Data Science

Artificial Inteligence

Machine Learning

Blockchain

Task 2 (Python for Data Science): Investigate a dataset using Python.

You will be assigned a dataset and need to execute Python code to answer three questions. As a data analyst, your task is to help the business to better address the concerns/ issues by providing the proper solution based on the outcome of the Python code. Please make sure that you follow the instructions below:

i. You may execute the Python code in any environment.
ii. Your final submission should be in a report format.
iii. You need to show the steps of Data Wrangling, Data cleaning, analysis, and conclusion.

Data source: https://www.kaggle.com/datasets/bhavikjikadara/google-play-store-applications/code
Data source file name: googleplaystore.csv

Importing necessary libraries.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Loading data and viewing first five rows.

```
[5] df = pd.read_csv('googleplaystore.csv')

# Display first five rows

df.head()
```

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	7-Jan-18	1.0.0	4.0.3 and up
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	15-Jan-18	2.0.0	4.0.3 and up
2	2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	1-Aug-18	1.2.4	4.0.3 and up
3	3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	8-Jun-18	Varies with device	4.2 and up
4	4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	20-Jun-18	1.1	4.4 and up

Displaying last five rows.

```
# display last 5 rows

df.tail()
```

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
10836	10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	Education	25-Jul-17	1.48	4.1 and up
10837	10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	Education	6-Jul-18	1	4.1 and up
10838	10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0	Everyone	Medical	20-Jan-17	1	2.2 and up
10839	10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0	Mature 17+	Books & Reference	19-Jan-15	Varies with device	Varies with device
10840	10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0	Everyone	Lifestyle	25-Jul-18	Varies with device	Varies with device

Viewing general information of the columns.

```
# display information about the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      10841 non-null  int64
 1   App             10841 non-null  object
 2   Category        10841 non-null  object
 3   Rating          9367 non-null   float64
 4   Reviews         10841 non-null  object
 5   Size            10841 non-null  object
 6   Installs        10841 non-null  object
 7   Type            10840 non-null  object
 8   Price           10841 non-null  object
 9   Content Rating  10840 non-null  object
 10  Genres          10841 non-null  object
 11  Last Updated    10841 non-null  object
 12  Current Ver     10833 non-null  object
 13  Android Ver     10838 non-null  object
dtypes: float64(1), int64(1), object(12)
memory usage: 1.2+ MB
```

Finding each column’s data types.

```
# checking data types of each column

df.dtypes
```

```
Unnamed: 0          int64
App                object
Category           object
Rating            float64
Reviews            object
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

Displaying general statistics summary of the data set such as count, min, max mean, standard deviation etc.

```
# display main mathematical charachteristics

df.describe()
```

	Unnamed: 0	Rating
count	10841.000000	9367.000000
mean	5420.000000	4.193338
std	3129.671468	0.537431
min	0.000000	1.000000
25%	2710.000000	4.000000
50%	5420.000000	4.300000
75%	8130.000000	4.500000
max	10840.000000	19.000000

Finding categorical variables.

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :\n\n', categorical)
```

```
There are 12 categorical variables

The categorical variables are :

 ['App', 'Category', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver']
```

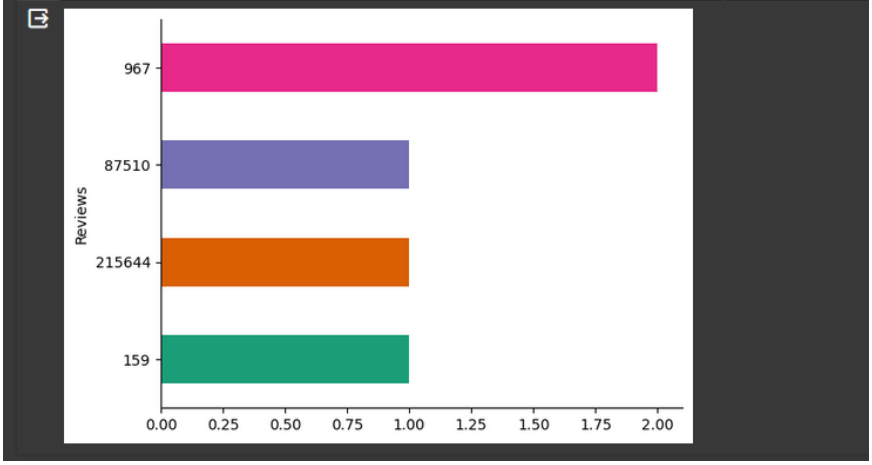
Viewing first five rows only having categorical values / columns.

```
# view the categorical variables

df[categorical].head()
```

	App	Category	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	159	19M	10,000+	Free	0	Everyone	Art & Design	7-Jan-18	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	15-Jan-18	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	1-Aug-18	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	215644	25M	50,000,000+	Free	0	Teen	Art & Design	8-Jun-18	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	20-Jun-18	1.1	4.4 and up

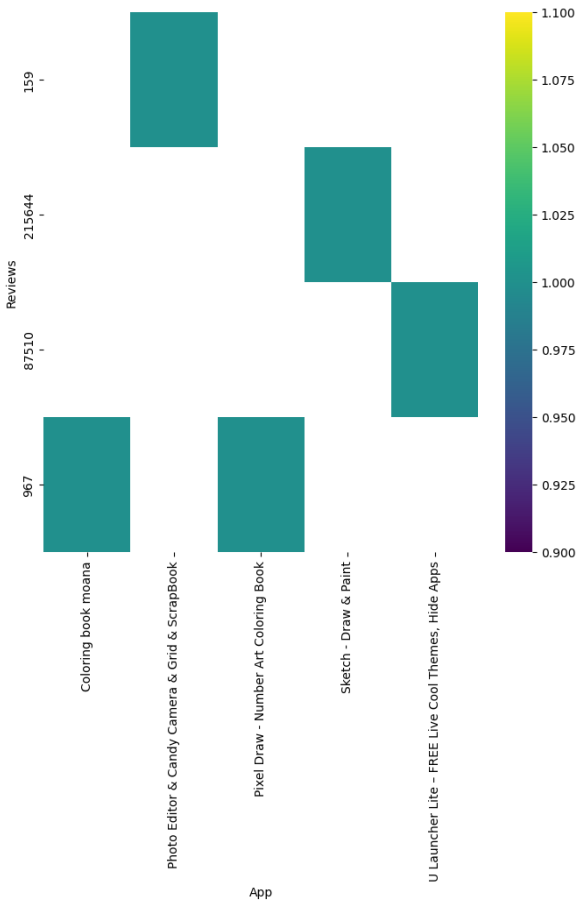
Generating a plot to view the distribution of number of Reviews under 8 levels (0.25, 0.50, …., 2.00)

```
from matplotlib import pyplot as plt
import seaborn as sns

_df_1.groupby('Reviews').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Plotting the first five rows (Apps') distribution against Reviews.

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['Reviews'].value_counts()
    for x_label, grp in _df_4.groupby('App')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('App')
_ = plt.ylabel('Reviews')
```



Finding total number of null values in categorical variables.

```
# view null values total in categorical vars

df[categorical].isnull().sum()
```

```
App              0
Category         0
Reviews          0
Size             0
Installs         0
Type             1
Price            0
Content Rating   1
Genres           0
Last Updated     0
Current Ver      8
Android Ver      3
dtype: int64
```

Viewing the frequency distribution of categorical variables.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization purposes
import seaborn as sns # for statistical data visualization
%matplotlib inline


# view frequency distribution of categorical variables

for var in categorical:

    print(df[var].value_counts()/float(len(df)))
```

```
ROBLOX                                                0.000830
CBS Sports App - Scores, News, Stats & Watch Live     0.000738
ESPN                                                  0.000646
Duolingo: Learn Languages Free                        0.000646
Candy Crush Saga                                      0.000646
                                                        ...
Meet U - Get Friends for Snapchat, Kik & Instagram    0.000092
U-Report                                              0.000092
U of I Community Credit Union                         0.000092
Waiting For U Launcher Theme                          0.000092
iHoroscope - 2018 Daily Horoscope & Astrology         0.000092
Name: App, Length: 9660, dtype: float64
FAMILY                 0.181902
GAME                   0.105525
TOOLS                  0.077760
MEDICAL                0.042708
BUSINESS               0.042432
PRODUCTIVITY           0.039111
PERSONALIZATION        0.036159
COMMUNICATION          0.035698
SPORTS                 0.035421
LIFESTYLE              0.035237
FINANCE                0.033761
HEALTH_AND_FITNESS     0.031455
PHOTOGRAPHY            0.030901
SOCIAL                 0.027212
NEWS_AND_MAGAZINES     0.026105
SHOPPING               0.023983
TRAVEL_AND_LOCAL       0.023799
DATING                 0.021585
BOOKS_AND_REFERENCE    0.021308
VIDEO_PLAYERS          0.016142
EDUCATION              0.014390
ENTERTAINMENT          0.013744
MAPS_AND_NAVIGATION    0.012637
FOOD_AND_DRINK         0.011715
HOUSE_AND_HOME         0.008117
LIBRARIES_AND_DEMO     0.007841
AUTO_AND_VEHICLES      0.007841
WEATHER                0.007564
ART_AND_DESIGN         0.005996
EVENTS                 0.005904
PARENTING              0.005535
COMICS                 0.005535
BEAUTY                 0.004889
```

Check unique values (classes) of categorical variables (Type, Content Rating, Current Ver and Android Ver)

```
[20] # check labels in Type variable

     df['Type'].unique()

     array(['Free', 'Paid', nan, '0'], dtype=object)

[21] # check labels in Content Rating variable

     df['Content Rating'].unique()

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

[22] # check labels in Current Ver variable

     df['Current Ver'].unique()

     array(['1.0.0', '2.0.0', '1.2.4', ..., '1.0.612928', '0.3.4', '2.0.148.0'],
           dtype=object)

     # check labels in Android Ver variable

     df['Android Ver'].unique()

     array(['4.0.3 and up', '4.2 and up', '4.4 and up', '2.3 and up',
            '3.0 and up', '4.1 and up', '4.0 and up', '2.3.3 and up',
            'Varies with device', '2.2 and up', '5.0 and up', '6.0 and up',
            '1.6 and up', '1.5 and up', '2.1 and up', '7.0 and up',
            '5.1 and up', '4.3 and up', '4.0.3 - 7.1.1', '2.0 and up',
            '3.2 and up', '4.4W and up', '7.1 and up', '7.0 - 7.1.1',
            '8.0 and up', '5.0 - 8.0', '3.1 and up', '2.0.1 and up',
            '4.1 - 7.1.1', nan, '5.0 - 6.0', '1.0 and up', '2.2 - 7.1.1',
            '5.0 - 7.1.1'], dtype=object)
```

Displaying all columns (Rating is the target column)

```
# displaying all columns / features & target var

df.columns

Index(['Unnamed: 0', 'App', 'Category', 'Rating', 'Reviews', 'Size',
       'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated',
       'Current Ver', 'Android Ver'],
      dtype='object')
```

Viewing diagnosis.

```
# each class how many records (Diagnosis)

df.value_counts()
```

```
Unnamed: 0  App                                        Category        Rating  Reviews  Size              Installs      Type  Price  Content Rating  Genres        Last Updated  Current Ver         Android Ver
0           Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN  4.1     159      19M               10,000+       Free  0      Everyone        Art & Design  7-Jan-18      1.0.0               4.0.3 and up          1
6860        Bacterial Vaginosis Symptoms               MEDICAL         4.1     698      2.4M              100,000+      Free  0      Everyone 10+    Medical       3-Feb-15      1.1                 3.0 and up            1
6825        BU Students' Rep. Council                  FAMILY          4.7     38       5.3M              500+          Free  0      Everyone        Education     5-Jul-18                          5.45.0_713          4.0 and up            1
6827        BU Calculator                              FAMILY          4.6     69       3.1M              1,000+        Free  0      Everyone        Education     24-Jul-17     2.4                 4.0.3 and up          1
6830        BU Syllabus                                FAMILY          4.5     38       6.5M              1,000+        Free  0      Everyone        Education     21-Sep-17     1.0.3               4.0.3 and up          1

..
3263        Phone                                      TOOLS           3.6     45483    Varies with device  10,000,000+  Free  0      Everyone        Tools         30-Jul-18     Varies with device  Varies with device    1
3264        HTC Lock Screen                            TOOLS           4.1     28250    Varies with device  10,000,000+  Free  0      Everyone        Tools         21-Mar-17     Varies with device  Varies with device    1
3265        Gboard - the Google Keyboard               TOOLS           4.2     1859115  Varies with device  500,000,000+  Free  0      Everyone        Tools         31-Jul-18     Varies with device  Varies with device    1
3266        Google Korean Input                        TOOLS           3.5     74819    Varies with device  100,000,000+  Free  0      Everyone        Tools         25-Jun-18     Varies with device  7.1 and up            1
10840       iHoroscope - 2018 Daily Horoscope & Astrology  LIFESTYLE   4.5     398307   19M               10,000,000+   Free  0      Everyone        Lifestyle     25-Jul-18     Varies with device  Varies with device    1
Length: 9360, dtype: int64
```

Removing unwanted columns and displaying the remaining columns.

```
[80] # drop unwanted columns

     df. drop('Unnamed: 0', axis=1, inplace=True)
```

```
[81] # drop unwanted columns

     df. drop('Size', axis=1, inplace=True)
```

```
# displaying all columns once again to check existing columns after dropping unnecessary columns

df.columns
```

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

Cleaning the Installs column.

Finding invalid rows and removing them from the Install column and converting the remaining rows to numeric.

```
# Finding rows where 'Installs' cannot be converted to numeric
invalid_installs = df['Installs'].str.contains('^[a-zA-Z]', regex=True)
invalid_installs_rows = df[invalid_installs]

# Dropping these rows
df_cleaned = df[~invalid_installs]

# Try to convert once again
df_cleaned['Installs'] = df_cleaned['Installs'].str.replace('+', '').str.replace(',', '').astype(float)

# Checking the data types
df_cleaned.dtypes
```

```
<ipython-input-33-9e7227ba4a88>:9: FutureWarning: The default value of regex will change from True to False in a future version. In addition,
  df_cleaned['Installs'] = df_cleaned['Installs'].str.replace('+', '').str.replace(',', '').astype(float)
<ipython-input-33-9e7227ba4a88>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Installs'] = df_cleaned['Installs'].str.replace('+', '').str.replace(',', '').astype(float)
App               object
Category          object
Rating           float64
Reviews           object
Installs         float64
Type              object
Price             object
Content Rating    object
Genres            object
dtype: object
```

Viewing the cleaned Installs column data set.

```
# cleaned data

df_cleaned.head()
```

	App	Category	Rating	Reviews	Installs	Type	Price	Content Rating	Genres
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10000.0	Free	0	Everyone	Art & Design
1	Coloring book moana	ART_AND_DESIGN	3.9	967	500000.0	Free	0	Everyone	Art & Design;Pretend Play
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	5000000.0	Free	0	Everyone	Art & Design
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	50000000.0	Free	0	Teen	Art & Design
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	100000.0	Free	0	Everyone	Art & Design;Creativity

Displaying the invalid row from the Installs column.

```
# invalid data rows from the installs column

invalid_installs_rows
```

	App	Category	Rating	Reviews	Installs	Type	Price	Content Rating	Genres
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	Free	0	Everyone	NaN	11-Feb-18

Removing the $ sign and converting the Price column into float data type.

```
[90] # Price column has $ sign hence before convert it to float, needs to remove the $ sign and then convert to float type

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

     <ipython-input-90-47dc16c00bd1>:3: FutureWarning: The default value of regex will change from True to False in a future ve
       df_cleaned['Price_float'] = df_cleaned['Price'].str.replace('$', '').astype(float)
     <ipython-input-90-47dc16c00bd1>:3: SettingWithCopyWarning:
     A value is trying to be set on a copy of a slice from a DataFrame.
     Try using .loc[row_indexer,col_indexer] = value instead

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
       df_cleaned['Price_float'] = df_cleaned['Price'].str.replace('$', '').astype(float)
```

Converting the Last Updated column into datetime format.

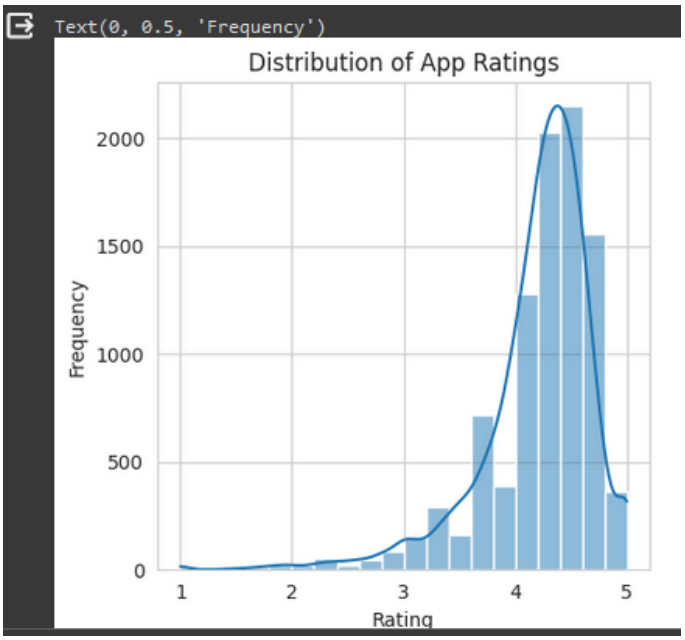
```
# Convert 'Last Updated' to datetime format
df_cleaned['Last Updated_dt'] = pd.to_datetime(df_cleaned['Last Updated'], errors='coerce', format='%d-%b-%y')

<ipython-input-91-6d6551ef30e5>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
  df_cleaned['Last Updated_dt'] = pd.to_datetime(df_cleaned['Last Updated'], errors='coerce', format='%d-%b-%y')
```

Transforming Installs column values into log values, adding it as a new column to the data set, and checking any null values of the new column.

```
[98] # apply the log transformation with NumPy

     df_cleaned['Installs_log'] = df_cleaned['Installs'].apply(lambda x: np.log(x + 1))

     <ipython-input-98-31423d80ee93>:3: SettingWithCopyWarning:
     A value is trying to be set on a copy of a slice from a DataFrame.
     Try using .loc[row_indexer,col_indexer] = value instead

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
       df_cleaned['Installs_log'] = df_cleaned['Installs'].apply(lambda x: np.log(x + 1))

     # checking any null values

     df_cleaned['Installs_log'].isnull().sum()

     0
```

Plotting Rating column's data distribution.

```
# set the aesthetic style of the plots

sns.set_style("whitegrid")

fig, ax = plt.subplots(2, 2, figsize=(10, 10), dpi=100)

# histogram of App Ratings
sns.histplot(df_cleaned['Rating'].dropna(), bins=20, kde=True, ax=ax[0, 0])  # Drop NA values for clean plotting
ax[0, 0].set_title('Distribution of App Ratings')
ax[0, 0].set_xlabel('Rating')
ax[0, 0].set_ylabel('Frequency')
```

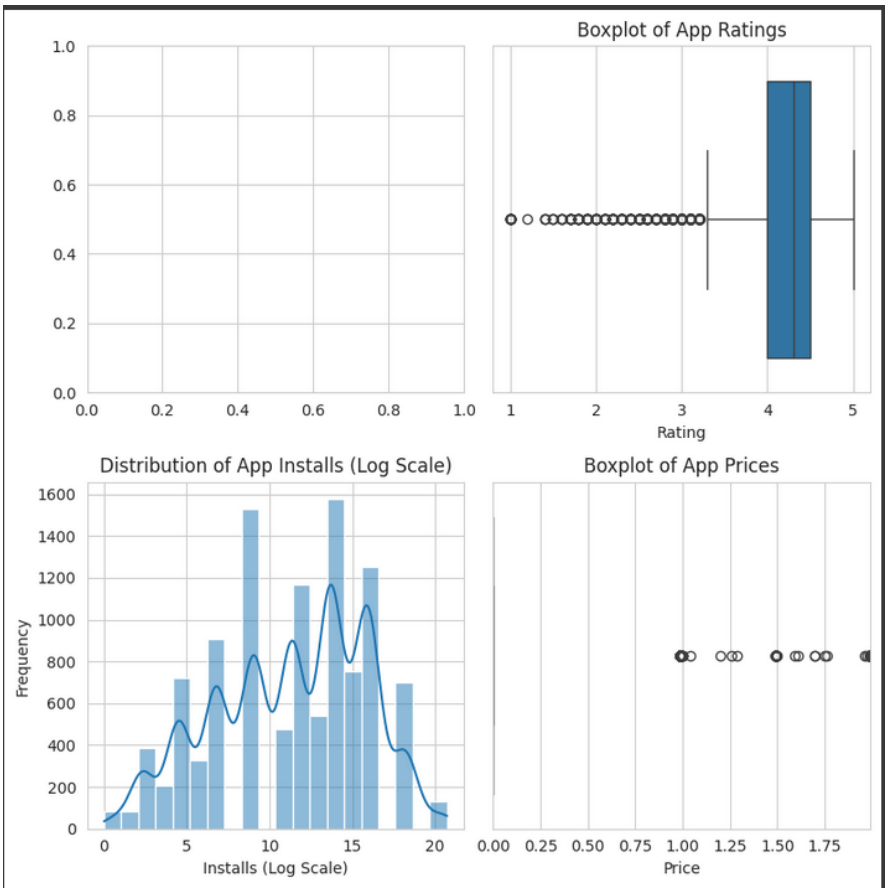


As per the above plot, it is left skewed.

Checking availability of outliers in the Rating and Price_float columns,

Viewing the frequency distribution of Installs_log column.

```
# boxplot of app Ratings

fig, ax = plt.subplots(2, 2, figsize=(8, 8), dpi=100)

sns.boxplot(x='Rating', data=df_cleaned, ax=ax[0, 1])
ax[0, 1].set_title('Boxplot of App Ratings')
ax[0, 1].set_xlabel('Rating')


# histogram of App Installs (Using Log-Transformed Data)

sns.histplot(df_cleaned['Installs_log'], bins=20, kde=True, ax=ax[1, 0])
ax[1, 0].set_title('Distribution of App Installs (Log Scale)')
ax[1, 0].set_xlabel('Installs (Log Scale)')
ax[1, 0].set_ylabel('Frequency')

# Boxplot of App Prices (Adjusting for visualization)

sns.boxplot(x='Price_float', data=df_cleaned, ax=ax[1, 1])
ax[1, 1].set_title('Boxplot of App Prices')
ax[1, 1].set_xlabel('Price')
ax[1, 1].set_xlim(0, df_cleaned['Price_float'].quantile(0.95))  # Limiting x-axis to 95th percentile for better visualization

plt.tight_layout()
plt.show()
```



As can be observed, there are significantly more outliers in the Price_float data than in the Rating variable. Additionally, the data in the Installs_log field exhibits a modest normal distribution.

Finding unique values and their absolute counts in the Type column.

```
[120] # finding unique values in Type column

      df_cleaned['Type'].unique()

      array(['Free', 'Paid', nan], dtype=object)

[121] # above same result can be generated using groupby function
      # find number of instances (rows) that belong to 'Type' feature/variable (absolute count)

      df_cleaned.groupby('Type').size()

      Type
      Free    10039
      Paid      800
      dtype: int64
```

Utilizing map function to transform Type column's values into numeric values.

```
[122] # using map function to encode categorical variable values into numeric values

      # Define a mapping dictionary
      mapping_dict = {'Free': 0, 'Paid': 1}

      # Mapping the values in the 'Type' column using the dictionary
      df_cleaned['Type_Encoded'] = df_cleaned['Type'].map(mapping_dict)

      <ipython-input-122-b8930baf54df>:7: SettingWithCopyWarning:
      A value is trying to be set on a copy of a slice from a DataFrame.
      Try using .loc[row_indexer,col_indexer] = value instead

      See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
        df_cleaned['Type_Encoded'] = df_cleaned['Type'].map(mapping_dict)
```

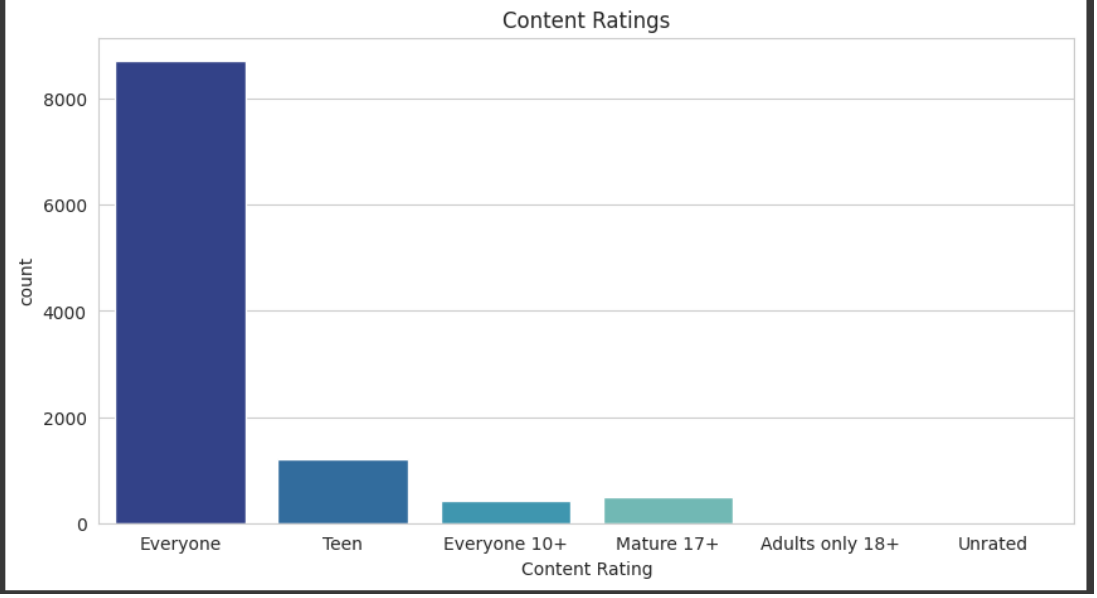
Finding Content Rating column's unique values and finding their total counts.

```
[123] # finding unique values in Content Rating column

      df_cleaned['Content Rating'].unique()

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

[124] # count 'Content Rating' column's classes

      df_cleaned['Content Rating'].value_counts()

      Everyone           8714
      Teen               1208
      Mature 17+          499
      Everyone 10+        414
      Adults only 18+       3
      Unrated               2
      Name: Content Rating, dtype: int64
```

Visualizing the distribution of Content Rating variable.

```
# visualizing the distribution of Content Rating

plt.figure(figsize=(10,5))
sns.countplot(x=df_cleaned["Content Rating"], palette="YlGnBu_r")
plt.title("Content Ratings")
plt.show()
```

```
<ipython-input-275-0e99f3e2a580>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` varia

  sns.countplot(x=df_cleaned["Content Rating"], palette="YlGnBu_r")
```



From the above plot, it can observed that in the Google Play Store, there are high number of android applications for all kinds of users and secondly the highest number of applications available for teenagers.

Encoding the above column's strings to numeric values using the map function.

```
[125] # using map function to encode categorical variable values into numeric values

      # Define a mapping dictionary
      mapping_dict = {'Everyone': 1, 'Teen':2, 'Everyone 10+' :3, 'Mature 17+' : 4, 'Adults only 18+' : 5, 'Unrated' : 0}

      # Mapping the values in the 'Type' column using the dictionary
      df_cleaned['Content Rating_Encoded'] = df_cleaned['Content Rating'].map(mapping_dict)
```

```
<ipython-input-125-e08aa123993c>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Content Rating_Encoded'] = df_cleaned['Content Rating'].map(mapping_dict)
```

Viewing Genres column's unique values.

```
# finding unique values in Genres column

df_cleaned['Genres'].unique()
```

```
array(['Art & Design', 'Art & Design;Pretend Play',
       'Art & Design;Creativity', 'Art & Design;Action & Adventure',
       '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',
```

Converting Genres column's categorical classes into numeric values using the LabelEncoder. (the number of classes are many)

```
[127] # count 'Genres' column's classes

      df_cleaned['Genres'].value_counts()
```

```
Tools                                842
Entertainment                        623
Education                            549
Medical                              463
Business                             460
                                    ...
Parenting;Brain Games                  1
Travel & Local;Action & Adventure      1
Lifestyle;Pretend Play                 1
Tools;Education                        1
Strategy;Creativity                    1
Name: Genres, Length: 119, dtype: int64
```

```
[128] # using sklearn to encode categorical variable values into numeric values

      from sklearn.preprocessing import LabelEncoder

      # Creating a LabelEncoder object
      labelEncoderObj = LabelEncoder()

      # Genres column has many uniques values / classes hence auto encoder is the best method instead of map

      # Fit and transforming the categorical variable to numeric values
      df_cleaned['Genres_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Genres'])
```

```
<ipython-input-128-c81eed01f7fc>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Genres_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Genres'])
```

Displaying unique values of the Category column and finding their total counts.

```
[129] # finding unique values in the Category column

      df_cleaned['Category'].unique()

      array(['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'],
            dtype=object)

[130] # count Category column classes

      df_cleaned['Category'].value_counts()

      FAMILY                1972
      GAME                  1144
      TOOLS                  843
      MEDICAL                463
      BUSINESS               460
      PRODUCTIVITY           424
      PERSONALIZATION        392
      COMMUNICATION          387
      SPORTS                 384
      LIFESTYLE              382
      FINANCE                366
      HEALTH_AND_FITNESS     341
      PHOTOGRAPHY            335
      SOCIAL                 295
      NEWS_AND_MAGAZINES     283
      SHOPPING               260
      TRAVEL_AND_LOCAL       258
      DATING                 234
      BOOKS_AND_REFERENCE    231
      VIDEO_PLAYERS          175
      EDUCATION              156
      ENTERTAINMENT          149
```

Encoding the Category column and adding them to a new column.

```
[131] # Encoding Category column values to numeric values

      # Fit and transforming the categorical variable to numeric values
      df_cleaned['Category_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Category'])

      <ipython-input-131-f0bc7ffe41bf>:4: SettingWithCopyWarning:
      A value is trying to be set on a copy of a slice from a DataFrame.
      Try using .loc[row_indexer,col_indexer] = value instead

      See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
        df_cleaned['Category_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Category'])
```

Counting App column’s classes and viewing their names.

```
[132] # count App column classes

      df_cleaned['App'].value_counts()
```

```
ROBLOX                                               9
CBS Sports App - Scores, News, Stats & Watch Live    8
ESPN                                                 7
Duolingo: Learn Languages Free                       7
Candy Crush Saga                                     7
                                                    ..
Meet U - Get Friends for Snapchat, Kik & Instagram   1
U-Report                                             1
U of I Community Credit Union                        1
Waiting For U Launcher Theme                         1
iHoroscope - 2018 Daily Horoscope & Astrology        1
Name: App, Length: 9659, dtype: int64
```

```
[133] # labels in 'App' variable

      df_cleaned.App.unique()
```

```
array(['Photo Editor & Candy Camera & Grid & ScrapBook',
       'Coloring book moana',
       'U Launcher Lite – FREE Live Cool Themes, Hide Apps', ...,
       'Parkinson Exercices FR', 'The SCP Foundation DB fr nn5n',
       'iHoroscope - 2018 Daily Horoscope & Astrology'], dtype=object)
```

Encoding App variable’s classes.

```
# Encoding App column values to numeric values

# Fit and transforming the categorical variable to numeric values
df_cleaned['App_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['App'])
```

```
<ipython-input-134-3cf6d0ea62d7>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['App_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['App'])
```

Encoding the Current Ver column.

```
[142] # labels in 'Current Ver' variable

      df_cleaned['Current Ver'].unique()
```

```
array(['1.0.0', '2.0.0', '1.2.4', ..., '1.0.612928', '0.3.4', '2.0.148.0'],
      dtype=object)
```

```
# Encoding Current Ver column values to numeric values

# Fit and transforming the categorical variable to numeric values
df_cleaned['Current Ver_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Current Ver'])
```

```
<ipython-input-144-e6219b04668e>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Current Ver_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Current Ver'])
```

Encoding the Android Ver column.

```
[145] # labels in 'Android Ver' variable

      df_cleaned['Android Ver'].unique()
```

```
array(['4.0.3 and up', '4.2 and up', '4.4 and up', '2.3 and up',
       '3.0 and up', '4.1 and up', '4.0 and up', '2.3.3 and up',
       'Varies with device', '2.2 and up', '5.0 and up', '6.0 and up',
       '1.6 and up', '1.5 and up', '2.1 and up', '7.0 and up',
       '5.1 and up', '4.3 and up', '4.0.3 - 7.1.1', '2.0 and up',
       '3.2 and up', '4.4W and up', '7.1 and up', '7.0 - 7.1.1',
       '8.0 and up', '5.0 - 8.0', '3.1 and up', '2.0.1 and up',
       '4.1 - 7.1.1', nan, '5.0 - 6.0', '1.0 and up', '2.2 - 7.1.1',
       '5.0 - 7.1.1'], dtype=object)
```

```
# Encoding Android Ver column values to numeric values

# Fit and transforming the categorical variable to numeric values
df_cleaned['Android Ver_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Android Ver'])
```

```
<ipython-input-146-2f49001c9b18>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Android Ver_Encoded'] = labelEncoderObj.fit_transform(df_cleaned['Android Ver'])
```

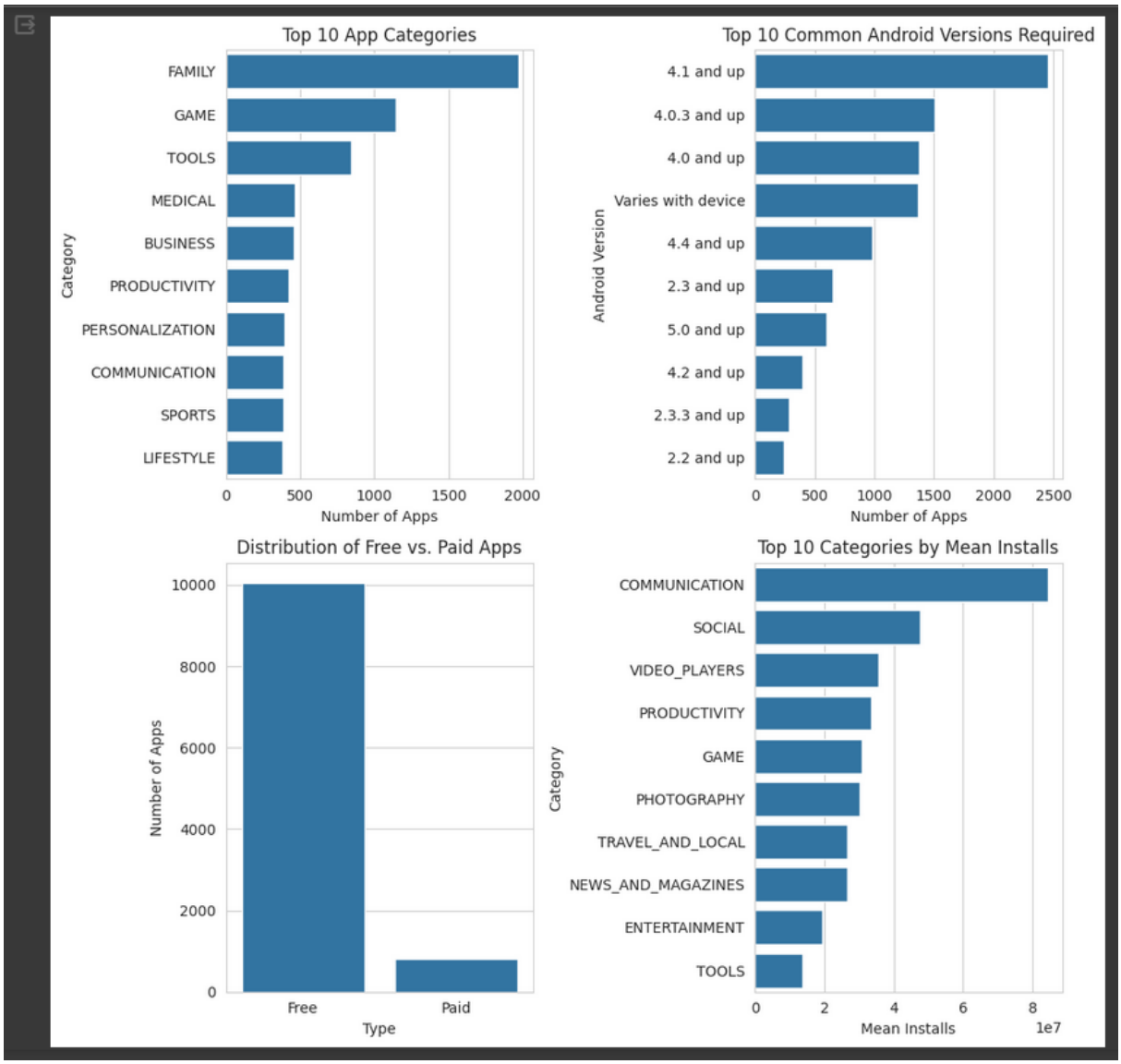
Converting the Reviews column values into float.

```
# adding a new column by converting the Reviews column to a float type

# df_cleaned['Reviews_Log'] = df_cleaned['Reviews'].astype(float).apply(np.log) # THIS LOG VALUES THROW AN ERROR AT THE TIME OF FIT THE MODEL

df_cleaned['Reviews_float'] = df_cleaned['Reviews'].astype(float)
```

```
<ipython-input-110-82fca8961f24>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Reviews_float'] = df_cleaned['Reviews'].astype(float)
```

Analysing original data.

```
# visualization setup
fig, ax = plt.subplots(2, 2, figsize=(10, 10), dpi=100)


# best categories
top_categories = df_cleaned['Category'].value_counts().head(10)
sns.barplot(x=top_categories.values, y=top_categories.index, ax=ax[0, 0])
ax[0, 0].set_title('Top 10 App Categories')
ax[0, 0].set_xlabel('Number of Apps')
ax[0, 0].set_ylabel('Category')


# common android versions
common_android_versions = df_cleaned['Android Ver'].value_counts().head(10)
sns.barplot(x=common_android_versions.values, y=common_android_versions.index, ax=ax[0, 1])
ax[0, 1].set_title('Top 10 Common Android Versions Required')
ax[0, 1].set_xlabel('Number of Apps')
ax[0, 1].set_ylabel('Android Version')


# free vs. paid apps distribution
free_vs_paid = df_cleaned['Type'].value_counts()
sns.barplot(x=free_vs_paid.index, y=free_vs_paid.values, ax=ax[1, 0])
ax[1, 0].set_title('Distribution of Free vs. Paid Apps')
ax[1, 0].set_xlabel('Type')
ax[1, 0].set_ylabel('Number of Apps')


# install ranges per category - mean installs
mean_installs_per_category = df_cleaned.groupby('Category')['Installs'].mean().sort_values(ascending=False).head(10)
sns.barplot(x=mean_installs_per_category.values, y=mean_installs_per_category.index, ax=ax[1, 1])
ax[1, 1].set_title('Top 10 Categories by Mean Installs')
ax[1, 1].set_xlabel('Mean Installs')
ax[1, 1].set_ylabel('Category')


plt.tight_layout()
plt.show()
```

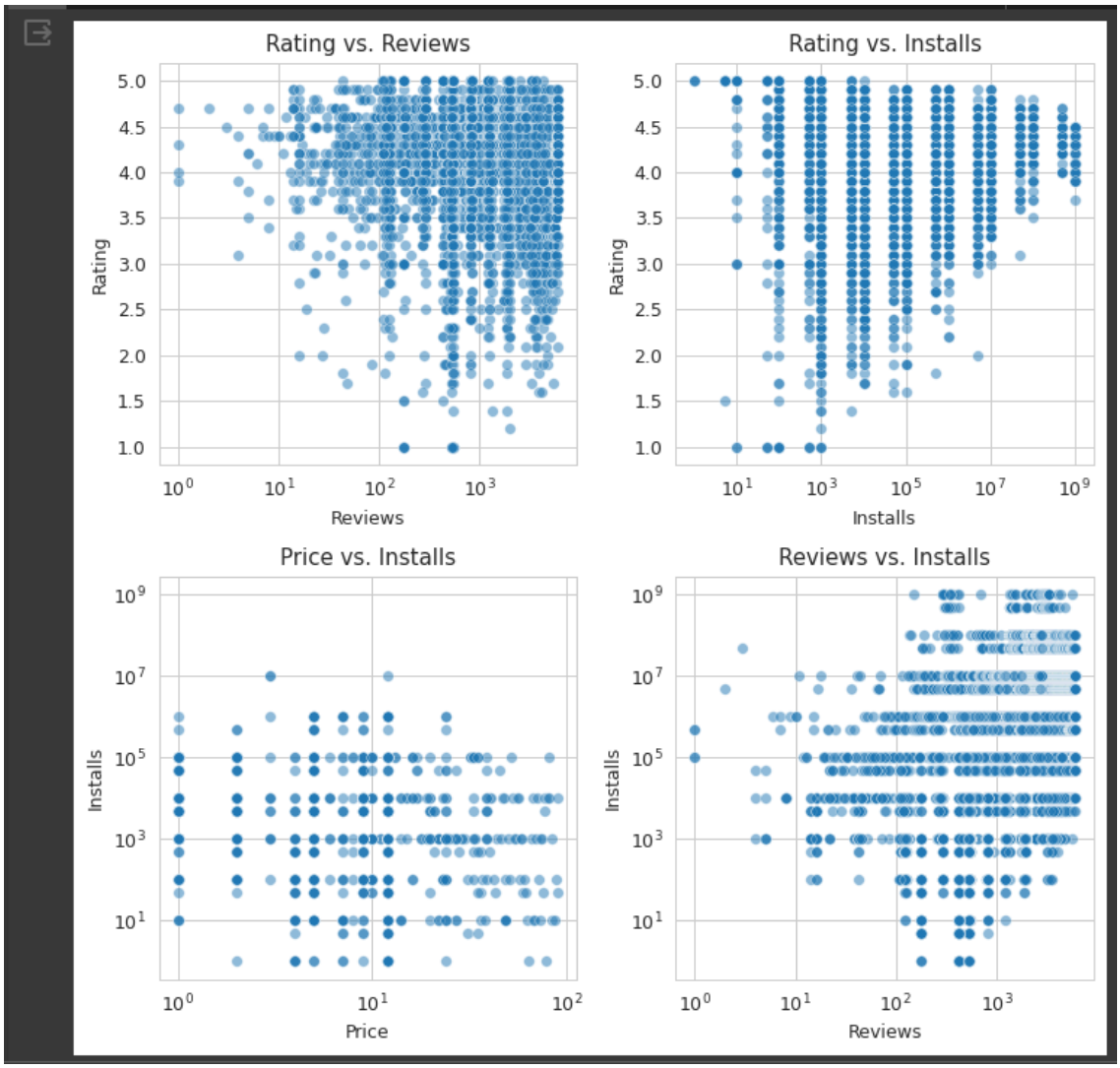


Observations:

- There are plenty of apps in the Google Playstore that fall into the Family and Gaming categories.
- The majority of apps require Android 4.1 or later as the operating system.
- Free programs are used by many individuals.
- Applications connected to communication are widely used (WhatsApp, FB messanger etc.).

Analysing original data.

```
fig, ax = plt.subplots(2, 2, figsize=(8, 8), dpi=90)

# rating vs. reviews
sns.scatterplot(x='Reviews', y='Rating', data=df_cleaned, ax=ax[0, 0], alpha=0.5)
ax[0, 0].set_title('Rating vs. Reviews')
ax[0, 0].set_xlabel('Reviews')
ax[0, 0].set_ylabel('Rating')
ax[0, 0].set_xscale('log')  # Using log scale due to wide range of values

# rating vs. installs
sns.scatterplot(x='Installs', y='Rating', data=df_cleaned, ax=ax[0, 1], alpha=0.5)
ax[0, 1].set_title('Rating vs. Installs')
ax[0, 1].set_xlabel('Installs')
ax[0, 1].set_ylabel('Rating')
ax[0, 1].set_xscale('log')  # Using log scale due to wide range of values

# price vs. installs
sns.scatterplot(x='Price', y='Installs', data=df_cleaned, ax=ax[1, 0], alpha=0.5)
ax[1, 0].set_title('Price vs. Installs')
ax[1, 0].set_xlabel('Price')
ax[1, 0].set_ylabel('Installs')
ax[1, 0].set_xscale('log')  # Using log scale due to wide range of values
ax[1, 0].set_yscale('log')  # Also log scale for better visualization

# reviews vs. installs
sns.scatterplot(x='Reviews', y='Installs', data=df_cleaned, ax=ax[1, 1], alpha=0.5)
ax[1, 1].set_title('Reviews vs. Installs')
ax[1, 1].set_xlabel('Reviews')
ax[1, 1].set_ylabel('Installs')
ax[1, 1].set_xscale('log')  # Using log scale due to wide range of values
ax[1, 1].set_yscale('log')  # Also log scale for better visualization

plt.tight_layout()
plt.show()
```



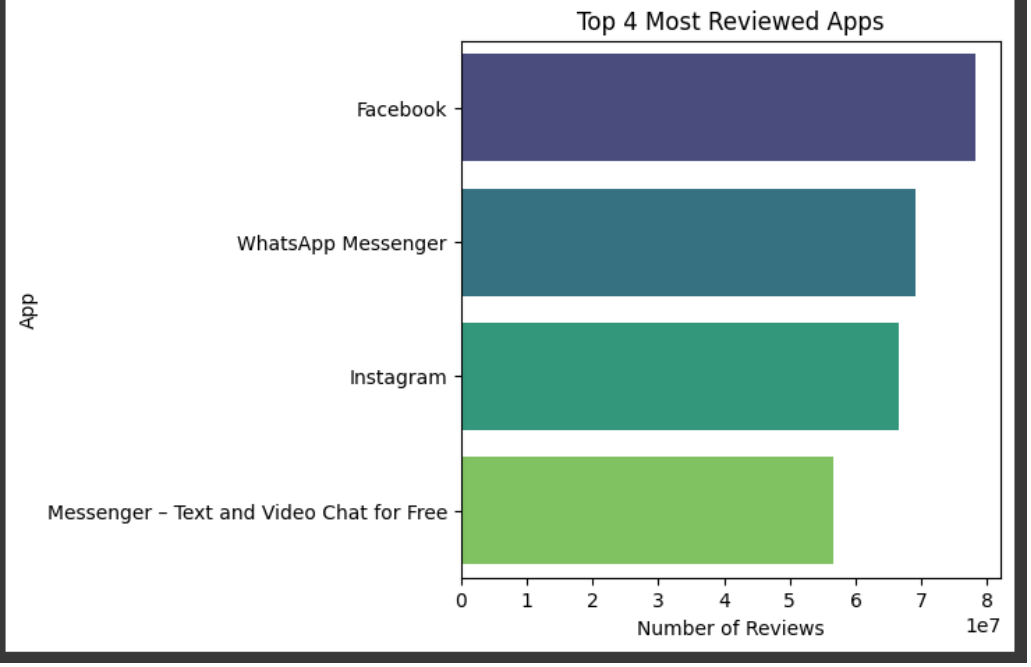
Observations:

- Both user ratings and reviews increased at the same time.
- The quantity of installations of Android apps increases when a large number of users rate them. Additionally, there are a lot of installations in the range of ratings 3.5 to 5.0, which indicates that a lot of users typically install Android apps in this range.
- Users are more likely to install apps when there are more reviews.

Analysing Reviews vs Android Apps.

```
# sorting the dataframe by the 'reviews' column and select the top 10 rows
top_10_apps = df_cleaned.sort_values('Reviews_float', ascending=False).head(10)

# creating a bar plot
plt.figure(figsize=(5, 5))
sns.barplot(data=top_10_apps, x='Reviews_float', y='App', palette='viridis')
plt.xlabel('Number of Reviews')
plt.ylabel('App')
plt.title('Top 4 Most Reviewed Apps')

plt.show()
```

```
<ipython-input-35-b3e2ce083dd3>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assi

  sns.barplot(data=top_10_apps, x='Reviews_float', y='App', palette='viridis')
```

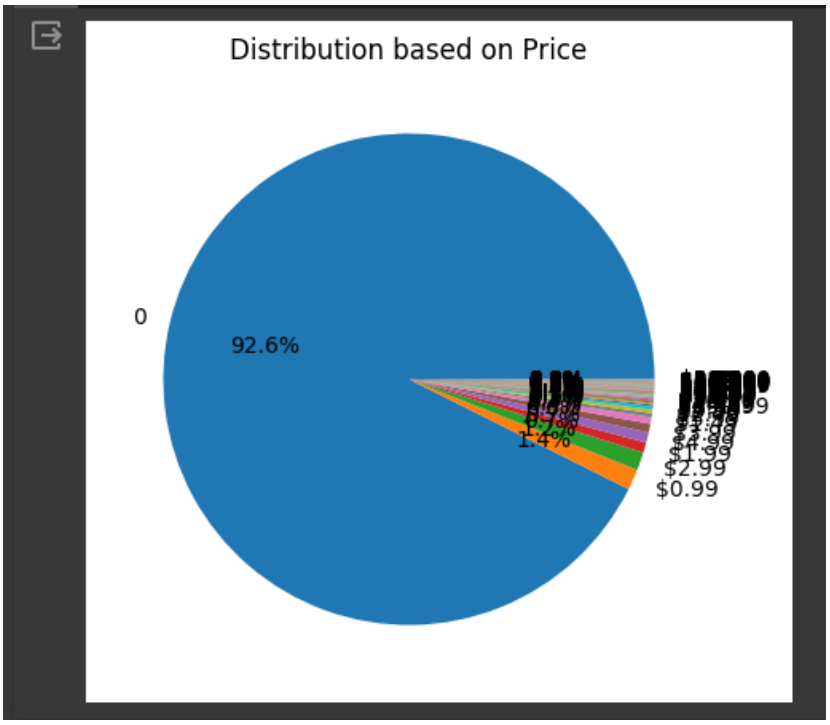


Observations:

- In the Google Apps Store, the Facebook app has received the most reviews.
- The amount of reviews for the Instagram App and WhatsApp Messenger are nearly equal.

Generating a Pie chart including the Price distribution.

```
# Plotting a pie chart to display the distribution of price.

downloads = df_cleaned['Price'].value_counts()
plt.figure(figsize=(5,5))
downloads.plot(kind='pie', autopct='%1.1f%%')
plt.title("Distribution based on Price")
plt.ylabel('')
plt.show()
```



Observation: It seems there are so many free android apps in the Google Apps Store.

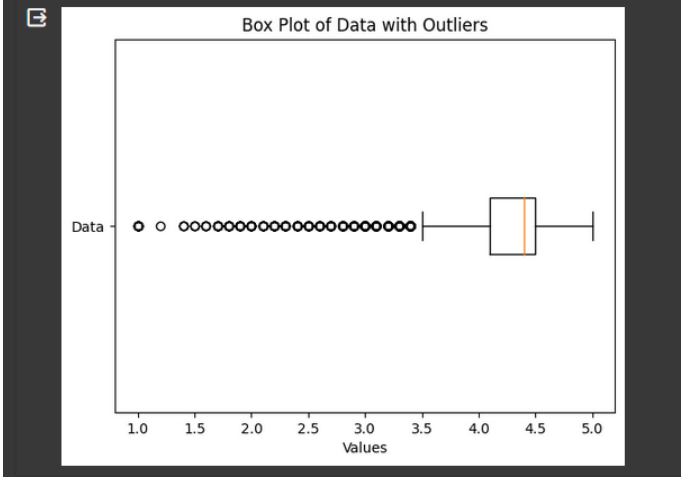
Removing missing values from the Rating_mis_handled column.

```
# Rating column has many missing values
# it is possible to use a ML model to predict missing categorical values based on other features in the dataset.

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='most_frequent')
df_cleaned['Rating_mis_handled'] = imputer.fit_transform(df_cleaned[['Rating']])
```

```
<ipython-input-153-17bf083eb21d>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Rating_mis_handled'] = imputer.fit_transform(df_cleaned[['Rating']])
```

Generating a box plot to find out liers.

```
# Generating a box plot to find outliers

plt.boxplot(df_cleaned['Rating_mis_handled'], vert=False, labels=['Data'])
plt.title('Box Plot of Data with Outliers')
plt.xlabel('Values')
plt.show()
```

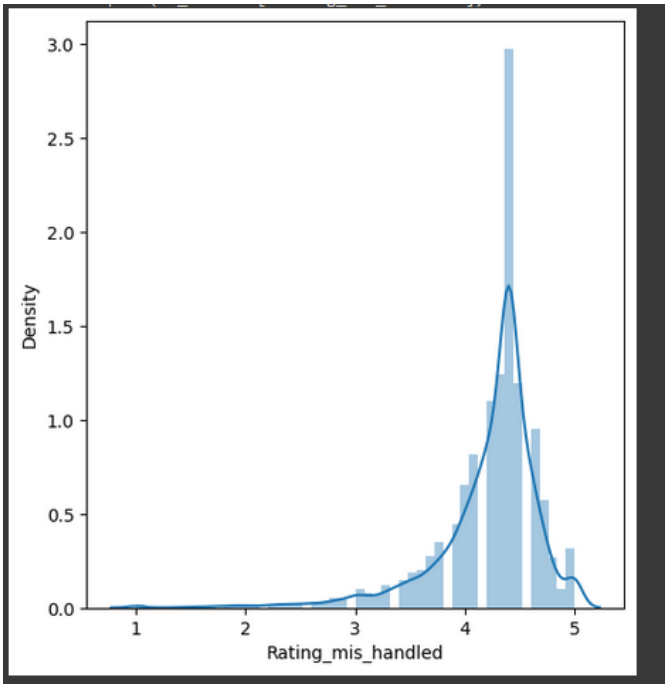


Finding the skewness of the above column.

```
[156] import matplotlib.pyplot as plt
      import seaborn as sns
```

```
# plotting the destribution of the column data to find the skewness

plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.distplot(df_cleaned['Rating_mis_handled'])

plt.show()
```

```
<ipython-input-157-923d8ac3d300>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df_cleaned['Rating_mis_handled'])
```




```
# finding the skewness
# negative value means the distribution of the column's data has been skewed to the left

df_cleaned['Rating_mis_handled'].skew()
```

```
-2.0902251713418454
```

The Pandas library offers a function called skew() that can be used to determine how skew a data set is. The asymmetry of a real-valued random variable's probability distribution with respect to its mean is measured by its skewness. A distribution that is skewed to the left is indicated by a negative skewness, and one that is skewed to the right by a positive skewness.

According to the above results, the data distribution of the column Rating_mis_handled skewed to the right.

```
# finding the bounadray values

print('Heigth allowed=',df_cleaned['Rating_mis_handled'].mean()+3*df_cleaned['Rating_mis_handled'].std())
print('Lowest allowed=',df_cleaned['Rating_mis_handled'].mean()-3*df_cleaned['Rating_mis_handled'].std())
```

```
Heigth allowed= 5.672665881781542
Lowest allowed= 2.767481719694473
```

```
# finding outliers

df_cleaned[(df_cleaned['Rating_mis_handled']>5.67 )|( df_cleaned['Rating_mis_handled']<2.77)]
```

Checking the dimension of the data set.

```
# checking the dimention of the data set

df_cleaned.shape
```

```
(10840, 24)
```

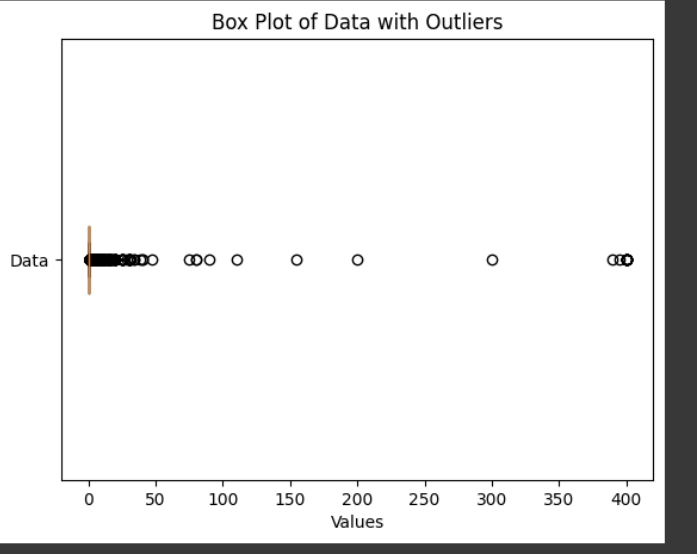
Trimming and re checking the dimension.

```
[162] # trimming

df_cleaned=df_cleaned[(df_cleaned['Rating_mis_handled']<5.31) & (df_cleaned['Rating_mis_handled']>3.28)]

# checking the dimention of the data set after trimming / removing outliers

df_cleaned.shape

(10337, 24)
```

It is observed that the outliers of the mentioned column have been removed.

Checking outliers in the Price_float variable.

```
# Generating a box plot to after removing outliers

plt.boxplot(df_cleaned['Price_float'], vert=False, labels=['Data'])
plt.title('Box Plot of Data with Outliers')
plt.xlabel('Values')
plt.show()
```



It seems that the Price_float variable has so many outliers hence better to not to use this feature to fit the model.

Checking na values in the Type_Encoded field.

```
# check na values

import numpy as np

np.isnan(df_cleaned['Type_Encoded']).sum()
```

```
1
```

Cleaning Type_Encoded column by removing non numerical values.

```
df_cleaned['Type_Encoded'].unique()
```

```
array([ 0.,  1., nan])
```

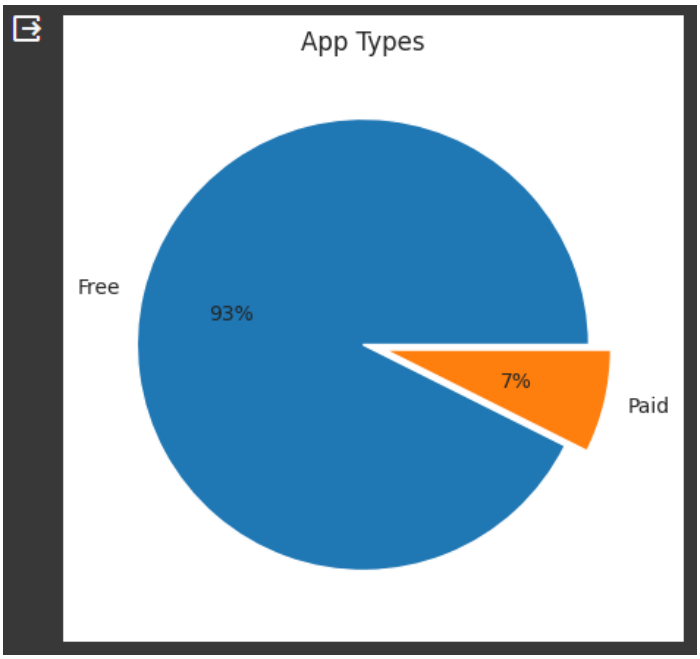
```
df_cleaned['Type_Encoded'].replace(np.NaN, 0, inplace=True)
```

```
# viewing total null of each column

df_cleaned.isnull().sum()
```

```
0
```

Displaying the distribution of Application Types using a pie chart.

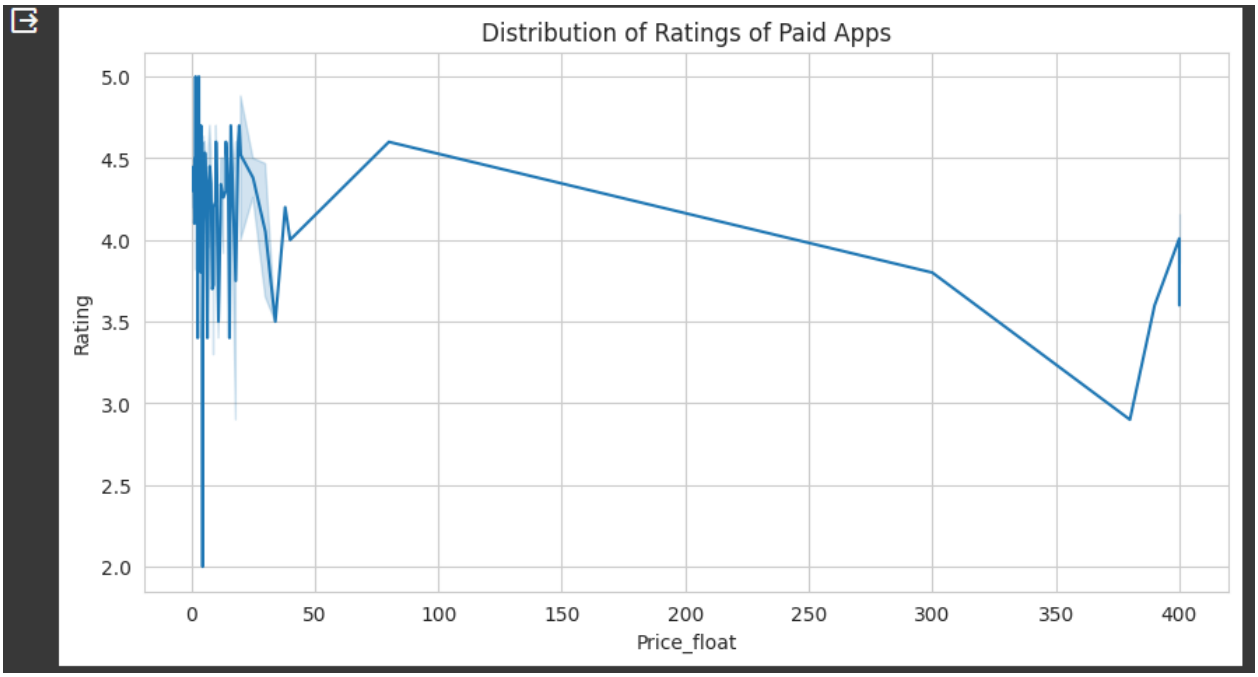
```
# visualizing App Types in a pie chart

plt.figure(figsize=(5,5))
fig = df_cleaned.groupby("Type")["Type"].count().plot(kind="pie", autopct='%1.0f%%', shadow=False, explode=(0,0.1))
fig.set_title("App Types")
fig.set_ylabel("")
plt.show()
```



It can be seen that when compared to Paid applications on the Google Play Store, a significantly high number of Free Android applications are available.

Visualizing the comparison between paid applications against user ratings on the Google Play Store.

```
# Generating a sns line plot to compare ratings vs paid apps

plt.figure(figsize=(10,5))
sns.lineplot(data=df_cleaned[df_cleaned["Type"] == "Paid"], x="Price_float", y="Rating")
plt.title("Distribution of Ratings of Paid Apps")
plt.show()
```



It's interesting to note that a lot of free apps have good ratings, and when prices of applications rise, ratings drop to about a 3.0 rating level.

Checking duplicate values and remove them from the App_Encoded column.

```
# finding duplicates

duplicates_count = df_cleaned.duplicated(subset=['App_Encoded']).sum()

print("Number of duplicate rows:", duplicates_count)
```

```
Number of duplicate rows: 1169
```

```
# Removing duplicates

df_cleaned.drop_duplicates(subset=['App_Encoded'], inplace=True)

print("Number of duplicate rows:", df_cleaned.duplicated().sum())
```

```
Number of duplicate rows: 0
```

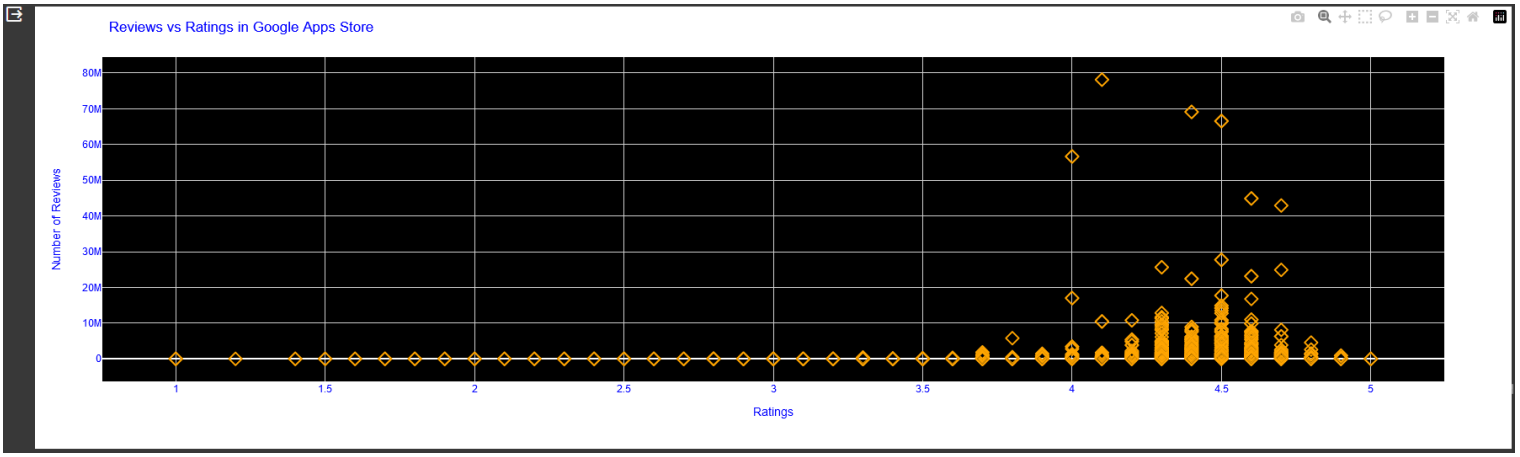
Generating an interactive scatter plot to display the data analysis between the Rating vs Reviews.

Note: A list of column names (App_Encoded, Category_Encoded, Installs_Encoded) or keys from the DataFrame is passed to the hover_data argument. These columns hold the data that must appear when hovering over the appropriate points on the scatter plot.

```
# creating an iteractive scatter plot to display reviews vs rating

import plotly.express as px

fig = px.scatter(df_cleaned, x='Rating', y='Reviews_float', hover_data=['App_Encoded', 'Category_Encoded', 'Installs_log'])

# marker size, shape, and border line customization
fig.update_traces(
    marker=dict(
        size=12,  # to increase the size of the data points
        symbol='diamond',  # to change the shape of the data points to a circle
        line=dict(
            color='orange',  # border line color
            width=2  # width of the border line
        ),
        color='rgba(0, 0, 0, 0)'  # marker color

    )
)
```


```
# creating an iteractive scatter plot to display reviews vs rating

import plotly.express as px

fig = px.scatter(df_cleaned, x='Rating', y='Reviews_float', hover_data=['App_Encoded', 'Category_Encoded', 'Installs_log'])

# marker size, shape, and border line customization
fig.update_traces(
    marker=dict(
        size=12,  # to increase the size of the data points
        symbol='diamond',  # to change the shape of the data points to a circle
        line=dict(
            color='orange',  # border line color
            width=2  # width of the border line
        ),
        color='rgba(0, 0, 0, 0)'  # marker color

    )
)
```

```python
# Customize the layout
fig.update_layout(
    title="Reviews vs Ratings in Google Apps Store",
    xaxis_title="Ratings",
    yaxis_title="Number of Reviews",
    font=dict(
        family="Arial, sans-serif",
        size=12,
        color="blue"
    ),
    plot_bgcolor='black',  # plot background color
    hoverlabel=dict(
        bgcolor="yellow",  # hover label background color
        font_size=14
    ),
    xaxis=dict(
        gridcolor='lightgrey'  # gridlines
    ),
    yaxis=dict(
        gridcolor='lightgrey'  # gridlines
    )
)

fig.show()
```



Observations:

- In average, 4.5 rating category has a high number of reviews on play store apps.
- From 3.75 rating to 4.9 rating range (approximately), reviews have increased and gradually decreased.

Defining x (6 features) and y (dependant / target variable), splitting the data set into train and test sets (80% to 20% ratio) and displaying their sizes.

.

Note: Due to RandomForestRegression classifier, all the features and the target variable must contain numerical values.

```
# Spliting data into two arrays: X (features) and y (labels).

feature_columns = ['Type_Encoded', 'Content Rating_Encoded', 'Genres_Encoded','Category_Encoded', 'App_Encoded', 'Reviews_float']

x = df_cleaned[feature_columns].values
y = df_cleaned['Rating_mis_handled'].values
```

```
# Split data into 80% of train data set and 20% of test data set

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=40)
```

```
# checking test and train set sizes

print(X_train.shape,
X_test.shape,
y_train.shape,
y_test.shape)
```

```
(7334, 6) (1834, 6) (7334,) (1834,)
```

Standardization.

In machine learning, standardizing features by taking the mean away and scaling to the variance is a standard preprocessing step. Through this method, a dataset's characteristics are altered to have a zero mean and a one standard deviation. Z-score normalization is another term for this.

```
# Standardize features by removing the mean and scaling to unit variance
# Standard Scaling for model efficiency

from sklearn.preprocessing import StandardScaler

sc = StandardScaler(with_std  = True ,with_mean = True, copy = True)
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Using RandomForestRegressor classifier.

n_estimators:

Functionality: in a random forest, this parameter defines the number of trees.

Default Value: 100.

Impact: Increasing the number of trees generally improves the performance of the random forest, but it also increases the computational cost.

random_state:

Functionality: To populate the random number generator, use this parameter. Reproducibility is ensured by using a fixed seed, thus if the user runs the model with the same random_state each time, the outcomes should be the same.

Default Value: If nothing is said, the random state is not fixed, and every time the model runs, it can yield a different set of outcomes.

Impact: Reproducing the results or requiring consistent outcomes for debugging or code sharing necessitates fixing the random state.

```
from sklearn.ensemble import RandomForestRegressor
# RandomForestRegressor is a classifier which expects float numbers (labels), but RandomForestClassifier expects discrete numbers (labels).

# Create another Random Forest Regressor model with a different parameters
rfr_model = RandomForestRegressor(n_estimators=100, random_state=52)
```

Fit the model with X_train and y_train data sets.

```
# Build a forest of trees from the training set (X, y).
# Fit the model on the training data
# Fit the model with X_Train

rfr_model.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=52)
```

Making predictions on the test data set.

```
# Assigning predictions that has captured from the features of the test set
# Make predictions on the test set

y_pred = rfr_model.predict(X_test)
```

Evaluating the model by finding the accuracy of the model.

```
# Evaluating the model

# thisis to match y_test values that has been captured earlier with the y_pred predicted values
# accuracy score measures the proportion of the correct predictions out of the total predictions
# accuracy = accuracy_score(y_test, y_pred) : accuracy_score function = this is for classification tasks only, therefore, score() function can be used as follow

rfr_model.score(X_train, y_train)
```

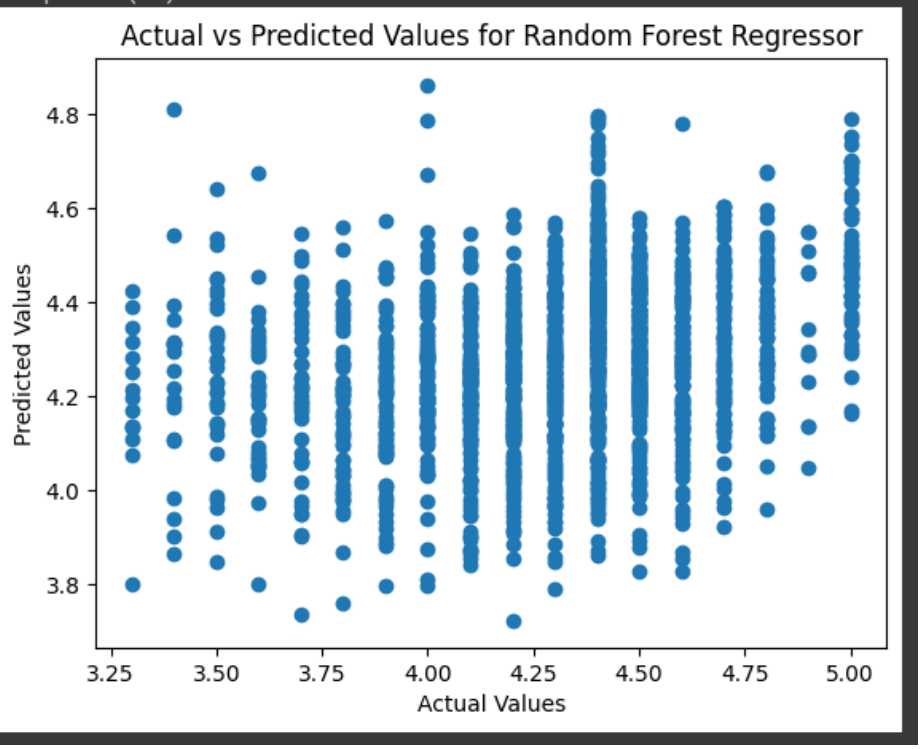
```
0.8636001540038032
```

From the above, the model has an accuracy score of 86% which is good for predictions on new data feeds.

Evaluating the model by plotting a scatter plot (Actual vs Predicted values).

```python
# Evaluating the model

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Display the regression report
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}') # # In regression tasks, it is good to use the metric R-squared (co-effecient of determination)

# Visualize predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values for Random Forest Regressor')
plt.show()
```

```
Mean Absolute Error (MAE): 0.2526810250817889
Mean Squared Error (MSE): 0.11208892529989097
R-squared (R2): 0.036350242206164673
```



Same as the score of accuracy, it can be observed that the model has a very good ratio of predicting values.

Experimenting to increase the performance of the model by including more features (9 features included).

TRYING TO INCREASE MODEL'S PERFORMANCE

```
# Spliting data into two arrays: X (features) and y (labels).

feature_columns = ['Installs_log','Type_Encoded', 'Content Rating_Encoded', 'Genres_Encoded','Category_Encoded', 'App_Encoded', 'Current Ver_Encoded',
       'Android Ver_Encoded','Reviews_float']

x = df_cleaned[feature_columns].values
y = df_cleaned['Rating_mis_handled'].values
```

```
# Evaluating the model

# thisis to match y_test values that has been captured earlier with the y_pred predicted values
# accuracy score measures the proportion of the correct predictions out of the total predictions
# accuracy = accuracy_score(y_test, y_pred) : accuracy_score function = this is for classification tasks only, therefore, score() function can be used as follow

rfr_model.score(X_train, y_train)
```

```
0.8838922397900871
```

When rerun the same above-mentioned steps, it came to notice that the accuracy of the model has been increased from 86% to 88% which is better than the previous model to predict values against new data.

Accuracy: It shows the percentage of cases in the dataset that have been accurately classified out of all instances. Stated differently, accuracy quantifies the model's capacity to accurately forecast the class labels.

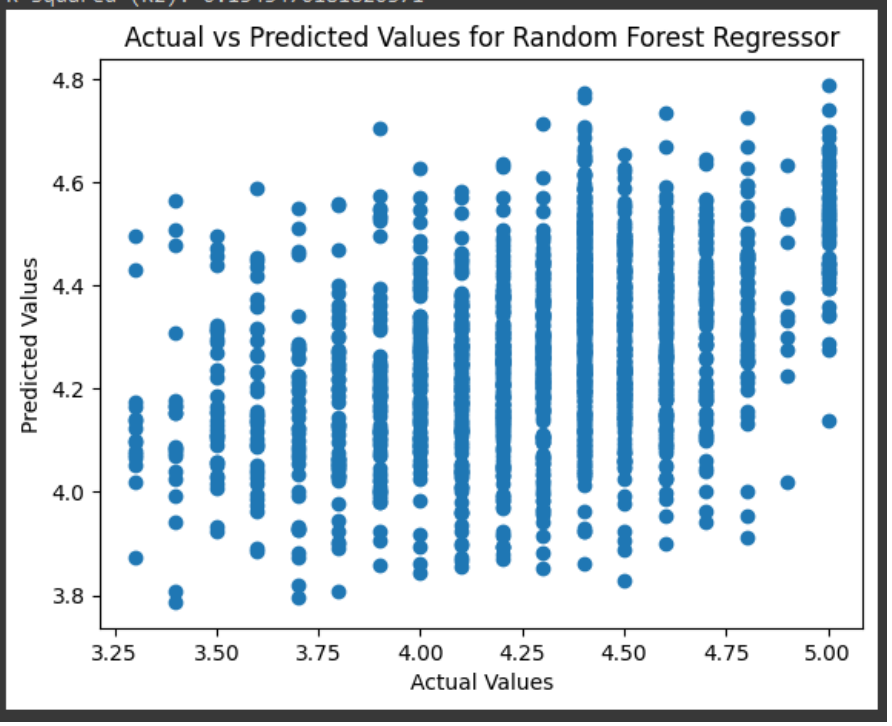
Mathematical equation:

$$\text{Accuracy} = \frac{\text{No of correct predictions}}{\text{Total number of predictions}}$$

Re-visualizing the values of Actual vs Predicted values.

```python
# Evaluating the model

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Display the regression report
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}') # # In regression tasks, it is good to use the metric R-squared (co-effecient of determination)

# Visualize predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values for Random Forest Regressor')
plt.show()
```

```
Mean Absolute Error (MAE): 0.23735823336968437
Mean Squared Error (MSE): 0.09836388495092696
R-squared (R2): 0.1543470181820571
```



Finding best features among others.

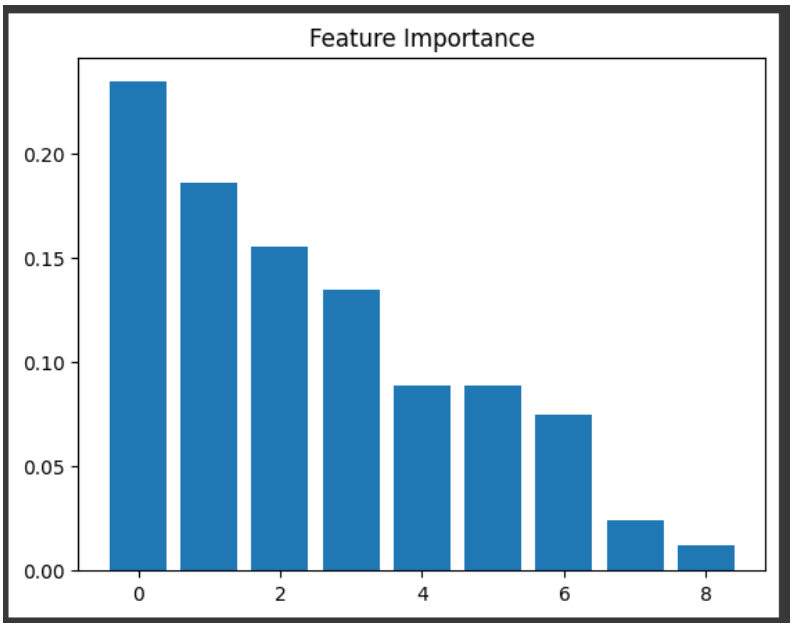
```
# For finding best features among others.
# To label the features form best to worst using above defined feature_columns
# Using RandomForestRegressor

from sklearn.ensemble import RandomForestRegressor

r_forest_r = RandomForestRegressor(n_estimators=500,random_state=1)
r_forest_r.fit(X_train, y_train)

importances = r_forest_r.feature_importances_

indices = np.argsort(importances)[::-1]

for f in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30, feature_columns[indices[f]], importances[indices[f]]))
```

```
 1) Reviews_float                0.234490
 2) App_Encoded                  0.185871
 3) Current Ver_Encoded          0.155397
 4) Installs_log                 0.134744
 5) Genres_Encoded               0.089090
 6) Android Ver_Encoded          0.088953
 7) Category_Encoded             0.074568
 8) Content Rating_Encoded       0.024387
 9) Type_Encoded                 0.012501
```

According to the above results, Rating and Android Applications, has a high effect on the Review rate.

Plotting the above results.

```
plt.title('Feature Importance')
plt.bar(range(X_train.shape[1]), importances[indices], align='center')

# plt.xticks(range(X_train.shape[1]), feature_columns[indices], rotation=90)
plt.tick_params(axis='x', labels=feature_columns[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()

plt.show()
```



So, the feature zero which is Reviews has a high impact on Ratings and in contrast Type has a very low impact on the same dependant variable.

Finding the difference between predictions and actual values.

```
# displaying predictions and actual values with their differences

for z in zip(y_test, y_pred):
    print(z, (z[0]-z[1]) /z[0] )
```

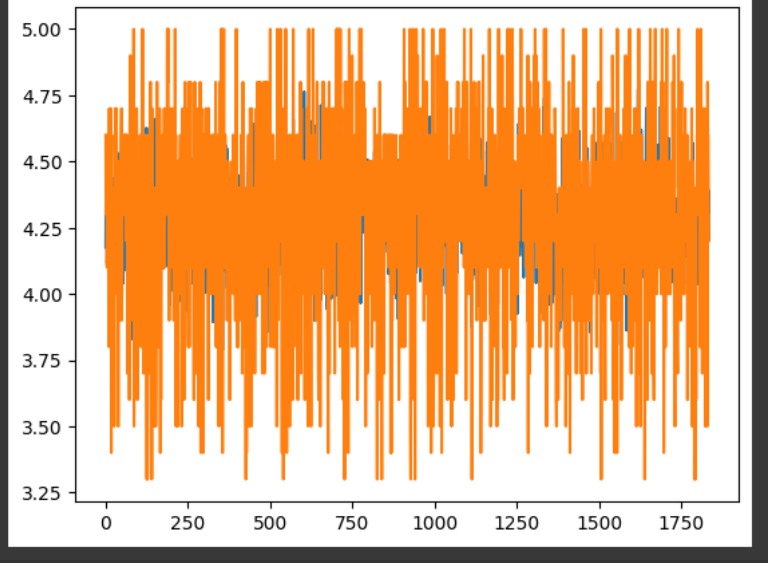
```
(3.9, 4.237000000000003) -0.08641025641025714
(4.4, 4.273000000000003) 0.028863636363635706
(4.5, 4.493000000000002) 0.0015555555555550882
(4.5, 4.266000000000002) 0.0519999999999996
(4.4, 4.3459999999999965) 0.012272727272728141
(4.7, 4.287999999999998) 0.08765957446808566
(3.5, 4.27) -0.2199999999999999
(4.4, 4.324000000000003) 0.01727272727272658
(4.4, 4.569999999999995) -0.03863636363636241
(4.5, 3.925000000000001) 0.12777777777777752
(3.9, 4.203999999999995) -0.07794871794871677
(4.2, 4.265999999999997) -0.01571428571428504
```

From the above differences says that the actual values vs predictions have a very small gap which is a good sign.

Plotting actual ratings and predicted ratings.

```
# plotting actual Ratings (orange line) and Predicted Ratings(blue line)

r = []
for pair in  zip(y_pred, y_test):
  r.append(pair)

plt.plot(r)
```

```
[<matplotlib.lines.Line2D at 0x7d9399fabd00>,
 <matplotlib.lines.Line2D at 0x7d9399faad10>]
```

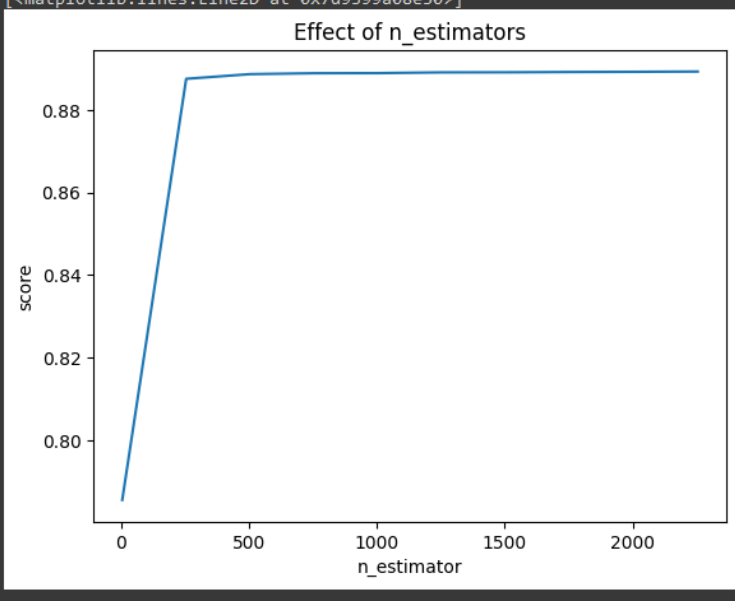


From the above plot says that the predicted values against the actual values are significantly closer each other.

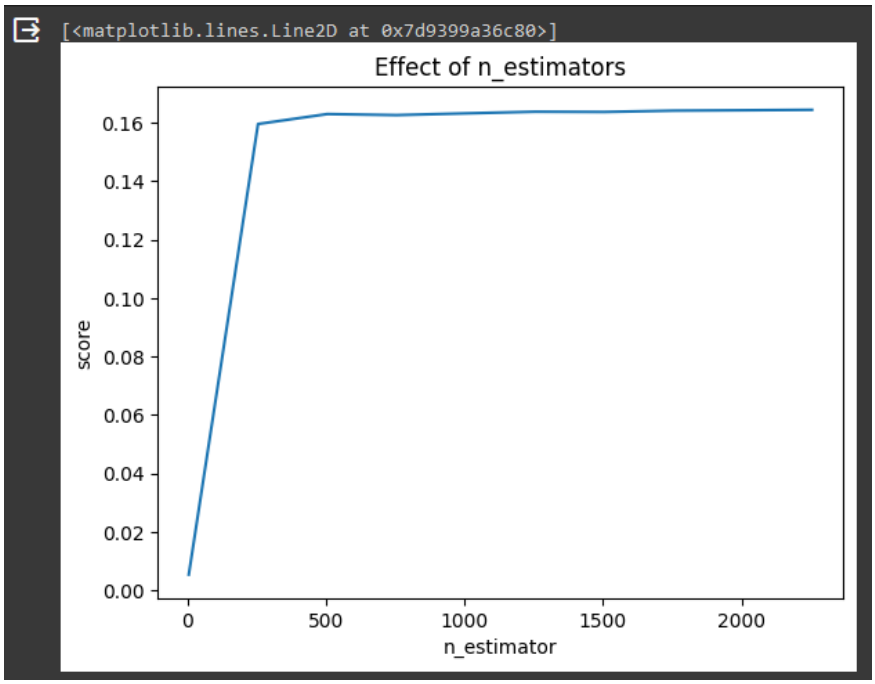
Visualizing the most efficient estimator value.

```
# Effect of estimators on score
# With np.range it is possible to find the most efficient estimator value.

estimators = np.arange(5, 2500, 250) # 0 to 2500 increased with 250
scores = []
for n in estimators:
    rfr_model.set_params(n_estimators=n)
    rfr_model.fit(X_train, y_train)
    scores.append(rfr_model.score(X_train, y_train))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```

```
[<matplotlib.lines.Line2D at 0x7d9399a08e50>]
```




```
stimators = np.arange(5, 2500, 250) # 5 to 2500 increased with 250
scores = []
for n in estimators:
    rfr_model.set_params(n_estimators=n)
    rfr_model.fit(X_train, y_train)
    scores.append(rfr_model.score(X_test, y_test))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```



From the above, around 250 is the highest estimator value.

Report on Google Play Store Analysis

Outliers and Data Distributions

The Price_float variable has substantially more outliers than the Rating variable, according to the dataset analysis. Moreover, there is a slight normal distribution seen in the data in the Installs_log field.

App Availability by User Type

There are many of Android apps available in the Google Play Store that appeal to different user types. Interestingly, there is a noticeable focus on teen-focused apps, suggesting a varied app ecosystem.

App Categories

There appears to be plenty of family-friendly and entertainment apps on the Google Play Store, as seen by the large number of apps in the Family and Gaming categories.

Operating System Requirements

Because most of the apps in the dataset require Android 4.1 or later as the operating system, it's critical to keep apps up to date in order for it to work properly.

Free and Communication Apps

A large number of people utilize free programs, and communication-related apps like Facebook Messenger and WhatsApp are very popular.

User Ratings, Reviews, and Installations

Over time, there is a concurrent increase in user ratings and reviews. Additionally, a relationship is shown between the number of installations and user ratings. This relationship is concentrated in the rating range of 3.5 to 5.0, suggesting that consumers typically install apps in this range.

Popular Apps

The Facebook app has accumulated the most reviews in the Google Play Store. Furthermore, there are about equal numbers of reviews for WhatsApp Messenger and the Instagram app.

Free vs. Paid Apps

One noteworthy finding in the Google Play Store is the large number of free Android applications. According to the analysis, a significant portion of free apps get positive reviews. On the other hand, ratings have decreased to a level of about 3.0 as application prices rise.

Rating Categories and Reviews

A favorable association between greater ratings and better user involvement is suggested by the high number of reviews found in apps with an average rating of 4.5. But the number of reviews steadily declines after reaching its high in the 3.75–4.9 rating range.

Feature Impact on Review Rate

The results of the model feature analysis show that the review rate is significantly influenced by both rating and Android applications. Notably, ratings are significantly impacted by the 'Reviews' option (which may be indicated as 'zero'), whilst the dependent variable is comparatively less affected by the 'Type' feature.

Model Prediction Comparison

The comparison between actual values and predictions reveals a very small gap, signifying the model's reliability. Furthermore, the predicted values against the actual values are significantly close to each other, demonstrating the effectiveness of the developed model.

Model Performance

The dataset's predictive model performs well, as evidenced by its first 86% accuracy score. The accuracy has increased to 88% as a result of subsequent upgrades, suggesting improved predictive skills for new data sources.

In conclusion, the analysis provides valuable insights into the characteristics of the Google Play Store dataset, user behavior, and the performance of a predictive model, offering a comprehensive understanding of app trends and user preferences in the Android ecosystem.

References

5 Basic Components of A Blockchain Networ. (n.d.). Retrieved from vietnamblockchain.asia: https://vietnamblockchain.asia/post/5666316/5-basic-components-of-blockchain

Afshine Amidi and Shervine Amidi. (n.d.). *Recurrent Neural Networks cheatsheet* . Retrieved from stanford.edu: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Agit Çelik. (2018). *Revenue Prediction using Random Forest Regressor*. Retrieved from www.kaggle.com: https://www.kaggle.com/code/celikagit/revenue-prediction-using-random-forest-regressor

Akansha Khandelwal. (2024, Feb 22). *Azure Machine Learning: A Step-by-Step Guide*. Retrieved from www.analyticsvidhya.com: https://www.analyticsvidhya.com/blog/2021/09/a-comprehensive-guide-on-using-azure-machine-learning/

Ali, T. (n.d.). *Comparative Analysis of Blockchain Architecture and Its Application.* IEEEAccess.

Allie Grace Garnett. (n.d.). *How smart contracts work with blockchain: A step-by-step guide*. Retrieved from www.britannica.com: https://www.britannica.com/money/how-smart-contracts-work

amazon. (n.d.). *What is a Neural Network?* Retrieved from aws.amazon.com: https://aws.amazon.com/what-is/neural-network/

amazon. (n.d.). *What is Deep Learning?* Retrieved from aws.amazon.com: https://aws.amazon.com/what-is/deep-learning/

amazon. (n.d.). *What Is RNN?* Retrieved from aws.amazon.com: https://aws.amazon.com/what-is/recurrent-neural-network/

Asaph Azaria; Ariel Ekblaw; Thiago Vieira; Andrew Lippman. (2016, Aug 22). *MedRec: Using Blockchain for Medical Data Access and Permission Management*. Retrieved from ieeexplore.ieee.org: https://ieeexplore.ieee.org/document/7573685

Biplav Kant . (2022). *How to Detect and Remove Outliers*. Retrieved from www.kaggle.com: https://www.kaggle.com/code/biplavkant/how-to-detect-and-remove-outliers

Components of Blockchain Network. (2022, Oct 02). Retrieved from www.geeksforgeeks.org: https://www.geeksforgeeks.org/components-of-blockchain-network/

Ella Creamer. (2023, Jul 05). *Authors file a lawsuit against OpenAI for unlawfully 'ingesting' their books*. Retrieved from www.theguardian.com: https://www.theguardian.com/books/2023/jul/05/authors-file-a-lawsuit-against-openai-for-unlawfully-ingesting-their-books

Harnessing the Power of GenAI. (2024, Jan 18). Retrieved from www.wsj.com: https://www.wsj.com/video/sponsored/harnessing-the-power-of-genai/4C47317C-0018-4DE2-AE4D-DA6B6B571ADD.html?utm_medium=content_discovery&utm_source=google-search&gad_source=1&gclid=CjwKCAiA2pyuBhBKEiwApLaIOzOxSOQlPCMX6m5Nj27Cq36Ipez5qN_T6s3FFLH4U6EcqynPjO

ibm. (n.d.). *IBM Food Trust* . Retrieved from www.ibm.com: https://www.ibm.com/products/supply-chain-intelligence-suite/food-trust

James Vincent. (2023, Feb 08). *Google's AI chatbot Bard makes factual error in first demo*. Retrieved from www.theverge.com: https://www.theverge.com/2023/2/8/23590864/google-ai-chatbot-bard-mistake-error-exoplanet-demo

Jessica Groopman. (2019, Oct 30). *AI, blockchain and IoT convergence improves daily applications*. Retrieved from www.techtarget.com: https://www.techtarget.com/iotagenda/tip/AI-blockchain-and-IoT-convergence-improves-daily-applications

linkedin. (n.d.). *How can you normalize data in ML models during data cleaning?* Retrieved from www.linkedin.com: https://www.linkedin.com/advice/0/how-can-you-normalize-data-ml-models-during-cleaning-edo4e

Marcel Isler. (2023, Oct 13). *DLT ADVANTAGES AND BENEFITS OF DISTRIBUTED LEDGER TECHNOLOGY*. Retrieved from imiblockchain.com: https://imiblockchain.com/dlt-advantages-and-benefits/

Mark Purdy and A. Mark Williams . (2023, Oct 26). *How AI Can Help Leaders Make Better Decisions Under Pressure* . Retrieved from hbr.org: https://hbr.org/2023/10/how-ai-can-help-leaders-make-better-decisions-under-pressure

Matt G. Southern . (2023, Feb 22). *Microsoft's AI-Powered Bing Search Now On Mobile* . Retrieved from www.searchenginejournal.com: https://www.searchenginejournal.com/microsofts-ai-powered-bing-search-now-on-mobile/480762/

Matthew Urwin. (2023, Mar 08). *Precision and Recall: How to Evaluate Your Classification Model*. Retrieved from builtin.com: https://builtin.com/data-science/precision-and-recall

Michael M. Grynbaum and Ryan Mac. (2023, Dec 27). *The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work*. Retrieved from www.nytimes.com: https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html

Microsoft. (2017, May 22). *Power BI Convert numbers to month*. Retrieved from community.fabric.microsoft.com: https://community.fabric.microsoft.com/t5/Desktop/Convert-numbers-to-month/m-p/179838

Microsoft. (2021, Nov 10). *Evaluate Model component*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/evaluate-model?view=azureml-api-2
Microsoft. (2021, Nov 04). *Normalize Data component*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/normalize-data?view=azureml-api-2
Microsoft. (2021, Nov 04). *Two-Class Logistic Regression component*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/two-class-logistic-regression?view=azureml-api-2
Microsoft. (2023, Oct 20). *Data Analysis Expressions (DAX) GROUPBY*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/dax/groupby-function-dax
Microsoft. (2023, Aug 16). *Manage Azure resource groups by using the Azure portal*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal
Microsoft. (2023, Aug 09). *Tutorial: Train a classification model with no-code AutoML in the Azure Machine Learning studio*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-first-experiment-automated-ml?view=azureml-api-2
Microsoft. (2023, Jun 07). *What is automated machine learning (AutoML)?* Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml?view=azureml-api-2
Microsoft. (2024, Jan 05). *Use tags to organize your Azure resources and management hierarchy*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/tag-resources?wt.mc_id=azuremachinelearning_inproduct_portal_utilities-tags-tab
Microsoft. (2024, Jan 17). *What is an Azure Machine Learning compute instance?* Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/concept-compute-instance?view=azureml-api-2
Microsoft. (2024, Feb 01). *Workspace Managed Virtual Network Isolation*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-managed-network?view=azureml-api-2&WT.mc_id=Portal-Microsoft_Azure_MLTeamAccounts&tabs=azure-cli
Microsoft. (n.d.). *Azure Machine Learning documentation*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/machine-learning/?view=azureml-api-2
Mohamed Sohail, Waseem Mohammad Fayed, Fidel Kaldas. (2018). *IOT AND BLOCKCHAIN - A WALLET*. Retrieved from education.dell.com: https://education.dell.com/content/dam/dell-emc/documents/en-us/2018KS_Sohail-IoT_and_Blockchain-a_wallet_of_secrets.pdf
Olivia Barber. (2023, Oct 19). *How artificial intelligence will change decision making*. Retrieved from indatalabs.com: https://indatalabs.com/blog/artificial-intelligence-decision-making
oracle. (n.d.). *What Is Natural Language Processing (NLP)?* Retrieved from www.oracle.com: https://www.oracle.com/sa/artificial-intelligence/what-is-natural-language-processing/
Prateek Majumder. (2023, Sep 25). *Guide to Create Interactive Plots with Plotly Python*. Retrieved from www.analyticsvidhya.com: https://www.analyticsvidhya.com/blog/2021/10/interactive-plots-in-python-with-plotly-a-complete-guide/
Ripple: how it works, and why it is different than other crypto. (n.d.). Retrieved from www.bots.io: https://www.bots.io/botspedia/ripple-how-it-works-and-why-it-is-different-than-other-crypto#:~:text=and%20finally%2c%20ripple%20is%20an,also%20a%20form%20of%20dlt.
SaiKumar Kalla. (n.d.). *Components of Blockchain*. Retrieved from mindmajix.com: https://mindmajix.com/components-of-blockchain

Smart Contract Challenges. (n.d.). Retrieved from hedera.com: https://hedera.com/learning/smart-contracts/smart-contract-challenges

Welcome Hyperledger Fabric 2.0: Enterprise DLT for Production. (2020, Jan 30). Retrieved from www.hyperledger.org: https://www.hyperledger.org/blog/2020/01/30/welcome-hyperledger-fabric-2-0-enterprise-dlt-for-production

What are smart contracts on blockchain? . (n.d.). Retrieved from www.ibm.com: https://www.ibm.com/topics/smart-contracts

What is IoT? (n.d.). Retrieved from www.oracle.com: https://www.oracle.com/sa/internet-of-things/what-is-iot/

What is the Internet of Things? (2022, Aug 17). Retrieved from www.mckinsey.com: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-internet-of-things

xolo. (n.d.). *What is Estonian e-Residency and how to take advantage of it?* Retrieved from www.xolo.io: https://www.xolo.io/zz-en/e-residency

Yusuf Mehdi. (2023, Feb 07). *Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web*. Retrieved from blogs.microsoft.com: https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/