```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import cm
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, r2_score
from tensorflow.keras.losses import mean_squared_error

data = pd.read_csv("googleplaystore.csv")

df=data

df.head()
```

```
                                                 App        Category
Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN
4.1
1                                 Coloring book moana  ART_AND_DESIGN
3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN
4.7
3                                Sketch - Draw & Paint  ART_AND_DESIGN
4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN
4.3

  Reviews  Size     Installs  Type Price Content Rating  \
0     159   19M      10,000+  Free     0       Everyone
1     967   14M     500,000+  Free     0       Everyone
2   87510  8.7M   5,000,000+  Free     0       Everyone
3  215644   25M  50,000,000+  Free     0           Teen
4     967  2.8M     100,000+  Free     0       Everyone

                      Genres      Last Updated         Current Ver  \
0               Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2               Art & Design    August 1, 2018               1.2.4
3               Art & Design      June 8, 2018  Varies with device
```

```
4    Art & Design;Creativity     June 20, 2018                  1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

Data Cleaning

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 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   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

```
# replace white spaces froh header.

df.columns = df.columns.str.replace(' ','_',regex=True)
df.columns
```

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs',
'Type',
       'Price', 'Content_Rating', 'Genres', 'Last_Updated',
'Current_Ver',
       'Android_Ver'],
      dtype='object')
```

```
df.describe()
```

```
             Rating
count  9367.000000
mean      4.193338
std       0.537431
min       1.000000
25%       4.000000
50%       4.300000
75%       4.500000
max      19.000000
```

```
df.dropna(axis = 0, inplace = True)
```

```
df.isnull().sum()
```

```
App               0
Category          0
Rating            0
Reviews           0
Size              0
Installs          0
Type              0
Price             0
Content_Rating    0
Genres            0
Last_Updated      0
Current_Ver       0
Android_Ver       0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             9360 non-null   object 
 1   Category        9360 non-null   object 
 2   Rating          9360 non-null   float64
 3   Reviews         9360 non-null   object 
 4   Size            9360 non-null   object 
 5   Installs        9360 non-null   object 
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   object 
 8   Content_Rating  9360 non-null   object 
 9   Genres          9360 non-null   object 
 10  Last_Updated    9360 non-null   object 
 11  Current_Ver     9360 non-null   object 
 12  Android_Ver     9360 non-null   object
```

```
dtypes: float64(1), object(12)
memory usage: 1023.8+ KB
```

```
df['Category'].describe()
```

```
count       9360
unique        33
top       FAMILY
freq        1746
Name: Category, dtype: object
```

```
print( len(df['Category'].unique()) , 'categories')
```

```
print("\n", df['Category'].unique())
```

```
33 categories
```

```
 ['ART_AND_DESIGN' 'AUTO_AND_VEHICLES' 'BEAUTY' 'BOOKS_AND_REFERENCE'
 'BUSINESS' 'COMICS' 'COMMUNICATION' 'DATING' 'EDUCATION'
'ENTERTAINMENT'
 'EVENTS' 'FINANCE' 'FOOD_AND_DRINK' 'HEALTH_AND_FITNESS'
'HOUSE_AND_HOME'
 'LIBRARIES_AND_DEMO' 'LIFESTYLE' 'GAME' 'FAMILY' 'MEDICAL' 'SOCIAL'
 'SHOPPING' 'PHOTOGRAPHY' 'SPORTS' 'TRAVEL_AND_LOCAL' 'TOOLS'
 'PERSONALIZATION' 'PRODUCTIVITY' 'PARENTING' 'WEATHER'
'VIDEO_PLAYERS'
 'NEWS_AND_MAGAZINES' 'MAPS_AND_NAVIGATION']
```

```
df['Reviews'].describe()
```

```
count     9360
unique    5990
top          2
freq        83
Name: Reviews, dtype: object
```

```
df['Reviews']
```

```
0           159
1           967
2         87510
3        215644
4           967
          ...
10834         7
10836        38
10837         4
10839       114
10840    398307
Name: Reviews, Length: 9360, dtype: object
```

```
# Convert to int
df['Reviews'] = df['Reviews'].apply(lambda x: int(x))
df['Reviews'].describe()
```

```
count    9.360000e+03
mean     5.143767e+05
std      3.145023e+06
min      1.000000e+00
25%      1.867500e+02
50%      5.955000e+03
75%      8.162750e+04
max      7.815831e+07
Name: Reviews, dtype: float64
```

```
df['Size'].describe()
```

```
count                   9360
unique                   413
top       Varies with device
freq                    1637
Name: Size, dtype: object
```

```
df['Size']
```

```
0                        19M
1                        14M
2                       8.7M
3                        25M
4                       2.8M
                ...
10834                   2.6M
10836                    53M
10837                   3.6M
10839    Varies with device
10840                    19M
Name: Size, Length: 9360, dtype: object
```

```
print( len(df['Size'].unique()) , "categories")

print("\n", df['Size'].unique())
```

```
413 categories

 ['19M' '14M' '8.7M' '25M' '2.8M' '5.6M' '29M' '33M' '3.1M' '28M'
'12M'
 '20M' '21M' '37M' '5.5M' '17M' '39M' '31M' '4.2M' '23M' '6.0M' '6.1M'
 '4.6M' '9.2M' '5.2M' '11M' '24M' 'Varies with device' '9.4M' '15M'
'10M'
 '1.2M' '26M' '8.0M' '7.9M' '56M' '57M' '35M' '54M' '201k' '3.6M'
'5.7M'
 '8.6M' '2.4M' '27M' '2.7M' '2.5M' '7.0M' '16M' '3.4M' '8.9M' '3.9M'
```

```
 '2.9M' '38M' '32M' '5.4M' '18M' '1.1M' '2.2M' '4.5M' '9.8M' '52M'
'9.0M'
 '6.7M' '30M' '2.6M' '7.1M' '22M' '6.4M' '3.2M' '8.2M' '4.9M' '9.5M'
 '5.0M' '5.9M' '13M' '73M' '6.8M' '3.5M' '4.0M' '2.3M' '2.1M' '42M'
'9.1M'
 '55M' '23k' '7.3M' '6.5M' '1.5M' '7.5M' '51M' '41M' '48M' '8.5M'
'46M'
 '8.3M' '4.3M' '4.7M' '3.3M' '40M' '7.8M' '8.8M' '6.6M' '5.1M' '61M'
'66M'
 '79k' '8.4M' '3.7M' '118k' '44M' '695k' '1.6M' '6.2M' '53M' '1.4M'
'3.0M'
 '7.2M' '5.8M' '3.8M' '9.6M' '45M' '63M' '49M' '77M' '4.4M' '70M'
'9.3M'
 '8.1M' '36M' '6.9M' '7.4M' '84M' '97M' '2.0M' '1.9M' '1.8M' '5.3M'
'47M'
 '556k' '526k' '76M' '7.6M' '59M' '9.7M' '78M' '72M' '43M' '7.7M'
'6.3M'
 '334k' '93M' '65M' '79M' '100M' '58M' '50M' '68M' '64M' '34M' '67M'
'60M'
 '94M' '9.9M' '232k' '99M' '624k' '95M' '8.5k' '41k' '292k' '80M'
'1.7M'
 '10.0M' '74M' '62M' '69M' '75M' '98M' '85M' '82M' '96M' '87M' '71M'
'86M'
 '91M' '81M' '92M' '83M' '88M' '704k' '862k' '899k' '378k' '4.8M'
'266k'
 '375k' '1.3M' '975k' '980k' '4.1M' '89M' '696k' '544k' '525k' '920k'
 '779k' '853k' '720k' '713k' '772k' '318k' '58k' '241k' '196k' '857k'
 '51k' '953k' '865k' '251k' '930k' '540k' '313k' '746k' '203k' '26k'
 '314k' '239k' '371k' '220k' '730k' '756k' '91k' '293k' '17k' '74k'
'14k'
 '317k' '78k' '924k' '818k' '81k' '939k' '169k' '45k' '965k' '90M'
'545k'
 '61k' '283k' '655k' '714k' '93k' '872k' '121k' '322k' '976k' '206k'
 '954k' '444k' '717k' '210k' '609k' '308k' '306k' '175k' '350k' '383k'
 '454k' '1.0M' '70k' '812k' '442k' '842k' '417k' '412k' '459k' '478k'
 '335k' '782k' '721k' '430k' '429k' '192k' '460k' '728k' '496k' '816k'
 '414k' '506k' '887k' '613k' '778k' '683k' '592k' '186k' '840k' '647k'
 '373k' '437k' '598k' '716k' '585k' '982k' '219k' '55k' '323k' '691k'
 '511k' '951k' '963k' '25k' '554k' '351k' '27k' '82k' '208k' '551k'
'29k'
 '103k' '116k' '153k' '209k' '499k' '173k' '597k' '809k' '122k' '411k'
 '400k' '801k' '787k' '50k' '643k' '986k' '516k' '837k' '780k' '20k'
 '498k' '600k' '656k' '221k' '228k' '176k' '34k' '259k' '164k' '458k'
 '629k' '28k' '288k' '775k' '785k' '636k' '916k' '994k' '309k' '485k'
 '914k' '903k' '608k' '500k' '54k' '562k' '847k' '948k' '811k' '270k'
 '48k' '523k' '784k' '280k' '24k' '892k' '154k' '18k' '33k' '860k'
'364k'
 '387k' '626k' '161k' '879k' '39k' '170k' '141k' '160k' '144k' '143k'
 '190k' '376k' '193k' '473k' '246k' '73k' '253k' '957k' '420k' '72k'
```

```
 '404k' '470k' '226k' '240k' '89k' '234k' '257k' '861k' '467k' '676k'
 '552k' '582k' '619k']
```

```
df.Size.value_counts().head()
```

```
Size
Varies with device    1637
14M                    165
12M                    161
15M                    159
11M                    159
Name: count, dtype: int64
```

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

```
1637
```

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

```
def convert_size_to_bytes(size_str):
    if size_str == 'Varies with device':
        return np.nan
    elif isinstance(size_str, str):
        if size_str.endswith('M'):
            return float(size_str[:-1]) * 1024 * 1024
        elif size_str.endswith('k'):
            return float(size_str[:-1]) * 1024
    return np.nan

df['Size'] = df['Size'].apply(convert_size_to_bytes)

print(df['Size'].dtype)
```

```
float64
```

```
print(df['Size'].unique()[:5])
```

```
[19922944.  14680064.   9122611.2 26214400.   2936012.8]
```

```
df.Size = pd.to_numeric(df.Size)

df['Size'].fillna(df.groupby('Category')
['Size'].transform('mean'),inplace = True)

df.Size.describe()
```

```
count    9.360000e+03
mean     2.358802e+07
std      2.272194e+07
min      8.704000e+03
25%      6.920602e+06
50%      1.677722e+07
```

```
75%      3.145728e+07
max      1.048576e+08
Name: Size, dtype: float64
```

```
print( len(df['Installs'].unique()) , "categories")

print("\n", df['Installs'].unique())
```

```
19 categories

 ['10,000+' '500,000+' '5,000,000+' '50,000,000+' '100,000+' '50,000+'
 '1,000,000+' '10,000,000+' '5,000+' '100,000,000+' '1,000,000,000+'
 '1,000+' '500,000,000+' '100+' '500+' '10+' '5+' '50+' '1+']
```

```
df.Installs.value_counts()
```

```
Installs
1,000,000+        1576
10,000,000+       1252
100,000+          1150
10,000+           1009
5,000,000+         752
1,000+             712
500,000+           537
50,000+            466
5,000+             431
100,000,000+       409
100+               309
50,000,000+        289
500+               201
500,000,000+        72
10+                 69
1,000,000,000+      58
50+                 56
5+                   9
1+                   3
Name: count, dtype: int64
```

```
df.Installs = df.Installs.apply(lambda x: x.replace(',',''))
df.Installs = df.Installs.apply(lambda x: x.replace('+',''))
df.Installs = df.Installs.apply(lambda x: int(x))
```

```
df.Installs.unique()
```

```
array([     10000,     500000,    5000000,   50000000,     100000,
           50000,    1000000,   10000000,       5000,  100000000,
      1000000000,       1000,  500000000,        100,        500,
              10,          5,         50,          1], dtype=int64)
```

```
df.Type.value_counts()
```

```
Type
Free    8715
Paid     645
Name: count, dtype: int64
```

```
df.Price.value_counts()
```

```
Price
0          8715
$2.99       114
$0.99       106
$4.99        70
$1.99        59
           ...
$1.29          1
$299.99        1
$379.99        1
$37.99         1
$1.20          1
Name: count, Length: 73, dtype: int64
```

```
print( len(df['Price'].unique()) , "categories")

print("\n", df['Price'].unique())
```

```
73 categories

 ['0' '$4.99' '$3.99' '$6.99' '$7.99' '$5.99' '$2.99' '$3.49' '$1.99'
 '$9.99' '$7.49' '$0.99' '$9.00' '$5.49' '$10.00' '$24.99' '$11.99'
 '$79.99' '$16.99' '$14.99' '$29.99' '$12.99' '$2.49' '$10.99' '$1.50'
 '$19.99' '$15.99' '$33.99' '$39.99' '$3.95' '$4.49' '$1.70' '$8.99'
 '$1.49' '$3.88' '$399.99' '$17.99' '$400.00' '$3.02' '$1.76' '$4.84'
 '$4.77' '$1.61' '$2.50' '$1.59' '$6.49' '$1.29' '$299.99' '$379.99'
 '$37.99' '$18.99' '$389.99' '$8.49' '$1.75' '$14.00' '$2.00' '$3.08'
 '$2.59' '$19.40' '$3.90' '$4.59' '$15.46' '$3.04' '$13.99' '$4.29'
 '$3.28' '$4.60' '$1.00' '$2.95' '$2.90' '$1.97' '$2.56' '$1.20']
```

```
df.Price = df.Price.apply(lambda x: x.replace('$',''))
df['Price'] = df['Price'].apply(lambda x: float(x))
```

```
df.Price.describe()
```

```
count    9360.000000
mean        0.961279
std        15.821640
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       400.000000
Name: Price, dtype: float64
```

```
df.Price.unique()
```

```
array([  0.  ,   4.99,   3.99,   6.99,   7.99,   5.99,   2.99,   3.49,
         1.99,   9.99,   7.49,   0.99,   9.  ,   5.49,  10.  ,  24.99,
        11.99,  79.99,  16.99,  14.99,  29.99,  12.99,   2.49,  10.99,
         1.5 ,  19.99,  15.99,  33.99,  39.99,   3.95,   4.49,   1.7 ,
         8.99,   1.49,   3.88, 399.99,  17.99, 400.  ,   3.02,   1.76,
         4.84,   4.77,   1.61,   2.5 ,   1.59,   6.49,   1.29, 299.99,
       379.99,  37.99,  18.99, 389.99,   8.49,   1.75,  14.  ,   2.  ,
         3.08,   2.59,  19.4 ,   3.9 ,   4.59,  15.46,   3.04,  13.99,
         4.29,   3.28,   4.6 ,   1.  ,   2.95,   2.9 ,   1.97,   2.56,
         1.2 ])
```

```
df['Content_Rating'].unique()
```

```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

```
df['Content_Rating'].value_counts()
```

```
Content_Rating
Everyone           7414
Teen               1084
Mature 17+          461
Everyone 10+        397
Adults only 18+       3
Unrated               1
Name: count, dtype: int64
```

```
print( len(df['Genres'].unique()) , "categories")
```

```
print("\n", df['Genres'].unique())
```

```
115 categories
```

```
 ['Art & Design' 'Art & Design;Pretend Play' 'Art & Design;Creativity'
 'Auto & Vehicles' 'Beauty' 'Books & Reference' 'Business' 'Comics'
 'Comics;Creativity' 'Communication' 'Dating' 'Education;Education'
 'Education' 'Education;Creativity' 'Education;Music & Video'
 'Education;Action & Adventure' 'Education;Pretend Play'
 'Education;Brain Games' 'Entertainment' 'Entertainment;Music & Video'
 'Entertainment;Brain Games' 'Entertainment;Creativity' 'Events'
'Finance'
 'Food & Drink' 'Health & Fitness' 'House & Home' 'Libraries & Demo'
 'Lifestyle' 'Lifestyle;Pretend Play' 'Adventure;Action & Adventure'
 'Arcade' 'Casual' 'Card' 'Casual;Pretend Play' 'Action' 'Strategy'
 'Puzzle' 'Sports' 'Music' 'Word' 'Racing' 'Casual;Creativity'
 'Casual;Action & Adventure' 'Simulation' 'Adventure' 'Board' 'Trivia'
 'Role Playing' 'Simulation;Education' 'Action;Action & Adventure'
 'Casual;Brain Games' 'Simulation;Action & Adventure'
 'Educational;Creativity' 'Puzzle;Brain Games' 'Educational;Education'
```

```
 'Card;Brain Games' 'Educational;Brain Games' 'Educational;Pretend
Play'
 'Entertainment;Education' 'Casual;Education' 'Music;Music & Video'
 'Racing;Action & Adventure' 'Arcade;Pretend Play'
 'Role Playing;Action & Adventure' 'Simulation;Pretend Play'
 'Puzzle;Creativity' 'Sports;Action & Adventure'
 'Educational;Action & Adventure' 'Arcade;Action & Adventure'
 'Entertainment;Action & Adventure' 'Puzzle;Action & Adventure'
 'Strategy;Action & Adventure' 'Music & Audio;Music & Video'
 'Health & Fitness;Education' 'Adventure;Education' 'Board;Brain
Games'
 'Board;Action & Adventure' 'Board;Pretend Play' 'Casual;Music &
Video'
 'Role Playing;Pretend Play' 'Entertainment;Pretend Play'
 'Video Players & Editors;Creativity' 'Card;Action & Adventure'
'Medical'
 'Social' 'Shopping' 'Photography' 'Travel & Local'
 'Travel & Local;Action & Adventure' 'Tools' 'Tools;Education'
 'Personalization' 'Productivity' 'Parenting' 'Parenting;Music &
Video'
 'Parenting;Brain Games' 'Parenting;Education' 'Weather'
 'Video Players & Editors' 'Video Players & Editors;Music & Video'
 'News & Magazines' 'Maps & Navigation'
 'Health & Fitness;Action & Adventure' 'Educational' 'Casino'
 'Adventure;Brain Games' 'Lifestyle;Education'
 'Books & Reference;Education' 'Puzzle;Education'
 'Role Playing;Brain Games' 'Strategy;Education' 'Racing;Pretend Play'
 'Communication;Creativity' 'Strategy;Creativity']
```

```
df.Genres.value_counts()
```

```
Genres
Tools                        732
Entertainment                533
Education                    468
Action                       358
Productivity                 351
                            ...
Parenting;Brain Games          1
Card;Brain Games               1
Tools;Education                1
Entertainment;Education        1
Strategy;Creativity            1
Name: count, Length: 115, dtype: int64
```

```
df['Genres'] = df['Genres'].str.split(';').str[0]
df['Genres'].replace('Music & Audio', 'Music',inplace = True)
```

```
print( len(df['Genres'].unique()) , "categories")

print("\n", df['Genres'].unique())
```

```
47 categories

 ['Art & Design' 'Auto & Vehicles' 'Beauty' 'Books & Reference'
'Business'
 'Comics' 'Communication' 'Dating' 'Education' 'Entertainment'
'Events'
 'Finance' 'Food & Drink' 'Health & Fitness' 'House & Home'
 'Libraries & Demo' 'Lifestyle' 'Adventure' 'Arcade' 'Casual' 'Card'
 'Action' 'Strategy' 'Puzzle' 'Sports' 'Music' 'Word' 'Racing'
 'Simulation' 'Board' 'Trivia' 'Role Playing' 'Educational'
 'Video Players & Editors' 'Medical' 'Social' 'Shopping' 'Photography'
 'Travel & Local' 'Tools' 'Personalization' 'Productivity' 'Parenting'
 'Weather' 'News & Magazines' 'Maps & Navigation' 'Casino']
```

```
df['Last_Updated']
```

```
0         January 7, 2018
1        January 15, 2018
2          August 1, 2018
3            June 8, 2018
4           June 20, 2018
               ...
10834       June 18, 2017
10836       July 25, 2017
10837        July 6, 2018
10839    January 19, 2015
10840       July 25, 2018
Name: Last_Updated, Length: 9360, dtype: object
```

```
df['new'] = pd.to_datetime(df['Last_Updated'])
df['new'].describe()
```

```
count                             9360
mean     2017-11-29 13:00:55.384615424
min                2010-05-21 00:00:00
25%                2017-10-09 00:00:00
50%                2018-06-01 00:00:00
75%                2018-07-24 00:00:00
max                2018-08-08 00:00:00
Name: new, dtype: object
```

```
df['lastupdate'] = (df['new'] -  df['new'].max()).dt.days
df['lastupdate'].head()
```

```
0   -213
1   -205
2     -7
```

```
3    -61
4    -49
Name: lastupdate, dtype: int64

df.head()

                                                 App        Category
Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN
4.1
1                                Coloring book moana  ART_AND_DESIGN
3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN
4.7
3                              Sketch - Draw & Paint  ART_AND_DESIGN
4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN
4.3

   Reviews        Size  Installs  Type  Price Content_Rating
Genres  \
0      159  19922944.0     10000  Free    0.0       Everyone  Art &
Design
1      967  14680064.0    500000  Free    0.0       Everyone  Art &
Design
2    87510   9122611.2   5000000  Free    0.0       Everyone  Art &
Design
3   215644  26214400.0  50000000  Free    0.0           Teen  Art &
Design
4      967   2936012.8    100000  Free    0.0       Everyone  Art &
Design

       Last_Updated         Current_Ver   Android_Ver        new
lastupdate
0   January 7, 2018               1.0.0  4.0.3 and up 2018-01-07
-213
1  January 15, 2018               2.0.0  4.0.3 and up 2018-01-15
-205
2    August 1, 2018               1.2.4  4.0.3 and up 2018-08-01
-7
3      June 8, 2018  Varies with device    4.2 and up 2018-06-08
-61
4     June 20, 2018                 1.1    4.4 and up 2018-06-20
-49
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9360 entries, 0 to 10840
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   App             9360 non-null   object        
 1   Category        9360 non-null   object        
 2   Rating          9360 non-null   float64       
 3   Reviews         9360 non-null   int64         
 4   Size            9360 non-null   float64       
 5   Installs        9360 non-null   int64         
 6   Type            9360 non-null   object        
 7   Price           9360 non-null   float64       
 8   Content_Rating  9360 non-null   object        
 9   Genres          9360 non-null   object        
 10  Last_Updated    9360 non-null   object        
 11  Current_Ver     9360 non-null   object        
 12  Android_Ver     9360 non-null   object        
 13  new             9360 non-null   datetime64[ns]
 14  lastupdate      9360 non-null   int64         
dtypes: datetime64[ns](1), float64(3), int64(3), object(8)
memory usage: 1.1+ MB
```

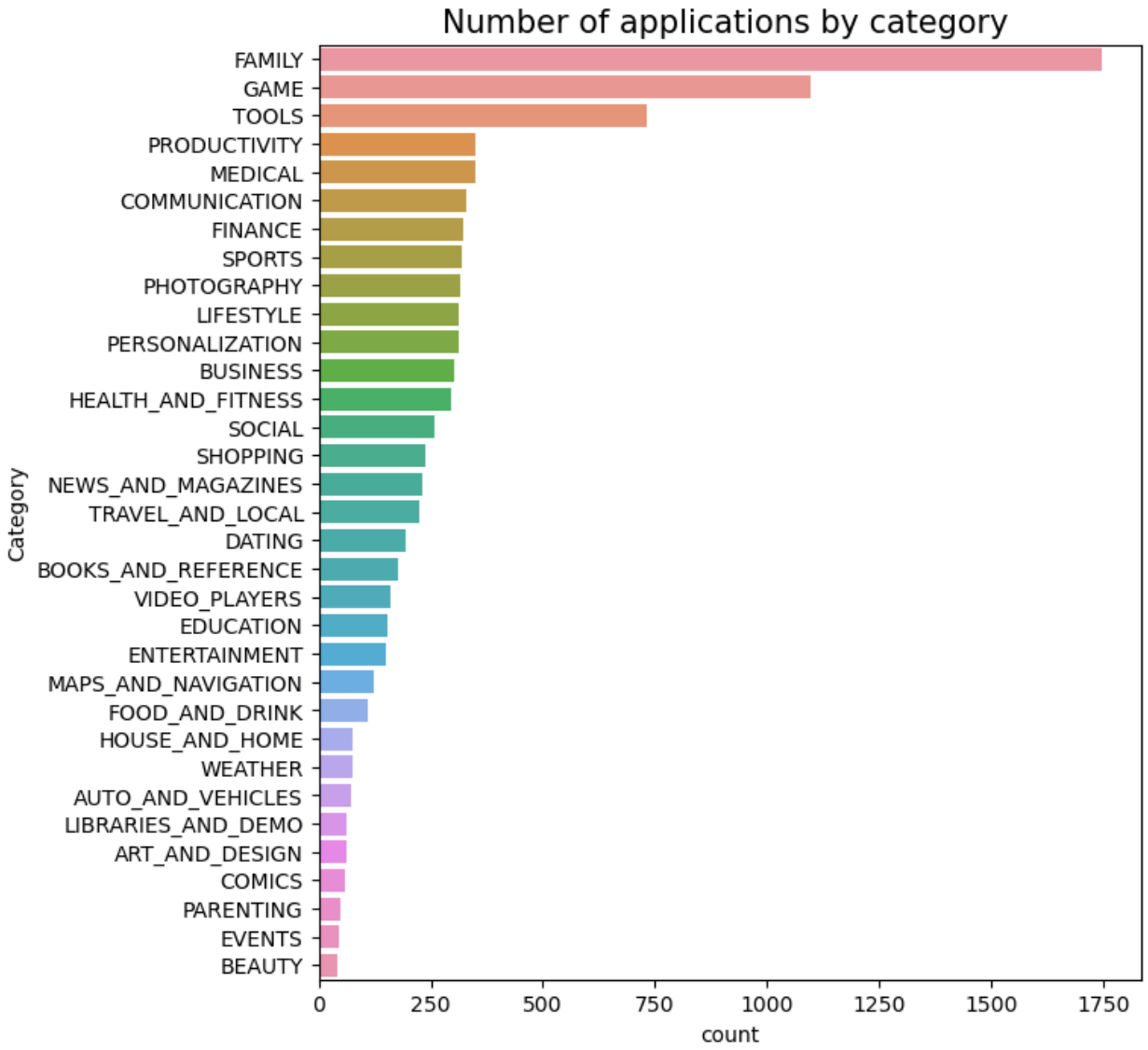
```
df.describe()
```

```
             Rating       Reviews          Size      Installs  \
Price  \
count  9360.000000  9.360000e+03  9.360000e+03  9.360000e+03   
9360.000000   
mean      4.191838  5.143767e+05  2.358802e+07  1.790875e+07   
0.961279   
min       1.000000  1.000000e+00  8.704000e+03  1.000000e+00   
0.000000   
25%       4.000000  1.867500e+02  6.920602e+06  1.000000e+04   
0.000000   
50%       4.300000  5.955000e+03  1.677722e+07  5.000000e+05   
0.000000   
75%       4.500000  8.162750e+04  3.145728e+07  5.000000e+06   
0.000000   
max       5.000000  7.815831e+07  1.048576e+08  1.000000e+09   
400.000000   
std       0.515263  3.145023e+06  2.272194e+07  9.126637e+07
```

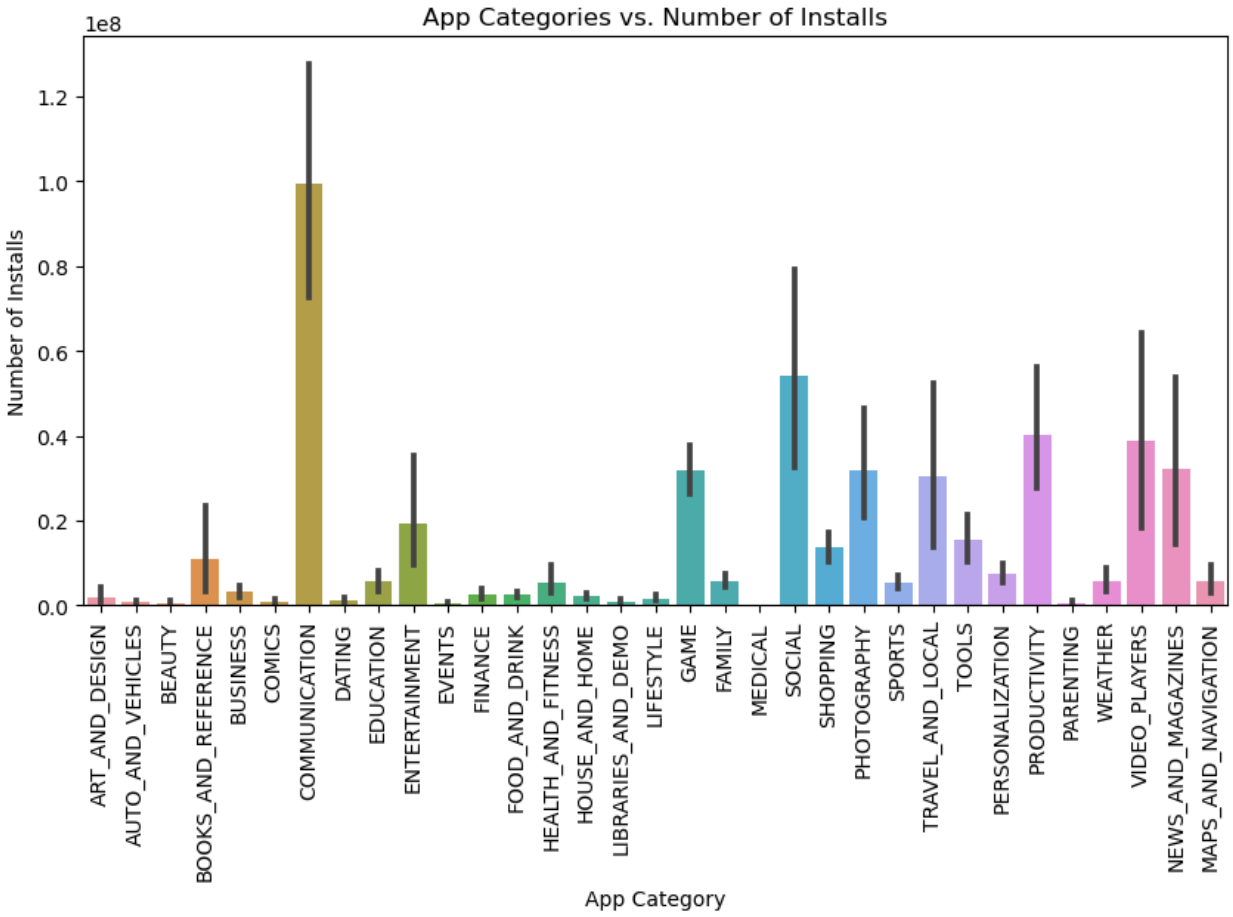
```
15.821640
```

```
                                new   lastupdate
count                          9360  9360.000000
mean   2017-11-29 13:00:55.384615424  -251.457692
min              2010-05-21 00:00:00 -3001.000000
25%              2017-10-09 00:00:00  -303.000000
50%              2018-06-01 00:00:00   -68.000000
75%              2018-07-24 00:00:00   -15.000000
max              2018-08-08 00:00:00     0.000000
std                              NaN   396.167305
```

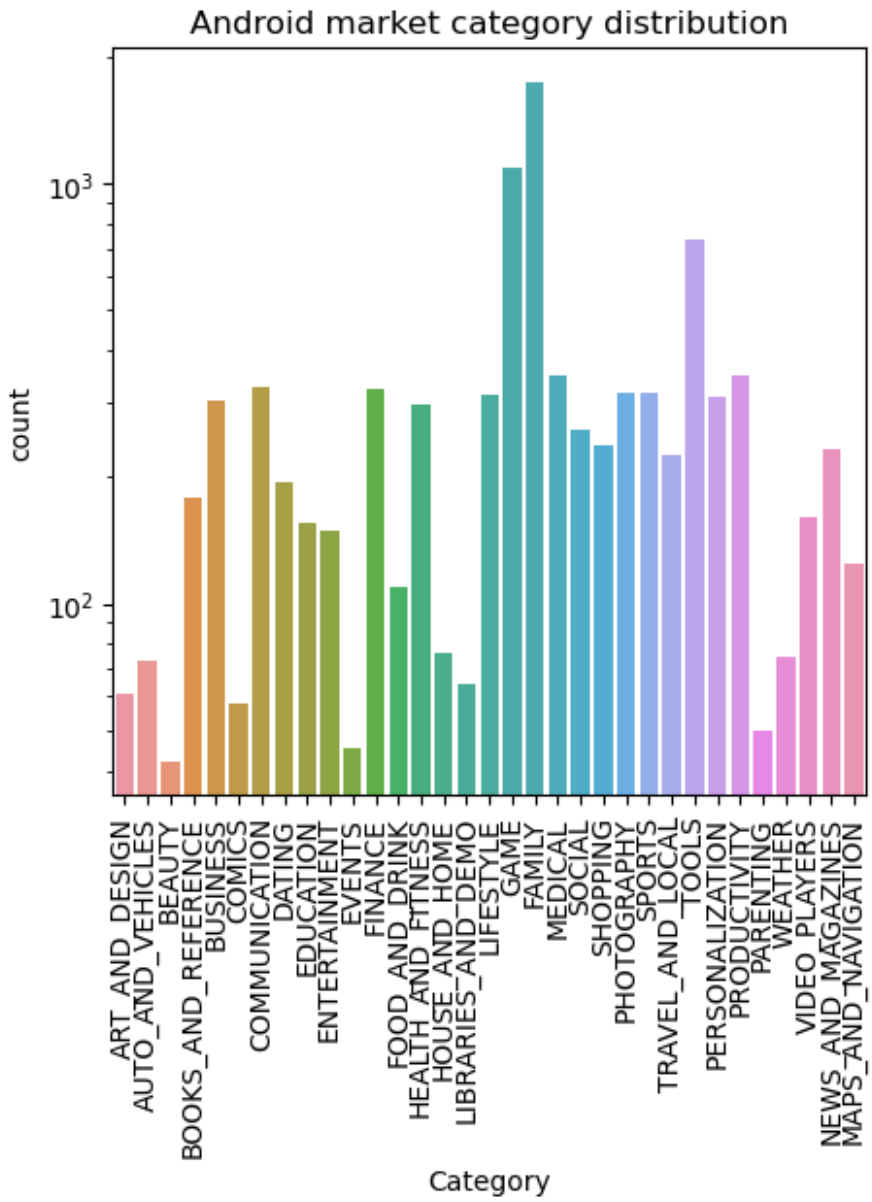
```
plt.figure(figsize=(7,8))
sns.countplot(y='Category',
              data=df,
              order = df['Category'].value_counts().index).set_title(
    label = "Number of applications by category ",
    fontsize = 15);
```



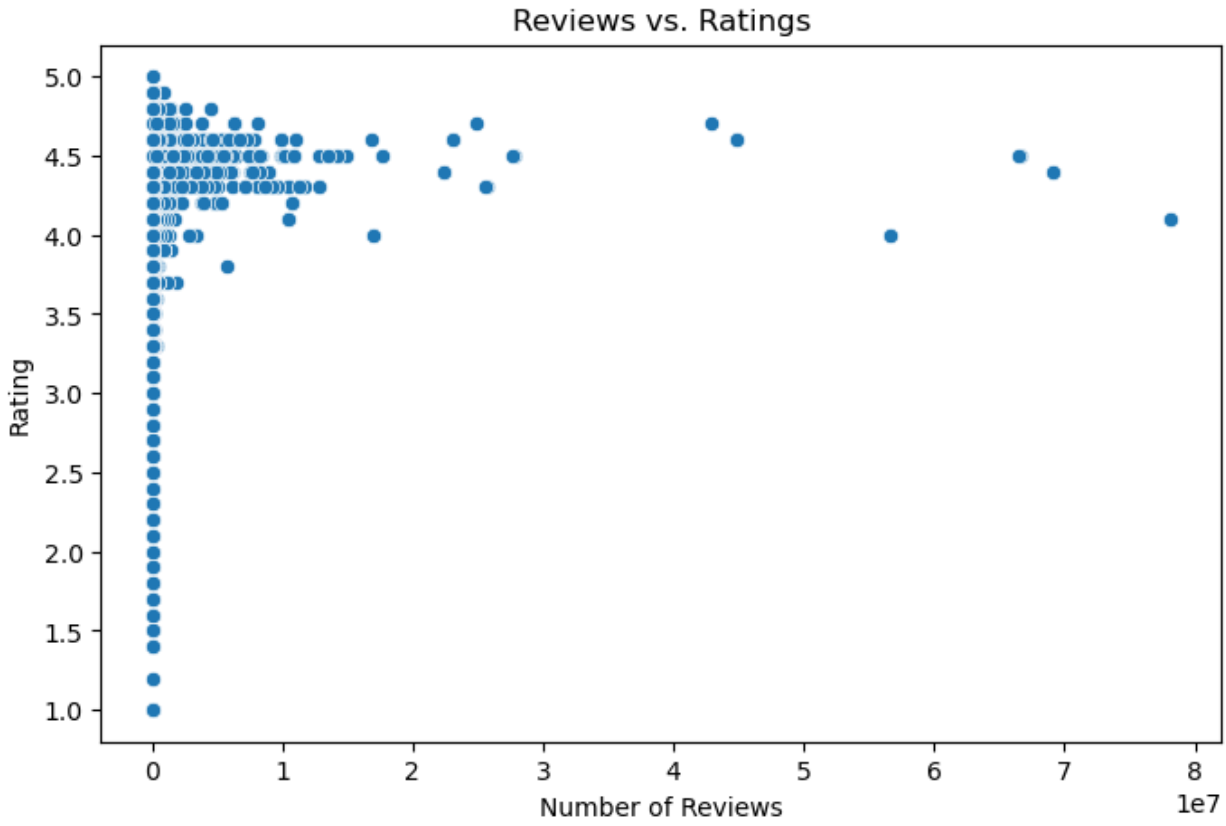

```
plt.figure(figsize=(10, 5))
sns.barplot(x='Category', y='Installs', data=df)
plt.xticks(rotation=90)
plt.title('App Categories vs. Number of Installs')
plt.xlabel('App Category')
plt.ylabel('Number of Installs')
plt.show()
```



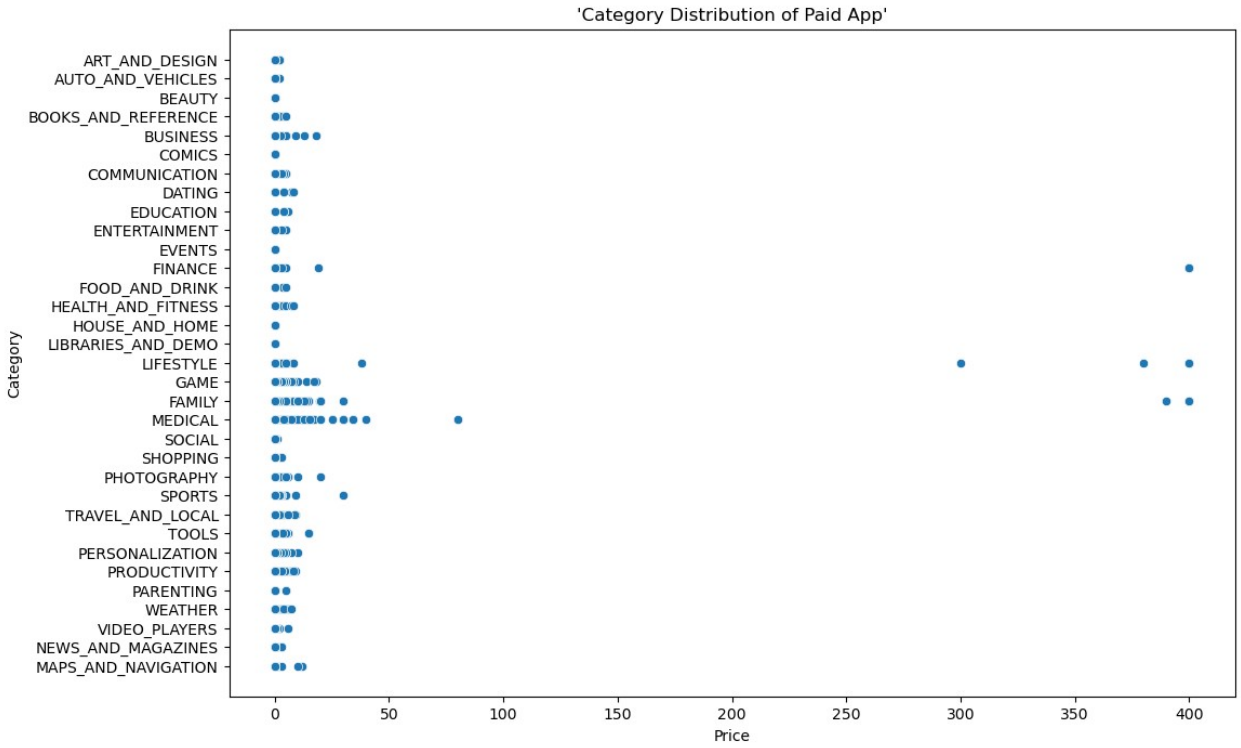

```
plt.figure(figsize=(5, 5))
sns.countplot(x=df.Category , log=True)
plt.xticks(rotation='vertical')
plt.title("Android market category distribution")
plt.show()
```



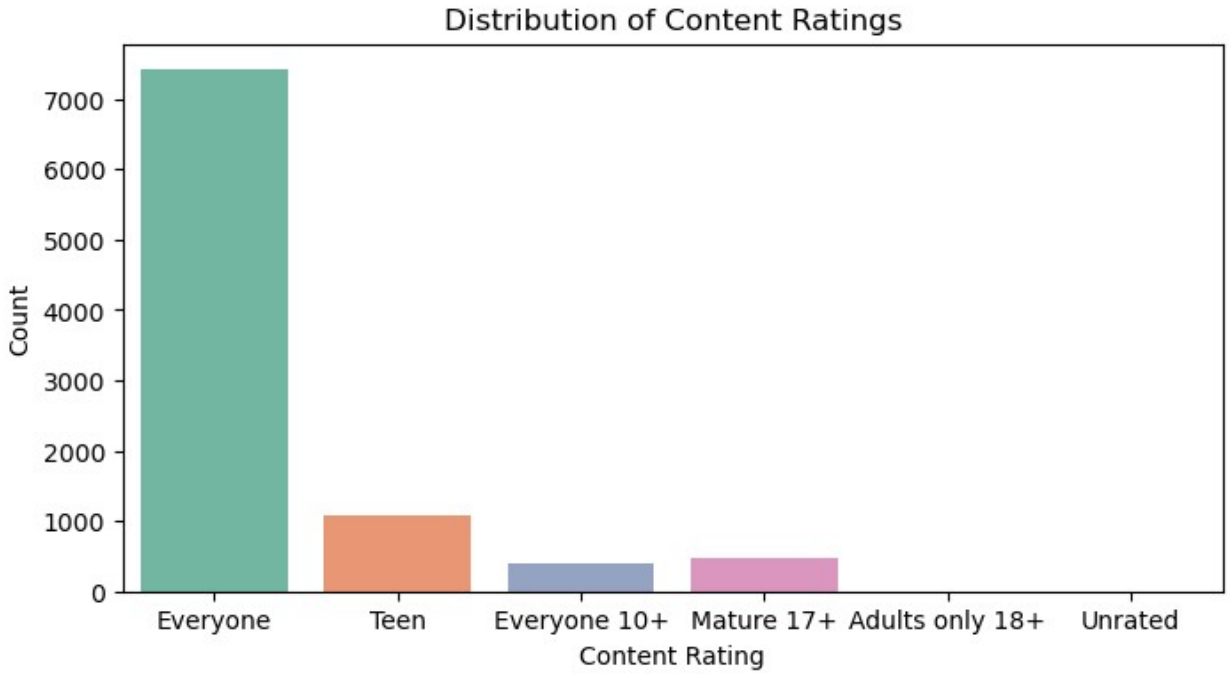

```
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Reviews', y='Rating', data=df)
plt.title('Reviews vs. Ratings')
plt.xlabel('Number of Reviews')
plt.ylabel('Rating')
plt.show()
```



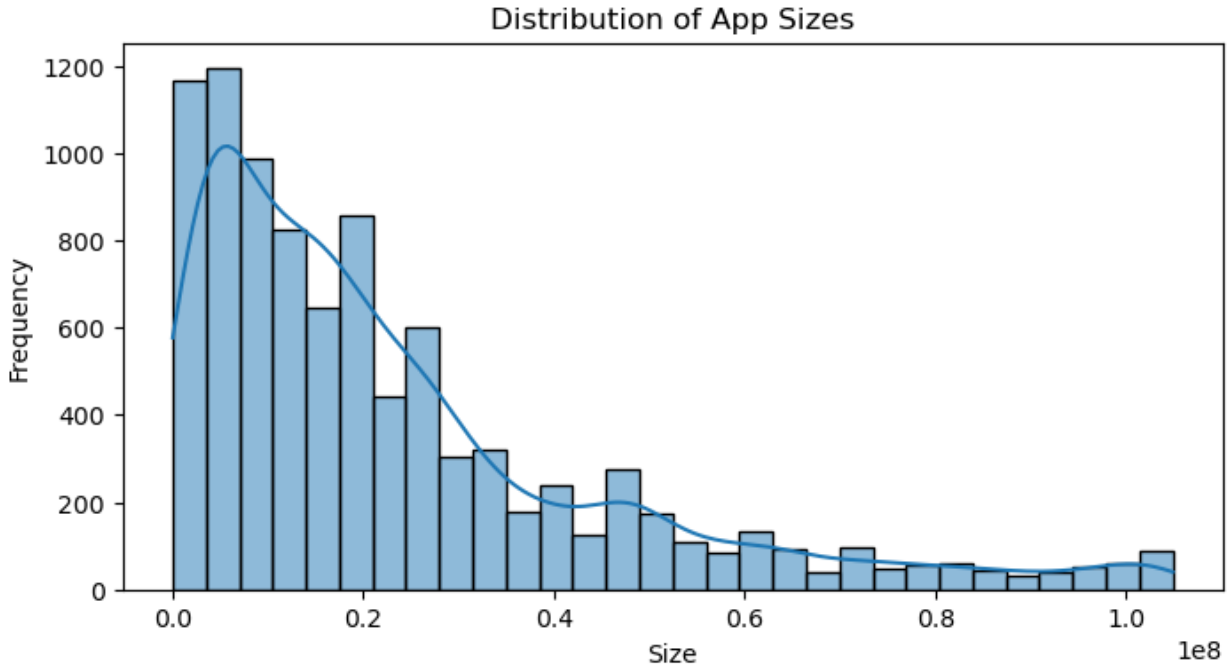

```
plt.figure(figsize=(12,8))
sns.scatterplot(x='Price',
                y='Category',
                data=df).set_title("'Category Distribution of Paid
App'");
```




```
plt.figure(figsize=(8, 4))
sns.countplot(x='Content_Rating', data=df, palette='Set2')
plt.title('Distribution of Content Ratings')
plt.xlabel('Content Rating')
plt.ylabel('Count')
plt.show()
```

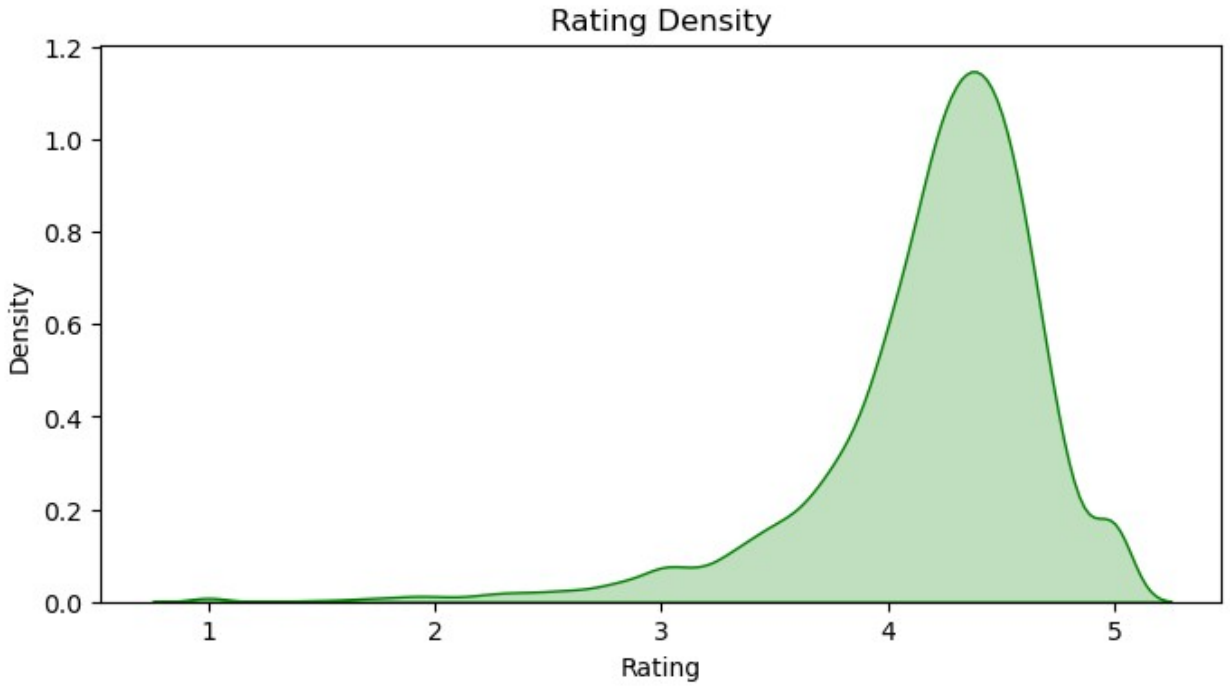



```
plt.figure(figsize=(8, 4))
sns.histplot(df['Size'], bins=30, kde=True)
plt.title('Distribution of App Sizes')
plt.xlabel('Size')
plt.ylabel('Frequency')
plt.show()
```

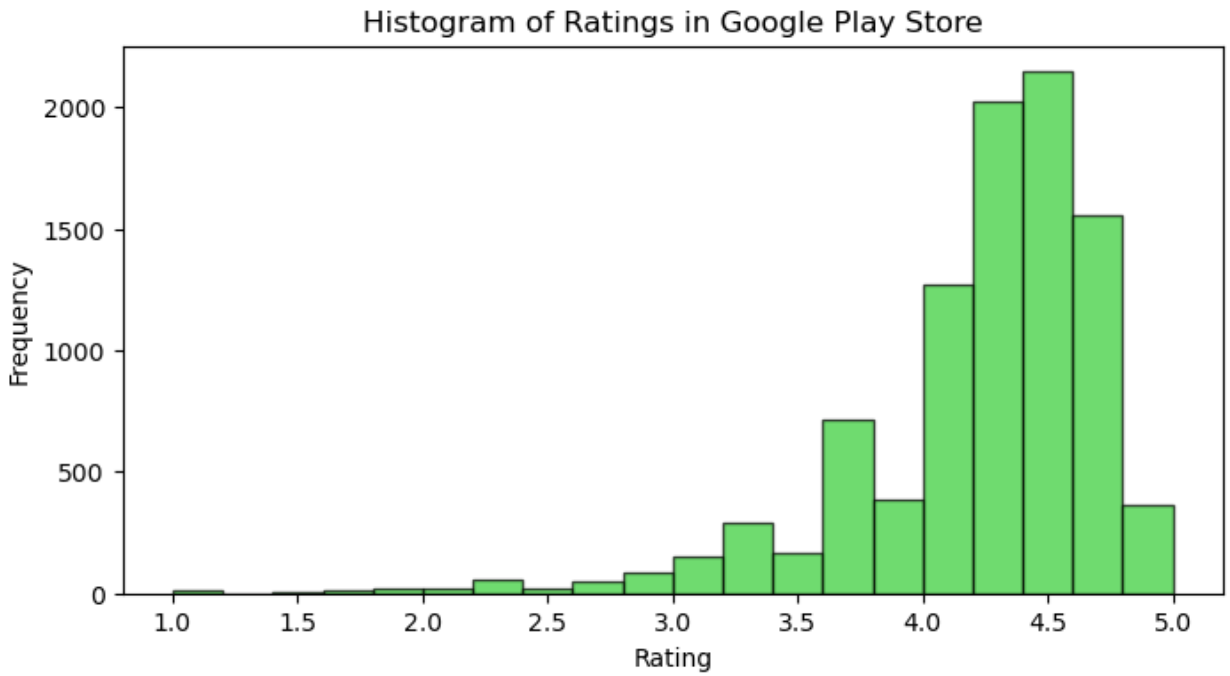


```
plt.figure(figsize=(8, 4))
sns.kdeplot(df['Rating'], color="green", shade = True);
plt.title('Rating Density')
```

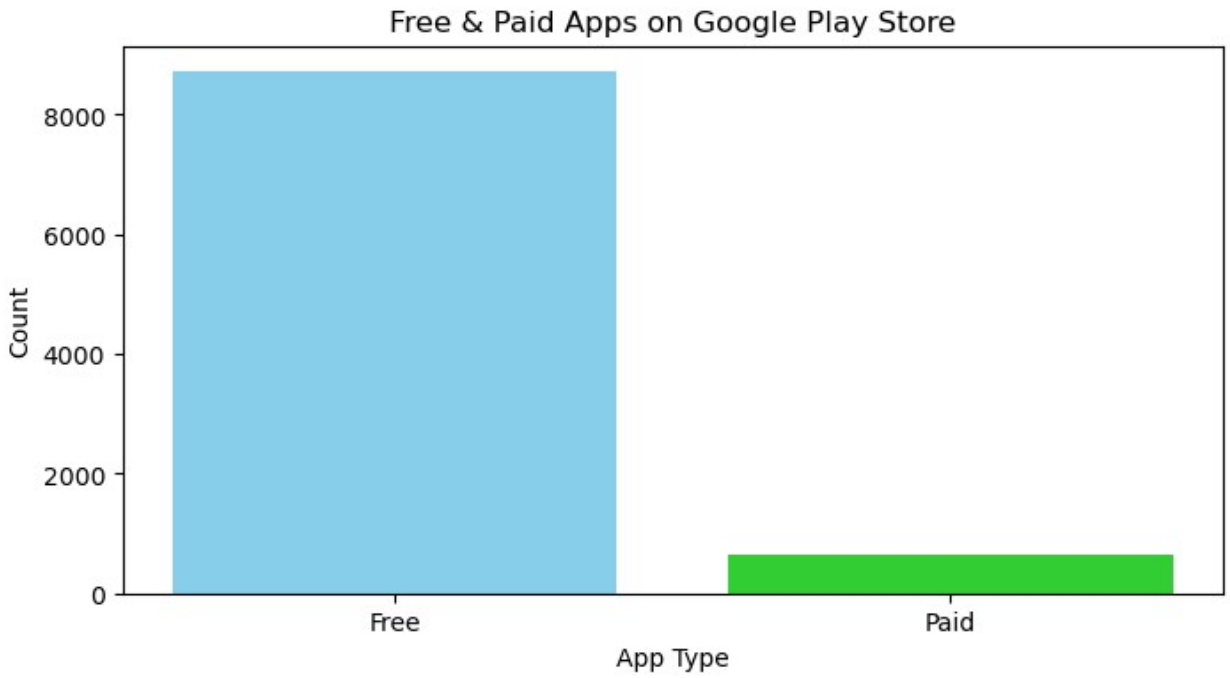
```
Text(0.5, 1.0, 'Rating Density')
```



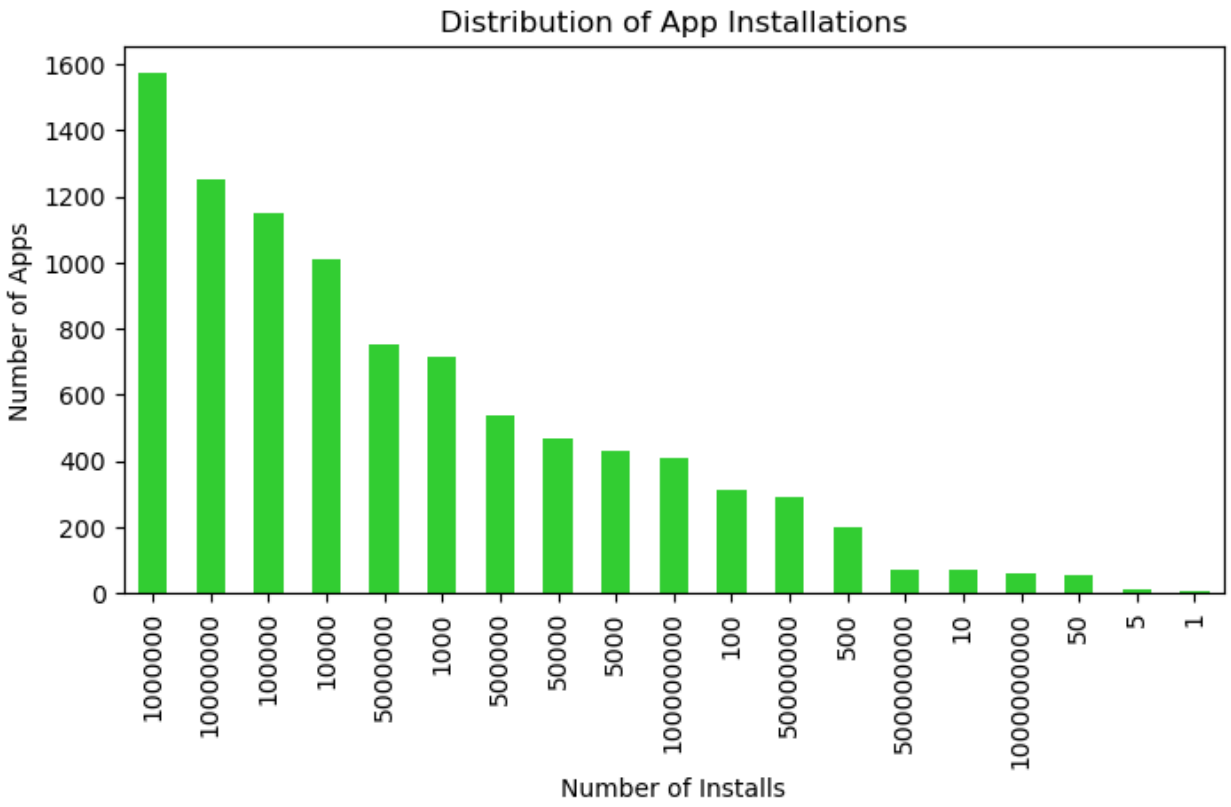

```
plt.figure(figsize=(8, 4))
plt.hist(df['Rating'], bins=20, edgecolor='k', alpha=0.7,
color='limegreen')
plt.title('Histogram of Ratings in Google Play Store')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```




```
type_counts = df["Type"].value_counts()
plt.figure(figsize=(8, 4))
plt.bar(type_counts.index, type_counts.values, color=['skyblue',
'limegreen'])
plt.title("Free & Paid Apps on Google Play Store")
plt.xlabel("App Type")
plt.ylabel("Count")
plt.xticks(type_counts.index, ["Free", "Paid"])
plt.show()
```




```
install_counts = df['Installs'].value_counts()
plt.figure(figsize=(8, 4))
install_counts.plot(kind='bar', color="limegreen")
plt.xlabel('Number of Installs')
plt.ylabel('Number of Apps')
plt.title('Distribution of App Installations')
plt.show()
```




```
numeric_df = df.select_dtypes(include='number')

plt.figure(figsize=(8, 4))
sns.heatmap(numeric_df.corr(), annot=True, linewidths=1.5, fmt='.2f')
plt.title("Correlation", size=15)
plt.show()
```


```
df.columns
```

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs',
'Type',
       'Price', 'Content_Rating', 'Genres', 'Last_Updated',
'Current_Ver',
       'Android_Ver', 'new', 'lastupdate'],
      dtype='object')
```

```
df = df.drop(columns=['App', 'Last_Updated', 'Current_Ver',
'Android_Ver', 'new', 'lastupdate'])
df = pd.get_dummies(df)
X = df.drop(columns=['Rating'])
y = df['Rating']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

model = LinearRegression()
model = model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
from sklearn.metrics import r2_score
```

```
# Calculate R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)
```

```
Mean Squared Error: tf.Tensor(0.24991163114804751, shape=(),
dtype=float64)
R-squared: 0.030489423391992898
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

import numpy as np
from sklearn.preprocessing import StandardScaler

# Assuming X_train is your training data
# Fit the StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

# Save the mean and scale parameters
np.save("scaler_mean.npy", scaler.mean_)
np.save("scaler_scale.npy", scaler.scale_)

# Artifical Neural network

model = Sequential([
    Dense(128, activation='relu',
input_shape=(X_train_scaled.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='sigmoid'),
    Dense(16, activation='tanh'),
    Dense(units=1)
])


# Compile the model with the standard loss function
model.compile(optimizer='adam', loss=mean_squared_error)

history = model.fit(X_train_scaled, y_train, epochs=5,
validation_data=(X_test_scaled, y_test), verbose=2)
```

```
Epoch 1/5
205/205 - 3s - 13ms/step - loss: 1.2007 - val_loss: 0.2524
Epoch 2/5
205/205 - 1s - 4ms/step - loss: 0.2610 - val_loss: 0.2502
Epoch 3/5
```

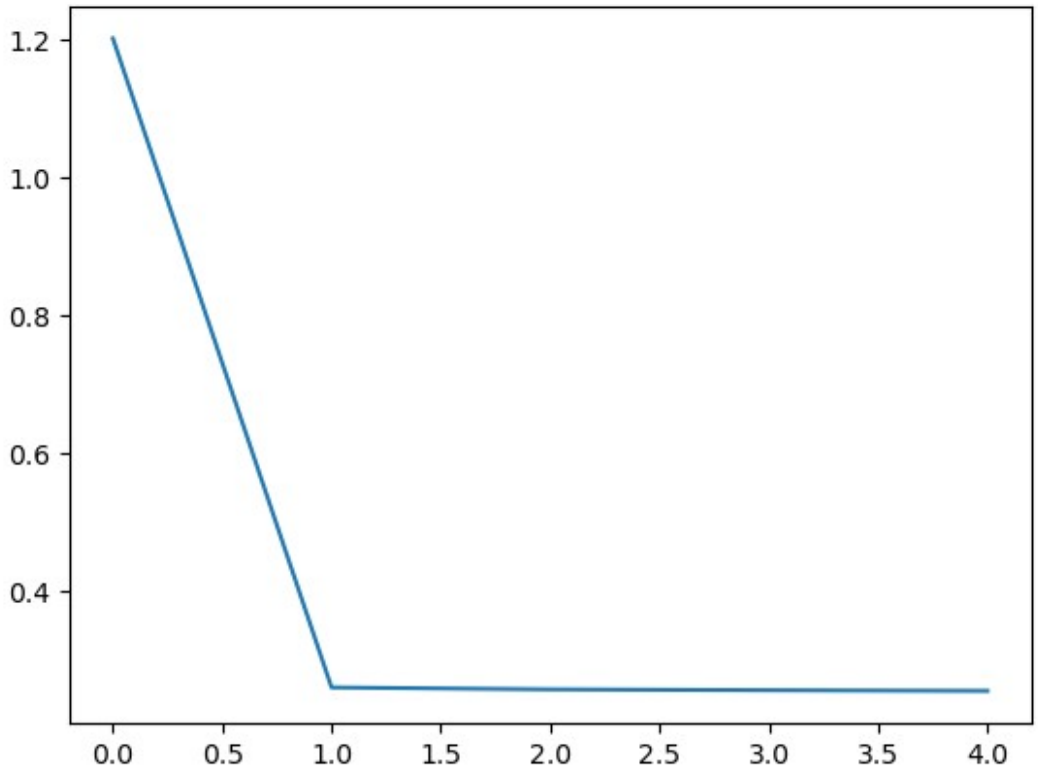
```
205/205 - 1s - 3ms/step - loss: 0.2582 - val_loss: 0.2520
Epoch 4/5
205/205 - 1s - 3ms/step - loss: 0.2570 - val_loss: 0.2506
Epoch 5/5
205/205 - 1s - 3ms/step - loss: 0.2560 - val_loss: 0.2509
```

```
model.metrics_names
```

```
['loss']
```

```
loss = model.history.history['loss']
sns.lineplot(x=range(len(loss)),y=loss)
```

```
<Axes: >
```




```
# Evaluate the model
mse = model.evaluate(X_test_scaled, y_test)
print("Mean Squared Error on test set:", mse)
```

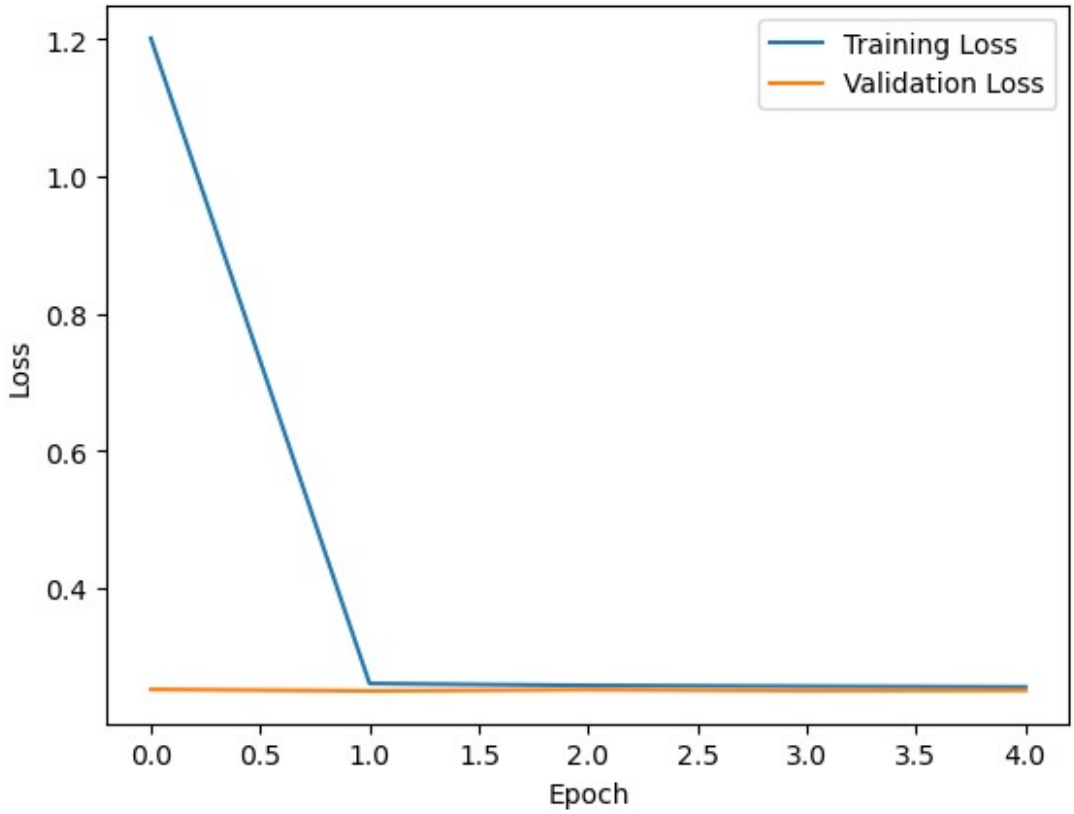
```
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2670
Mean Squared Error on test set: 0.25088563561439514
```

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

```
plt.legend()
plt.show()

# predictions
predictions = model.predict(X_test_scaled)

comparison = pd.DataFrame({'Actual': y_test, 'Predicted':
predictions.flatten()})
comparison['Predicted'] = comparison['Predicted'].round(1)
print(comparison.head(10))
```




```
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
      Actual  Predicted
5560     4.1        4.2
2895     4.1        4.2
7836     4.3        4.2
9116     4.1        4.3
598      4.0        4.1
6465     4.5        4.3
721      3.9        4.4
5902     4.7        4.2
5698     4.1        4.3
7938     4.0        4.2
```

```python
# Predictions
from sklearn.metrics import mean_squared_error
predictions = model.predict(X_test_scaled)

# Evaluate the model
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
rmse = np.sqrt(mean_squared_error(y_test, predictions))

print("Mean Absolute Error:", mae.round(4))
print("R-squared (R2) Score:", r2.round(4))
```

```
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Mean Absolute Error: 0.3417
R-squared (R2) Score: 0.0267
```

```python
accuracy = 1 - (rmse / np.mean(y_test))
accuracy_formatted = "{:.2f}%".format(accuracy)
print("Accuracy:", accuracy_formatted)
```

```
Accuracy: 0.88%
```

```python
# model.save("google_playstore_rating_prediction_model.h5")
```