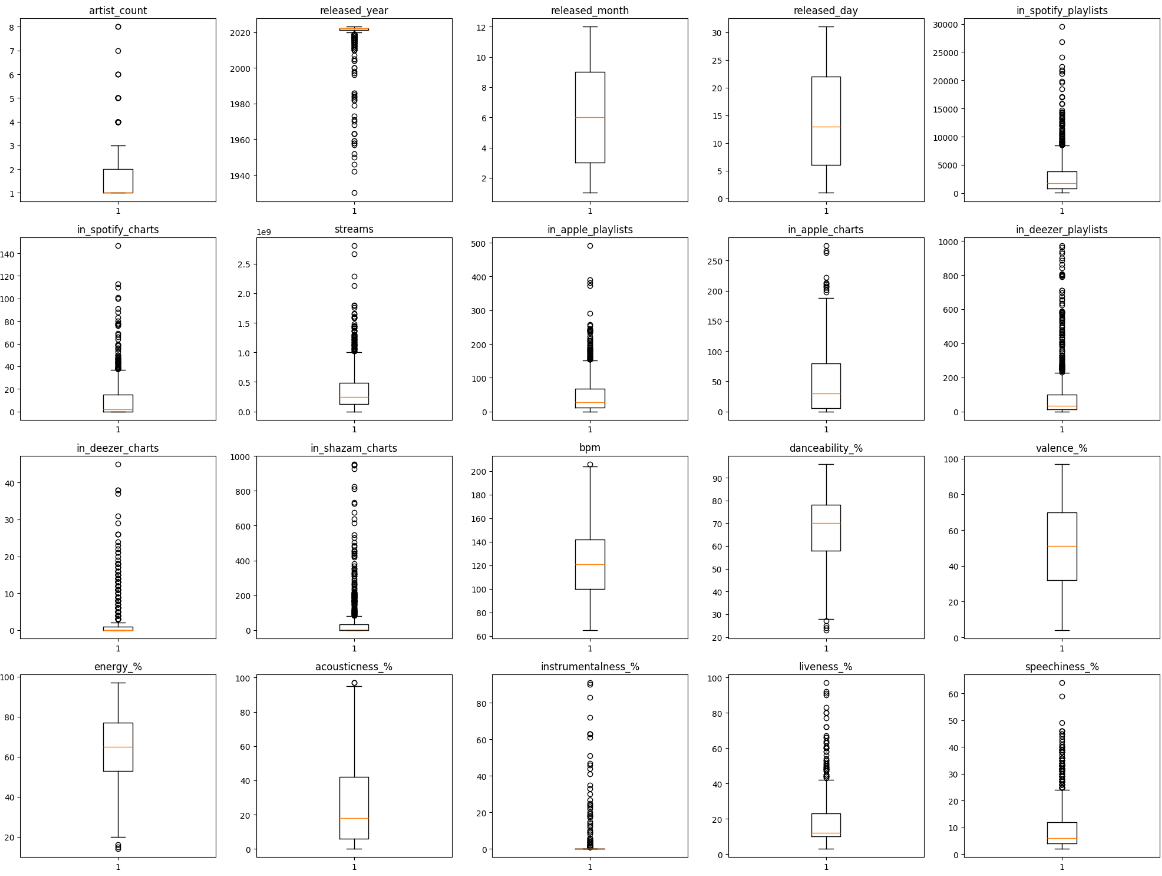


Рисунок 6. Графики выбросов для всех числовых признаков

Сильные выбросы можно заметить на графиках для *in\_spotify\_charts*, *in\_apple\_playlists* и *in\_deezer\_charts.* Удалим эти выбросы:

df['in\_spotify\_charts'].replace(df['in\_spotify\_charts'].max(), df['in\_spotify\_charts'].quantile(0.99), inplace=True)

df['in\_apple\_playlists'].replace(df['in\_apple\_playlists'].max(), df['in\_apple\_playlists'].quantile(0.99), inplace=True)

df['in\_deezer\_charts'].replace(df['in\_deezer\_charts'].max(), df['in\_deezer\_charts'].quantile(0.99), inplace=True)

Снова выполним код выше для построения графиков и проверим, удалось ли избавиться от выбросов.

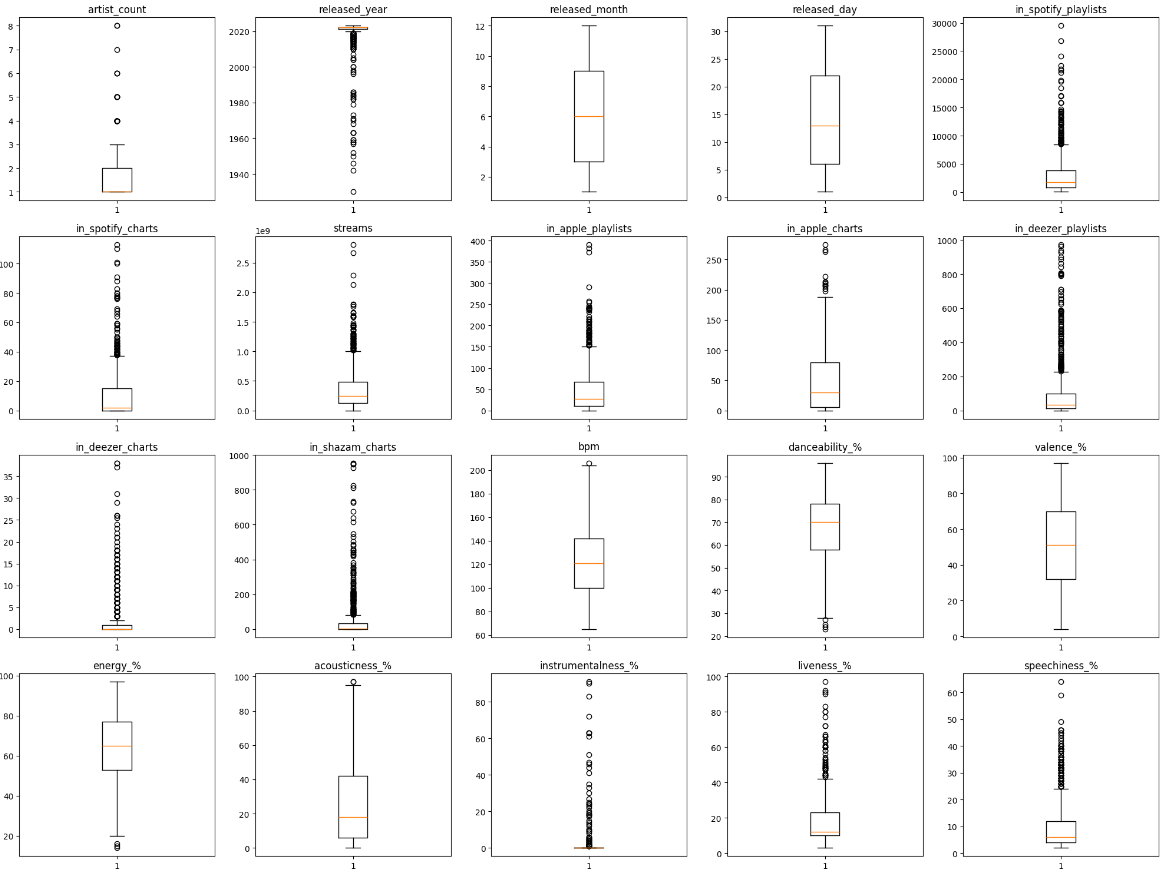


Рисунок 7. Графики выбросов после обработки

От самых заметных выбросов получилось избавиться.

Проверим данные на нормальность распределения. Для начала построим гистограммы по всем числовым признакам.

Гистограмма — это графическое представление распределения числовых данных в виде столбцов. Каждый столбец показывает количество (или частоту) значений, попадающих в определённый диапазон (интервал).

Гистограмма строится по следующему алгоритму:

* Определяются диапазоны (интервалы) значений,
* Подсчитывается количество данных, попадающих в каждый интервал,
* Строятся столбцы: высота каждого столбца соответствует частоте значений в соответствующем интервале.



Рисунок 8. Гистограмма нормального распределения

fig, axes = plt.subplots(4, 5, figsize=(20, 15))

# Перебираем все столбцы и создаем гистограмму на каждой оси

for i, col in enumerate(df.columns):

ax = axes[i // 5, i % 5] # Определяем расположение на сетке

ax.hist(df[col], bins=30, linewidth=1, edgecolor="black")

ax.set\_title(col) # Добавляем название столбца

plt.tight\_layout() # Делаем оформление компактным

plt.show()

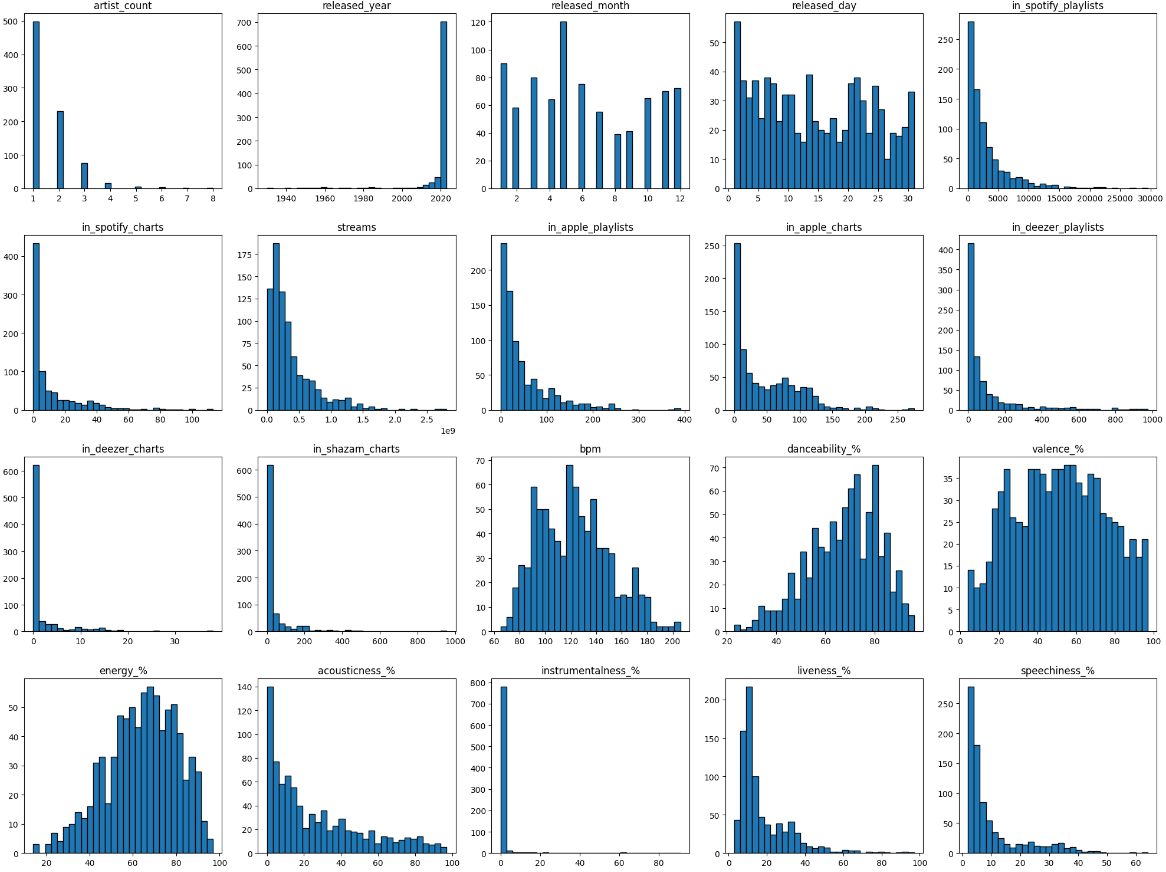


Рисунок 9. Гистограммы числовых признаков

Визуально сложно понять, имеет ли какой-либо из признаков нормальное распределение. Для более точной оценки проверим столбцы с помощью статистических критериев.

Критерий Шапиро-Уилка — один из статистических тестов, которые проверяют, соответствует ли распределение выборки нормальному.

Вычисляется статистика W, основанная на соотношении дисперсий данных. Если W близка к 1, выборка, скорее всего, нормально распределена. Также выдается значение p:

* Если p > 0.05, гипотеза о нормальности принимается,
* Если p ≤ 0.05, гипотеза отвергается (распределение ненормальное).

for col in df.columns:

print(f'{col}, W: {stats.shapiro(df[col])[0]:.3f} pvalue: {stats.shapiro(df[col])[1]}')

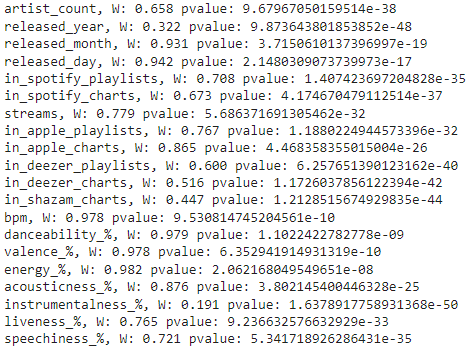


Рисунок 10. Результаты теста Шапиро-Уилка

Хоть для некоторых столбцов W статистика близка к значению 1, значение p для всех столбцов меньше 0.05, следовательно на 5% уровне значимости теория о нормальности распределения какого-либо из столбцов отвергается.

### Этап 3. Корреляционный анализ данных

Корреляционный анализ — это методы статистики, используемые для оценки степени взаимосвязи между двумя переменными. Он помогает определить, насколько сильно одна переменная изменяется при изменении другой.

Парный коэффициент корреляции показывает степень линейной зависимости между двумя переменными X и Y.

Если переменные связаны линейно, связь выражается коэффициентом корреляции. Значение коэффициента лежит в диапазоне от −1 до 1, связь характеризуется следующим образом:

* r∈[0.7,1] или [−0.7,−1]: сильная связь.
* r∈[0.3,0.7) или [−0.3,−0.7): умеренная связь.
* r∈[0,0.3) или [0,−0.3): слабая связь.

Выведем тепловую карту корреляции наших признаков:

corr\_matrix = df.corr()

plt.figure(figsize=(15, 8))

sns.heatmap(corr\_matrix, annot=True)

plt.title("Корреляционная матрица")

plt.show()

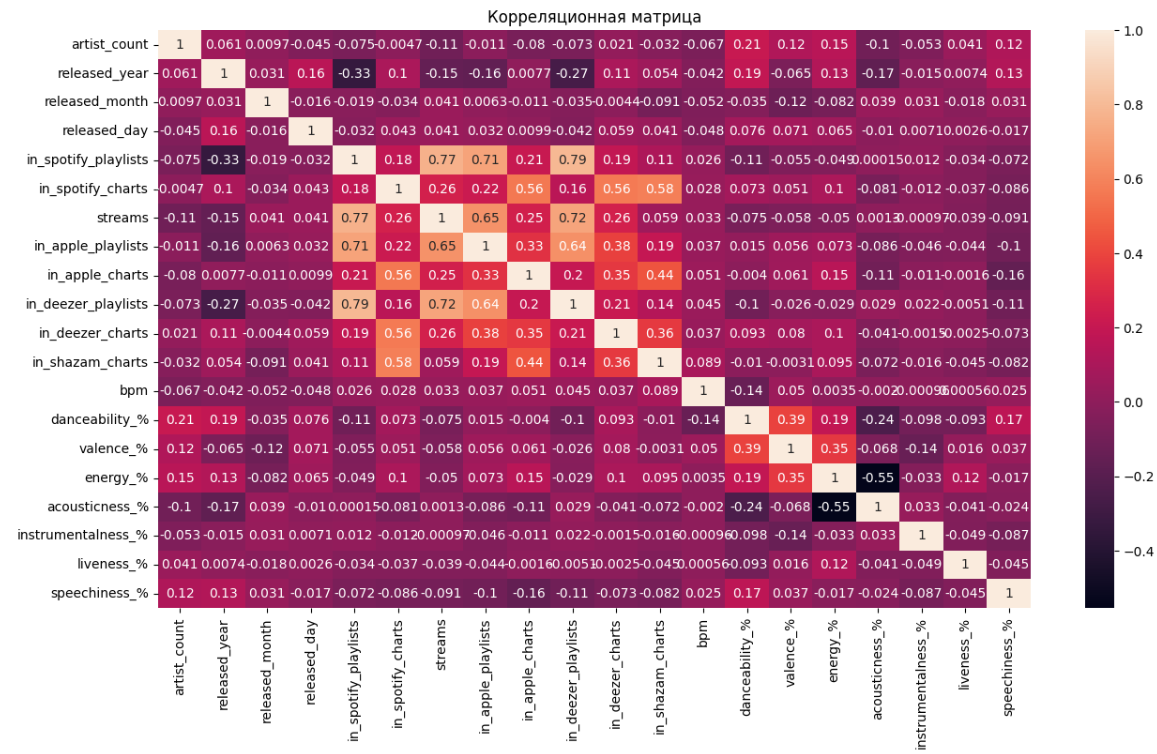


Рисунок 11. Тепловая карта корреляционной матрицы

В качестве целевой функции возьмем количество прослушиваний *streams*, так как это ключевой показатель, характеризующий популярность исполнителя.

Выведем признаки, наиболее коррелирующие с целевой функцией:

print(corr\_matrix['streams'].sort\_values(ascending=False))

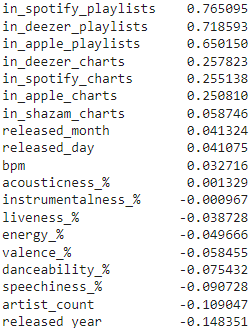


Рисунок 12. Коэффициенты корреляции с целевой функцией

Проверим данные в целевом столбце на упорядоченность:

Y = df['streams']

print(f'pvalue: {statsmodels.sandbox.stats.runs.runstest\_1samp(Y, cutoff="median")[1]}')

Результат: pvalue: 1.925906048356783e-06

Значение p близко к 0, следовательно с достаточно большой долей вероятности можно сказать, что выборка не случайна. Перемешаем таблицу случайным образом:

df = df.sample(frac=1, random\_state=42).reset\_index(drop=True)

Таким образом на третьем этапе была выдвинута гипотеза о зависимости количества прослушиваний песни на Spotify от количества испонителей, дня и месяца релиза, количества присутствий песни в плейлистах и чартах на стриминговых площадках и характеристик песни.

### Этап 4. Моделирование

Выделим целевую функцию и факторные показатели в перемешанной таблице и разобъем данные на обучающую и тестовую выборки в отношении 7 к 3:

Y = df['streams']

X = df.drop(columns=['streams'])

# Разделяем на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.3, random\_state=42)

Построим модель множественной линейной регрессии сначала используя все числовые признаки:

# Выбираем модель линейной регресси и обучаем

model = LinearRegression()

model.fit(X\_train, y\_train)

# Получаем прогнозы

y\_train\_pred = model.predict(X\_train)

Вычислим коэффициент детерминации и среднеквадратическую ошибку на обучающей выборке:

r2\_train = r2\_score(y\_train, y\_train\_pred)

rmse\_train = root\_mean\_squared\_error(y\_train, y\_train\_pred)

print("Метрики на обучающей выборке:")

print(f"R2: {r2\_train:.3f}")

print(f"RMSE: {rmse\_train:.0f}")

Результат:  
R2: 0.712  
RMSE: 206480850

Значения не очень высокие, проверим модель на значимость (ее коэффициент детерминации R2 значимо не равен 0), для этого используем критерий Фишера:

df1, df2 = 19, len(X\_train) - 20

F\_test = (r2\_train / (1 - r2\_train) \* (df2 / df1)) >= stats.f.ppf(1 - 0.05, dfn=df1, dfd=df2)

print('Модель значима:', F\_test)

Результат: Модель значима: True

Следовательно на уровне значимости 5% можно сделать вывод, что модель значима. Однако количество факторных признаков построенной модели велико (19), попробуем сократить их количество, взяв 3 самых значимых:

X = df[['in\_spotify\_playlists', 'in\_deezer\_playlists', 'in\_apple\_playlists']] # Коэффициент корреляции > 0.5

# Разделяем на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.3, random\_state=42)

# Выбираем модель линейной регресси и обучаем

model.fit(X\_train, y\_train)

# Получаем прогнозы

y\_train\_pred = model.predict(X\_train)

# Вычисляем метрики

r2\_train = r2\_score(y\_train, y\_train\_pred)

rmse\_train = root\_mean\_squared\_error(y\_train, y\_train\_pred)

print("Метрики на обучающей выборке:")

print(f"R2: {r2\_train:.3f}")

print(f"RMSE: {rmse\_train:.0f}")

Получившиеся метрики:  
R2: 0.668  
RMSE: 221721972

Судя по метрикам качество модели при уменьшении числа факторов снижается, но незначительно. Также проверим модель на значимость:

df1, df2 = 3, len(X\_train) - 4

F\_test = (r2\_train / (1 - r2\_train) \* (df2 / df1)) >= stats.f.ppf(1 - 0.05, dfn=df1, dfd=df2)

print('Модель значима:', F\_test)

Результат: Модель значима: True

Мы смоги значительно уменьшить количество факторных признаков, при этом не сильно снизив качество модели. Выведем коэффициенты модели:

print("Коэффициенты модели:")

for i in range(len(model.coef\_)):

print(f'{X.columns[i]}:\t{model.coef\_[i]:.3f}')

print(f'Свободный член:\t\t{model.intercept\_:.0f}')

Результат:  
in\_spotify\_playlists: 36317.863  
in\_deezer\_playlists: 713450.389  
in\_apple\_playlists: 1354373.510  
Свободный член: 128175048

Модель зависимости количества прослушиваний имеет вид:  
***streams*** = 128175048 + 36317.863 \* *in\_spotify\_playlists* + 713450.389 \* *in\_deezer\_playlists* + 1354373.510 \* *in\_apple\_playlists*

Интерпретация коэффициентов:

* Свободный член: при нулевых значениях факторных переменных количество прослушиваний составит 128175048.
* Коэффициент при *in\_spotify\_playlists*: при добавлении песни еще один плейлист на Spotify количество прослушиваний увеличивается на 36317.863,
* Коэффициент при *in\_deezer\_playlists*: при добавлении песни еще один плейлист на Deezer количество прослушиваний увеличивается на 713450.389,
* Коэффициент при *in\_apple\_playlists*: при добавлении песни еще один плейлист на Apple Music количество прослушиваний увеличивается на 1354373.510.

Выглядит итоговая формула и коэффициенты нереалистично, однако объяснить это можно тем, что в наборе данных взяты показатели самых популярных песен с огромными значениями прослушиваний у каждой.

### Этап 5. Прогнозирование

С помощью получившейся модели сделаем прогноз на тестовой выборке и вычислим для него метрики:

y\_test\_pred = model.predict(X\_test)

r2\_test = r2\_score(y\_test, y\_test\_pred)

rmse\_test = root\_mean\_squared\_error(y\_test, y\_test\_pred)

print("\nМетрики на тестовой выборке:")

print(f"R2: {r2\_test:.3f}")

print(f"RMSE: {rmse\_test:.0f}")

Результат:  
Метрики на тестовой выборке:  
R2: 0.512  
RMSE: 229285892

# Заключение

В результате проведенного регрессионного анализа была проверена гипотеза о том, что количество прослушиваний песни на Spotify зависит от количества исполнителей, дня и месяца выпуска, количества присутствий песни в плейлистах и чартах на стриминговых площадках и характеристик песни. Анализ показал, что такая зависимость действительно существует, и эти показатели могут быть использованы для предсказания прослушиваний.

Наибольшее влияние на количество прослушиваний оказывает количество присутствий песни в плейлистах на платформах Spotify, Deezer и Apple Music, что подтверждается значением коэффициентов модели. Несмотря на то, что точность модели оказалась невысокой, результаты F-теста на уровне значимости 5% свидетельствуют о её статистической значимости. Это указывает на то, что включённые в анализ факторы оказывают значимый эффект, даже если предсказательная мощность модели ограничена.

# Список использованных источников и литературы

1. Google Диск : файл : 954 записи. URL: https://drive.google.com/uc?id=1ACZWi\_ZwPqKYUpQ0Zt0N9j3-CV6eGD86 (дата обращения: 9.12.2024).
2. Протодьяконов, А. В. Алгоритмы Data Science и их практическая реализация на Python : учебное пособие / А. В. Протодьяконов, П. А. Пылов, В. Е. Садовников. — Москва, Вологда : Инфра-Инженерия, 2022. — 392 c. — ISBN 978-5-9729-1006-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт].
3. Exploding Topics : official site. URL: <https://explodingtopics.com/blog/music-streaming-stats> (дата обращения: 9.12.20224).

# Приложения

Приложение 1

Программный код

Программный код доступен для просмотра и выполнения на платформе Google Colab по ссылке: <https://colab.research.google.com/drive/1Gg_W3wTXhOEbppV9nrIYDH4mMxkmGRrt?usp=sharing>