|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» (ПГНИУ)  Региональный институт непрерывного образования (РИНО ПГНИУ)  Цифровая кафедра | | | | | |
|  | |  |  | | |
|  |  | | | |  |
|  | |  | |  | |
| Выпускная аттестационная (квалификационная) работа  по курсу профессиональной переподготовки «Анализ данных и машинное обучение»  АНАЛИЗ ФАКТОРОВ, ВЛИЯЮЩИХ НА ПРОДАЖИ ВИДЕОИГР | | | | | |
|  | | |  | | |
|  | | | Разработчики проекта:  Базеев Александр Константинович,  Никитин Иван Викторович,  Язик Мария Денисовна | | |
|  | |  | |  | |
| Пермь 2024 | | | | | |

Оглавление

[ПАСПОРТ ПРОЕКТА 3](#_Toc185820390)

[СОДЕРЖАНИЕ ПРОЕКТА АНАЛИЗ ПРОБЛЕМЫ ИССЛЕДОВАНИЯ 4](#_Toc185820391)

[1 Исходные данные 5](#_Toc185820392)

[2 Реализация проекта 6](#_Toc185820393)

[Этап 1. Выбор средств проектирования системы 6](#_Toc185820394)

[Этап 2. Предварительный анализ данных 7](#_Toc185820395)

[Этап 3. Корреляционный анализ данных 11](#_Toc185820396)

[Этап 4. Моделирование и прогнозирование 13](#_Toc185820397)

[Заключение 20](#_Toc185820398)

[Библиографический список 21](#_Toc185820399)

[Приложение А Диаграмма прецендентов 23](#_Toc185820400)

Паспорт проекта

**Название проекта**: Анализ факторов, влияющих на продажи видеоигр

**Сведения об авторах**: Базеев Александр Константинович, Никитин Иван Викторович, Язик Мария Денисовна

**Цель**: выполнить анализ данных о продажах видеоигр и построить модель зависимости продаж от различных факторов с помощью регрессии, позволяющую делать прогнозы с высокой точностью.

**Задачи:**

1. Провести анализ проблемы и обосновать её актуальность.
2. Загрузить данные и подготовить их для количественного анализа, включая обработку пропущенных значений.
3. Выполнить предварительный анализ данных, включая выявление и обработку выбросов, проверку нормальности распределения данных и проведение корреляционного анализа.
4. Построить модель зависимости целевого признака от факторов с использованием линейной регрессии, подобрать наилучшую модель, оценить её качество и выполнить прогнозирование.
5. Проинтерпретировать полученные результаты и сделать выводы о достижении поставленной цели.

**Краткое описание проекта:**

Требуется проанализировать данные по продажам видеоигр и определить, возможно ли описать зависимость общих продаж по миру от имеющихся факторных переменных с помощью регрессии. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построенная регрессионная модель и рекомендации по ее использованию.

содержание проекта  
Анализ проблемы исследования

С каждым годом видеоигры становятся все более популярными среди молодежи. Игровая индустрия демонстрирует стремительный рост и развитие, что подтверждается значительными экономическими показателями. В 2022 году объем рынка видеоигр достиг 184,4 млрд долларов. Это привлекает внимание не только игроков, но и инвесторов, разработчиков и исследователей. Важно понимать, какие факторы способствуют такому быстрому росту, а также как видеоигры влияют на молодежь и общество в целом.

Рост популярности видеоигр привлекает значительные инвестиции. Что способствует увеличению качества и разнообразие игр от одного издателя. С помощью дополнительного финансирования разработчики могут внедряют новые механики, делая игровой процесс все более увлекательным.

Уже давно видеоигры перестали быть только развлечением, где нет особого сюжета. Современные видеоигры часто имеют сложные и оригинальные сюжеты, которые могут соперничать с фильмами-лауреатами престижных премий. На сегодняшний день стали популярны киноадаптации известных игр.

Растущая популярность специальных премий, таких как The Game Awards, отражает серьезное отношение к игровой индустрии. Данные премии помогают знакомить аудиторию с новыми проектами и привлекать еще больше спонсоров на развитие проектов.

Анализ проблемы исследования показывает, что видеоигры и игровая индустрия представляют собой сложную и многоаспектную область, требующую глубокого изучения. Экономический рост, развитие технологий, социальное и культурное влияние, а также признание в виде премий и киноадаптаций — все эти факторы взаимосвязаны и влияют на развитие отрасли. Понимание этих аспектов поможет не только разработчикам и инвесторам, но и исследователям и обществу в целом лучше осознать значение видеоигр в современном мире.

Анализ данной ситуации и определяет актуальность темы исследования. Таком образом, необходимо проанализировать данные по продажам видеоигр и определить, возможно ли описать зависимость общих продаж по миру от имеющихся факторных переменных с помощью регрессии

*Цель*: выполнить анализ данных о продажах видеоигр и построить модель зависимости продаж от различных факторов с помощью регрессии, позволяющую делать прогнозы с высокой точностью.

*Задачи:*

1. Провести анализ проблемы и обосновать её актуальность.
2. Загрузить данные и подготовить их для количественного анализа, включая обработку пропущенных значений.
3. Выполнить предварительный анализ данных, включая выявление и обработку выбросов, проверку нормальности распределения данных и проведение корреляционного анализа.
4. Построить модель зависимости целевого признака от факторов с использованием линейной регрессии, подобрать наилучшую модель, оценить её качество и выполнить прогнозирование.
5. Проинтерпретировать полученные результаты и сделать выводы о достижении поставленной цели.
6. Исходные данные

В данной работе анализируется список рейтинг самых лучших по продажам по всему миру видеоигр.

Список колонок анализируемого набора данных:

1. Rank – позиция игры в рейтинге по глобальным продажам.
2. Name – название игры.
3. Platform – платформа, на которой была выпущена игра.
4. Year – год выпуска видеоигры.
5. Genre – жанр видеоигры.
6. Publisher – издатель видеоигры.
7. NA\_Sales – продажи игры в Северной Америке.
8. EU\_Sales – продажи игры в Европе.
9. JP\_Sales – продажи игры в Японии
10. Other\_Sales – продажи игры в других регионах.
11. Global\_Sales – общие продажи игры по всему миру.

Необходимо проанализировать данные по играм и определить, можно ли описать зависимость количество продаж от имеющихся факторных переменных различными методами регрессии.

Выдвинем гипотезу исследования: глобальное количество продаж видеоигр можно предсказать с высокой точностью зная жанр, издателя и платформу, на которой выпускается игра.

1. Реализация проекта
2. Выбор средств проектирования системы

Подключим необходимые библиотеки и установим seaborn по умолчанию для отрисовки графиков:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

Смонтируем диск и загрузим датасет

from google.colab import drive

drive.mount('/content/drive/')

games = pd.read\_csv('/content/drive/MyDrive/vgsales.csv')

Выведем несколько первых строчек датасета:

games.head()

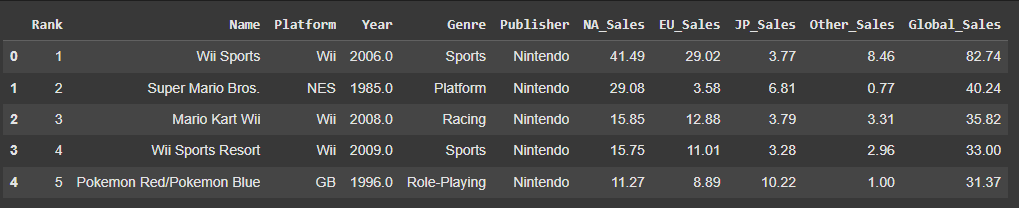


Рисунок 1 – Исходный датасет

Выведем сводную статистику всех числовых столбцов:

games.describe()

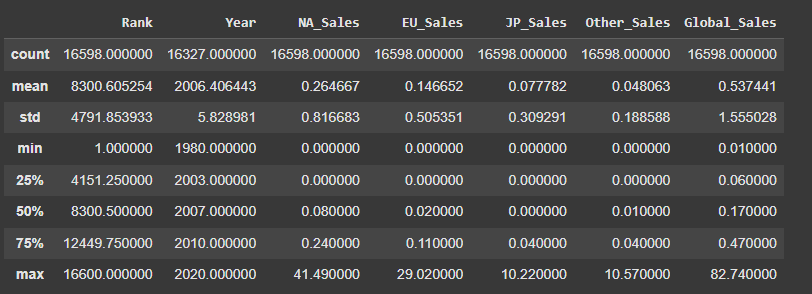


Рисунок 2 – Статистика числовых столбцов

Выведем сводную статистику всех столбцов с объектным типом данных:

games.describe(include=['O'])



Рисунок 3 – Статистика столбцов с объектным типом данных

Выведем информацию о типах данных колонок:

games.info()

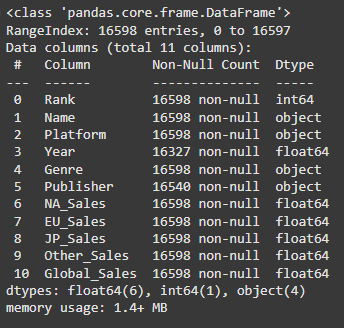


Рисунок 4 – Типы данных колонок

Проверим данные на наличие пустых значений:

games.isnull().sum()



Рисунок 5 – Количество пустых значений

Видим пустые значения в некоторых столбцах, построим тепловую карту:

cols = games.columns

# определяем цвета

# желтый - пропущенные данные, синий - не пропущенные

colours = ['#000099', '#ffff00']

sns.heatmap(games[cols].isnull(), cmap=sns.color\_palette(colours))

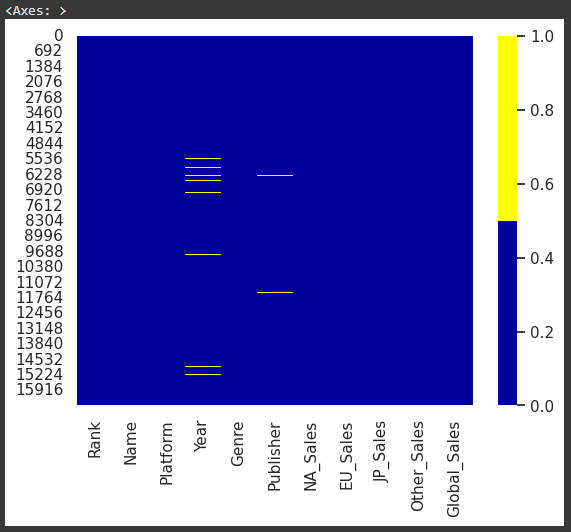


Рисунок 6 – Тепловая карта с пустыми значениями

Заполняем пропущенный значения в столбике «Year» значением медианы по этому столбцу:

med = games['Year'].median()

games['Year'] = games['Year'].fillna(med)

Удалим строчки с пустыми значениями в столбце «Publisher»:

games.dropna(axis=0, inplace=True)

Выведем заново тепловую карту:

cols = games.columns

# определяем цвета

# желтый - пропущенные данные, синий - не пропущенные

colours = ['#ffff00', '#000099']

sns.heatmap(games[cols].isnull(), cmap=sns.color\_palette(colours))

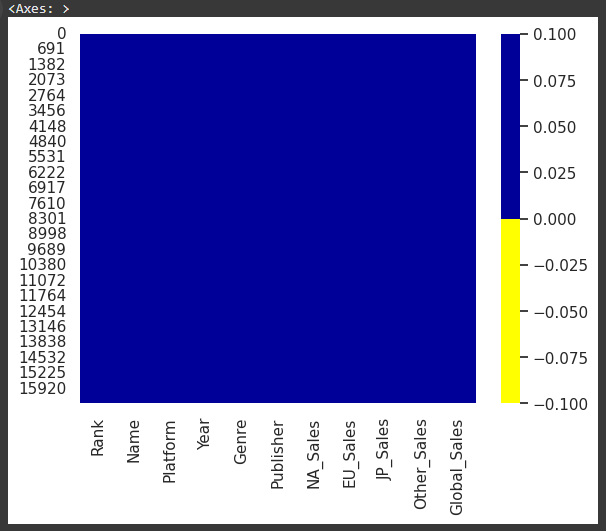


Рисунок 7 - Тепловая карта без пустых значений

1. Предварительный анализ данных

Вычислим среднее, моду, медиану, стандартное отклонение и квартили.

Для начала разберемся в сути этих статистик. Среднее арифметическое это сумма значений всех вариант выборки деленное на объем выборки. Мода - это значение признака, которое наиболее часто встречается в выборке. Медиана – значение признака, которое находится в середине ранжированного ряда.

numeric\_games = games.select\_dtypes(include=['number'])

numeric\_games.drop("Rank",axis=1,inplace=True)

#Среднее

mean = numeric\_games.mean()

# Мода

mode = numeric\_games.mode().iloc[0]

# Медиана

median = numeric\_games.median()

# Стандартное отклонение

std\_dev = numeric\_games.std()

# Квартили

quartiles = numeric\_games.quantile([0.25, 0.5, 0.75])

# Объединение результатов в один DataFrame

descriptive\_stats = pd.DataFrame({

    'Mean': mean,

    'Mode': mode,

    'Median': median,

    'Standard Deviation': std\_dev,

    'Q1': quartiles.loc[0.25],

    'Q2': quartiles.loc[0.5],

    'Q3': quartiles.loc[0.75]

})

print(descriptive\_stats)

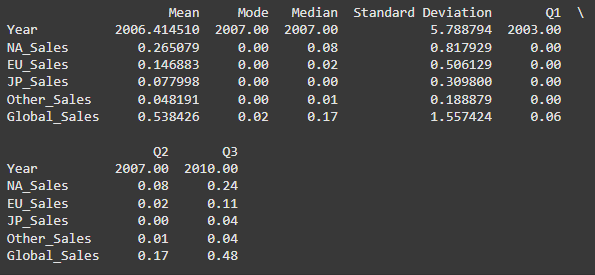


Рисунок 8 – Описательные статистики по колонкам

Далее проверим данные на наличие выбросов, используя диаграмму «ящик с усами» (boxplot).

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

axes = axes.flatten()

for i, col in enumerate(numeric\_games.columns):

    games.boxplot(column=[col], ax=axes[i])

    axes[i].set\_title(col)

plt.tight\_layout()

plt.show()

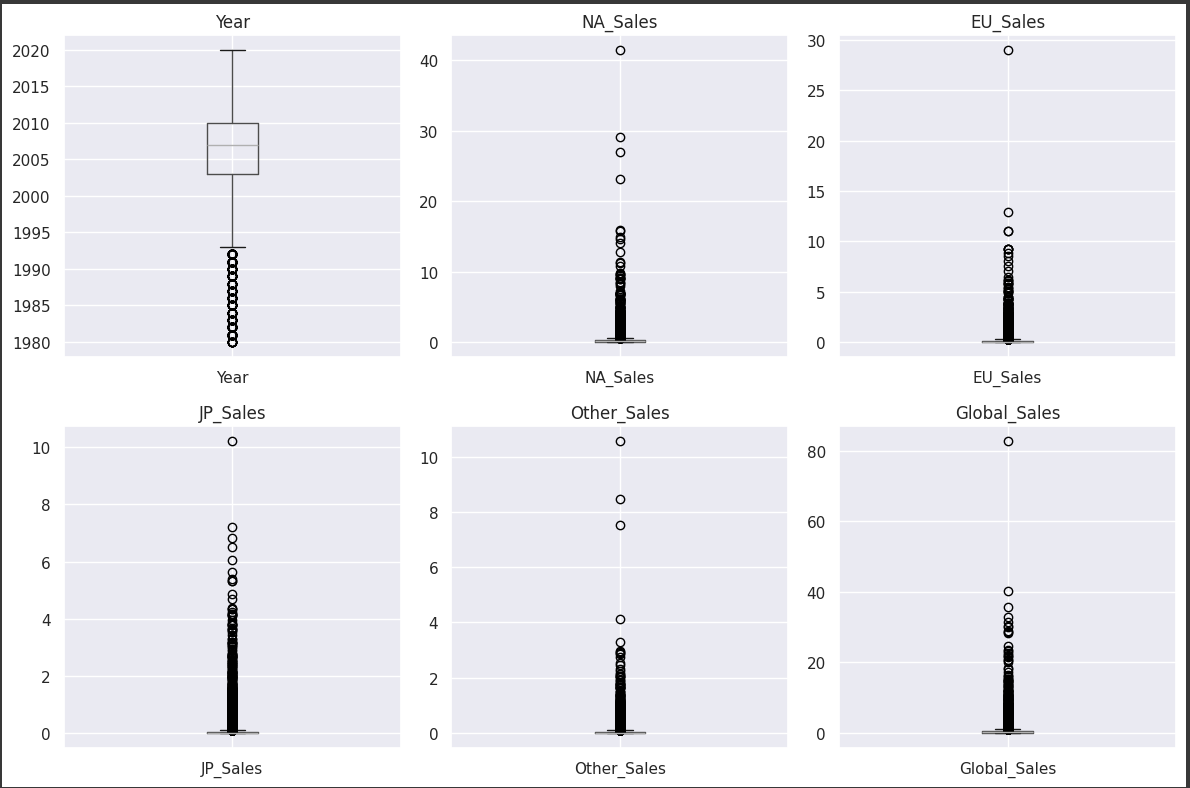


Рисунок 9 – Выбросы

Смотрим распределение по каждой колонке:

numeric\_games.hist(figsize=(15, 10), bins=10)

plt.suptitle('Histograms of Columns', fontsize=16)

plt.show()

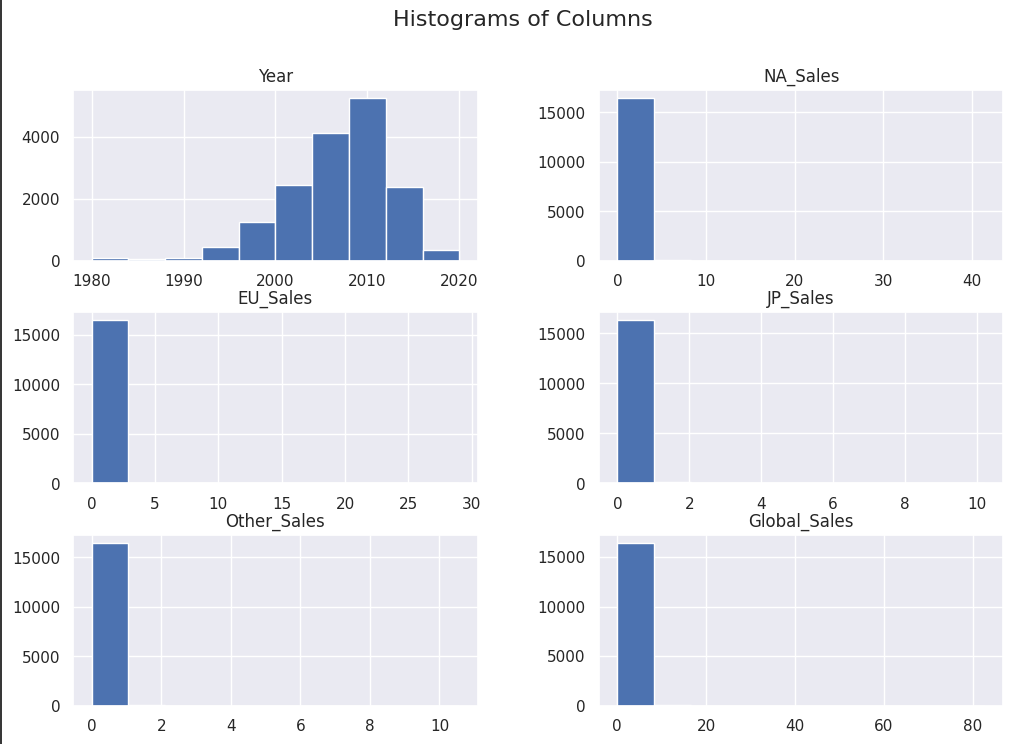


Рисунок 10 – Распределение по каждой колонке

Из-за года не видно распределения продаж, поэтому построим график без года и зададим ограничение по y:

fig, axes = plt.subplots(len(numeric\_games.drop("Year", axis=1).columns), 1, figsize=(15, 10))

for i, col in enumerate(numeric\_games.drop("Year", axis=1).columns):

    axes[i].hist(numeric\_games[col], bins=10)

    axes[i].set\_ylim(0, 100)

    axes[i].set\_title(f'Histogram of {col}')

plt.suptitle('Histograms of Columns', fontsize=16)

plt.tight\_layout()

plt.show()

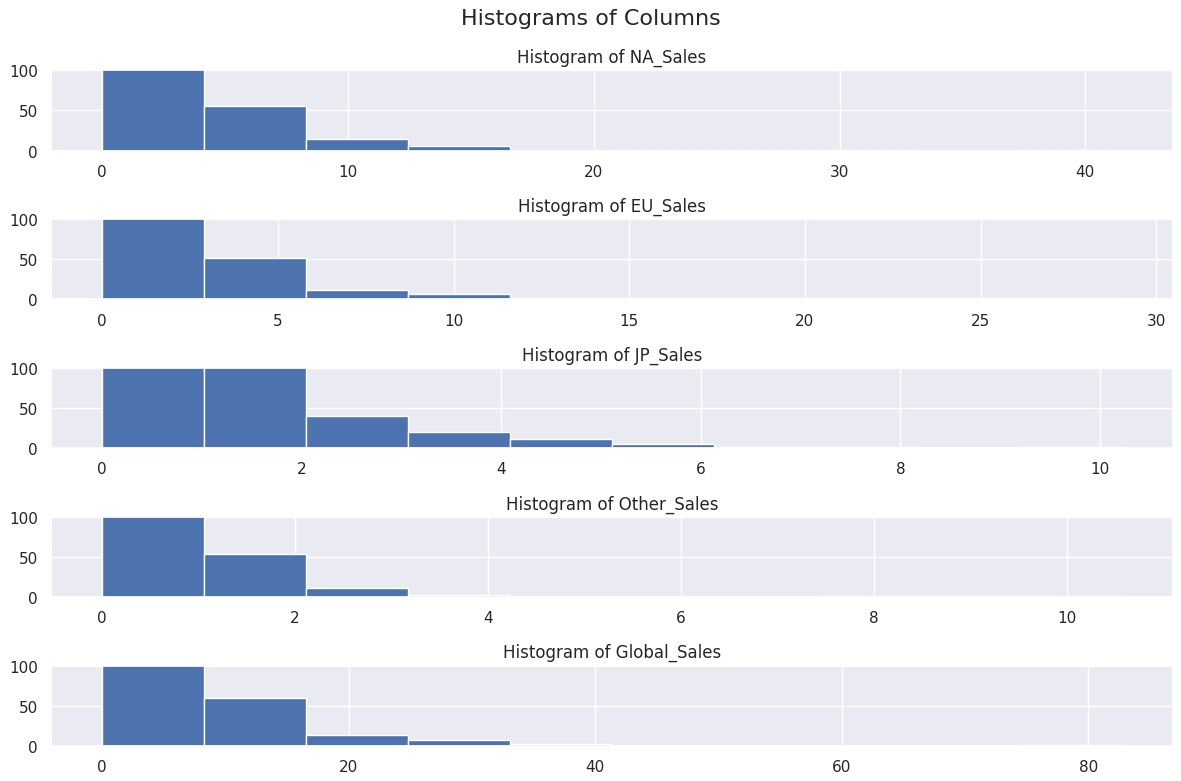


Рисунок 11 – Распределение по продажам

На нормальное распределение похож столбец год, поэтому проверим, нормально ли в нём распределены данные:

import scipy.stats as stats

for col in numeric\_games.columns:

  data = numeric\_games[col]

  mu = np.mean(data)

  sigma = np.std(data)

  # Тест Колмогорова-Смирнова

  alpha = 0.05

  d\_statistic, p\_value = stats.kstest(data, 'norm', args=(mu, sigma))

  print('p-значение:', p\_value)

  if p\_value > alpha:

      print('Не удалось отвергнуть гипотезу о нормальности')

  else:

      print('Гипотеза о нормальности отвергнута')

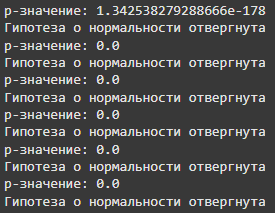


Рисунок 12 – Опровержение гипотезы о нормальном распределении столбца год

В датасете необходимые для обучения модели данные являются категориальными. Заменим их на количественные:

from sklearn.preprocessing import OrdinalEncoder

enc = OrdinalEncoder()

games[['Platform', 'Genre', 'Publisher']] = enc.fit\_transform(games[['Platform', 'Genre', 'Publisher']])

games.head()

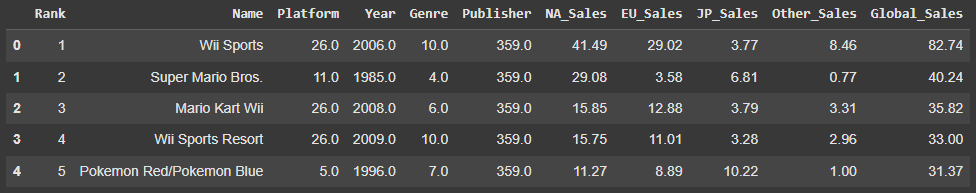


Рисунок 13 – Датасет после замены данных

1. Корреляционный анализ данных

Корреляционный анализ данных представляет собой статистический метод, применяемый для изучения зависимости между двумя или более переменными. Основная задача этого анализа — выявить степень взаимосвязи между изменениями различных переменных. Так как данные всех колонок не имеют нормального распределения, поэтому используем ранговый коэффициент Спирмена. Матрицу корреляции строим с помощью диаграммы «тепловая карта» (heatmap).

correlation\_matrix = games.drop(["Name", "Rank", "NA\_Sales", "EU\_Sales", "JP\_Sales", "Other\_Sales"], axis=1).corr(method='spearman')

# Визуализация с помощью тепловой карты

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, cbar\_kws={"shrink": .8})

plt.title('Матрица корреляции (Спирмен)')

plt.show()

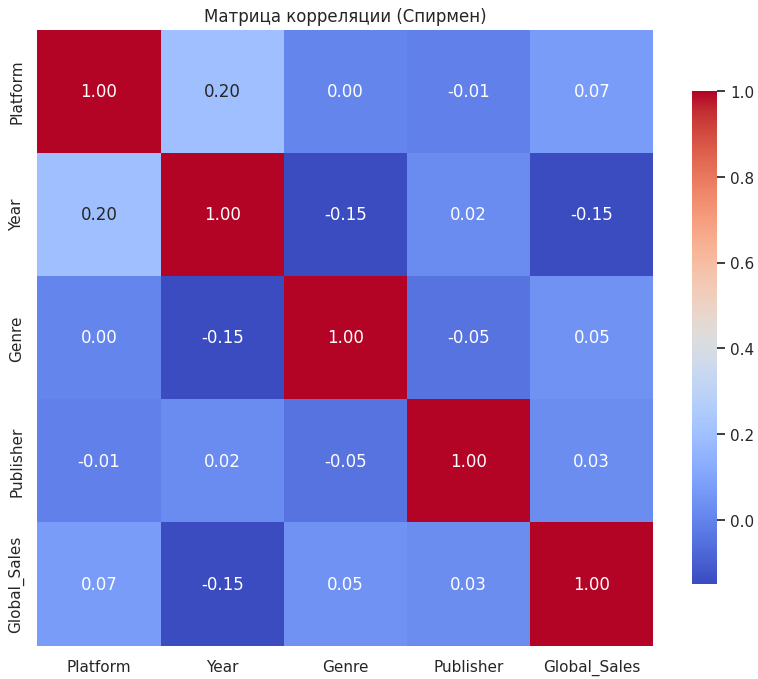


Рисунок 14 – Тепловая карта

Есть подозрение на тренд, проверим с помощью теста Манна-Кендалла на упорядоченность. В случае, если данные упорядочены, перемешаем их.

from scipy.stats import kendalltau

from sklearn.utils import shuffle

correlation, p\_value = kendalltau(range(len(games)), games['Global\_Sales'])

alpha = 0.05

print(f"Корреляция Манна-Кендалла: {correlation:.3f}")

print(f"p-значение: {p\_value:.3f}")

if p\_value < alpha:

    print("Данные, вероятно, упорядочены.")

    games = shuffle(games)

    print("Данные перемешаны.")

else:

    print("Данные, вероятно, не упорядочены.")

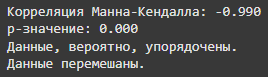


Рисунок 15 – Проверяем упорядоченность данных

Выведем данные и проверим, как они перемешались:

games.head()

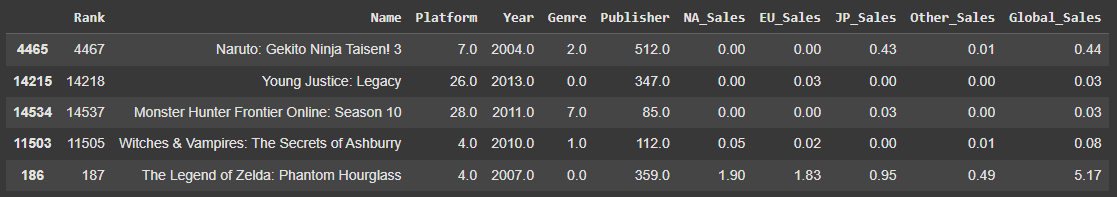


Рисунок 16 – Датасет после перемешки

1. Моделирование и прогнозирование

Используем функцию, которая вычислит метрики для моделей. Они определяют, насколько хорошо модель регрессии соответствует набору данных RMSE – типичное расстояние между прогнозируемым значением, полученным с помощью модели регрессии и фактическим значением, а R2 – показывает насколько хорошо переменный-предикторы могут объяснить вариацию в целевой переменной.

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

from collections import Counter

def Reg\_Models\_Evaluation\_Metrics (model,X\_train,y\_train,X\_test,y\_test,y\_pred):

    cv\_score = cross\_val\_score(estimator = model, X = X\_train, y = y\_train, cv = 10)

    r2 = model.score(X\_test, y\_test)

    n = X\_test.shape[0]

    p = X\_test.shape[1]

    adjusted\_r2 = 1-(1-r2)\*(n-1)/(n-p-1)

    RMSE = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))

    R2 = model.score(X\_test, y\_test)

    CV\_R2 = cv\_score.mean()

    print(f'RMSE:  {round(RMSE,4)}\n  R2: {round(R2,4)}\n Adjusted R2: {round(adjusted\_r2, 4)}\n Cross Validated R2: {round(cv\_score.mean(),4)}')

    return R2, adjusted\_r2, CV\_R2, RMSE

Удаляем большие выбросы (игры, продавшиеся тиражом больше 20ти миллионов), потому что мы посчитали, что такой коммерческий успех является единичными случаями и его невозможно предсказать:

min\_threshold = 0

max\_threshold = 20

games = games[(games['Global\_Sales'] >= min\_threshold) & (games['Global\_Sales'] <= max\_threshold)]

games.head()

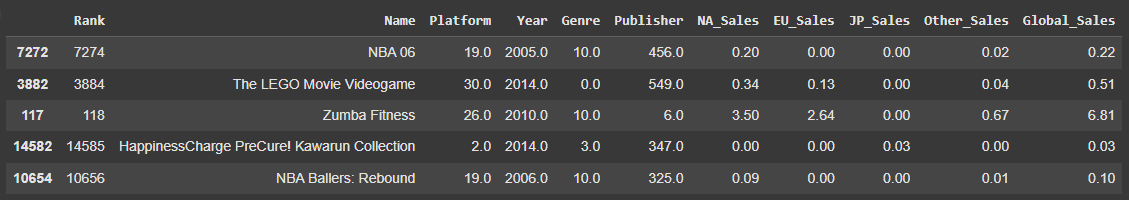


Рисунок 17 – Датасет после удаления больших выбросов

Нормализуем данные, чтобы данные распределялись примерно на одном промежутке:

from sklearn.preprocessing import RobustScaler

# Создание экземпляра RobustScaler

scaler = RobustScaler()

# Применение RobustScaler к данным

games[['rsPlatform','rsGenre', 'rsPublisher', 'rsGlobal\_Sales']] = scaler.fit\_transform(games[['Platform','Genre', 'Publisher', 'Global\_Sales']])

games.head()

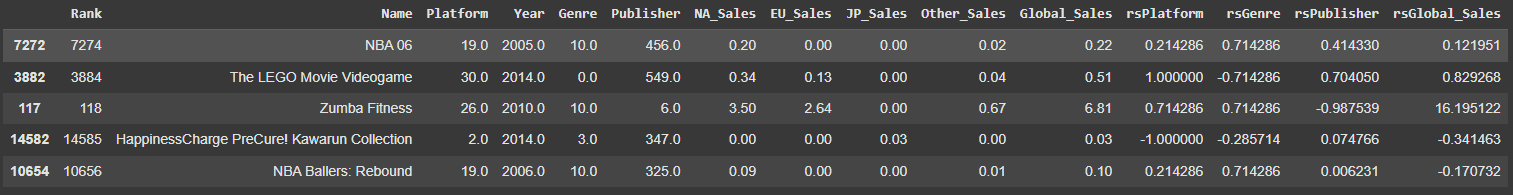


Рисунок 18 – Датасет после скалирования данных

Логарифмируем целевую переменную из-за большого количества выбросов и делим выборку на тестовую и тренировочную:

games\_copy = games

games\_copy['rsGlobal\_Sales'] = np.log1p(games\_copy['rsGlobal\_Sales'])

X = games\_copy[['rsPlatform', 'rsGenre', 'rsPublisher']]

y = games\_copy['rsGlobal\_Sales']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)



Рисунок 19 – Количество строк и столбцов в выборках

Создаем и обучаем модель линейной регрессии:

from sklearn.linear\_model import LinearRegression

# Создадим модель линейной регрессии

modelLR = LinearRegression()

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

y\_pr\_LR = modelLR.predict(X\_test)

# Проверим метрики

\_ = Reg\_Models\_Evaluation\_Metrics(modelLR, X\_train, y\_train, X\_test, y\_test, y\_pr\_LR)

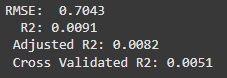


Рисунок 20 – Метрики модели линейной регрессии

Создаем и обучаем модель полиномиальной регрессии:

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, include\_bias=False)

poly\_features = poly.fit\_transform(X)

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

modelPR = LinearRegression()

modelPR.fit(X\_p\_train, y\_p\_train)

y\_pr\_PR = modelPR.predict(X\_p\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelPR, X\_p\_train, y\_p\_train, X\_p\_test, y\_p\_test, y\_pr\_PR)

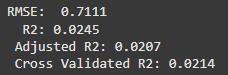


Рисунок 21 – Метрики модели полиномиальной регрессии

Создаем и обучаем модель регрессии дерева принятия решений:

from sklearn.tree import DecisionTreeRegressor

# Create a stepwise regression model

modelDT = DecisionTreeRegressor()

# Fit the model to the data

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

# Predict the response for a new data point

y\_dt\_pred = modelDT.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelDT, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

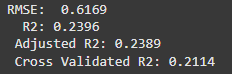


Рисунок 22 – Метрики модели регрессии дерева принятия решений

Создаем и обучаем модель регрессии случайного леса:

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Create a stepwise regression model

modelRF = RandomForestRegressor(n\_estimators=100, random\_state=14, max\_depth=10,min\_samples\_split=4, min\_samples\_leaf=3)

# Fit the model to the data

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

# Predict the response for a new data point

y\_rf\_pred = modelRF.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_rf\_pred)

\_ = Reg\_Models\_Evaluation\_Metrics(modelRF, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

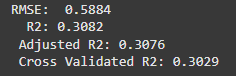


Рисунок 23 – Метрики модели регрессии случайного леса

Построим график распределения ошибок модели регрессии случайного леса:

errors = y\_test - y\_rf\_pred

# Гистограмма распределения ошибок

plt.figure(figsize=(10, 6))

sns.histplot(errors, bins=30, kde=True)

plt.title("Распределение ошибок")

plt.xlabel("Ошибка (Фактические - Предсказанные)")

plt.ylabel("Частота")

plt.axvline(0, color='red', linestyle='--')  # Линия нуля

plt.show()

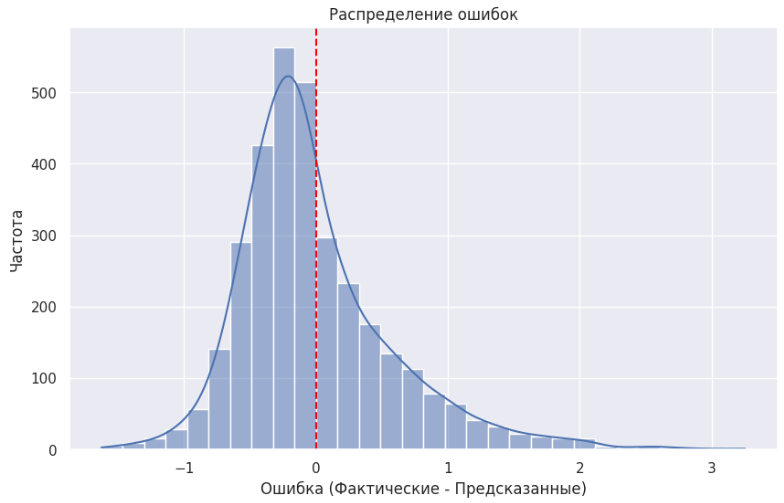


Рисунок 24 – График распределения ошибок

После обучения моделей, можно сделать вывод, что гипотеза неверна. Максимальная точность предсказания глобальных продаж, которую мы можем получить, зная только жанр, платформу и издателя видеоигры – 30% с помощью модели регрессии случайного леса.

В таком случае выдвинем новую гипотезу. Предположим, что мы теперь знаем еще и продажи в Европе, например. Тогда новая гипотеза будет такова: глобальное количество продаж видеоигр можно предсказать с высокой точностью зная жанр, издателя, платформу, на которой выпускается игра и продажи в Европе.

Аналогично обучаем модели регрессий, но уже с добавленным новым полем.

games\_copy = games

games\_copy['Global\_Sales'] = np.log1p(games\_copy['Global\_Sales'])

X = games\_copy[['Platform', 'Genre', 'Publisher', 'EU\_Sales']]

y = games\_copy['Global\_Sales']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)



Рисунок 25 – Размер выборки

Результат обучение модели линейной регрессии:

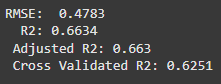


Рисунок 26 – Линейная регрессия

Результат обучения модели полиномиальной регрессии:

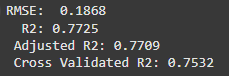


Рисунок 27 – Полиномиальная регрессия

Результат обучения модели регрессии дерева принятия решений:

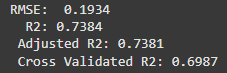


Рисунок 28 – Регрессия дерева принятия решений

Результат обучения модели регрессии случайного леса:

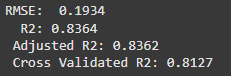


Рисунок 29 – Регрессия случайного леса

Визуализируем регрессию случайного леса на параметре «Platform»:

X2 = games\_copy['Platform'].values[:,np.newaxis]

y2 = games\_copy['Global\_Sales'].values

plt.scatter(X2, y2,color='g')

plt.plot(X\_train['Platform'], modelRF.predict(X\_train),color='k')

plt.ylabel('Sales', size=12)

plt.xlabel('Platform', size=12)

plt.show()

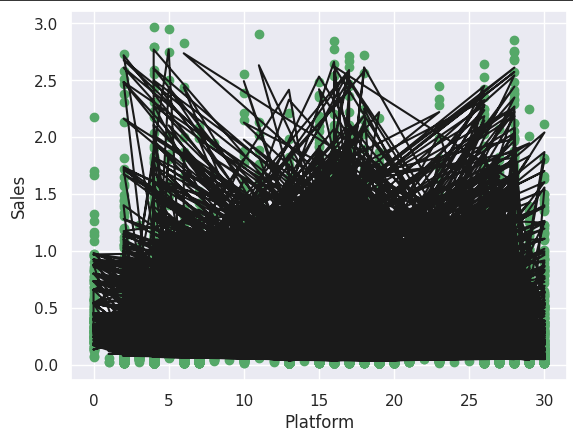


Рисунок 30 – Визуализация регрессии случайного леса

Заключение

На основе проведенного регрессионного анализа можно сделать вывод, что первоначальная гипотеза не подтвердилась: зная только жанр, издателя и платформу видеоигры можно с высокой точностью предсказать глобальное количество продаж.

Можно было предположить, что этих данных будет недостаточно, что и подтвердилось в процессе анализа. Поэтому была выдвинута новая гипотеза: глобальное количество продаж видеоигр можно предсказать с высокой точностью зная жанр, издателя, платформу, на которой выпускается игра и продажи в Европе, которая успешно подтвердилась.

Таким образом, поставленная цель была достигнута: был произведен анализ данных и построена модель регрессии случайного леса, которая позволяет прогнозировать глобальные продажи с высокой точностью.

Для успешного анализа данных и построения модели были решены следующие задачи:

* выполнен анализ проблемы;
* данные были подготовлены к анализу;
* произведен предварительный анализ данных;
* осуществлено моделирование зависимости целевого признака методом регрессии случайного леса

В ходе выполнения работы все поставленные задачи выполнены, что означает успешное достижение цели.

Библиографический список

1. StandardScaler, MinMaxScaler and RobustScaler techniques – ML [Электронный ресурс]. – URL: https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/ (дата обращения 22.12.2024).
2. Scikit-learn [Электронный ресурс]. – URL: https://scikit-learn.org/stable/modules/preprocessing.html/ (дата обращения 22.12.2024).
3. Pandas [Электронный ресурс]. – URL: https://pandas.pydata.org/pandas-docs/stable/index.html (дата обращения 22.12.2024).
4. Vitmo. Выброс [Электронный ресурс]. – URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81
5. Яндекс образования, Метрики классификации и регрессии [Электронный ресурс]. – URL: https://education.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii
6. Блог Практикума. Как привести данные в форму: что такое нормализация и зачем она нужна. [Электронный ресурс]. – URL: https://practicum.yandex.ru/blog/chto-takoe-normalizaciya-dannyh/

Приложение А  
Программный код

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set() # устанавливаем seaborn по умолчанию для отрисовки графиков

from google.colab import drive

drive.mount('/content/drive/')

games = pd.read\_csv('/content/drive/MyDrive/vgsales.csv')

games.head()

games.describe()

games.describe(include=['O'])

games.info()

games.isna().sum() / len(games)

games.isnull().sum()

cols = games.columns

# определяем цвета

# желтый - пропущенные данные, синий - не пропущенные

colours = ['#000099', '#ffff00']

sns.heatmap(games[cols].isnull(), cmap=sns.color\_palette(colours))

med = games['Year'].median()

print(med)

games['Year'] = games['Year'].fillna(med)

games.dropna(axis=0, inplace=True)

games.info()

cols = games.columns

# определяем цвета

# желтый - пропущенные данные, синий - не пропущенные

colours = ['#ffff00', '#000099']

sns.heatmap(games[cols].isnull(), cmap=sns.color\_palette(colours))

numeric\_games = games.select\_dtypes(include=['number'])

numeric\_games.drop("Rank",axis=1,inplace=True)

#Среднее

mean = numeric\_games.mean()

# Мода

mode = numeric\_games.mode().iloc[0]

# Медиана

median = numeric\_games.median()

# Стандартное отклонение

std\_dev = numeric\_games.std()

# Квартили

quartiles = numeric\_games.quantile([0.25, 0.5, 0.75])

# Объединение результатов в один DataFrame

descriptive\_stats = pd.DataFrame({

'Mean': mean,

'Mode': mode,

'Median': median,

'Standard Deviation': std\_dev,

'Q1': quartiles.loc[0.25],

'Q2': quartiles.loc[0.5],

'Q3': quartiles.loc[0.75]

})

print(descriptive\_stats)

games.Global\_Sales.hist();

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

axes = axes.flatten()

for i, col in enumerate(numeric\_games.columns):

games.boxplot(column=[col], ax=axes[i])

axes[i].set\_title(col)

plt.tight\_layout()

plt.show()

numeric\_games.hist(figsize=(12, 8), bins=10)

plt.suptitle('Histograms of Columns', fontsize=16)

plt.show()

fig, axes = plt.subplots(len(numeric\_games.drop("Year", axis=1).columns), 1, figsize=(12, 8))

for i, col in enumerate(numeric\_games.drop("Year", axis=1).columns):

axes[i].hist(numeric\_games[col], bins=10)

axes[i].set\_ylim(0, 100)

axes[i].set\_title(f'Histogram of {col}')

plt.suptitle('Histograms of Columns', fontsize=16)

plt.tight\_layout()

plt.show()

import scipy.stats as stats

for col in numeric\_games.columns:

data = numeric\_games[col]

mu = np.mean(data)

sigma = np.std(data)

# Тест Колмогорова-Смирнова

alpha = 0.05

d\_statistic, p\_value = stats.kstest(data, 'norm', args=(mu, sigma))

print('p-значение:', p\_value)

if p\_value > alpha:

print('Не удалось отвергнуть гипотезу о нормальности')

else:

print('Гипотеза о нормальности отвергнута')

from sklearn.preprocessing import OrdinalEncoder

enc = OrdinalEncoder()

games[['Platform', 'Genre', 'Publisher']] = enc.fit\_transform(games[['Platform', 'Genre', 'Publisher']])

games.head()

correlation\_matrix = games.drop(["Name", "Rank", "NA\_Sales", "EU\_Sales", "JP\_Sales", "Other\_Sales"], axis=1).corr(method='spearman')

# Визуализация с помощью тепловой карты

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, cbar\_kws={"shrink": .8})

plt.title('Матрица корреляции (Спирмен)')

plt.show()

from scipy.stats import kendalltau

from sklearn.utils import shuffle

correlation, p\_value = kendalltau(range(len(games)), games['Global\_Sales'])

alpha = 0.05

print(f"Корреляция Манна-Кендалла: {correlation:.3f}")

print(f"p-значение: {p\_value:.3f}")

if p\_value < alpha:

print("Данные, вероятно, упорядочены.")

games = shuffle(games)

print("Данные перемешаны.")

else:

print("Данные, вероятно, неупорядочены.")

games.head()

correlation, p\_value = kendalltau(range(len(games)), games['Global\_Sales'])

alpha = 0.05

print(f"Корреляция Манна-Кендалла: {correlation:.3f}")

print(f"p-значение: {p\_value:.3f}")

if p\_value < alpha:

print("Данные, вероятно, упорядочены.")

else:

print("Данные, вероятно, неупорядочены.")

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

from collections import Counter

def Reg\_Models\_Evaluation\_Metrics (model,X\_train,y\_train,X\_test,y\_test,y\_pred):

cv\_score = cross\_val\_score(estimator = model, X = X\_train, y = y\_train, cv = 10)

r2 = model.score(X\_test, y\_test)

n = X\_test.shape[0]

p = X\_test.shape[1]

adjusted\_r2 = 1-(1-r2)\*(n-1)/(n-p-1)

RMSE = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))

R2 = model.score(X\_test, y\_test)

CV\_R2 = cv\_score.mean()

print(f'RMSE: {round(RMSE,4)}\n R2: {round(R2,4)}\n Adjusted R2: {round(adjusted\_r2, 4)}\n Cross Validated R2: {round(cv\_score.mean(),4)}')

return R2, adjusted\_r2, CV\_R2, RMSE

min\_threshold = 0

max\_threshold = 20

games = games[(games['Global\_Sales'] >= min\_threshold) & (games['Global\_Sales'] <= max\_threshold)]

games.head()

games.info()

from sklearn.preprocessing import RobustScaler

# Создание экземпляра RobustScaler

scaler = RobustScaler()

# Применение RobustScaler к данным

games[['rsPlatform','rsGenre', 'rsPublisher', 'rsGlobal\_Sales']] = scaler.fit\_transform(games[['Platform','Genre', 'Publisher', 'Global\_Sales']])

games.head()

games\_copy = games

games\_copy['rsGlobal\_Sales'] = np.log1p(games\_copy['rsGlobal\_Sales'])

X = games\_copy[['rsPlatform', 'rsGenre', 'rsPublisher']]

y = games\_copy['rsGlobal\_Sales']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)

from sklearn.linear\_model import LinearRegression

# Создадим модель линейной регрессии

modelLR = LinearRegression()

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

y\_pr\_LR = modelLR.predict(X\_test)

# Проверим метрики

\_ = Reg\_Models\_Evaluation\_Metrics(modelLR, X\_train, y\_train, X\_test, y\_test, y\_pr\_LR)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, include\_bias=False)

poly\_features = poly.fit\_transform(X)

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

# Create a linear regression model

modelPR = LinearRegression()

# Fit the model to the data

modelPR.fit(X\_p\_train, y\_p\_train)

# Predict the response for a new data point

y\_pr\_PR = modelPR.predict(X\_p\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelPR, X\_p\_train, y\_p\_train, X\_p\_test, y\_p\_test, y\_pr\_PR)

from sklearn.tree import DecisionTreeRegressor

# Create a stepwise regression model

modelDT = DecisionTreeRegressor()

# Fit the model to the data

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

# Predict the response for a new data point

y\_dt\_pred = modelDT.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelDT, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Create a stepwise regression model

modelRF = RandomForestRegressor(n\_estimators=100, random\_state=14, max\_depth=10,min\_samples\_split=4, min\_samples\_leaf=3)

# Fit the model to the data

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

# Predict the response for a new data point

y\_rf\_pred = modelRF.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_rf\_pred)

\_ = Reg\_Models\_Evaluation\_Metrics(modelRF, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

errors = y\_test - y\_rf\_pred

# Гистограмма распределения ошибок

plt.figure(figsize=(10, 6))

sns.histplot(errors, bins=30, kde=True)

plt.title("Распределение ошибок")

plt.xlabel("Ошибка (Фактические - Предсказанные)")

plt.ylabel("Частота")

plt.axvline(0, color='red', linestyle='--') # Линия нуля

plt.show()

errors = np.expm1(y\_test) - np.expm1(y\_rf\_pred)

plt.figure(figsize=(10, 6))

plt.scatter(np.expm1(y\_test), errors, alpha=0.5)

plt.axhline(0, color='red', linestyle='--') # Линия нуля

plt.title("График рассеяния ошибок")

plt.xlabel("Фактические значения")

plt.ylabel("Ошибка (Фактические - Предсказанные)")

plt.show()

games\_copy = games

games\_copy['Global\_Sales'] = np.log1p(games\_copy['Global\_Sales'])

X = games\_copy[['Platform', 'Genre', 'Publisher', 'EU\_Sales']]

y = games\_copy['Global\_Sales']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)

from sklearn.linear\_model import LinearRegression

# Создадим модель линейной регрессии

modelLR = LinearRegression()

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

y\_pr\_LR = modelLR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelLR, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, include\_bias=False)

poly\_features = poly.fit\_transform(X)

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

# Create a linear regression model

modelPR = LinearRegression()

# Fit the model to the data

modelPR.fit(X\_p\_train, y\_p\_train)

# Predict the response for a new data point

y\_pr\_PR = modelPR.predict(X\_p\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelPR, X\_p\_train, y\_p\_train, X\_p\_test, y\_p\_test, y\_pr\_PR)

from sklearn.tree import DecisionTreeRegressor

# Create a stepwise regression model

modelDT = DecisionTreeRegressor()

# Fit the model to the data

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

# Predict the response for a new data point

y\_dt\_pred = modelDT.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelDT, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Create a stepwise regression model

modelRF = RandomForestRegressor(n\_estimators=100, random\_state=14, max\_depth=10,min\_samples\_split=4, min\_samples\_leaf=3)

# Fit the model to the data

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

# Predict the response for a new data point

y\_rf\_pred = modelRF.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_rf\_pred)

\_ = Reg\_Models\_Evaluation\_Metrics(modelRF, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

X2 = games\_copy['Platform'].values[:,np.newaxis]

y2 = games\_copy['Global\_Sales'].values

plt.scatter(X2, y2,color='g')

plt.plot(X\_train['Platform'], modelRF.predict(X\_train),color='k')

plt.ylabel('Sales', size=12)

plt.xlabel('Platform', size=12)

plt.show()