МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.Баумана

Отчет по лабораторной работе №2

по курсу «Технологии машинного обучения»

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

Подготовил

Ионов С.А.

ИУ5-62Б

1. Описание задания
2. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.)
3. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   * обработку пропусков в данных;
   * кодирование категориальных признаков;
   * масштабирование данных.
4. Текст программы

import numpy as np

import scipy as sc

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from pylab import rcParams # для того, чтобы задавать размер диаграмм

%matplotlib inline

from sklearn.impute import SimpleImputer

from sklearn.impute import MissingIndicator

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.preprocessing import MinMaxScaler

data = pd.read\_csv('data/googleplaystore.csv', sep=",")

data.head()

data.info()

Вычисляем количество и процент пропусков в датасете.

mising\_count = data.isnull().sum()

all\_count = data.isnull().count()

pd.concat([mising\_count.sort\_values(), (mising\_count/all\_count\*100).sort\_values()],

axis=1, keys=['Количество пропусков', 'Процент пропусков']).tail(6)

**1) Обработка пропусков в данных**

rcParams['figure.figsize'] = 7,7

g = sns.kdeplot(data=data, x="Rating", shade=True)

g.set\_xlabel("Rating", size = 15)

g.set\_ylabel("Frequency", size = 15)

plt.title('Distribution of Rating', size = 18)

indicator = MissingIndicator()

mask\_missing\_values\_only = indicator.fit\_transform(data[['Rating']])

imp\_num = SimpleImputer(strategy='most\_frequent')

data\_num\_imp = imp\_num.fit\_transform(data[['Rating']])

data['Rating'] = data\_num\_imp

filled\_data = data\_num\_imp[mask\_missing\_values\_only]

print('Rating', 'most\_frequent', filled\_data.size, filled\_data[0], filled\_data[filled\_data.size-1], sep='; ')

# Ищем строки, в которых атрибут "Reviews" не является численным

data[~data['Reviews'].str.isnumeric()]

data.head()

# выравниваем строку в соответствии со столбцами

for i in range(data.shape[1]-1, 1, -1):

data.iloc[10472, i] = data.iloc[10472, i-1]

print(f"Category:\n{data['Category'].unique()}")

print(f"\nGenres:\n{data['Genres'].unique()}")

data.loc[10472, "Category"], data.loc[10472, "Genres"] = 'PHOTOGRAPHY', 'Photography'

data['Rating'] = data['Rating'].astype('float')

data['Reviews'] = data['Reviews'].astype('int')

# Ищем строки, в которых атрибут "Type" содержит пропуски

data[data['Type'].isnull()]

data['Type'].unique()

data['Type'].fillna('Free', inplace = True)

# Ищем строки, в которых атрибут "Current Ver" содержит пропуски

data[data['Current Ver'].isnull()]

# Ищем самые применяемые значения для данного атрибута

data['Current Ver'].value\_counts()

# Заменяем пропущенные значения на "Varies with device"

imp = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

data['Current Ver'] = imp.fit\_transform(data[['Current Ver']])

# Ищем строки, в которых атрибут "Android Ver" содержит пропуски

data[data['Android Ver'].isnull()]

# Ищем самые применяемые значения для данного атрибута

data['Android Ver'].value\_counts()

data['Android Ver'].fillna(value="Varies with device", inplace=True)

**2) Преобразование категориальных признаков в числовые**

data['Type'].unique()

le = LabelEncoder()

data['Type'] = le.fit\_transform(data['Type'])

print(f"Количество уникальных записей атрибута 'App' = {data['App'].nunique()} из {data.shape[0]}")

data.drop\_duplicates(inplace = True)

# проверяем уникальные значения признака "Category"

data['Category'].unique()

# кодирование категорий one-hot encoding

pd.get\_dummies(data, columns = ['Category'])

# проверяем уникальные значения признака "Size"

data['Size'].unique()

len(data[data.Size == 'Varies with device'])

data['Size'].replace('Varies with device', np.nan, inplace = True )

data['Size'] = data['Size'].str.replace('M', 'e+6').str.replace('k', 'e+3').astype('float')

g = sns.kdeplot(data=data, x="Size", shade=True)

g.set\_xlabel("Size", size = 15)

g.set\_ylabel("Frequency", size = 15)

plt.title('Distribution of Size', size = 18)

indicator = MissingIndicator()

mask\_missing\_values\_only = indicator.fit\_transform(data[['Size']])

imp\_num = SimpleImputer(strategy='median')

data\_num\_imp = imp\_num.fit\_transform(data[['Size']])

data['Size'] = data\_num\_imp

filled\_data = data\_num\_imp[mask\_missing\_values\_only]

print('Size', 'median', filled\_data.size, filled\_data[0], filled\_data[filled\_data.size-1], sep='; ')

# проверяем уникальные значения признака "Installs"

data['Installs'].unique()

data['Installs'] = data['Installs'].str.replace(',', '').str.replace('+', '').astype('int')

# проверяем уникальные значения признака "Price"

data['Price'].unique()

data['Price'] = data['Price'].str.replace('$', '').astype('float')

# проверяем уникальные значения признака "Content Rating"

data['Content Rating'].unique()

# кодирование категорий one-hot encoding

pd.get\_dummies(data, columns = ['Content Rating']).head()

**3) Масштабирование данных**

fig, axes = plt.subplots(6, figsize=(20, 30))

axes[0].set\_title('Rating', size=18)

axes[0].hist(data['Rating'])

axes[1].set\_title('Reviews', size=18)

axes[1].hist(data['Reviews'])

axes[2].set\_title('Size', size=18)

axes[2].hist(data['Size'])

axes[3].set\_title('Installs', size=18)

axes[3].hist(data['Installs'])

axes[4].set\_title('Type', size=18)

axes[4].hist(data['Type'])

axes[5].set\_title('Price', size=18)

axes[5].hist(data['Price'])

sc = MinMaxScaler()

data['Rating'] = sc.fit\_transform(data[['Rating']])

data['Reviews'] = sc.fit\_transform(data[['Reviews']])

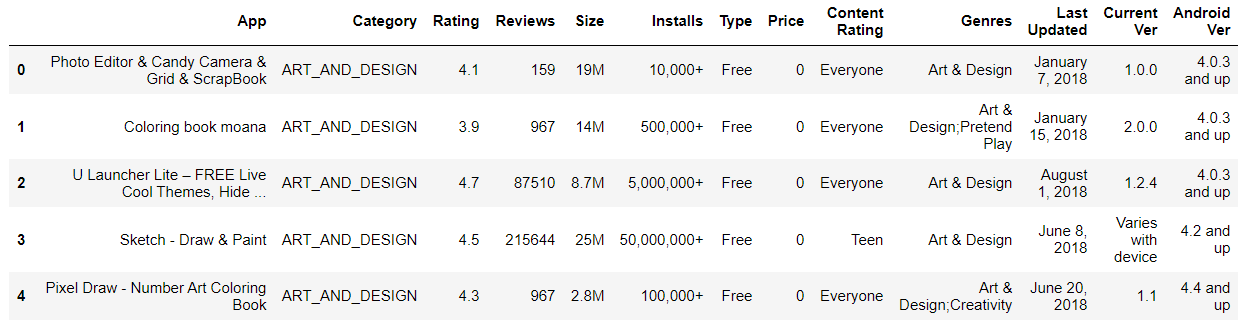
data['Size'] = sc.fit\_transform(data[['Size']])

data['Installs'] = sc.fit\_transform(data[['Installs']])

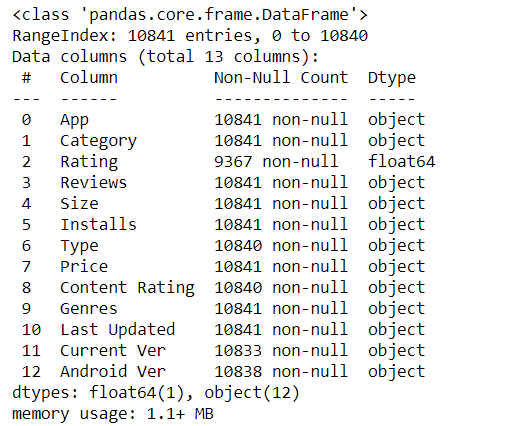
data['Price'] = sc.fit\_transform(data[['Price']])

1. Экранные формы с примерами выполнения программы
2. **Обработка пропусков в данных**

* Для лабораторной работы был выбран [Google Play Store Apps](https://www.kaggle.com/lava18/google-play-store-apps)
* Первые 5 строк набора данных имеют вид:



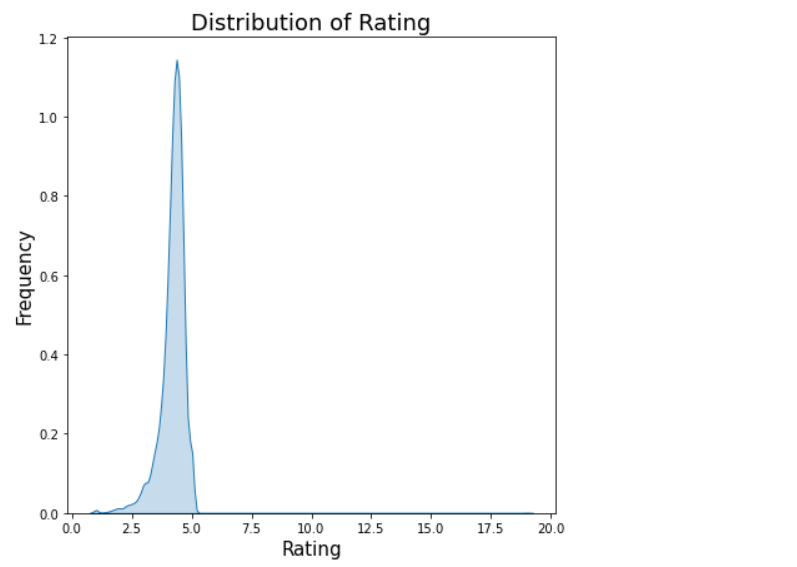
* Набор данных имеет 10841 строк и 13 столбцов:



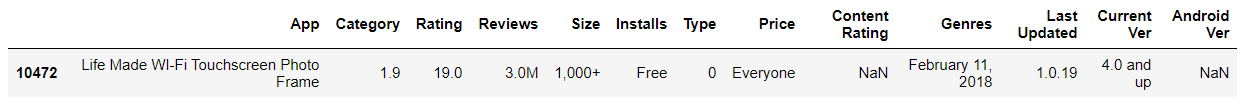
* Количество и процент пропусков в датасете:



Хотя мы имеем около 13% пропусков в колонке "Rating", что выше идеального порога 5% для восстановления, мы попытаемся все равно восстановить значения, так как мы имеем мало признаков. Для этого строим гистограмму распределения:



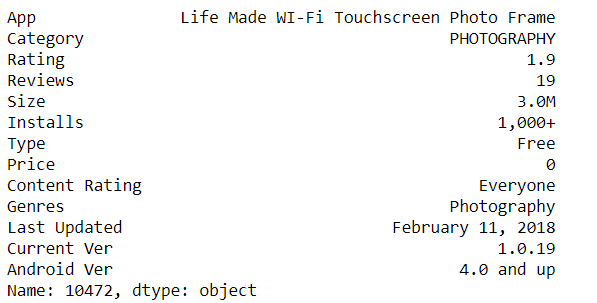
Гистограмма имеет практически идеальное одномодальное распределение, поэтому для заполнения пропусков можно использовать моду. Атрибут «Review» по смыслу должен быть численным. Ищем строки, в которых это не выполняется:



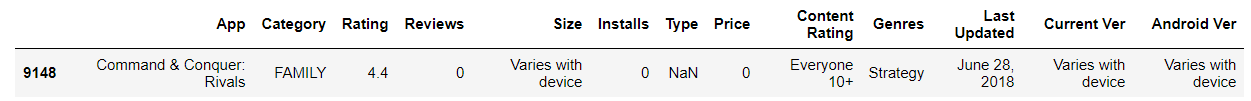
Можно заметить, что строка, в которой атрибут "Review" не является численным, сдвинута влево относительно столбцов (начиная с атрибута "Category").

После приведения к нормальному виду атрибуты "Genres" и "Category" станут неопределенными. Но их можно подобрать, исходя из названия приложения. Наиболее подходящими являются категория "PHOTOGRAPHY" и жанр "Photography".

В итоге запись 10472 принимает следующий вид:

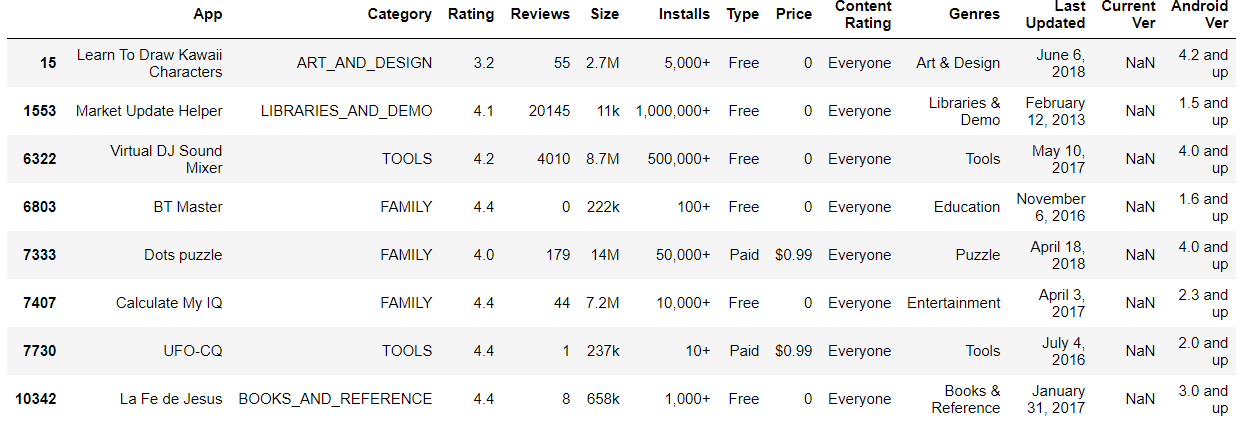


Ищем строки, в которых атрибут «Type» содержит пропуски:

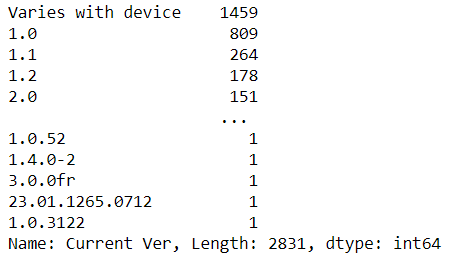


Строка, в которой есть пропуск у атрибута "Type", имеет атрибут "Price", равный нулю. Поэтому тип данного приложения можно классифицировать, как бесплатный (Free).

Аналогично, атрибут «Current Ver» содержит пропуски:



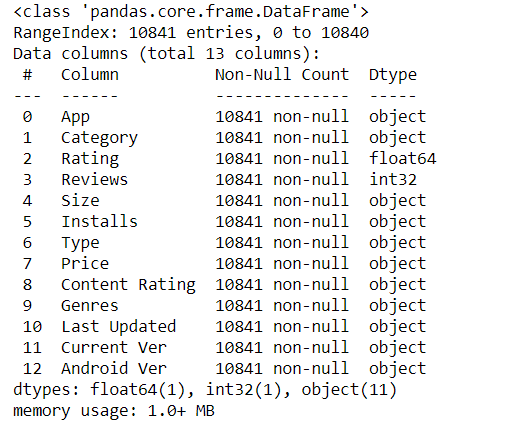
Ищем самые применяемые значения для данного атрибута:



Заменяем пропущенные значения на "Varies with device":

Аналогично присвоим пропущенным строкам атрибута "Android Ver" значения "Varies with device". В следующем разделе будем заменять это значение на максимальную версию за год, в котором было выпущено последнее обновление приложения.

* Теперь датасет не содержит пустых значений:



1. **Преобразование категориальных признаков в числовые**

Атрибут "Type" подходит для кодирования целочисленными значениями.

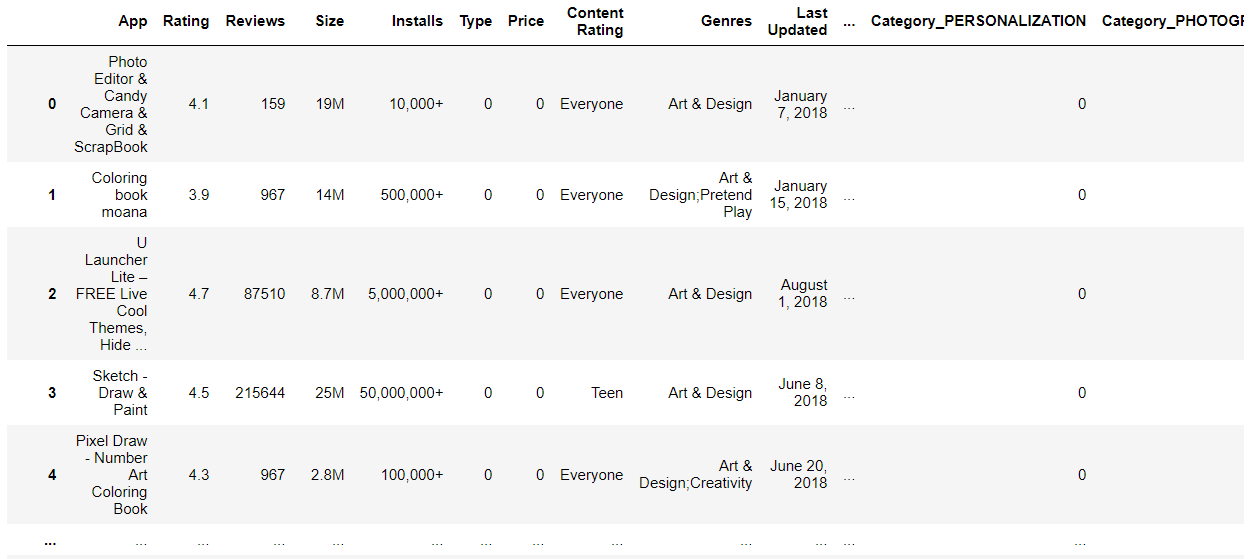
Определяем количество уникальных элементов атрибута "App":



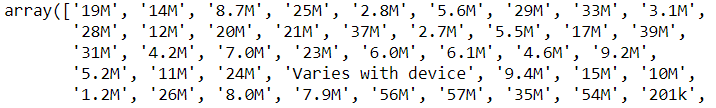
Отсюда следует, что атрибут содержит дублированные данные, которые следует устранить:



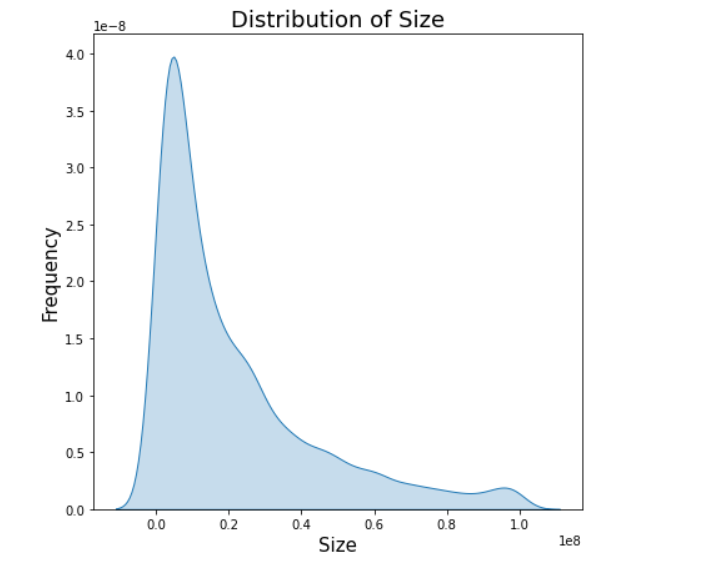
Кодируем атрибут «Category» методом «one-hot encoding»:



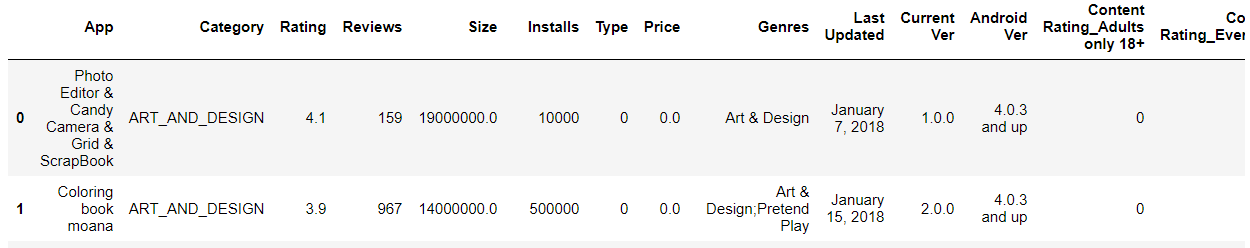
Признак «Size» можно привести к числовому, заменив строковые части: «M», «k» на соответствующие приставки: 1000000 и 1000, а также значения «Varies with device» каким-либо среднестатистическим значением:



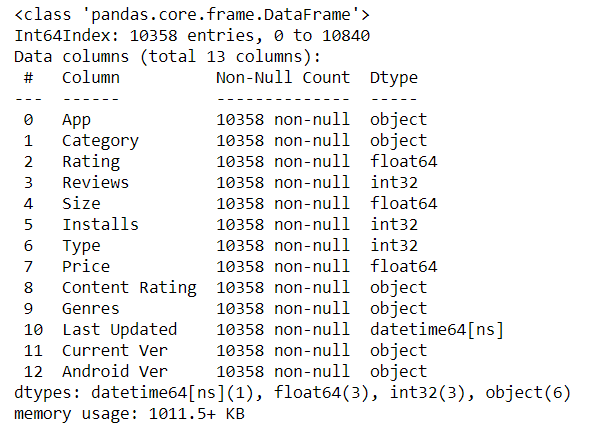
Исходя из распределения для заполнения значений будем использовать медиану:



Аналогично устраняем строковые компоненты в атрибутах «Installs», «Price» и приводим их числовому типу, выполняем «one-hot» кодирование для «Content Rating»:

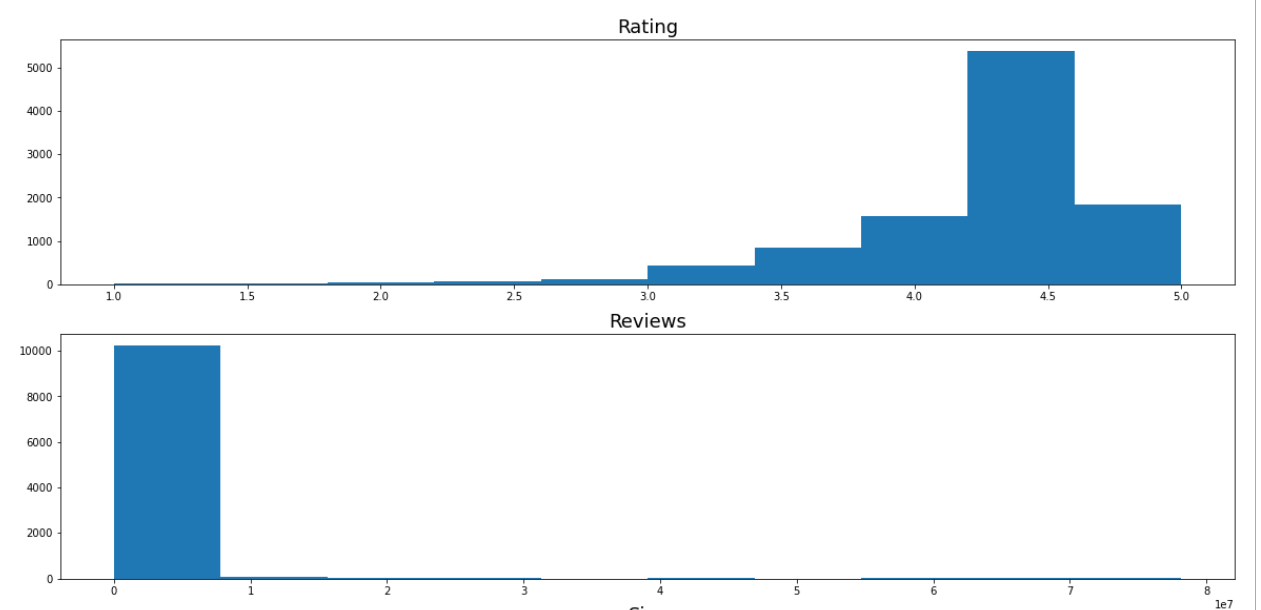


После всех преобразований получаем следующий датасет:



1. **Масштабирование данных**

Масштабируем числовые признаки к диапазону (0; 1). До масштабирования диаграммы выглядели следующим образом:



После масштабирования диаграммы стали выглядеть следующим образом:

