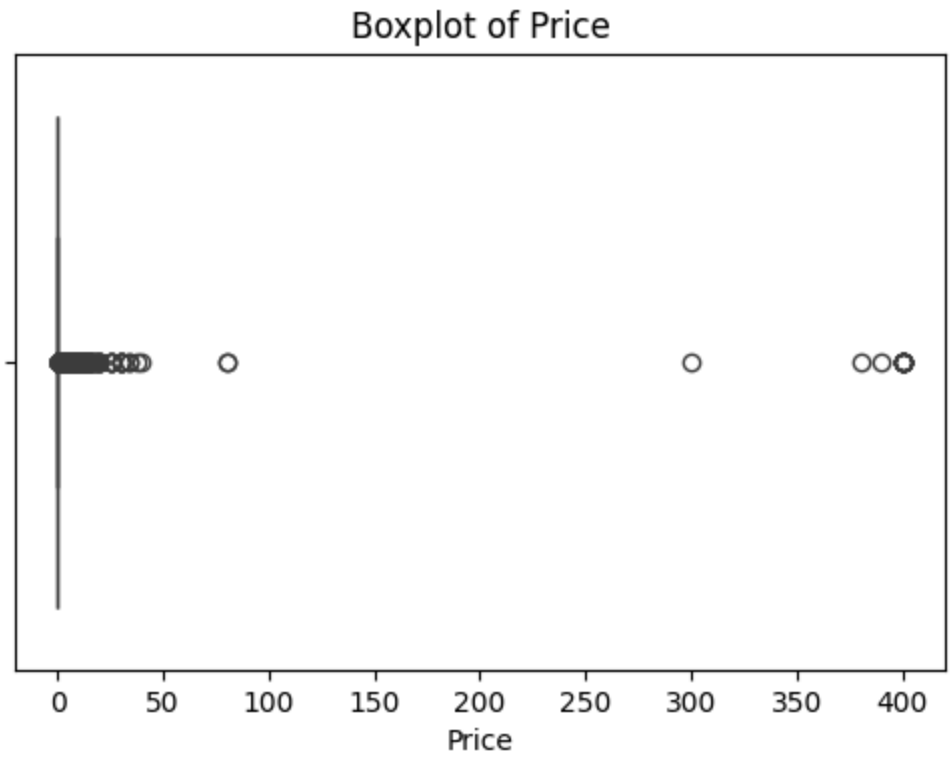


```
# ========================================================
# APP RATING PREDICTION - COMPLETE COURSE END PROJECT
# ========================================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
# Optional: prevent plot blocking in some IDEs
plt.switch_backend('TkAgg')

# ------------------------------------------------------
# 1. Load Dataset
# ------------------------------------------------------

df = pd.read_csv("googleplaystore.csv")
print("Initial Shape:", df.shape)

# ------------------------------------------------------
# 2. Check & Drop Null Values
# ------------------------------------------------------

print("\nNull Values:\n", df.isnull().sum())
df.dropna(inplace=True)
print("After Dropping Nulls:", df.shape)

# ------------------------------------------------------
# 3. Data Cleaning & Formatting
# ------------------------------------------------------

# Remove 'Varies with device'
df = df[df['Size'] != 'Varies with device']

# Fix Size (Convert MB -> KB by multiplying 1000)
def convert_size(size):
```

```
    if 'M' in size:
        return float(size.replace('M', '')) * 1000
    elif 'k' in size:
        return float(size.replace('k', ''))
    else:
        return np.nan

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

# Convert Reviews
df['Reviews'] = pd.to_numeric(df['Reviews'],
errors='coerce')

# Clean Installs
df['Installs'] = df['Installs'].str.replace('[+,]', '',
regex=True)
df['Installs'] = pd.to_numeric(df['Installs'],
errors='coerce')

# Clean Price
df['Price'] = df['Price'].str.replace('$', '',
regex=False)
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

# Remove new nulls
df.dropna(inplace=True)

print("After Cleaning:", df.shape)
```

```
# ----------------------------------------------------
# 4. Sanity Checks
# ----------------------------------------------------

# Rating between 1 and 5
df = df[(df['Rating'] >= 1) & (df['Rating'] <= 5)]

# Reviews <= Installs
df = df[df['Reviews'] <= df['Installs']]

# Free apps should not have price > 0
df = df[~((df['Type'] == 'Free') & (df['Price'] > 0))]

print("After Sanity Checks:", df.shape)

# ----------------------------------------------------
# 5. UNIVARIATE ANALYSIS
# ----------------------------------------------------

plt.figure(figsize=(6,4))
sns.boxplot(x=df['Price'])
plt.title("Boxplot of Price")
plt.show()

plt.figure(figsize=(6,4))
sns.boxplot(x=df['Reviews'])
plt.title("Boxplot of Reviews")
```

```
plt.show()

plt.figure(figsize=(6,4))
sns.histplot(df['Rating'], bins=20)
plt.title("Histogram of Rating")
plt.show()

plt.figure(figsize=(6,4))
sns.histplot(df['Size'], bins=20)
plt.title("Histogram of Size")
plt.show()

# ----------------------------------------------------
# 6. Outlier Treatment
# ----------------------------------------------------

# Remove Price > 200
df = df[df['Price'] <= 200]

# Remove Reviews > 2M
df = df[df['Reviews'] <= 2000000]

# Remove Installs above 99th percentile
install_threshold = df['Installs'].quantile(0.99)
df = df[df['Installs'] <= install_threshold]

print("After Outlier Treatment:", df.shape)
```

```
# ------------------------------------------------------
# 7. BIVARIATE ANALYSIS
# ------------------------------------------------------

sns.scatterplot(x='Price', y='Rating', data=df)
plt.title("Rating vs Price")
plt.show()

sns.scatterplot(x='Size', y='Rating', data=df)
plt.title("Rating vs Size")
plt.show()

sns.scatterplot(x='Reviews', y='Rating', data=df)
plt.title("Rating vs Reviews")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x='Content Rating', y='Rating', data=df)
plt.xticks(rotation=45)
plt.title("Rating vs Content Rating")
plt.show()

plt.figure(figsize=(12,5))
sns.boxplot(x='Category', y='Rating', data=df)
plt.xticks(rotation=90)
plt.title("Rating vs Category")
plt.show()
```

```
# ------------------------------------------------------
# 8. Data Preprocessing
# ------------------------------------------------------

inp1 = df.copy()

# Log transformation
inp1['Reviews'] = np.log1p(inp1['Reviews'])
inp1['Installs'] = np.log1p(inp1['Installs'])

# Drop unnecessary columns
inp1 = inp1.drop(['App','Last Updated','Current
Ver','Android Ver'], axis=1)

# Dummy encoding
inp2 = pd.get_dummies(
   inp1,
   columns=['Category','Genres','Content Rating','Type'],
   drop_first=True
)

# ------------------------------------------------------
# 9. Train-Test Split (70-30)
# ------------------------------------------------------

df_train, df_test = train_test_split(inp2, test_size=0.3,
random_state=42)
```

```
X_train = df_train.drop('Rating', axis=1)
y_train = df_train['Rating']

X_test = df_test.drop('Rating', axis=1)
y_test = df_test['Rating']

# Ensure no object columns
print("\nChecking Data Types:")
print(X_train.dtypes.value_counts())

# ------------------------------------------------------
# 10. Linear Regression Model
# ------------------------------------------------------

model = LinearRegression()
model.fit(X_train, y_train)

# ------------------------------------------------------
# 11. Model Evaluation
# ------------------------------------------------------

train_pred = model.predict(X_train)
test_pred = model.predict(X_test)

print("\n=========== MODEL PERFORMANCE ===========")
print("Train R2 Score:", r2_score(y_train, train_pred))
print("Test R2 Score :", r2_score(y_test, test_pred))
```

OUTPUT:

```
✅ All features are numeric. Training model...

----- MODEL PERFORMANCE -----
Train R2 Score: 0.161268430908607 43
Test R2 Score : 0.11681895751120863

[Done] exited with code=0 in 1.459 seconds

[Running] python -u
"/Users/rsh/Documents/AppRatingPrediction/app_rating_pred
iction.py"
Initial Shape: (10841, 13)

Null Values:
App                  0
Category             0
Rating            1474
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
After Dropping Nulls: (9360, 13)
After Cleaning: (7723, 13)
After Sanity Checks: (7717, 13)
```