



**Федеральное государственное бюджетное  
образовательное учреждение  
высшего образования  
«Московский государственный  
технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)**

---

Факультет «Информатика и вычислительная техника»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технология машинного обучения»

Отчет по лабораторной работе №1  
«Разведочный анализ данных. Исследование и визуализация данных.»

Выполнил:

студент группы ИУ5-61Б  
Коновалов М. В.

Подпись и дата:

Проверил:  
преподаватель каф.

ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2023 г.

## **Определение данных для анализа**

В качестве данных для анализа был выбран Dataset «Приложения Google Play Store».

Сегодня пользователям доступно для загрузки 1,85 миллиона различных приложений. У пользователей Android есть из чего выбирать: 2,56 миллиона доступны в магазине Google Play. Эти приложения стали играть огромную роль в том, как мы живем сегодня.

Google Play Store использует сложные современные методы (например, динамическую загрузку страниц) с использованием JQuery, что усложняет парсинг.

Каждое приложение (строка) имеет значения категории, рейтинга, размера и т. д.

Данные о приложениях в Play Store обладают огромным потенциалом для успеха компаний, занимающихся разработкой приложений. Разработчики могут получить полезную информацию для работы и захвата рынка Android!

## **Описание данных**

- Приложение – название
- Категория – жанры
- Рейтинг – оценка
- Отзывы – количество отзывов
- Размер приложения
- Установки – количество установок
- Тип – платный / бесплатный
- Цена
- Рейтинг содержания – для кого предназначен
- Последнее обновление
- Текущая версия
- Андроид версия

## **Формулирование гипотез**

Гипотеза 1:

Большинство игр в Google Play Store – бесплатные

Гипотеза 2:

H0: Средняя скачиваемость приложений с высоким и низким рейтингом одинаковые

H1: Средняя скачиваемость приложений с высоким и низким рейтингом разные

Гипотеза 3:

Приложения, у которых недавно было выпущено обновления продаются лучше, чем те, что со старым обновлением.

## Предобработка данных

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import math as mth
import matplotlib.patches as patches
from scipy import stats as st
plt.rcParams.update({'figure.max_open_warning': 0})
import plotly.graph_objects as go
import plotly.express as px
```

Рисунок 4.1 – загрузка библиотек

```
df = pd.read_csv('googleplaystore.csv')
```

Рисунок 4.2 – чтение датасета в переменной

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone

Рисунок 4.3 – чтение первых 5 строк датасета

Заметим, что Category и Genres дублируют одни и те же данные, поэтому удалим их из датафрейма.

```
df.drop(['Genres'], axis=1, inplace=True)
```

Рисунок 4.4 – удаление столбца

```

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   App               10841 non-null   object  
 1   Category          10841 non-null   object  
 2   Rating             9367 non-null   float64 
 3   Reviews            10841 non-null   object  
 4   Size               10841 non-null   object  
 5   Installs           10841 non-null   object  
 6   Type               10840 non-null   object  
 7   Price              10841 non-null   object  
 8   Content Rating    10840 non-null   object  
 9   Last Updated       10841 non-null   object  
 10  Current Ver       10833 non-null   object  
 11  Android Ver       10838 non-null   object  
dtypes: float64(1), object(11)
memory usage: 1016.5+ KB

```

Рисунок 4.5 – информация о столбцах

## Обработка пропусков и дубликатов

```

def draw_missing(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()).sort_values(as
missing_data = pd.concat([total, percent], axis=1, keys=[ 'Total'
return missing_data

```

Рисунок 4.6 – функция для подсчета пропущенных значений

```
draw_missing(df).round(1)
```

	Total	Percent
<b>Rating</b>	1474	13.6
<b>Current Ver</b>	8	0.1
<b>Android Ver</b>	3	0.0
<b>Type</b>	1	0.0
<b>Content Rating</b>	1	0.0
<b>App</b>	0	0.0
<b>Category</b>	0	0.0
<b>Reviews</b>	0	0.0
<b>Size</b>	0	0.0
<b>Installs</b>	0	0.0
<b>Price</b>	0	0.0
<b>Last Updated</b>	0	0.0

Рисунок 4.7 – таблица пропущенных значений

Т.к пропущенные значения не превышают 20% удалим их из датафрейма без потери важной информации

```
df.duplicated().sum()
```

474

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

0

Рисунок 4.8 – обработка дубликатов

### Проверка названий колонок

```
df.columns
```

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs',
'Type',
       'Price', 'Content Rating', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

Рисунок 4.9 – названия колонок

### Изменение типов данных и обработка числовых значений

```
def convert_currency(val):
    new_val = val.replace('+', '').replace(',', '')
    return float(new_val)
```

Рисунок 4.10 – функция для обработки столбца

```
df.Installs= df["Installs"].apply(convert_currency)
```

Рисунок 4.11 – применение функции к столбцу

```
def convert_currency_price(val):
    new_val = val.replace('$', '')
    return float(new_val)
```

Рисунок 4.12 – функция для обработки столбца

```
df[ "Price"]= df[ "Price"].apply(convert_currency_price)
```

Рисунок 4.13 – применение функции к столбцу

```
def convert_currency_size(val):
    new_val = val.replace('M', '').replace('k', '').replace('Varies w
    return float(new_val)
```

Рисунок 4.14 – функция для обработки столбца

```
df.Size= df[ "Size"].apply(convert_currency_size)
```

Рисунок 4.13 – применение функции к столбцу

## Вывод

- 1) Изменили типы данных у числовых значений на float
- 2) Привели к правильному типу данных значения даты и сделали срез по году

В описании данных видна аномалия в столбце price большая часть значений 0.0 - скорее всего это связано с тем, что большая часть приложений бесплатные, убедимся в этом дальше и в столбце Size -1 - это из-за замены данных Varies with device на -1

# EDA

## Категоризация и анализ данных

```
Ввод [22]: fig = plt.figure()
fig.suptitle('Boxplot', fontsize=14, fontweight='bold')

ax = fig.add_subplot(111)
ax.boxplot(df['Rating'])
ax.set_ylabel('Rating')

plt.show()
```

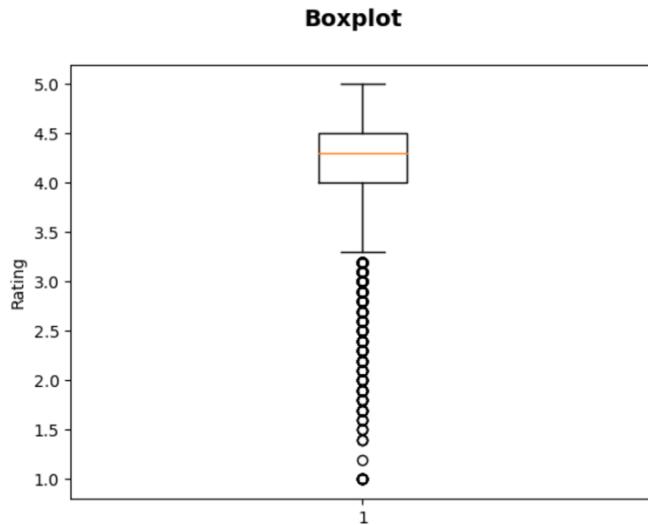


Рисунок 5.1 – диаграмма выбросов для столбца рейтинг

Заметим, что среднее значение рейтинга приложений лежит между 4.0 и 4.5, а все, что ниже 3.5 считаются выбросами. Проведем категоризацию данных.

```
: def rating_category(row):
    if row >=0 and row <3:
        return 'low'
    if row >= 3 and row <4:
        return 'ordinary'
    return 'high'

df['Rating_category'] = df['Rating'].apply(rating_category)
```

Рисунок 5.2 – функция для категоризации данных

```
rating_data = df['Rating_category'].value_counts().reset_index().ren  
sns.barplot(x=rating_data['index'], y=rating_data["count"]);
```

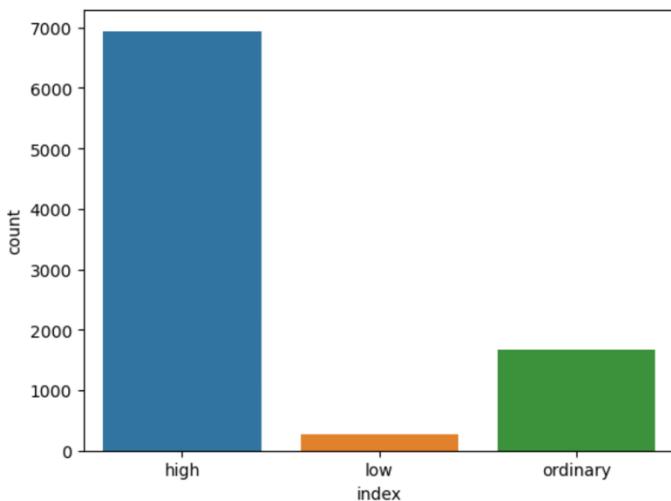


Рисунок 5.3 – рейтинг игр

Видно, что в данном датасете преобладают приложения с высоким рейтингом

## Категоризация обновлений

```
df['Last Updated']=pd.to_datetime(df['Last Updated'])
```

```
df['Last Updated_Year']=pd.DatetimeIndex(df['Last Updated']).year
```

Рисунок 5.4 – вычленение года из даты

```
Бывод [27]: years_data = df['Last Updated_Year'].value_counts().reset_index().re  
t1 = go.Scatter(x=years_data['index'], y=years_data["count"], name="  
data2 = [t1]  
layout = go.Layout(title="Новые обновления, появившиеся за эти годы", lege  
fig = go.Figure(data2, layout=layout)  
fig.show()
```

Новые обновления, появившиеся за эти годы

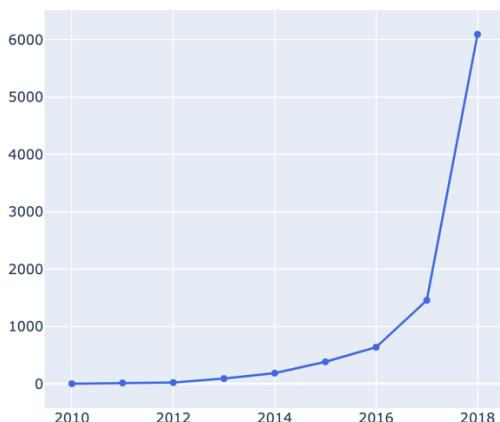


Рисунок 5.5 – диаграмма обновлений по годам

```

def update_category(row):
    if row >= 2010 and row < 2014:
        return 'old'
    if row >= 2014 and row < 2016:
        return 'ordinary'
    return 'new'

df['update_category'] = df['Last Updated Year'].apply(update_category)

```

Рисунок 5.6 – категоризация данных

```

plt.title("Категоризация данных по году обновления приложений")

up_data = df['update_category'].value_counts().reset_index().rename(
    columns={'index': 'index', 'count': 'count'});
sns.barplot(x=up_data['index'], y=up_data["count"]);

```

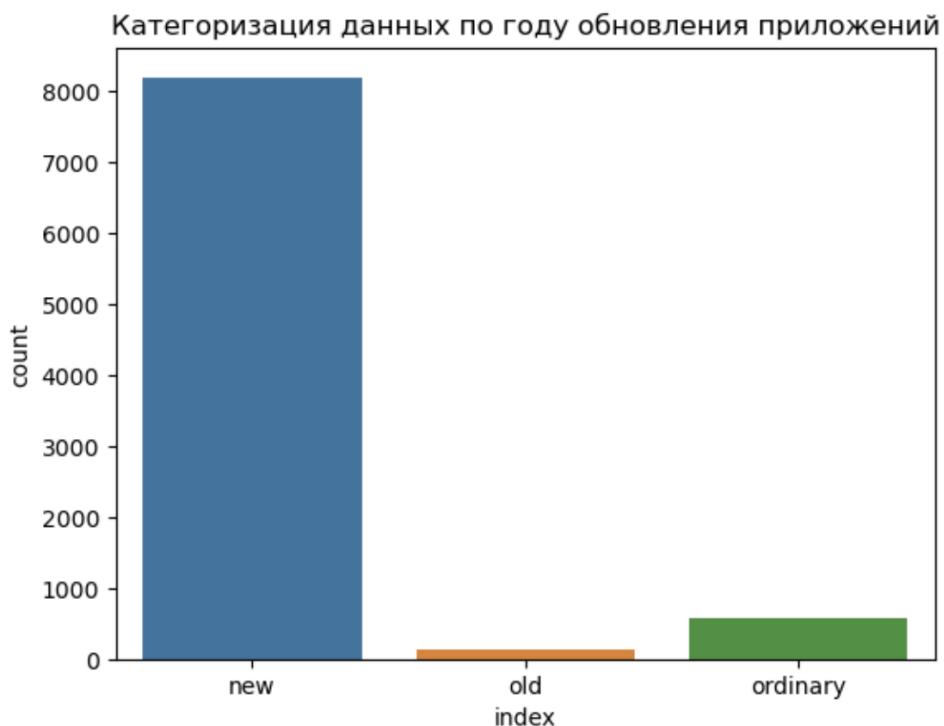


Рисунок 5.7 – рейтинг обновления игр

Что по линейному графику, что по гистограмме видно, что в датасете преобладают данных по играм с новым обновлением.

### Анализ рейтинга контента

```
|: content_data = df['Content Rating'].value_counts().reset_index().rename(columns={ 'index': 'Content Rating', 'count': 'Count' })
plt.figure(figsize = (15,5))
plt.title("Количество приложений по контенту")
sns.barplot(x=content_data['Content Rating'], y=content_data["Count"]);
```

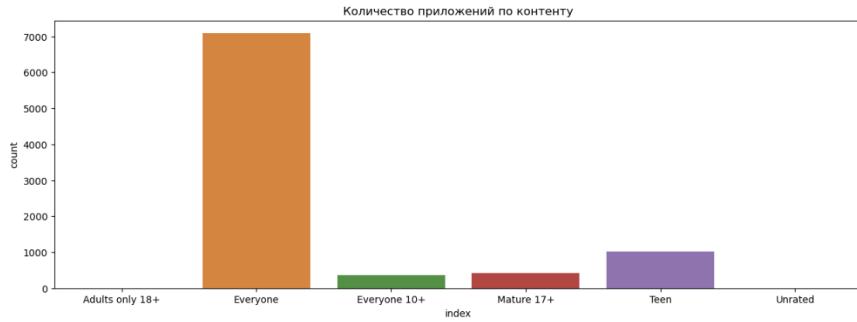


Рисунок 5.8 – анализ рейтинга

## Анализ категорий игр

```
plt.figure(figsize = (10,15))
plt.title("Количество приложений по жанру")
sns.countplot(y = 'Category', data = df);
```

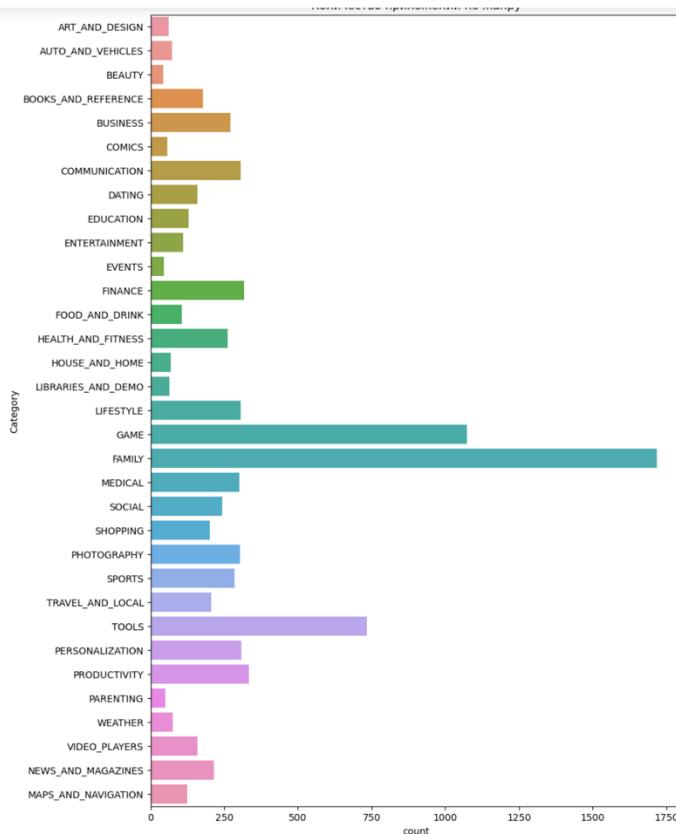


Рисунок 5.9 – анализ категорий игр

Гистограмма по жанрам дает нам возможность выявить топ-3 самых популярных категорий приложений:

1) Family

2) Game

3) Tools

**Посмотрим на линейную взаимосвязь между данными**

```
plt.figure(figsize=(10,5))
corr = df.corr()
sns.heatmap(corr, annot=True)
plt.show()
```

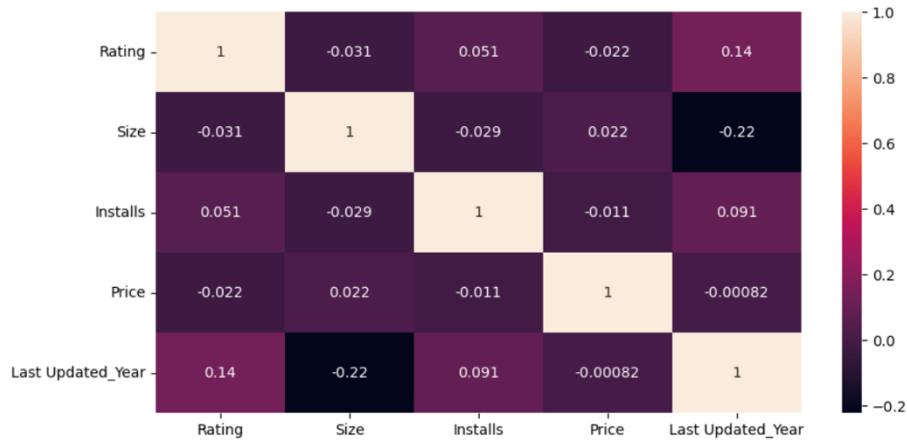


Рисунок 5.10 – тепловая карта корреляции

```
print(df[['Rating', 'Reviews',
          'Price']].corr())
pd.plotting.scatter_matrix(df[['Rating', 'Reviews',
          'Price']], figsize=(15,15));
```

```
Rating      Price
Rating  1.000000 -0.022371
Price   -0.022371  1.000000
```

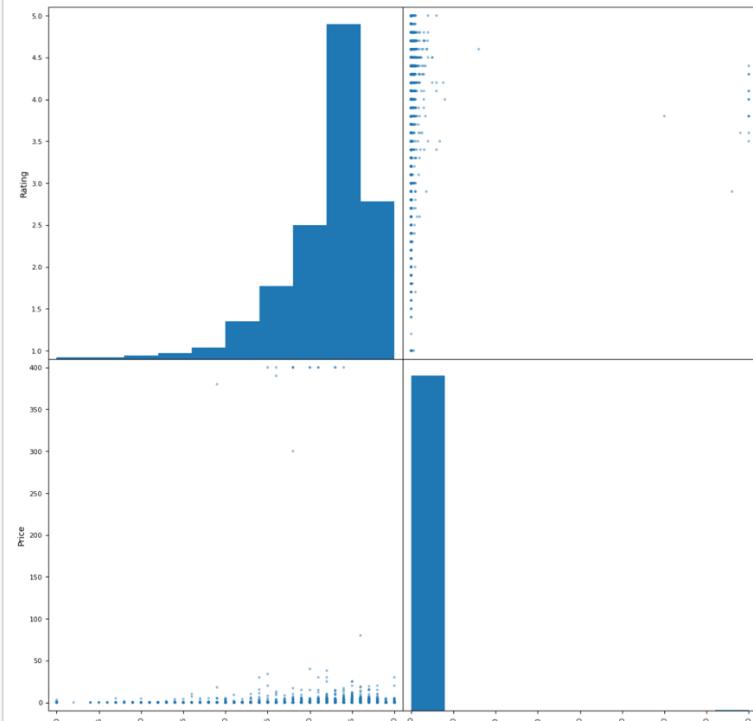


Рисунок 5.11 – матрица

Заметим, что между всеми значениями слабая линейная связь, тк значения корреляции близко к нулю.

**Проанализируем самые устанавливающиеся игры и их рейтинг**

```
top_installed.drop(['Price', 'Rating_category', 'Last Updated_Year', 'u'])
```

	App	Category	Rating	Reviews	Size	Content Rating	Last Updated	Current Ver	And
2604	Instagram	SOCIAL	4.5	66577446	-1.0	Teen	2018-07-31	Varies with device	Varies with device
2545	Instagram	SOCIAL	4.5	66577313	-1.0	Teen	2018-07-31	Varies with device	Varies with device
3909	Instagram	SOCIAL	4.5	66509917	-1.0	Teen	2018-07-31	Varies with device	Varies with device
1872	Subway Surfers	GAME	4.5	27725352	76.0	Everyone 10+	2018-07-12	1.90.0	4.1
1750	Subway Surfers	GAME	4.5	27724094	76.0	Everyone 10+	2018-07-12	1.90.0	4.1
1700	Subway Surfers	GAME	4.5	27723193	76.0	Everyone 10+	2018-07-12	1.90.0	4.1
1654	Subway Surfers	GAME	4.5	27722264	76.0	Everyone 10+	2018-07-12	1.90.0	4.1
3896	Subway Surfers	GAME	4.5	27711703	76.0	Everyone 10+	2018-07-12	1.90.0	4.1
2884	Google Photos	PHOTOGRAPHY	4.5	10859051	-1.0	Everyone	2018-08-06	Varies with device	Varies with device
2808	Google Photos	PHOTOGRAPHY	4.5	10858556	-1.0	Everyone	2018-08-06	Varies with device	Varies with device

Рисунок 5.12 – Таблица самых устанавливаемых игр и их рейтинг

## Проанализируем платные приложения

```
df_price = df.query('Price > 2')
```

```
df_price.describe().T
```

	count	mean	std	min	25%	50%	75%
<b>Rating</b>	405.0	4.243457	0.541165	1.00	4.10	4.40	4.60
<b>Size</b>	405.0	41.151605	120.549040	-1.00	2.60	12.00	36.00
<b>Installs</b>	405.0	109852.370370	717841.213368	10.00	1000.00	10000.00	50000.00
<b>Price</b>	405.0	20.436420	73.271307	2.49	2.99	3.99	5.99
<b>Last Updated_Year</b>	405.0	2016.856790	1.424410	2011.00	2016.00	2017.00	2018.00

```
df_price['Rating_category'].value_counts()
```

```
high      319
ordinary   71
low       15
Name: Rating_category, dtype: int64
```

Рисунок 5.13 – Анализ платных приложений

## Гипотезы

6.1 Большинство игр в Google Play Store – бесплатные

```
: plt.title("Количество значений по типам приложений ")

type_data = df[ 'Type' ].value_counts().reset_index().rename(columns =
    {"index": "index", "Type": "count"})

sns.barplot(x=type_data[ 'index' ], y=type_data[ "count" ]);
```

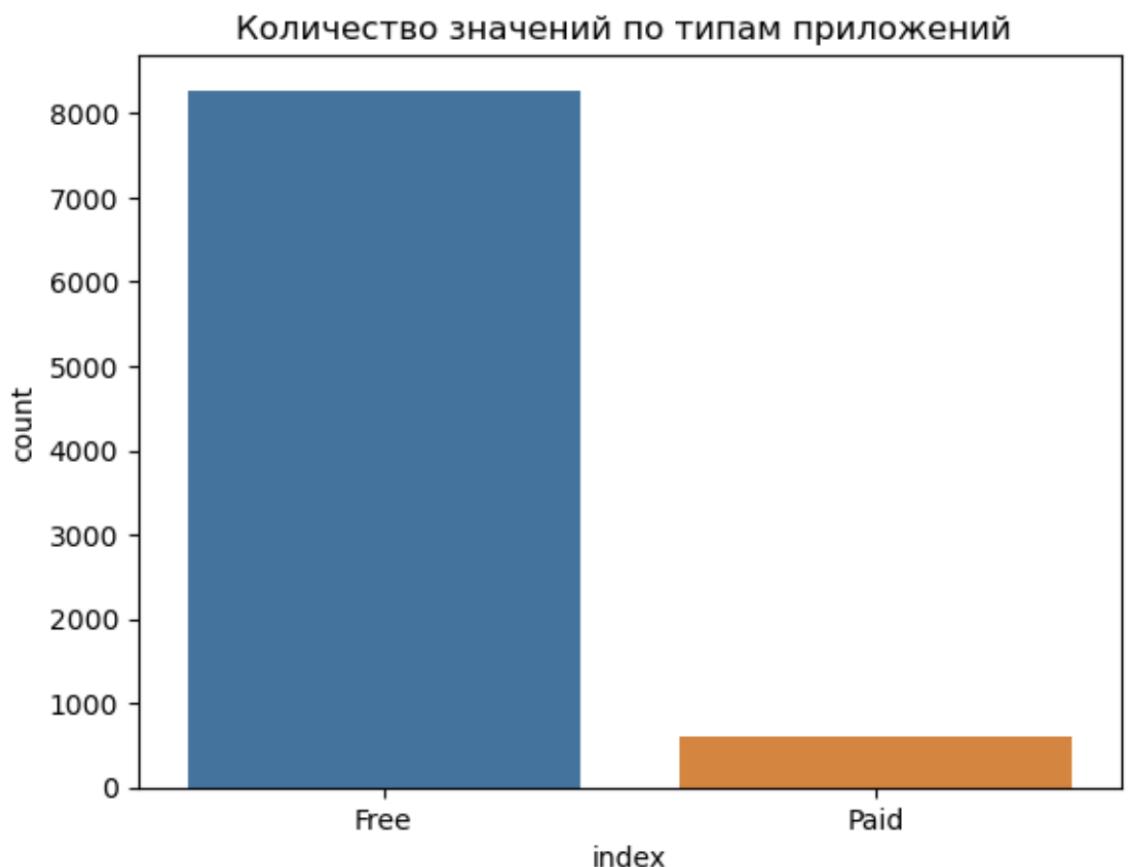


Рисунок 6.1 – анализ гипотезы 1

Видно, что в датасете преобладают бесплатные игры - этим самым объясняется ранее выявленная аномалия про 0.0 стоимость. Гипотеза подтверждена

6.2

H0: Средняя скачиваемость приложений с высоким и низким рейтингом одинаковые

H1: Средняя скачиваемость приложений с высоким и низким рейтингом разные

```

alpha = .05
result_first = st.ttest_ind(df_price.query('Rating_category == "high"')
                           , df_price.query('Rating_category == "ordinary"')
                           , equal_var=False)

prob_first = result_first.pvalue
print('p-значение: ', prob_first)
if prob_first < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")

```

р-значение: 0.0072213366905475055

Отвергаем нулевую гипотезу

Рисунок 6.2 – Метод статистические тесты значимости для проверки гипотез

```

high=df_price.loc[df_price['Rating_category']=='high']
ordinary=df_price.loc[df_price['Rating_category']=='ordinary']

```

```

plt.figure(figsize=(12,6))
sns.kdeplot(high['Installs'],label='high');
sns.kdeplot(ordinary['Installs'],label='ordinary');
plt.legend();

```

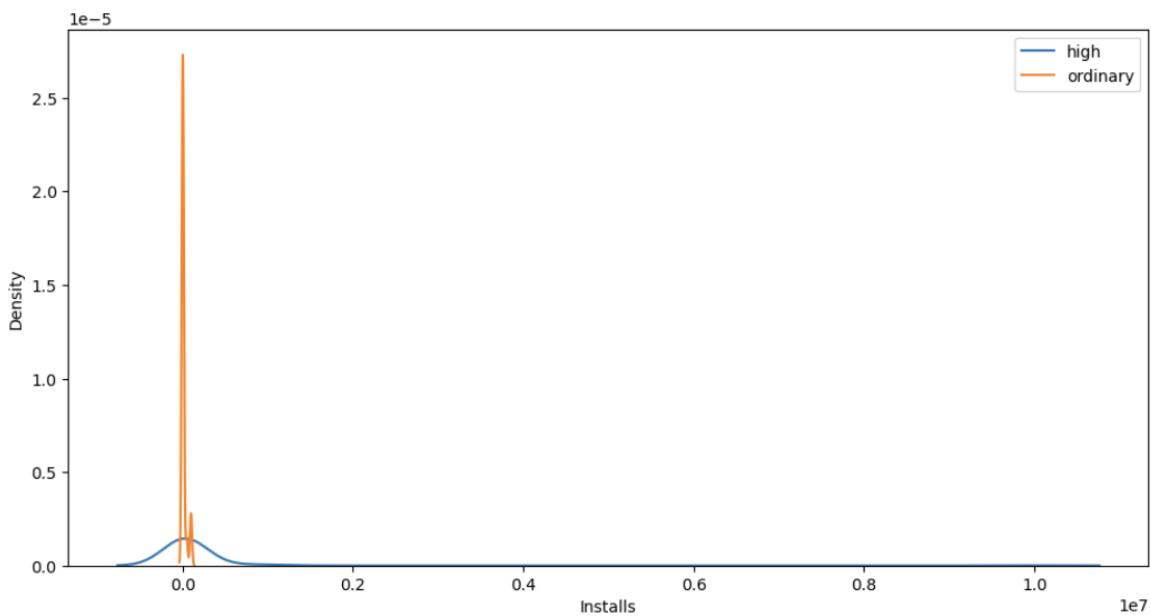


Рисунок 6.3 – Визуализация гипотезы №2

Интересный факт заключается в том, что скачиваемость у приложений со среднем рейтингом выше, чем с высоким, возможно это связано с высокой ценой за приложение.

6.3 Приложения, у которых недавно было выпущено обновление продаются лучше, чем те, что со старым обновлением.

```
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Price', y = 'Size', hue = 'update_category', palette='Set1')
plt.show()
```

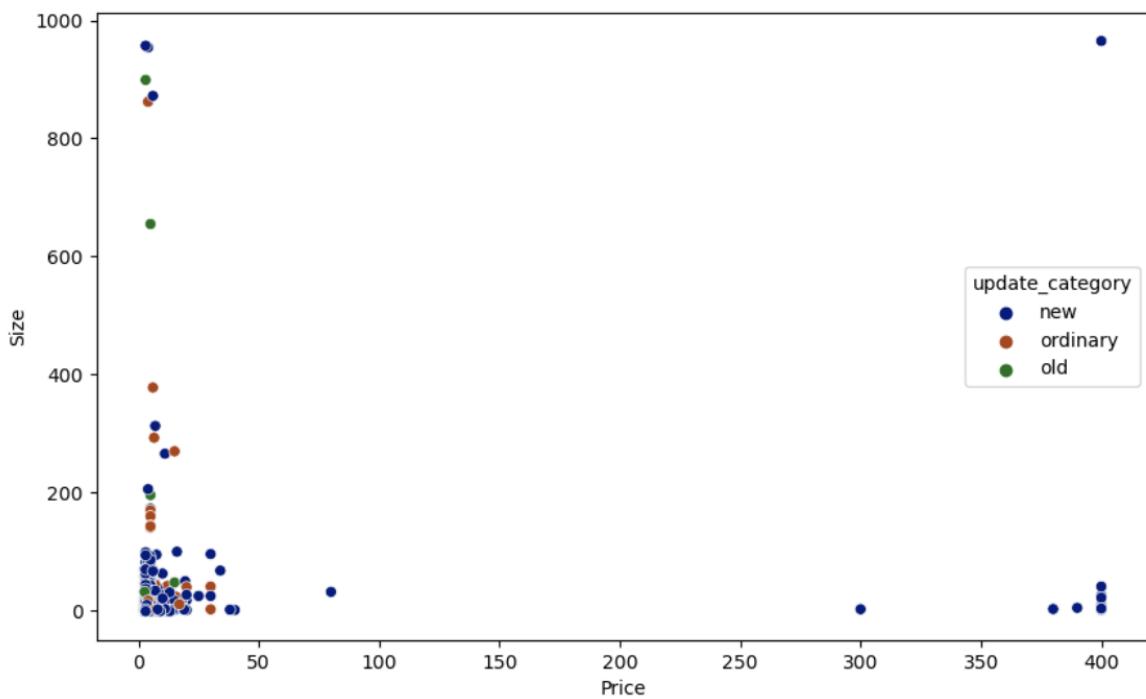


Рисунок 6.4 – Визуализация связи между размером и ценой

```
def dist(x,color = None,data=df_price, log = False, nbins=None ):
    print(f'Средний рейтинг приложения {x} = {round(data[x].mean(),2)}')
    print(f'Наиболее частый рейтинг приложение {x} = {round(data[x].mode()[0],2)}')
    fig = px.histogram(data, x=x, color=color, log_y=log, nbins= nbins)
    fig.show()
```

Рисунок 6.5 – Функция для построения гистограммы

```
dist('Rating', 'update_category')
```

Средний рейтинг приложения Rating = 4.24  
Наиболее частый рейтинг приложение Rating = 4.6

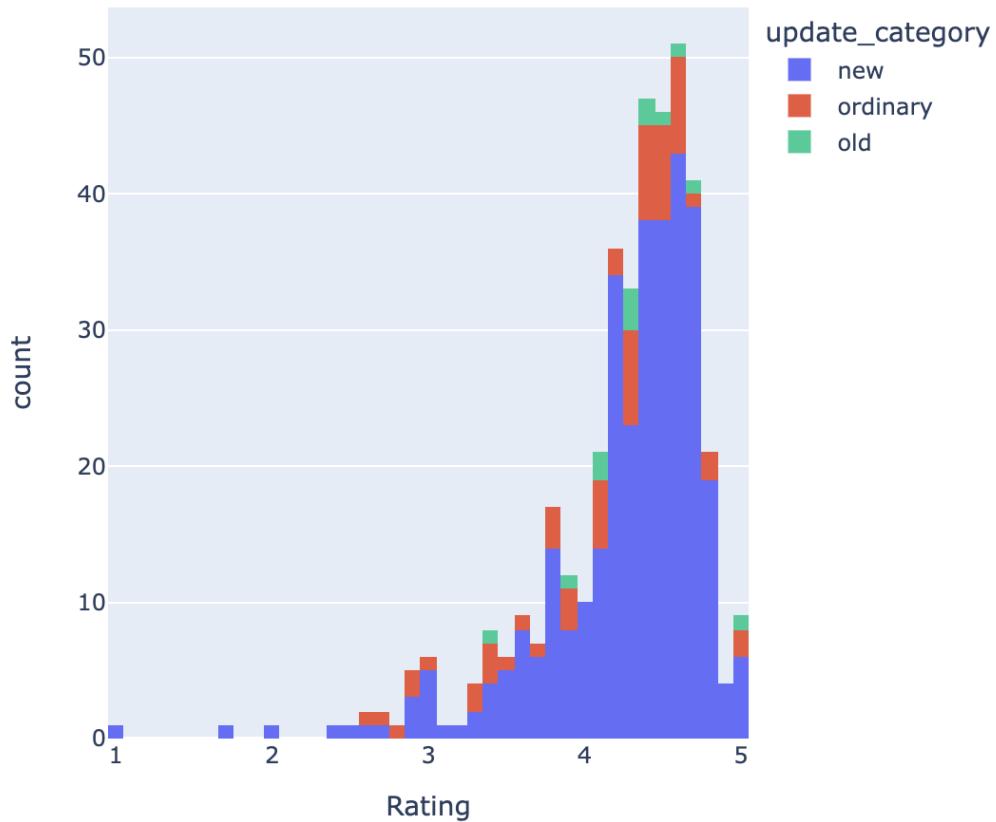


Рисунок 6.6 – Гистограмма зависимости рейтинга и обновления

```
def precentages(cols,tops, data = df_price):
    grouped = data[cols].groupby(cols).size().sort_values(ascending = False)
    list_of_groups = []
    for col,top in zip(cols[:-1],tops):
        for unique in data[col].unique():
            list_of_groups.append(grouped[grouped[col] == unique].head(top))
    grouped_t_c = pd.concat(list_of_groups)

    fig = px.sunburst(grouped_t_c, path=cols, values = 0)

    fig.update_layout(autosize=False, width=700, height=700)
    fig.show()
```

Рисунок 6.7 – функция для построения дерева решений

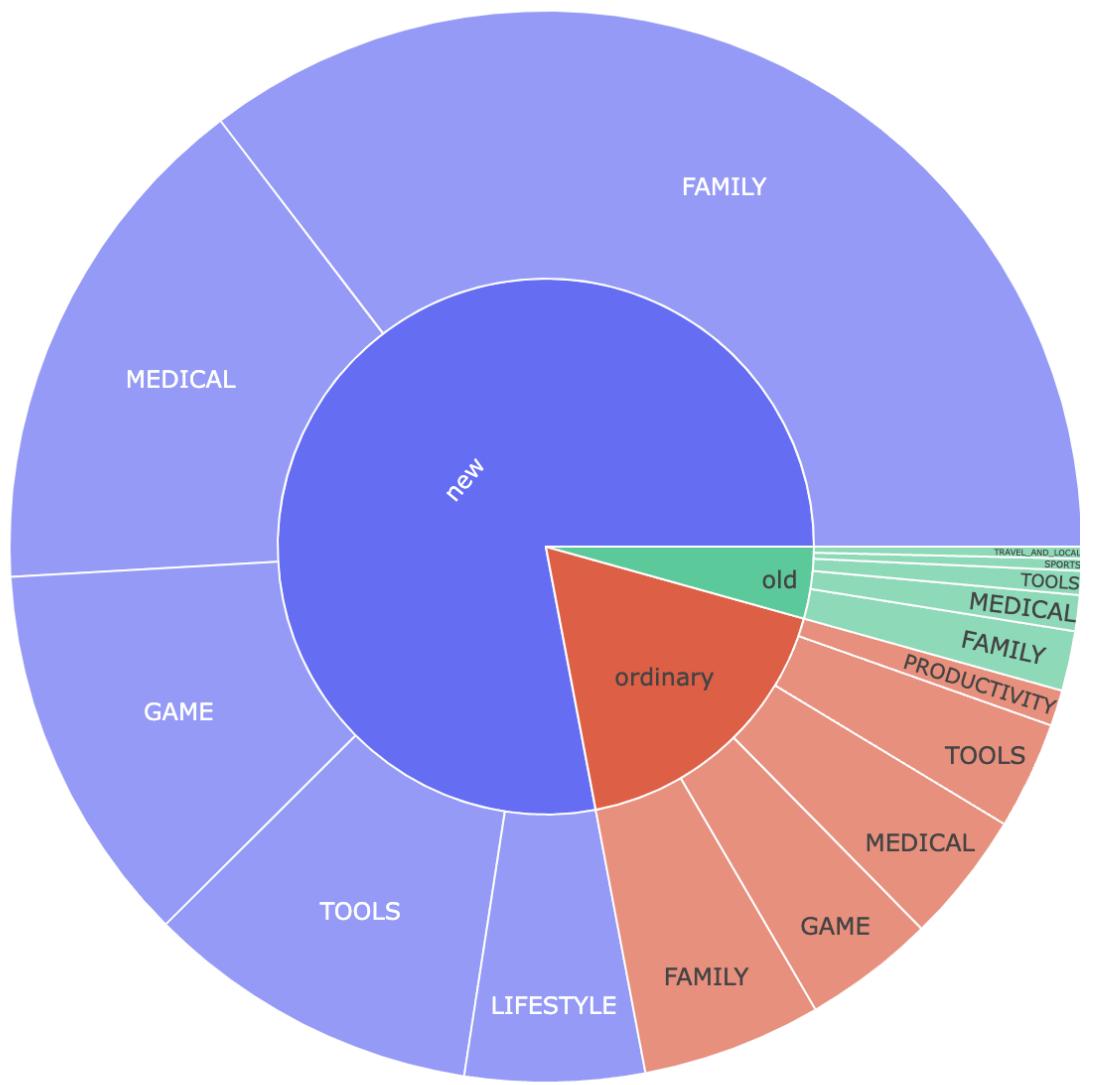


Рисунок 6.8 – дерево решений

Заметим, что действительно продаются игры с новыми обновлениями, при этом по графикам можно сделать несколько выводов

- самый популярный жанр у платных приложений - family и medical — это может быть связано с тем, что чаще всего такие приложения покупают люди старше 30 лет, те, кому семья и здоровье интереснее игр.
  - средний рейтинг у игр с новым, старым и недавнем обновлением одинаковый

## **Вывод**

Магазин Google Play - крупнейший рынок приложений в мире. Он генерирует более чем в два раза больше загрузок, чем Apple App Store, но зарабатывает лишь половину денег, чем App Store. Итак, мы собрали данные из Play Store, чтобы провести исследование. В Play Store есть основная проблема: отслеживание выпуска обновлений и уведомление разработчиков, если давно не было обновлений, поможет избежать резкого падения приложения с низким количеством установок.

Параметр успеха для приложения, основанный на количестве установок, распределении оценок 4 и 5 относительно общего количества оценок, соотношении установок к рейтингу, содержании.

## **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы был проведен анализ базы данных, выявлены важные статистические данные.

Для выполнения данной работы было изучено новое программное обеспечение: «Pandas» и «Seaborn» на базе языка программирования Python.

## **Список используемых источников**

- 1) Методические указания по программному обеспечению «Pandas»;
- 2) Методические указания по программному обеспечению «Seaborn»;
- 3) «Pandas. Работа с данными» (2020), Автор: Абдрахманов М. И.;
- 4) «Python. Визуализация данных: Matplotlib, Seaborn, Mayavi»;
- 5) <https://devpractice.ru>;
- 6) <https://coderoad.wiki>;
- 7) <https://www.delftstack.com>;
- 8) <https://www.machinelearningmastery.ru>;
- 9) <https://pythonru.com>.