Steps are followed to make the project "App Rating Prediction "

1. Load the data file using pandas.

```
# Load the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Load the dataset
data = pd.read_csv('googleplaystore.csv')
```

2. Check for null values in the data. Get the number of null values for each column.

```
# 1- check for null values
null_counts = data.isnull().sum()
print(null_counts)
```

3. Drop records with nulls in any of the columns.

```
# 2- Drop records with null
data = data.dropna()
data.info()
```

4. Variables seem to have incorrect type and inconsistent formatting. You need to fix them:

1. Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.
    1. Extract the numeric value from the column
    2. Multiply the value by 1,000, if size is mentioned in Mb

```
# 3- Correct data types and format
# a- convert size
def convert_size(Size):
    if 'M' in Size:
        return float(Size.replace('M',''))*1000
    elif 'K' in Size:
        return float(Size.replace('K',''))
    return None
data['Size'] = data['Size'].apply(convert_size)
```

2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).

```
# b- convert reviews
data['Reviews'] = pd.to_numeric(data['Reviews'], errors= 'coerce')
```

3. Installs field is currently stored as string and has values like 1,000,000+.
    1. Treat 1,000,000+ as 1,000,000

2. remove ‘+’, ‘,’ from the field, convert it to integer

```
# c- convert installs
data['Installs'] = data['Installs'].str.replace('+','').str.replace(',','').astype(int)
```

4. Price field is a string and has $ symbol. Remove ‘$’ sign, and convert it to numeric.

```
# d- convert price
data['Price'] = data['Price'].str.replace('$','').astype(float)
```

5. Sanity checks:

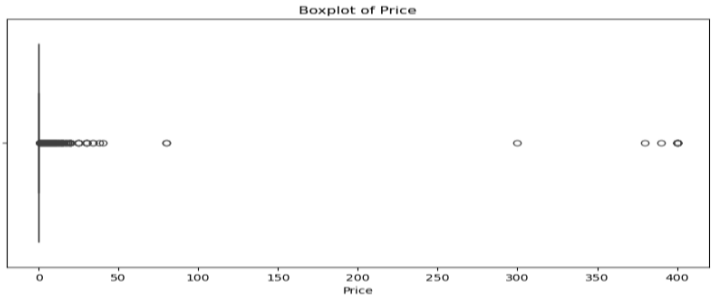
1. Average rating should be between 1 and 5 as only these values are allowed on the play store. Drop the rows that have a value outside this range.
2. Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.
3. For free apps (type = “Free”), the price should not be >0. Drop any such rows.

```
# 4- Sanity Checks

# a- Drop invalid Ratings
data = data[(data['Rating'] >= 1) & (data['Rating'] <= 5)]

# b- Drop invalid reviews
data = data[data['Reviews'] <= data['Installs']]

# c- Drop invalid prices for free apps
data = data[~((data['Type'] == 'Free') & (data['Price'] > 0))]
```

5. Performing univariate analysis:

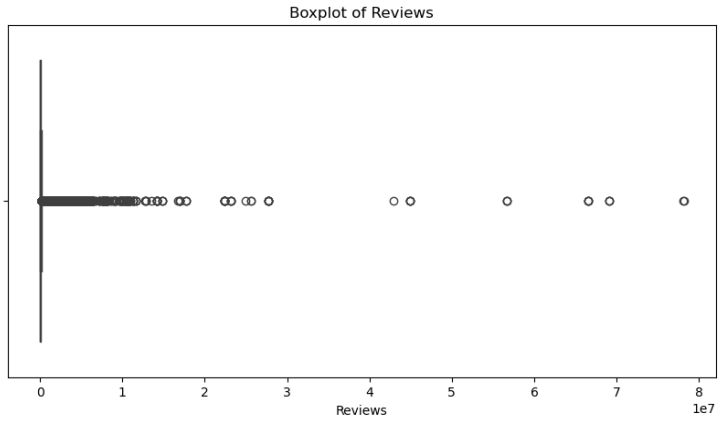
- Boxplot for Price
  - Are there any outliers? Think about the price of usual apps on Play Store.

```
# 5- Univariate Analysis

# Boxplot for Price
plt.figure(figsize=(10, 5))
sns.boxplot(x=data['Price'])
plt.title('Boxplot of Price')
plt.show()
```



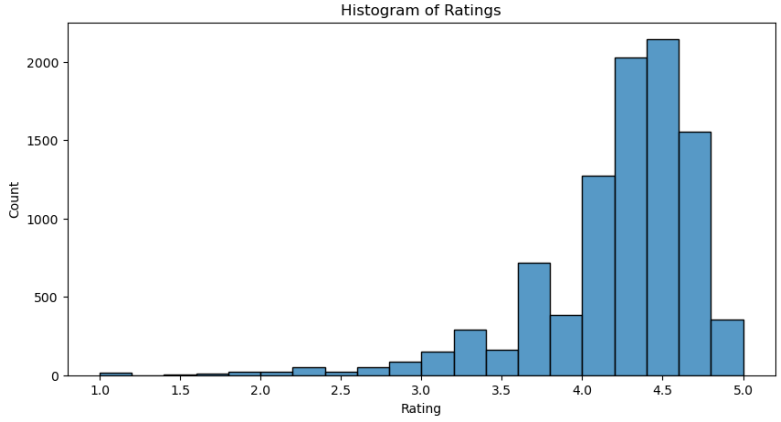
- Boxplot for Reviews
    - Are there any apps with very high number of reviews? Do the values seem right?

```
# Boxplot for Reviews
plt.figure(figsize=(10, 5))
sns.boxplot(x=data['Reviews'])
plt.title('Boxplot of Reviews')
plt.show()
```



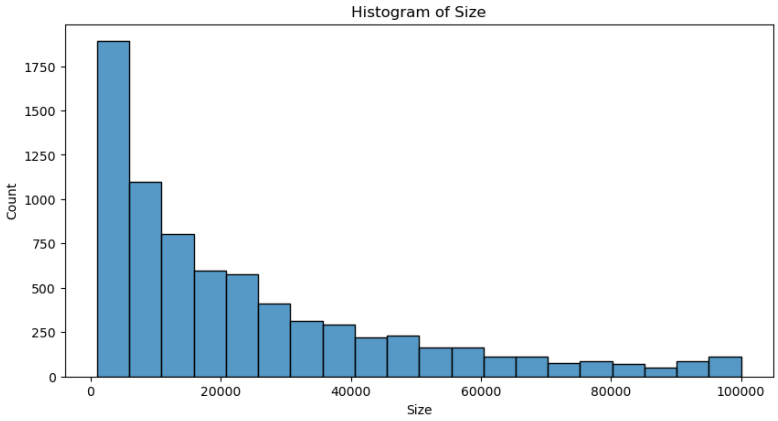
- Histogram for Rating
    - How are the ratings distributed? Is it more toward higher ratings?

```
# Histogram for Rating
plt.figure(figsize=(10, 5))
sns.histplot(data['Rating'], bins=20)
plt.title('Histogram of Ratings')
plt.show()
```



- Histogram for Size

```
# Histogram for Size
plt.figure(figsize=(10, 5))
sns.histplot(data['Size'], bins=20)
plt.title('Histogram of Size')
plt.show()
```



Note down your observations for the plots made above. Which of these seem to have outliers?

6. Outlier treatment:

1. Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!
    1. Check out the records with very high price
        1. Is 200 indeed a high price?
    2. Drop these as most seem to be junk apps
2. Reviews: Very few apps have very high number of reviews. These are all star apps that don't help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.
3. Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.
    1. Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99
    2. Decide a threshold as cutoff for outlier and drop records having values more than that

```
# Step 6: Outlier treatment
# Drop high-priced apps (above $200)
data = data[data['Price'] < 200]
```

```
# Drop records having more than 2 million reviews
data = data[data['Reviews'] <= 2000000]
```

```
# Finding percentiles for 'Installs'
percentiles = data['Installs'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
print("Percentiles for Installs:\n", percentiles)
```

```
Percentiles for Installs:
 0.10         1000.0
0.25        10000.0
0.50       500000.0
0.70      1000000.0
0.90     10000000.0
0.95     10000000.0
0.99    100000000.0
Name: Installs, dtype: float64
```

```
# Decide a threshold (let's say 95th percentile)
threshold_installs = percentiles[0.95]
data = data[data['Installs'] <= threshold_installs]
```

7. Bivariate analysis: Let's look at how the available predictors relate to the variable of interest, i.e., our target variable rating. Make scatter plots (for numeric features) and box plots (for character features) to assess the relations between rating and the other features.

1. Make scatter plot/joinplot for Rating vs. Price

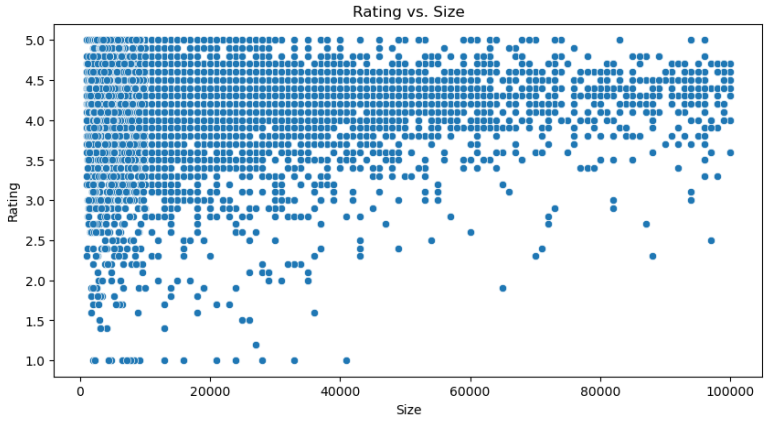
```
# Step 7: Bivariate analysis
# Scatter plot for Rating vs. Price
plt.figure(figsize=(10, 5))
sns.scatterplot(x='Price', y='Rating', data=data)
plt.title('Rating vs. Price')
plt.show()
```

1. What pattern do you observe? Does rating increase with price?

   No

2. Make scatter plot/joinplot for Rating vs. Size

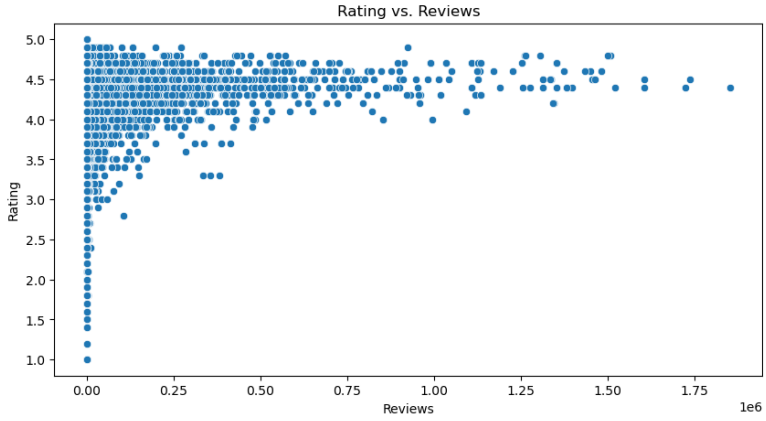
```
# Scatter plot for Rating vs. Size
plt.figure(figsize=(10, 5))
sns.scatterplot(x='Size', y='Rating', data=data)
plt.title('Rating vs. Size')
plt.show()
```



1. Are heavier apps rated better? Yes

3. Make scatter plot/joinplot for Rating vs. Reviews

```
# Scatter plot for Rating vs. Reviews
plt.figure(figsize=(10, 5))
sns.scatterplot(x='Reviews', y='Rating', data=data)
plt.title('Rating vs. Reviews')
plt.show()
```

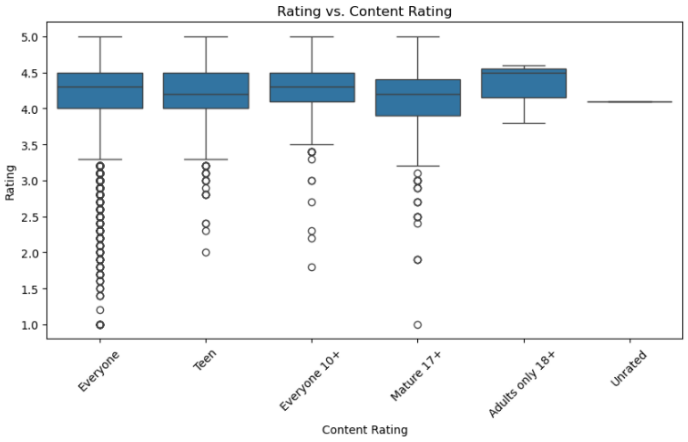


1. Does more review mean a better rating always?

   Yes

4. Make boxplot for Rating vs. Content Rating

```
# Boxplot for Rating vs. Content Rating
plt.figure(figsize=(10, 5))
sns.boxplot(x='Content Rating', y='Rating', data=data)
plt.title('Rating vs. Content Rating')
plt.xticks(rotation=45)
plt.show()
```

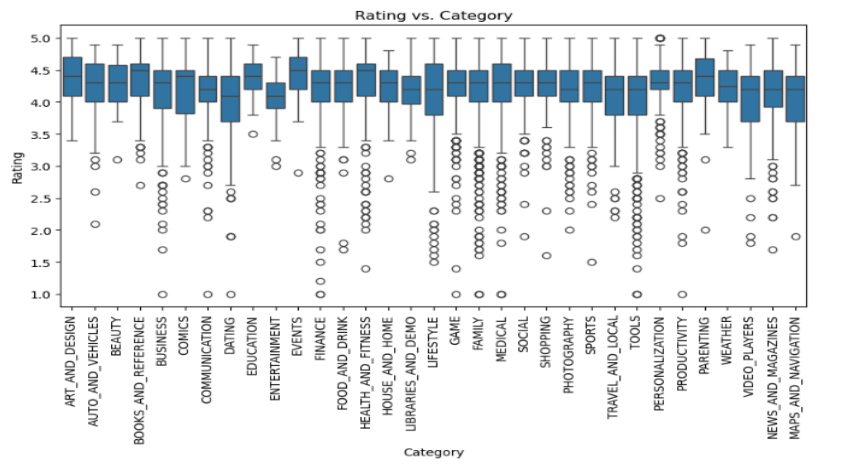


1. Is there any difference in the ratings? Are some types liked better?

   Adults Only 18+ liked better

5. Make boxplot for Ratings vs. Category

```
# Boxplot for Rating vs. Category
plt.figure(figsize=(10, 5))
sns.boxplot(x='Category', y='Rating', data=data)
plt.title('Rating vs. Category')
plt.xticks(rotation=90)
plt.show()
```



1. Which genre has the best ratings?

   Events and Parenting have the best ratings

For each of the plots above, note down your observation.

8. Data preprocessing

For the steps below, create a copy of the dataframe to make all the edits. Name it inp1.

1. Reviews and Install have some values that are still relatively very high. Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.
2. Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.
3. Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data, and all data should be numeric. Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be **inp2**.

```
# Step 8: Data preprocessing
inp1 = data.copy()  # Create a copy for further processing
```

```
# Apply log transformation
inp1['Reviews'] = np.log1p(inp1['Reviews'])
inp1['Installs'] = np.log1p(inp1['Installs'])
```

```
# Drop unnecessary columns
inp1 = inp1.drop(columns=['App', 'Last Updated', 'Current Ver', 'Android Ver'])
```

```
# Get dummy variables
inp2 = pd.get_dummies(inp1, columns=['Category', 'Genres', 'Content Rating'], drop_first=True)
```

9. Train test split and apply 70-30 split. Name the new dataframes df_train and df_test.

```
# Step 9: Train-test split
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(inp2, test_size=0.3, random_state=42)
```

10. Separate the dataframes into X_train, y_train, X_test, and y_test.

```
# Step 10: Separate into features and target
X_train = df_train.drop(columns=['Rating'])
y_train = df_train['Rating']
X_test = df_test.drop(columns=['Rating'])
y_test = df_test['Rating']
```

11 . Model building

- Use linear regression as the technique
- Report the R2 on the train set

```
# Step 11: Model building
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Report R2 on the train set
train_r2 = model.score(X_train, y_train)
print("R2 score on train set:", train_r2)
```

12. Make predictions on test set and report R2.

```
# Step 12: Predictions on test set
test_r2 = model.score(X_test, y_test)
print("R2 score on test set:", test_r2)
```