B.M.S. COLLEGE OF ENGINEERING

(Autonomous college under VTU)

Bull Temple Rd, Basavanagudi, Bengaluru, Karnataka 560019

2023-2025

Department of Computer Applications

Report is submitted for fulfillment of Lab Task in the subject

"Machine Learning"
(22MCA2PCML)

By

HRUTHIK CHAVAN D
1BM23MC038

Under the Guidance

Prof. K. P. Shailaja
(Assistant Professor)

B. M. S. COLLEGE OF ENGINEERING, BANGALORE – 19

(Autonomous Institute, Affiliated to VTU)

Department of Computer Applications

(Accredited by NBA for 5 years 2019-2024)

LABORATORY CERTIFICATE

This is to certify that **HRUTHIK CHAVAN D(1BM23MC038)** has satisfactorily completed the course of practical in **“Machine Learning - 22MCA2PCML”** Laboratory prescribed by B.M.S. College of Engineering (Autonomous College under VTU) 2nd Semester MCA Course in this college during the year 2023-2024.

Signature of Batch Incharge — Signature of HoD

Prof. K. P. Shailaja — Dr. Ch. Ram Mohan Reddy

Examiner:

CONTENTS

1. Consider the Smart Phone dataset and perform exploratory data analysis
i. Identify the dimension, structure, and summary of the data set

```
#1. Consider the Smart Phone dataset and perform exploratory data analysis.
#i. Identify the dimension, structure, and summary of the data set
import pandas as pd

# Load the dataset into a Pandas DataFrame
df = pd.read_csv("D:\ML\smartphones_cleaned_v6.csv")

# 1. Dimension of the Dataset
num_rows, num_cols = df.shape
print(f"The dataset has {num_rows} rows and {num_cols} columns.")

# 2. Structure of the Dataset
print("\nColumn Names and Data Types:")
print(df.dtypes)

# 3. Summary of the Dataset
print("\nSummary Statistics:")
df.describe()
```

```
Column Names and Data Types:
brand_name                    object
model                         object
price                          int64
rating                       float64
has_5g                          bool
has_nfc                         bool
has_ir_blaster                  bool
processor_brand               object
num_cores                    float64
processor_speed              float64
battery_capacity             float64
fast_charging_available        int64
fast_charging                float64
ram_capacity                 float64
internal_memory              float64
screen_size                  float64
refresh_rate                   int64
num_rear_cameras               int64
num_front_cameras            float64
os                            object
primary_camera_rear          float64
primary_camera_front         float64
extended_memory_available      int64
extended_upto                float64
resolution_width               int64
resolution_height              int64
dtype: object

Summary Statistics:
```

:5]:

	price	rating	num_cores	processor_speed	battery_capacity	fast_charging_available	fast_charging
count	980.000000	879.000000	974.000000	938.000000	969.000000	980.000000	769.000000
mean	32520.504082	78.258248	7.772074	2.427217	4817.748194	0.854082	46.126138
std	39531.812669	7.402854	0.836845	0.464090	1009.540054	0.353205	34.277870
min	3499.000000	60.000000	4.000000	1.200000	1821.000000	0.000000	10.000000
25%	12999.000000	74.000000	8.000000	2.050000	4500.000000	1.000000	18.000000
50%	19994.500000	80.000000	8.000000	2.300000	5000.000000	1.000000	33.000000
75%	35491.500000	84.000000	8.000000	2.840000	5000.000000	1.000000	66.000000
max	650000.000000	89.000000	8.000000	3.220000	22000.000000	1.000000	240.000000

ii. Pre-process the dataset and treat them (like missing values, 'na'?). Justify the treatment

```
# Check for missing values

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

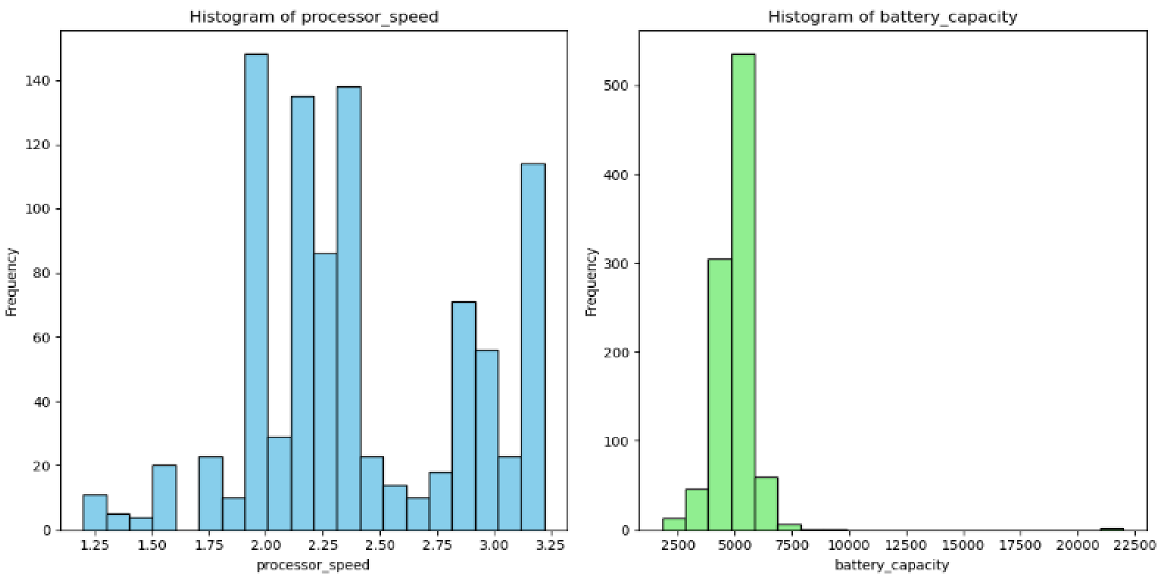
Output :

```
brand_name                  0
model                       0
price                       0
rating                    101
has_5g                      0
has_nfc                     0
has_ir_blaster              0
processor_brand            20
num_cores                   6
processor_speed            42
battery_capacity           11
fast_charging_available     0
fast_charging             211
ram_capacity                0
internal_memory             0
screen_size                 0
refresh_rate                0
resolution                  0
num_rear_cameras            0
num_front_cameras           4
os                         14
primary_camera_rear         0
primary_camera_front        5
extended_memory_available   0
```

iii. Plot the histogram for continuous variables (at least two) to analyse the data.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame with the Smart Phone data
df = pd.read_csv("D:\ML\smartphones_cleaned_v6.csv")
# Select two continuous variables for plotting histograms
```

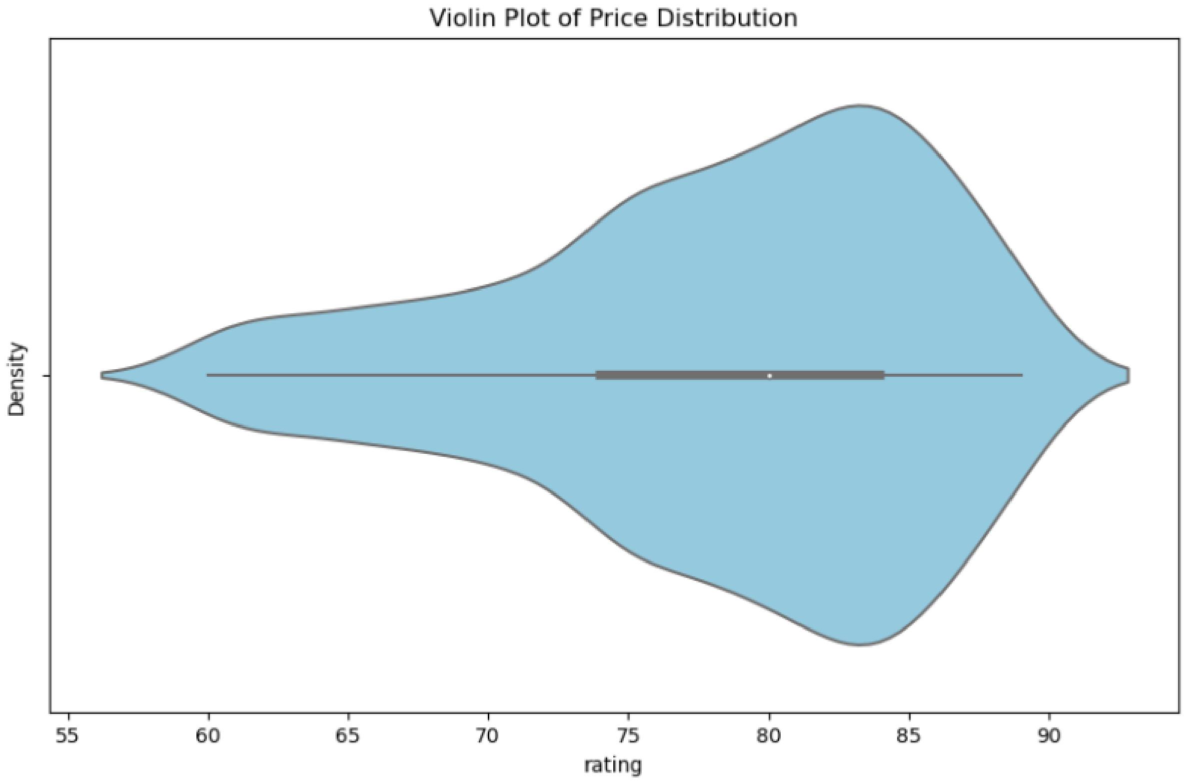
```
continuous_vars = ['processor_speed', 'battery_capacity']

# Plot histograms using Matplotlib
plt.figure(figsize=(12, 6))

# Plot histogram for Price
plt.subplot(1, 2, 1)
plt.hist(df['processor_speed'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of processor_speed')
plt.xlabel('processor_speed')
plt.ylabel('Frequency')

# Plot histogram for RAM
plt.subplot(1, 2, 2)
plt.hist(df['battery_capacity'], bins=20, color='lightgreen', edgecolor='black')
plt.title('Histogram of battery_capacity')
plt.xlabel('battery_capacity')
plt.ylabel('Frequency')

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



iv. Draw a violin plot do describe the distribution of a numerical variable to analyse the data

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

# Assuming df is your DataFrame with the Smart Phone data
df = pd.read_csv("D:\ML\smartphones_cleaned_v6.csv")
# Plotting a violin plot for the 'Price' variable
plt.figure(figsize=(10, 6))
sns.violinplot(x='rating', data=df, color='skyblue')
plt.title('Violin Plot of Price Distribution')
plt.xlabel('rating')
```

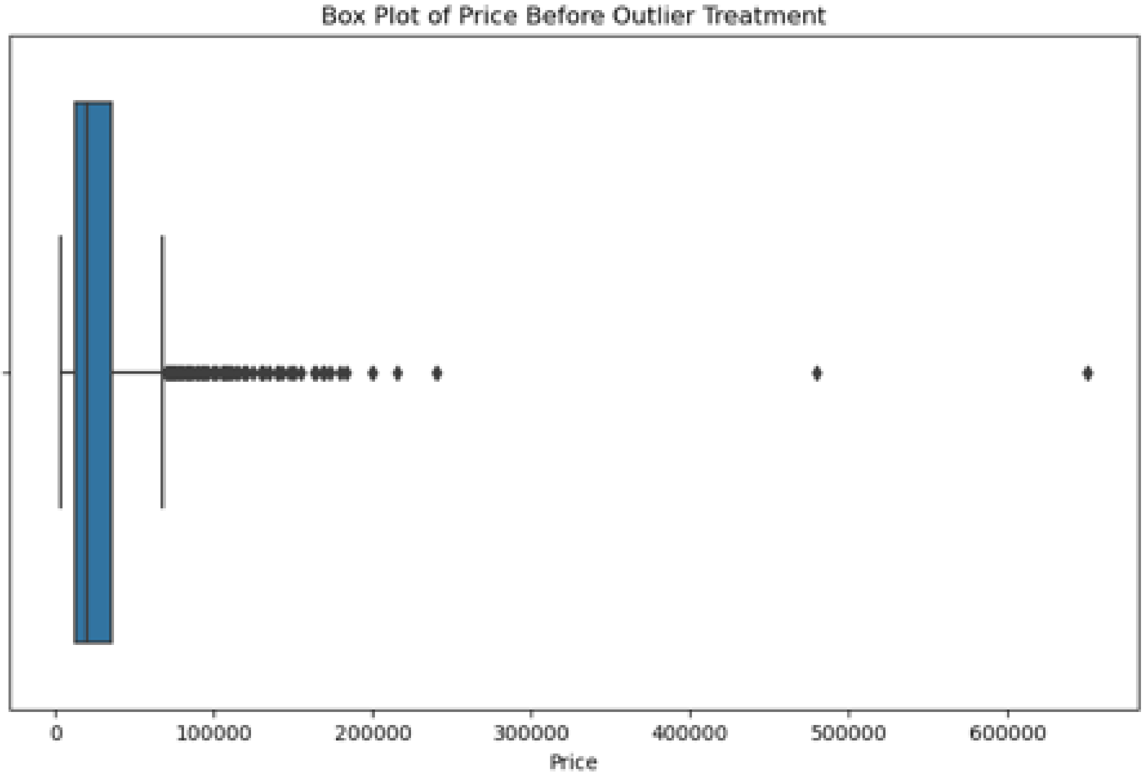
```
plt.ylabel('Density')
plt.show()
```

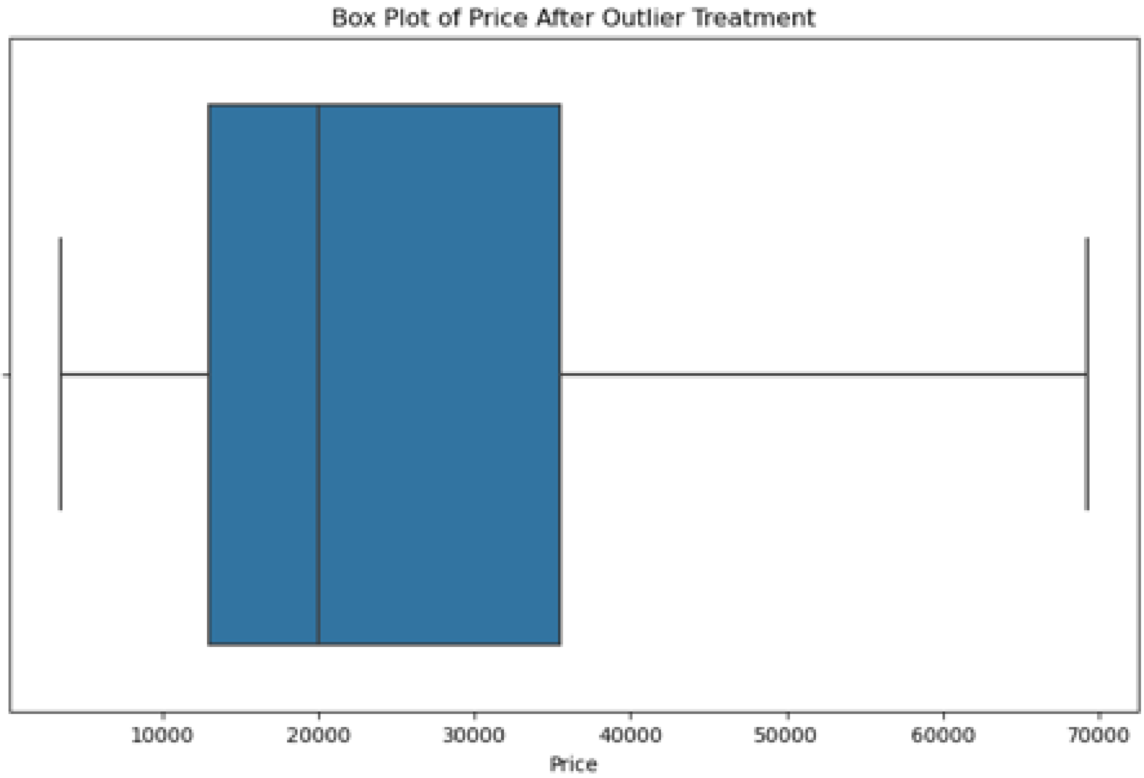


v. Recognize the outliers using box plot (Display the box plot before and after outlier treatment)

```
import numpy as np

variable = 'price'

plt.figure(figsize=(10, 6))
sns.boxplot(x=df[variable])
plt.title('Box Plot of Price Before Outlier Treatment')
plt.xlabel('Price')
plt.show()

Q1 = df[variable].quantile(0.25)
Q3 = df[variable].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df[variable] = np.where(df[variable] < lower_bound, lower_bound, df[variable])
df[variable] = np.where(df[variable] > upper_bound, upper_bound, df[variable])

plt.figure(figsize=(10, 6))
```

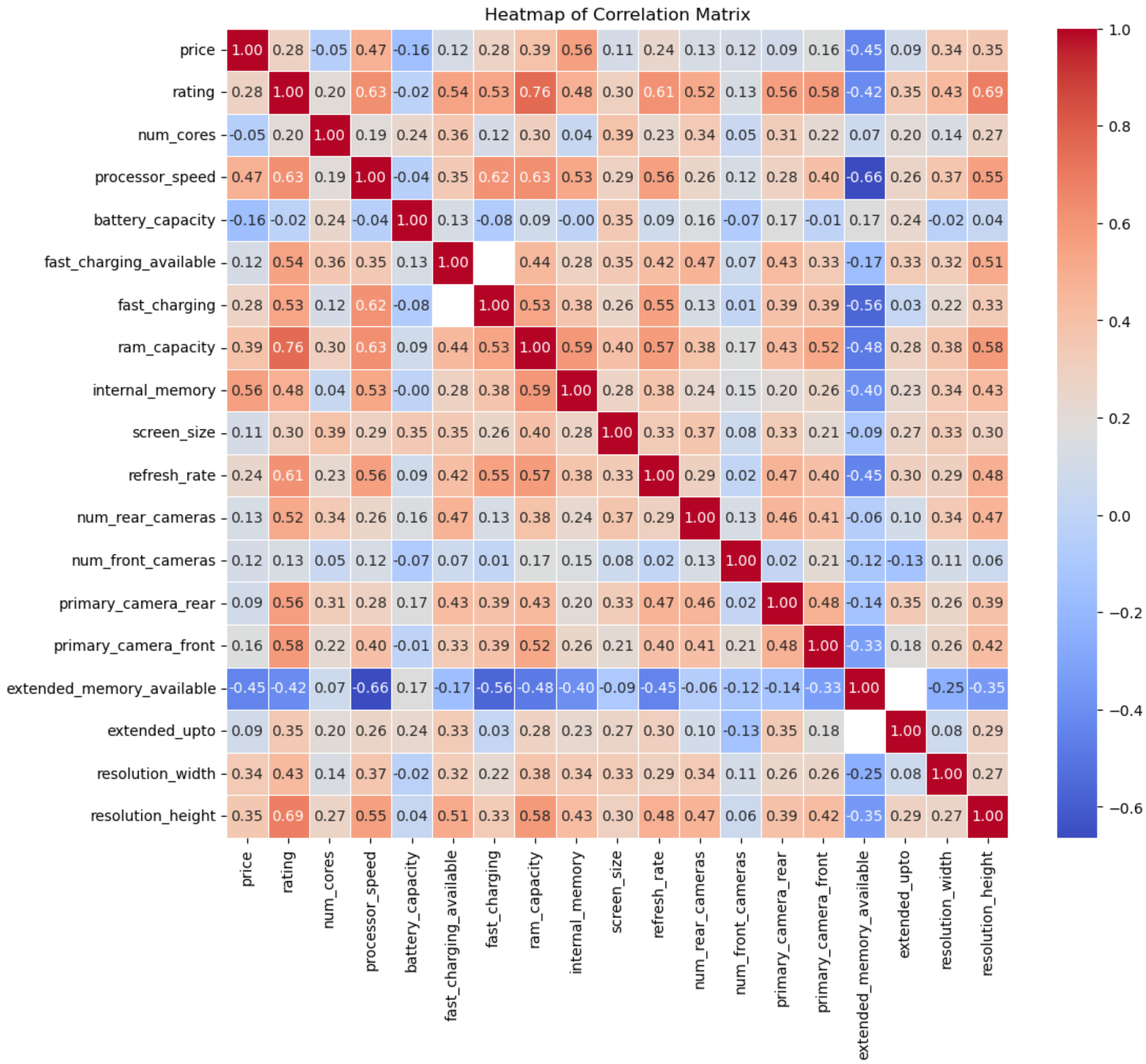
```
sns.boxplot(x=df[variable])
plt.title('Box Plot of Price After Outlier Treatment')
plt.xlabel('Price')
plt.show()
```





vi. Display a heat map to display the relationship among the attributes

```
file_path = 'D:/BMSCE/2nd sem/Machine Learning/ML
    Lab/archive/smartphones_cleaned_v6.csv'
df = pd.read_csv(file_path)

numerical_df = df.select_dtypes(include=['float64', 'int64'])

correlation_matrix = numerical_df.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
    linewidths=0.5)
plt.title('Heatmap of Correlation Matrix')
    plt.show()
```



vii. Standardize the continuous variable (if any)

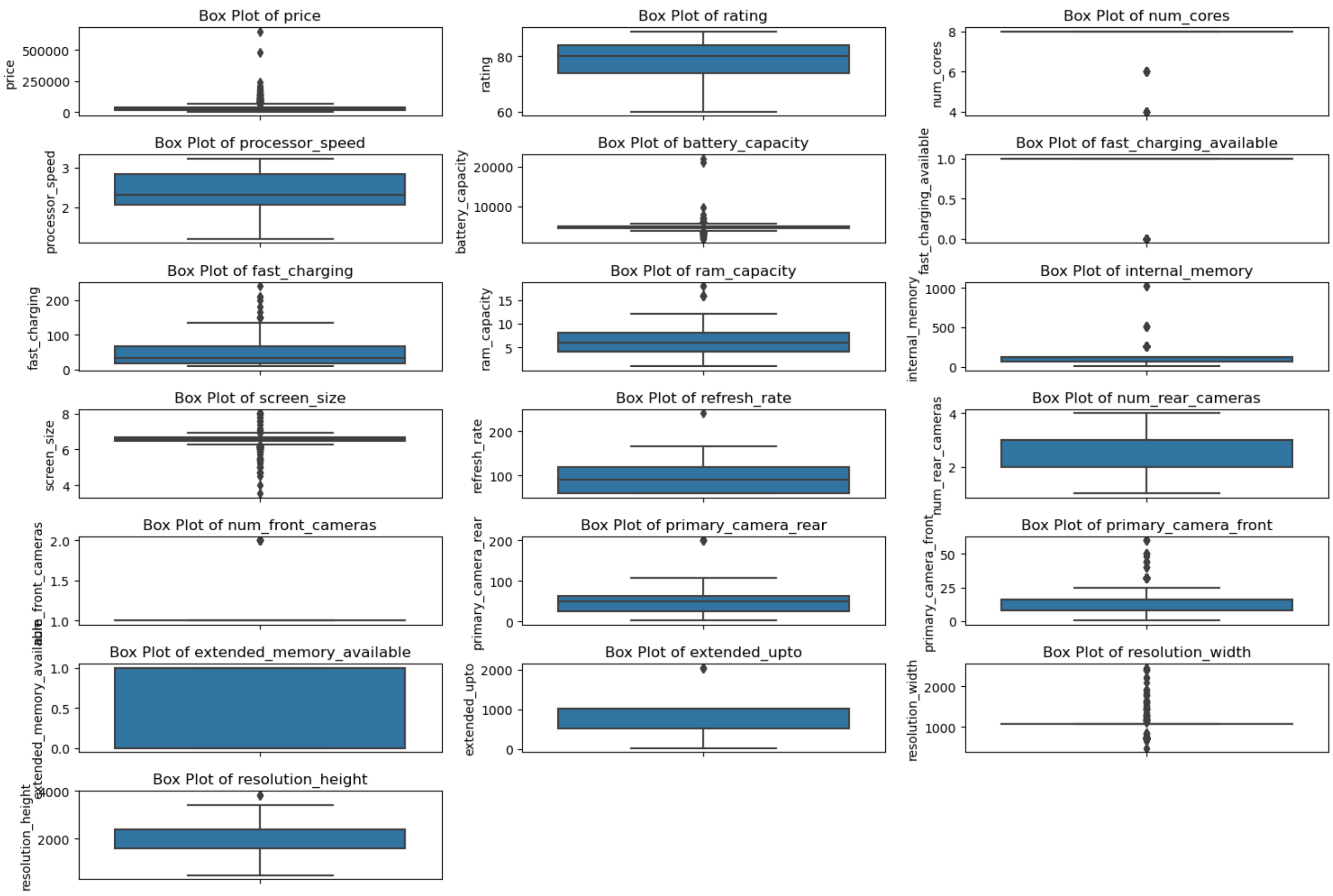
```
from sklearn.preprocessing import StandardScaler
numerical_df = df.select_dtypes(include=['float64', 'int64'])
scaler = StandardScaler()
standardized_values = scaler.fit_transform(numerical_df)
standardized_df = pd.DataFrame(standardized_values, columns=numerical_df.columns)
for col in numerical_df.columns:
    df[col] = standardized_df[col]
print(df.head())
```

```
  brand_name                      model     price    rating  has_5g  has_nfc  \
0    oneplus              OnePlus 11 5G  1.400820  1.532998    True     True
1    oneplus  OnePlus Nord CE 2 Lite 5G -0.388606  0.391286    True    False
2    samsung       Samsung Galaxy A14 5G -0.566987 -0.464998    True    False
3   motorola        Motorola Moto G62 5G -0.643655  0.391286    True    False
4     realme          Realme 10 Pro Plus -0.132536  0.534000    True    False

   has_ir_blaster processor_brand  num_cores  processor_speed  ...  \
0           False      snapdragon   0.273342         1.702938  ...
1           False      snapdragon   0.273342        -0.500706  ...
2           False          exynos   0.273342        -0.059978  ...
3           False      snapdragon   0.273342        -0.500706  ...
4           False       dimensity   0.273342         0.380751  ...

   refresh_rate  num_rear_cameras  num_front_cameras       os  \
0      0.957568          0.239309          -0.175353  android
1      0.957568          0.239309          -0.175353  android
2     -0.077869          0.239309          -0.175353  android
3      0.957568          0.239309          -0.175353  android
4      0.957568          0.239309          -0.175353  android

   primary_camera_rear  primary_camera_front  extended_memory_available  \
0            -0.009680             -0.054330                  -1.306592
1             0.414767             -0.054330                   0.765350
2            -0.009680             -0.330995                   0.765350
3            -0.009680             -0.054330                   0.765350
4             1.748742             -0.054330                  -1.306592

   extended_upto  resolution_width resolution_height
0       0.000000          1.255610          1.939746
1       1.099808          0.014302          0.382272
2       1.099808          0.014302          0.374523
3       1.099808          0.014302          0.359026
4       0.000000          0.014302          0.382272
```

2. For the data set in Q1,

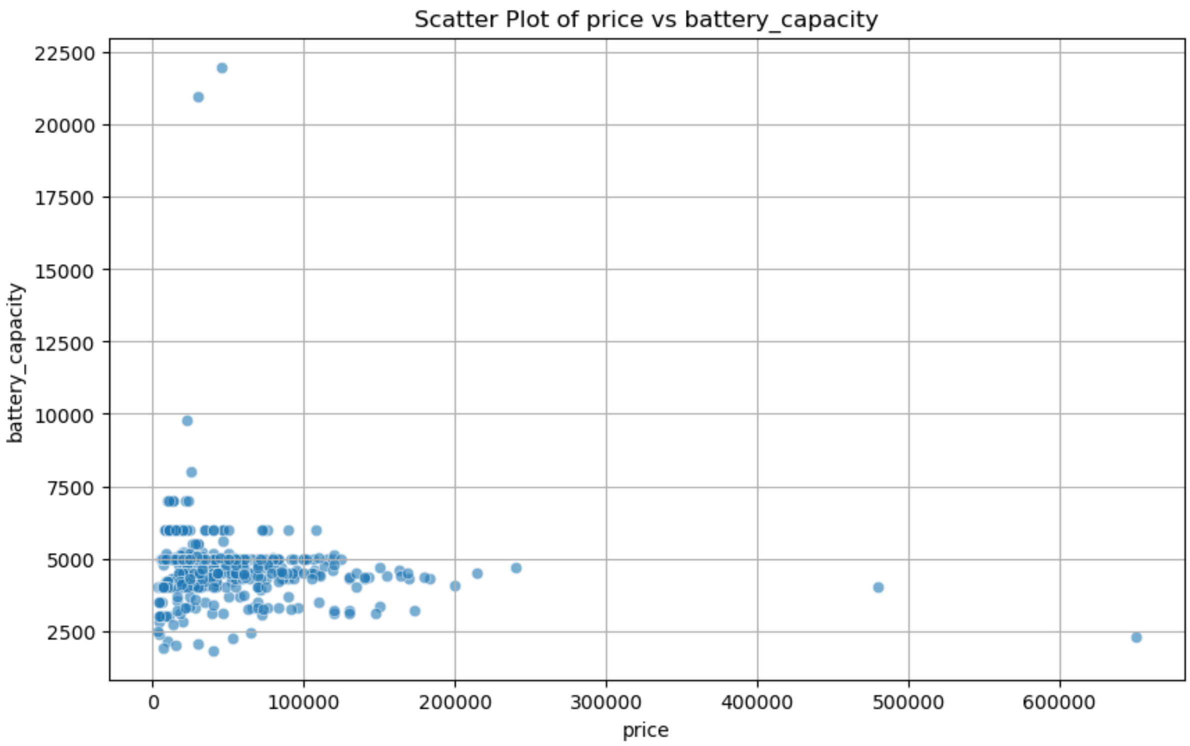
i. Show the distribution of continuous variables using Box Plot

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

file_path = 'D:/BMSCE/2nd sem/Machine Learning/ML
    Lab/archive/smartphones_cleaned_v6.csv'  # Update this path to your dataset location
df = pd.read_csv(file_path)

numerical_df = df.select_dtypes(include=['float64', 'int64'])

plt.figure(figsize=(15, 10))

for i, col in enumerate(numerical_df.columns):
    plt.subplot(len(numerical_df.columns) // 3 + 1, 3, i + 1)
    sns.boxplot(y=df[col])
    plt.title(f'Box Plot of {col}')

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



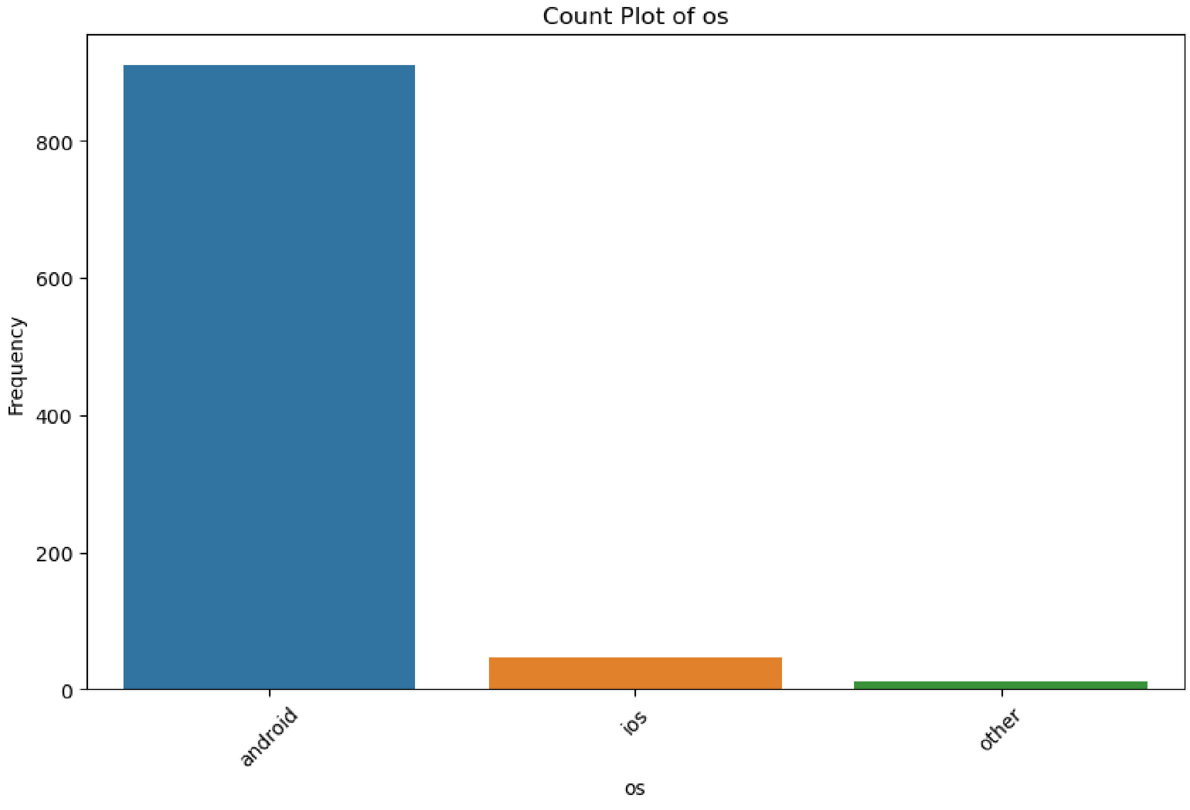
ii. Identify the relationship between two continuous variables using scatter plot

```
import pandas as pd
import matplotlib.pyplot as plt

file_path = 'D:/BMSCE/2nd sem/Machine Learning/ML
    Lab/archive/smartphones_cleaned_v6.csv'  # Update this path to your dataset location
df = pd.read_csv(file_path)

x_var = 'price'  # Replace with your chosen variable
y_var = 'battery_capacity'  # Replace with your chosen variable

plt.figure(figsize=(10, 6))
plt.scatter(df[x_var], df[y_var], alpha=0.6, edgecolors='w', linewidth=0.5)
plt.title(f'Scatter Plot of {x_var} vs {y_var}')
plt.xlabel(x_var)
plt.ylabel(y_var)
plt.grid(True)

plt.show()
```



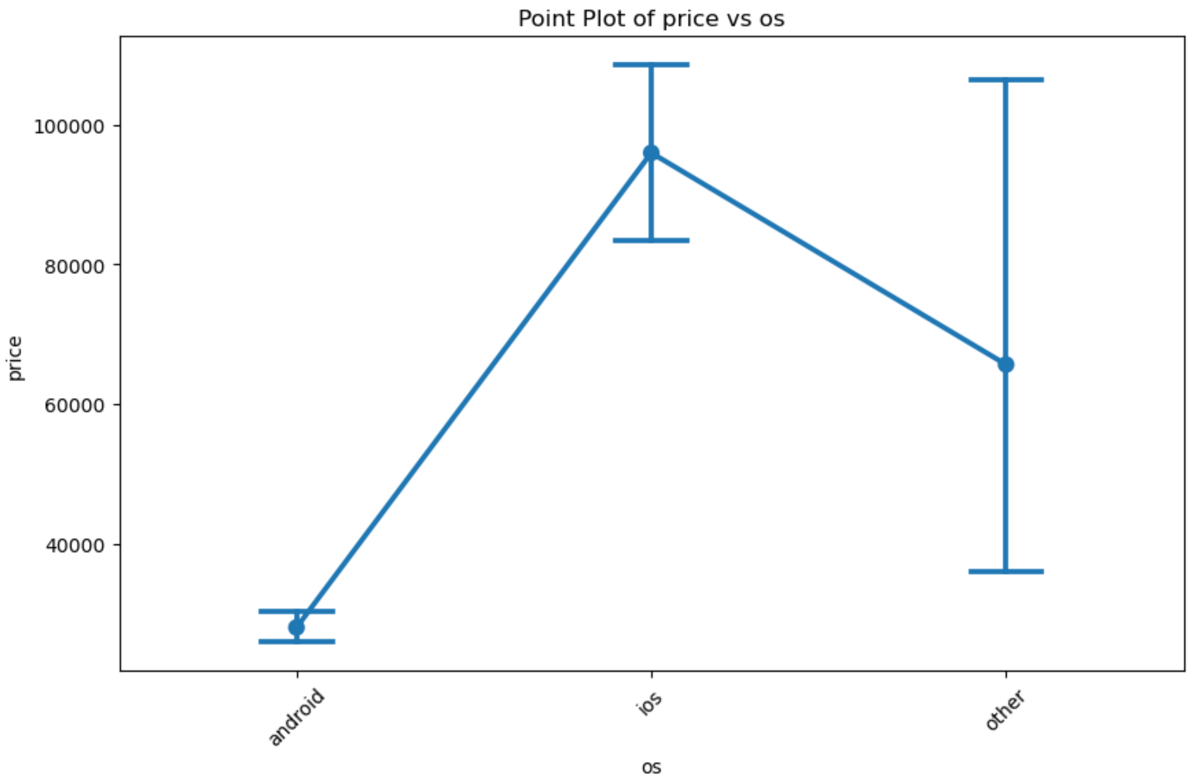
iii. Find and display the frequency of the categorical values using count plot

```
file_path = 'D:/BMSCE/2nd sem/Machine Learning/ML
    Lab/archive/smartphones_cleaned_v6.csv'  # Update this path to your dataset location
df = pd.read_csv(file_path)

categorical_var = 'os'  # Replace with your chosen categorical variable

plt.figure(figsize=(10, 6))
sns.countplot(x=df[categorical_var], order=df[categorical_var].value_counts().index)
plt.title(f'Count Plot of {categorical_var}')
plt.xlabel(categorical_var)
plt.ylabel('Frequency')
plt.xticks(rotation=45)  # Rotate labels if necessary for better readability
plt.show()
```



iv. Apply point plots to display one continuous and one categorical variable

```
file_path = 'D:/BMSCE/2nd sem/Machine Learning/ML
    Lab/archive/smartphones_cleaned_v6.csv'  # Update this path to your dataset
    location
df = pd.read_csv(file_path)

continuous_var = 'price'  # Replace with your chosen continuous variable
categorical_var = 'os'    # Replace with your chosen categorical variable

plt.figure(figsize=(10, 6))
sns.pointplot(x=categorical_var, y=continuous_var, data=df, capsize=0.2,
    markers='o', linestyles='-')
plt.title(f'Point Plot of {continuous_var} vs {categorical_var}')
plt.xlabel(categorical_var)
plt.ylabel(continuous_var)
plt.xticks(rotation=45)  # Rotate labels if necessary for better readability
plt.show()
```



3. For the Market-Basket dataset, apply Apriori algorithm and identify the best rules based on support and confidence.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df=pd.read_csv("basket.csv")
all_items=list(item for sublist in df.values.tolist() for item in sublist if pd.notna(item))
unique_items = list(set(all_items))[:200]
df_subset = df.head(200)
encoded_df=pd.DataFrame(0,index=range(len(df)),columns=unique_items)
for index,transaction in df_subset.iterrows():
    for item in transaction.dropna():
        encoded_df.loc[index,item]=1
min_support=0.3
min_confident=0.7
# Step 4: Apply the Apriori algorithm to find frequent itemsets
frequent_items = apriori(encoded_df, min_support=min_support, use_colnames=True)

# Step 5: Generate association rules based on the frequent itemsets
rules = association_rules(frequent_items, metric="confidence",
min_threshold=min_confident)

# Output the frequent itemsets and association rules
print("Frequent Itemsets:")
print(frequent_items)

print("\nAssociation Rules:")
print(rules)
```

sample dataset:

bread	milk	cookie	eggs	
bread	milk	cookie	soup	
bread	milk	cookie		
turkey	eggs			
eggs	cookies			
milk	diaper	bread		
bread	diaper			
bread	milk	cookie	avocado	
bread	milk	cookie		
bread	milk	cookie	eggs	

Output

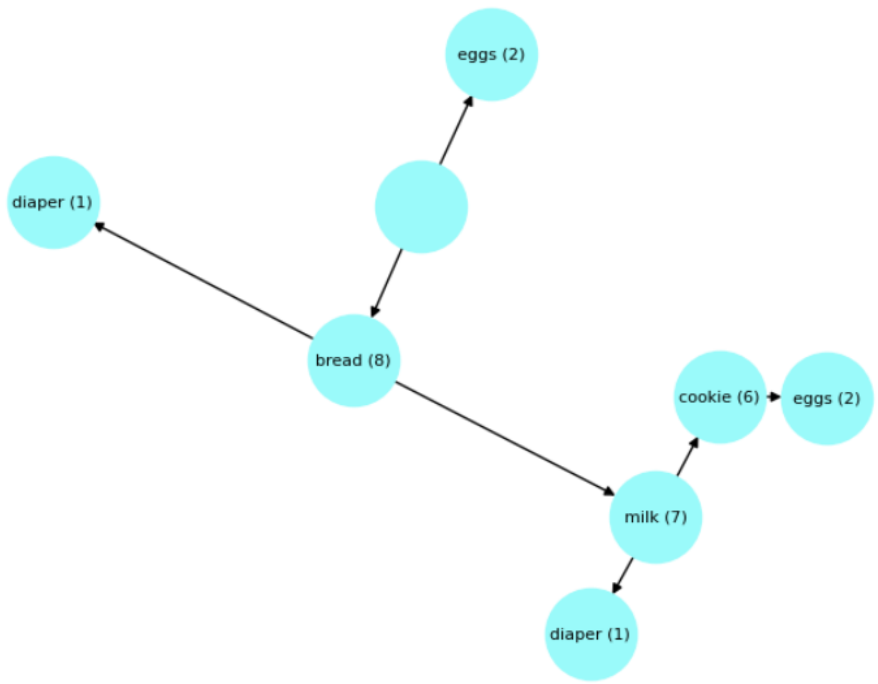
```
Frequent Itemsets:
    support                itemsets
0  0.500000                 (bread)
1  0.363636                (cookie)
2  0.454545                  (milk)
3  0.409091                  (eggs)
4  0.363636         (bread, cookie)
5  0.409091           (bread, milk)
6  0.363636          (milk, cookie)
7  0.363636   (bread, milk, cookie)

Association Rules:
        antecedents      consequents  antecedent support  consequent support
0           (bread)         (cookie)            0.500000            0.363636
1          (cookie)          (bread)            0.363636            0.500000
2           (bread)           (milk)            0.500000            0.454545
3            (milk)          (bread)            0.454545            0.500000
4            (milk)         (cookie)            0.454545            0.363636
5          (cookie)           (milk)            0.363636            0.454545
6     (bread, milk)         (cookie)            0.409091            0.363636
7   (bread, cookie)           (milk)            0.363636            0.454545
8    (milk, cookie)          (bread)            0.363636            0.500000
9           (bread)   (milk, cookie)            0.500000            0.363636
10           (milk)  (bread, cookie)            0.454545            0.363636
11         (cookie)    (bread, milk)            0.363636            0.409091
```

4. For the data set given in Q3, apply FP-tree algorithm, show the tree construction and identify the best rules based on support and confidence.

```
import pandas as pd
from mlxtend.frequent_patterns import fpgrowth

# Example dataset
data = [
    ['bread', 'milk', 'cookie', 'eggs'],
    ['bread', 'milk', 'cookie', 'soup'],
    ['bread', 'milk', 'cookie'],
    ['turkey', 'eggs'],
    ['eggs', 'cookies'],
    ['milk', 'diaper', 'bread'],
    ['bread', 'diaper'],
    ['bread', 'milk', 'cookie', 'avocado'],
    ['bread', 'milk', 'cookie'],
    ['bread', 'milk', 'cookie', 'eggs']
]

# Create a DataFrame with one-hot encoding
df = pd.DataFrame(data)
df = df.stack().reset_index().pivot_table(index='level_0', columns=0, aggfunc=lambda x: 1,
fill_value=0)

# Apply the FP-growth algorithm
frequent_itemsets = fpgrowth(df, min_support=0.2, use_colnames=True)

# Display frequent itemsets
print(frequent_itemsets)
```

	support	itemsets
0	0.8	((level_1, bread))
1	0.7	((level_1, milk))
2	0.6	((level_1, cookie))
3	0.4	((level_1, eggs))
4	0.2	((level_1, diaper))
5	0.7	((level_1, bread), (level_1, milk))
6	0.6	((level_1, cookie), (level_1, milk))
7	0.6	((level_1, cookie), (level_1, bread))
8	0.6	((level_1, cookie), (level_1, bread), (level_1...
9	0.2	((level_1, cookie), (level_1, eggs))
10	0.2	((level_1, eggs), (level_1, milk))
11	0.2	((level_1, eggs), (level_1, bread))
12	0.2	((level_1, cookie), (level_1, eggs), (level_1,...
13	0.2	((level_1, cookie), (level_1, eggs), (level_1,...
14	0.2	((level_1, eggs), (level_1, bread), (level_1, ...
15	0.2	((level_1, cookie), (level_1, eggs), (level_1,...
16	0.2	((level_1, bread), (level_1, diaper))

FP-tree structure:



5. For the Mall-Customer data set, implement K-means clustering algorithm and visualize the clusters.

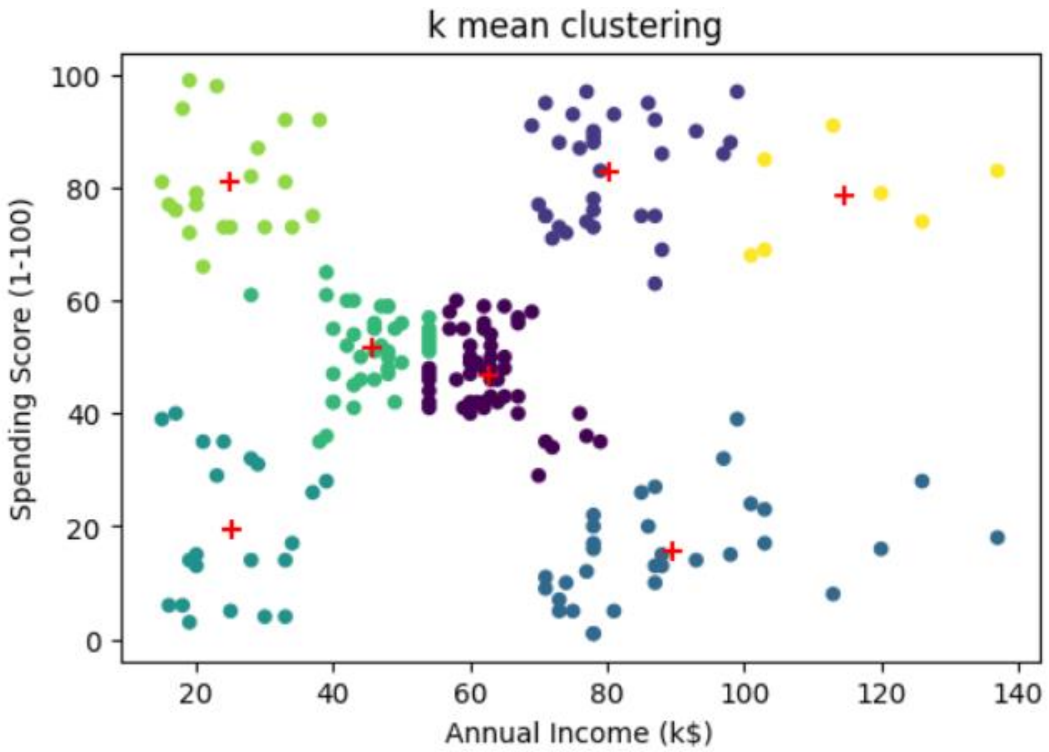
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv("Mall_Customers.csv")
#data.sample(5)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
22	23	Female	46	25	5
100	101	Female	23	62	41
36	37	Female	42	34	17
53	54	Male	59	43	60
191	192	Female	32	103	69

```
X=data[['Annual Income (k$)','Spending Score (1-100)']]
kmean=KMeans(n_clusters=7,random_state=0)
y_kmeans=kmean.fit_predict(X)
data['Cluster']=y_kmeans
#data.head(5)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	Male	19	15	39	3
1	2	Male	21	15	81	5
2	3	Female	20	16	6	3
3	4	Female	23	16	77	5
4	5	Female	31	17	40	3

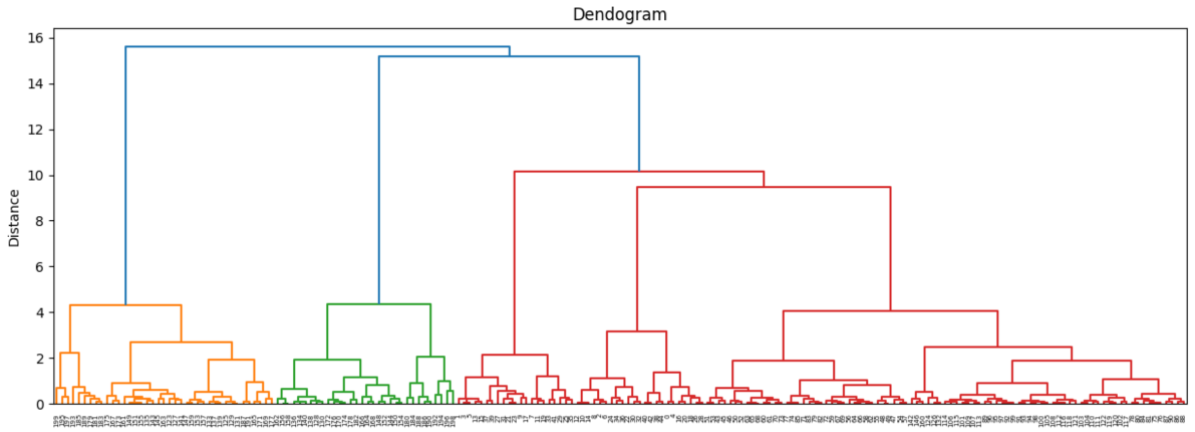
```
plt.figure(figsize=(8,6))
plt.scatter(X.iloc[:,0],X.iloc[:,1],c=y_kmeans,s=50)
centroids = kmean.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, marker='+', label='Centroids')
plt.title('k mean clustering ')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```


k mean clustering
Spending Score (1-100)
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income (k$)

6. For the Groceries dataset implement the Agglomerative clustering algorithm and visualize the clusters.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram,linkage
featues=data[['Annual Income (k$)','Spending Score (1-100)']]
#scalling the feature
scaler=StandardScaler()
scaled_feature=scaler.fit_transform(featues)
# Apply agglomerative clustering
agg_clustering=AgglomerativeClustering(n_clusters=5)
data['Cluster']=agg_clustering.fit_predict(scaled_feature)

linkage_matrix=linkage(scaled_feature,method='ward')
plt.figure(figsize=(20,17))
dendrogram(linkage_matrix)
plt.title('Dendogram')
plt.xlabel('levels')
plt.ylabel('Distance')
plt.show()
```

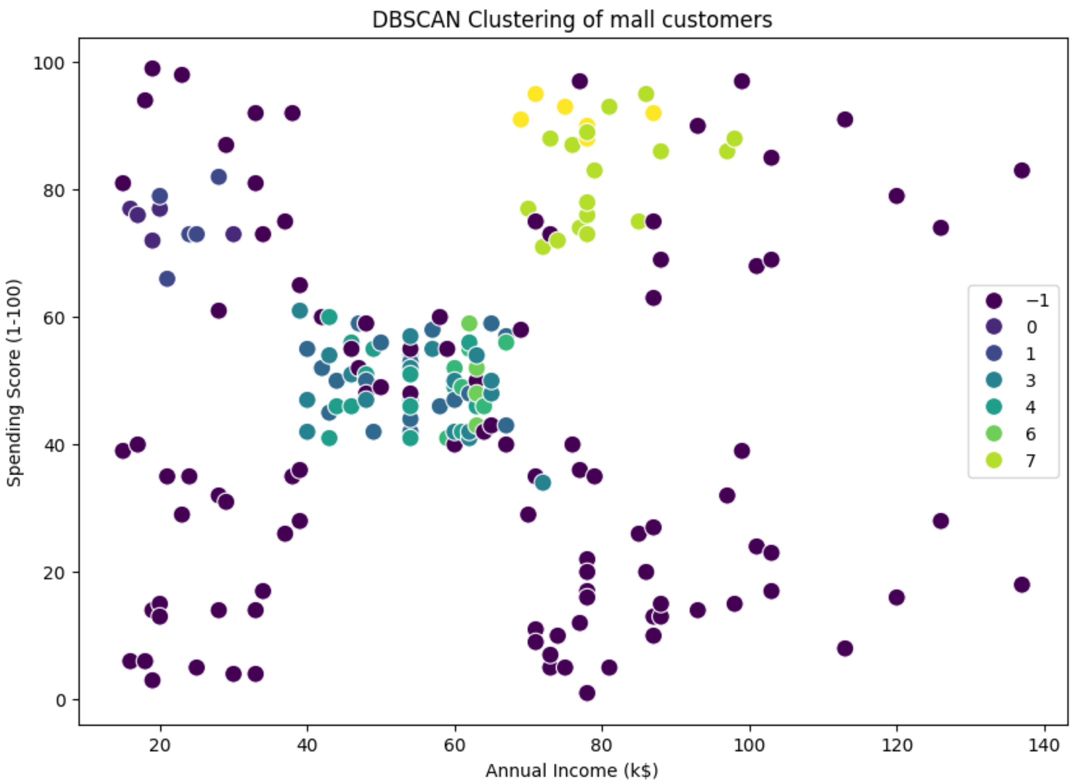
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
5	6	Female	22	17	76	3
11	12	Female	35	19	99	3
150	151	Male	43	78	17	0
67	68	Female	68	48	48	2
155	156	Female	27	78	89	1



7. For the Mall_Customers implement DBScan clustering algorithm and visualize the clusters.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
data=pd.read_csv('Mall_Customers.csv')

data['Gender']=data['Gender'].map({'Male':0,'Female':1})
data.sample(5)

features=data[['Gender','Age','Annual Income (k$)','Spending Score (1-100)']]

#standardizing the features to enure all variables are in same scallable
scalar=StandardScaler()
scaled_features=scalar.fit_transform(features)

#Applay db sacan clustering
dbscan=DBSCAN(eps=0.5,min_samples=5)
clusters=dbscan.fit_predict(scaled_features)

data['Cluster']=clusters
marks=['o','s','D','^','P','*']
plt.figure(figsize=(10,7))
sns.scatterplot(data=data,x="Annual Income (k$)",y="Spending Score (1-
100)",hue="Cluster",palette='viridis',s=100)
plt.title("DBSCAN Clustering of mall customers")
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

```
Cluster
-1    105
 3     18
 2     18
 7     17
 4     15
 5      7
 8      6
 0      5
 1      5
 6      4
Name: count, dtype: int64
```

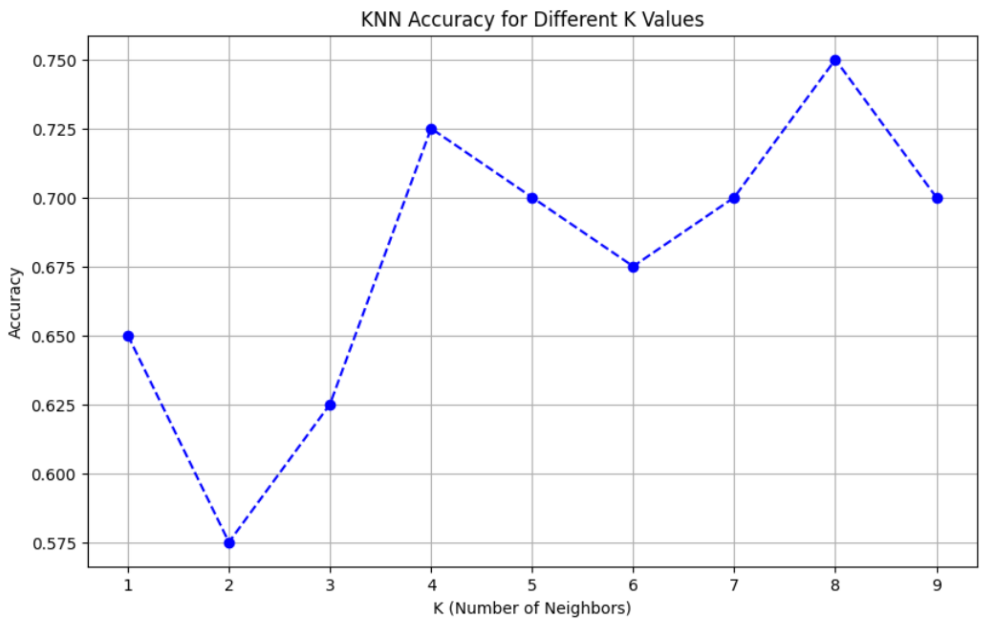


8. Implement KNN Classification algorithm on the Mall Customers. Analyse the model using different K values and display the performance of the model.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report,accuracy_score

data=pd.read_csv('Mall_Customers.csv')
data['Gender']=data['Gender'].map({'Male':0,'Female':1})

# Define the target variable (e.g., categorize Spending Score into low (0) and high (1))
# Assuming scores <= 50 are "low spenders" and > 50 are "high spenders"
data['Spending_Category'] = data['Spending Score (1-100)'].apply(lambda x: 1 if x > 50 else 
0)

X=data[['Gender','Age','Annual Income (k$)']]
y=data['Spending_Category']

#Standardize the feature
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2,random_state=42)

# Define the range of K values to tX_test
k_values=range(1,10)
accuracy_scores=[]

for k in k_values:
    # Initialize the KNN classifire
    knn=KNeighborsClassifier(n_neighbors=k)
    # Fit the data model on training data
    knn.fit(X_train,y_train)
    # predict on the testing data
    y_pred=knn.predict(X_test)
    accuracy=accuracy_score(y_test,y_pred)
    accuracy_scores.append(accuracy)
    print(f"\nK={k} Classification Report: ")
    print(classification_report(y_test,y_pred))

print("\nAccuracy scores for different K values:")
# Display the accuracy scores for different K values
for k in k_values:
    print(f"K={k}: {accuracy:.4f}")
```

```
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracy_scores, marker='o', linestyle='--', color='b')
plt.title('KNN Accuracy for Different K Values')
plt.xlabel('K (Number of Neighbors)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```

```
K=1 Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.70      0.70        23
           1       0.59      0.59      0.59        17

    accuracy                           0.65        40
   macro avg       0.64      0.64      0.64        40
weighted avg       0.65      0.65      0.65        40


K=2 Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.78      0.68        23
           1       0.50      0.29      0.37        17

    accuracy                           0.57        40
   macro avg       0.55      0.54      0.52        40
weighted avg       0.56      0.57      0.55        40
```




```
# Example custom input for a new customer: Female, 30 years old, Annual Income 70k
custom_data = pd.DataFrame([[1, 20, 20]],columns= ['Gender', 'Age', 'Annual Income (k$)']
```

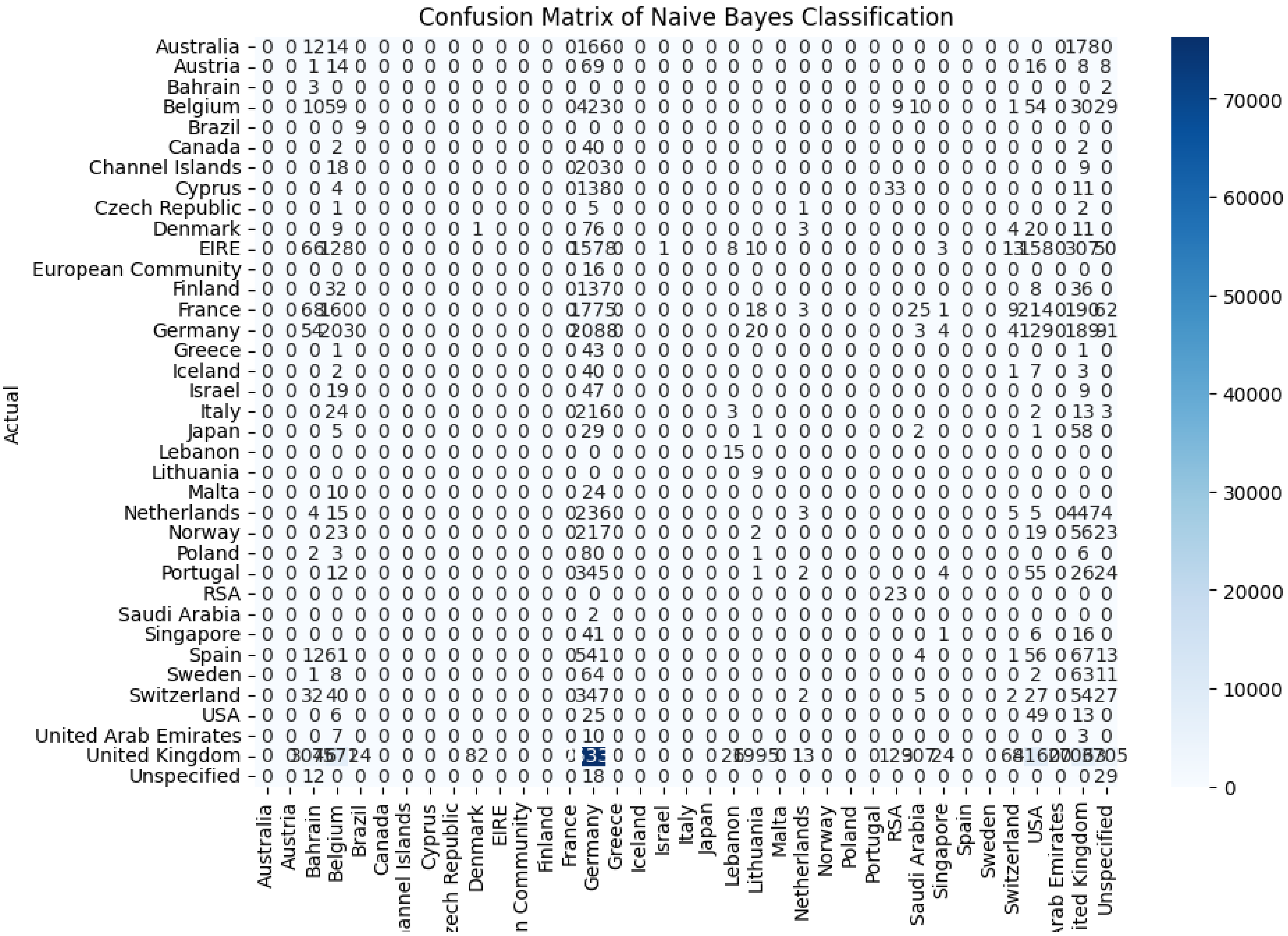
```
)  # Female, Age 30, Annual Income 70k

# Scale the custom data
custom_data_scaled = scaler.transform(custom_data)
# Predict the category for the custom data
predicted_category = knn.predict(custom_data_scaled)

# Output the predicted category
if predicted_category[0] == 1:
    print("Predicted Spending Category: High Spender")
else:
    print("Predicted Spending Category: Low Spender")
```

Output:
Predicted Spending Category: High Spender

9. Implement Naïve Bayes Classification algorithm on the Online Retail. Analyse the efficiency of the algorithm using different metrics.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv("Online_Retail.csv", encoding='latin1')

print(dataset.head())

# Dropping rows with missing CustomerID as we need it for classification
dataset = dataset.dropna(subset=['CustomerID'])

dataset['InvoiceDate'] = pd.to_datetime(dataset['InvoiceDate'])
dataset['InvoiceDay'] = dataset['InvoiceDate'].dt.day
dataset['InvoiceMonth'] = dataset['InvoiceDate'].dt.month
dataset['InvoiceHour'] = dataset['InvoiceDate'].dt.hour

X = dataset[['Quantity', 'UnitPrice', 'InvoiceDay', 'InvoiceMonth', 'InvoiceHour']]
y = dataset['Country']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(f"Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)

plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_,
yticklabels=model.classes_)
plt.title('Confusion Matrix of Naive Bayes Classification')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
  InvoiceNo StockCode                          Description  Quantity
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
Accuracy: 0.0105
                    precision    recall  f1-score   support

           Australia       0.00      0.00      0.00       370
             Austria       0.00      0.00      0.00       116
             Bahrain       0.00      0.60      0.00         5
             Belgium       0.01      0.09      0.01       625
              Brazil       0.27      1.00      0.43         9
              Canada       0.00      0.00      0.00        44
     Channel Islands       0.00      0.00      0.00       230
              Cyprus       0.00      0.00      0.00       186
      Czech Republic       0.00      0.00      0.00         9
             Denmark       0.01      0.01      0.01       124
                EIRE       0.00      0.00      0.00      2322
  European Community       0.00      0.00      0.00        16
             Finland       0.00      0.00      0.00       213
              France       0.00      0.00      0.00      2525
             Germany       0.02      0.75      0.05      2785
              Greece       0.00      0.00      0.00        45
             Iceland       0.00      0.00      0.00        53
              Israel       0.00      0.00      0.00        75
               Italy       0.00      0.00      0.00       261
               Japan       0.00      0.00      0.00        96
             Lebanon       0.29      1.00      0.45        15
           Lithuania       0.00      1.00      0.01         9
               Malta       0.00      0.00      0.00        34
         Netherlands       0.11      0.00      0.01       719
              Norway       0.00      0.00      0.00       340
              Poland       0.00      0.00      0.00        92
            Portugal       0.00      0.00      0.00       469
                 RSA       0.12      1.00      0.21        23
```


Confusion Matrix of Naive Bayes Classification
Actual
Australia
Austria
Bahrain
Belgium
Brazil
Canada
Channel Islands
Cyprus
Czech Republic
Denmark
EIRE
European Community
Finland
France
Germany
Greece
Iceland
Israel
Italy
Japan
Lebanon
Lithuania
Malta
Netherlands
Norway
Poland
Portugal
RSA
Saudi Arabia
Singapore
Spain
Sweden
Switzerland
USA
United Arab Emirates
United Kingdom
Unspecified
70000
60000
50000
40000
30000
20000
10000
0