Write Up

1) Load the data file using pandas-
```
import pandas as pd
import numpy as np
df=pd.read_csv("googleplaystore.csv")
```
2) Check for null values in the data. Get the number of null values for each column-
```
df.isna().sum()
```
3) Drop records with nulls in any of the columns.
```
df.dropna(inplace=True)
```
4) Variables seem to have incorrect type and inconsistent formatting. You need to fix them:
    - Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric and Multiply the value by 1,000, if size is mentioned in Mb

```
df["size"]=df.Size.str.extract('(^\d*)')
df["m/k"]=df["Size"].str[-1]
df["size"].replace(to_replace='',value='0',inplace=True)
df["size"]=df["size"].astype(float)
df["size"]=np.where(df["m/k"]=="M",df["size"]*1000,df["size"]*1)
df.drop(["Size","m/k"],axis=1,inplace=True)
```

    - Installs field is currently stored as string and has values like 1,000,000+. remove '+', ',' from the field, convert it to integer

```
columns=["Installs"]
for column in columns:
  df[column]=df[column].str.replace(r'\W',"")
```

    - Price field is a string and has $ symbol. Remove '$' sign, and convert it to numeric.
```
df["Price"]=df["Price"].str[1:]
df["Price"].replace(to_replace='',value='0',inplace=True)
```
    - Convert it to numeric (int/float).
```
columns=["Rating","Reviews","Installs"]
for column in columns:
   df[column]=df[column].astype(int)
df["Price"]=df["Price"].astype(float)
```
5) Sanity checks:
    - 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.
```
index=df[(df["Rating"]<1) & (df["Rating"]>5)].index
df.drop(index,axis=0,inplace=True)
```
    - Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.
```
index=df[df["Reviews"]>df["Installs"]].index
```

```
df.drop(index,axis=0,inplace=True)
```

- For free apps (type = “Free”), the price should not be >0. Drop any such rows.

```
index=df[(df["Type"]=="Free") & (df["Price"]>0)].index
df.drop(index,axis=0,inplace=True)
```

6) Performing univariate analysis (Boxplot for Price and Review and Histogram for Rating and Size)-

```
import matplotlib.pyplot as plt
fig,axes=plt.subplots(1,4,figsize=(20,5))
axes[1].boxplot(df["Price"])
axes[0].set_title("Price")
axes[1].boxplot(df["Reviews"])
axes[1].set_title("Reviews")
axes[2].hist(df["Rating"])
axes[2].set_title("Rating")
axes[3].hist(df["size"])
axes[3].set_title("Size")
```

7) Outlier treatment:

- A price of $200 for an application on the Play Store is very high and suspicious! Drop these as most seem to be junk apps.

```
index=df[df["Price"]>200].index
df.drop(index,axis=0,inplace=True)
```

- 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.

```
index=df[df["Reviews"]>2000000].index
df.drop(index,axis=0,inplace=True)
```

- Installs: There seems to be some outliers in this field too. Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99. Decide a threshold as cutoff for outlier and drop records having values more than that

```
print("10th percentile=",np.percentile(df["Installs"],10))
print("25th percentile=",np.percentile(df["Installs"],25))
print("50th percentile=",np.percentile(df["Installs"],50))
print("70th percentile=",np.percentile(df["Installs"],70))
print("90th percentile=",np.percentile(df["Installs"],90))
print("99th percentile=",np.percentile(df["Installs"],99))
index=df[df["Installs"]>10000000.0].index
df.drop(index,axis=0,inplace=True)
```

8) Bivariate analysis- scatter plot/joinplot for Rating vs. Price, Rating vs. Reviews, boxplot for Rating vs. Content Rating and Ratings vs. Category

```
fig,axes=plt.subplots(1,3,figsize=(27,9))
axes[0].scatter(df["Rating"],df["Price"])
axes[0].set_title("Rating vs Price")
axes[1].scatter(df["Rating"],df["size"])
axes[1].set_title("Rating vs Size")
```

```
axes[2].scatter(df["Rating"],df["Reviews"])
axes[2].set_title("Rating vs Reviews")
import seaborn as sns
sns.boxplot(x="Rating",y="Content Rating",data=df)
sns.boxplot(x="Rating",y="Category",data=df)
```

9) Data preprocessing-
   - Reviews and Install have some values that are still relatively very high so apply log transformation.
     ```
     inp1=df.copy()
     inp1["Reviews"]=np.log(inp1["Reviews"])
     inp1["Installs"]=np.log(inp1["Installs"])
     ```
   - Drop columns App, Last Updated, Current Ver, and Android Ver.
     ```
     inp1.drop(['App','Last Updated','Current Ver','Android
     Ver'],axis=1,inplace=True)
     ```
   - Get dummy columns for Category, Genres, and Content Rating
     ```
     inp2=pd.get_dummies(inp1,columns=['Category','Content Rating',
     'Genres',"Type"])
     ```
10) Train test split and apply 70-30 split. Separate the dataframes into X_train, y_train, X_test, and y_test.
    ```
    x=inp2.drop(["Rating"],axis=1).values
    y=inp2["Rating"].values
    from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
    ```
11) Model building. Use linear regression as the technique. Report the R2 on the train set.
    ```
    from sklearn.linear_model import LinearRegression
    reg=LinearRegression()
    reg.fit(x_train,y_train)
    from sklearn.metrics import r2_score
    y_pred_train=reg.predict(x_train)
    r_sq_train=r2_score(y_train,y_pred_train)
    print("R square of train set : ",r_sq_train)
    ```
12) Make predictions on test set and report R2.
    ```
    y_pred=reg.predict(x_test)
    r_sq_test=r2_score(y_test,y_pred)
    print("R square of test set : ",r_sq_test)
    y_pred_table=pd.DataFrame({"Actual Rating":y_test,"Predicted Rating":y_pred})
    ```