

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.**](https://bmsce.ac.in/home/facultyProfile/200/Dr-Ch-Ram-Mohan-Reddy) **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**

|  |  |  |
| --- | --- | --- |
| **SL.**  **No.** | **Programs** | **Page No.** |
| **1.** | **Consider the Smart Phone dataset and perform exploratory data analysis**   * Identify the dimension, structure, and summary of the data set * Pre-process the dataset and treat them (like missing values, ‘na’?). Justify the treatment * Plot the histogram for continuous variables (at least two) to analyze the data. * Draw a violin plot to describe the distribution of a numerical variable to analyse the data * Recognize the outliers using a box plot (Display the box plot before and after outlier treatment) * Display a heat map to display the relationship among the attributes * Standardize the continuous variable (if any) | **1** |
| **2.** | **For the data set in Q1,**   * Show the distribution of continuous variables using Box Plot * Identify the relationship between two continuous variables using a scatter plot * Find and display the frequency of the categorical values using a count plot * Apply point plots to display one continuous and one categorical variable | **9** |
| **3.** | **For the Market-Basket dataset, apply the Apriori algorithm and identify the best rules based on support and confidence.** | **13** |
| **4.** | **For the data set given in Q3, apply the FP-tree algorithm, show the tree construction, and identify the best rules based on support and confidence.** | **15** |
| **5.** | **For the Mall-Customer data set, implement the K-means clustering algorithm and visualize the clusters** | **17** |
| **6.** | **For the Groceries dataset** **implement an Agglomerative clustering algorithm and visualize the clusters.** | **19** |
| **7.** | **For the Mall\_Customers implement the DBScan clustering algorithm and visualize the clusters.** | **20** |
| **8.** | **Implement KNN Classification algorithm on the Mall Customers. Analyze the model using different K values and display the performance of the model.** | **22** |
| **9.** | **Implement Naïve Bayes Classification algorithm on the Online Retail. Analyse the efficiency of the algorithm using different metrics.** | **25** |

1. **Consider the Smart Phone dataset and perform exploratory data analysis**
   * 1. **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

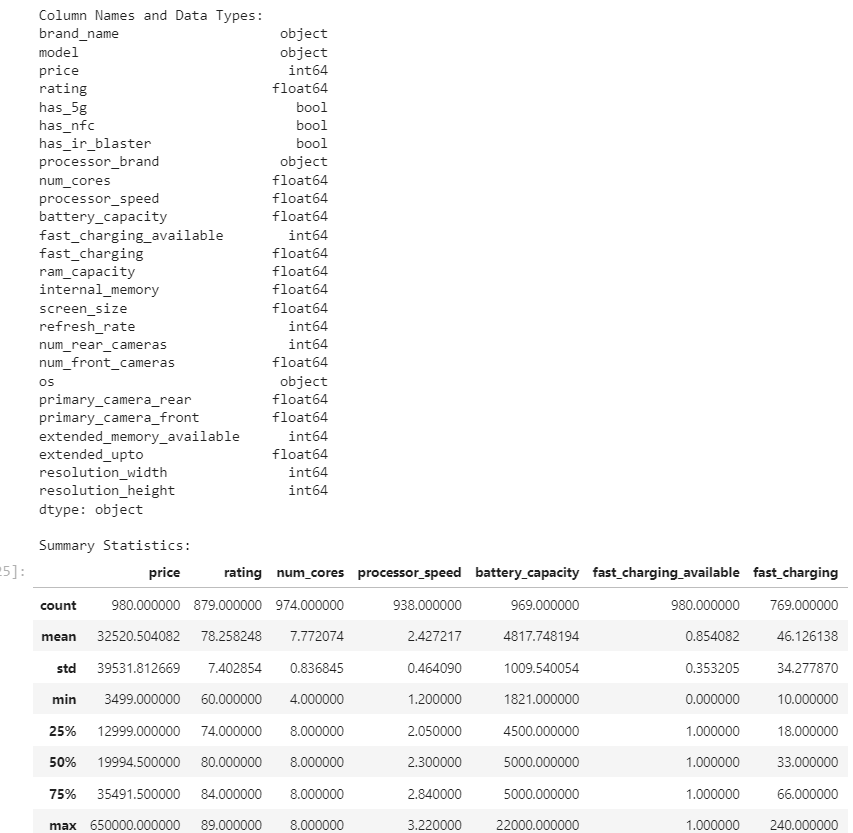
print("\nColumn Names and Data Types:")

print(df.dtypes)

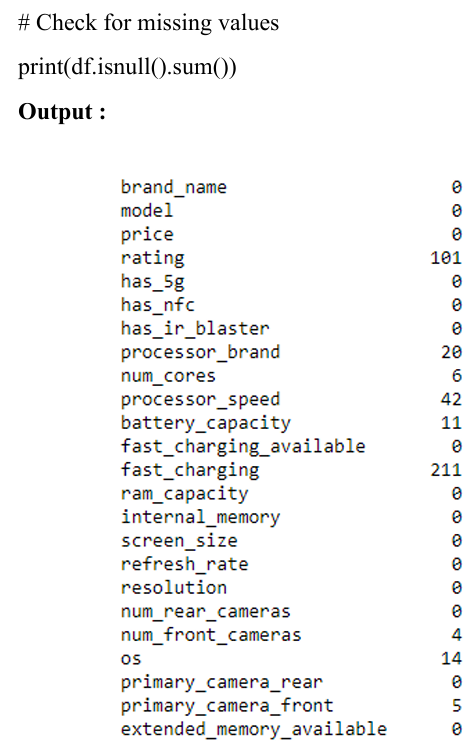
# 3. Summary of the Dataset

print("\nSummary Statistics:")

df.describe()



* + 1. **Pre-process the dataset and treat them (like missing values, ‘na’?). Justify the treatment**

****

* + 1. **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')

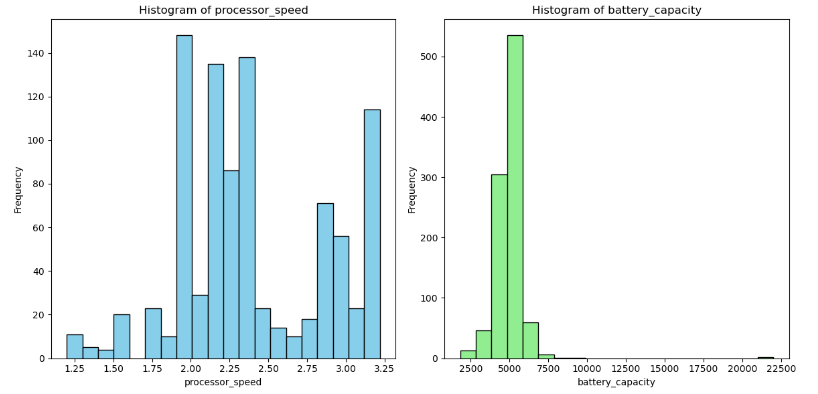
plt.title('Histogram of battery\_capacity')

plt.xlabel('battery\_capacity')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()



* + 1. **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))

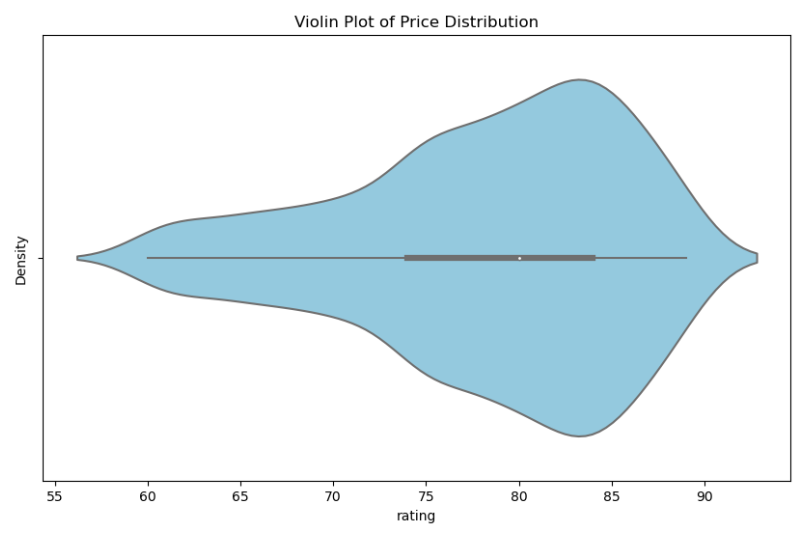
sns.violinplot(x='rating', data=df, color='skyblue')

plt.title('Violin Plot of Price Distribution')

plt.xlabel('rating')

plt.ylabel('Density')

plt.show()



* + 1. **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])

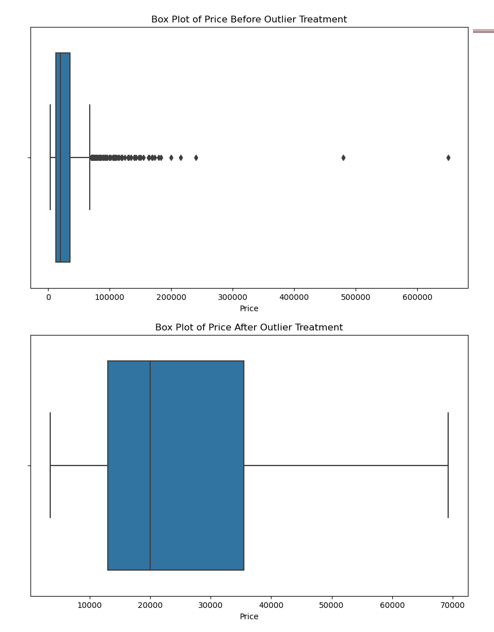
plt.figure(figsize=(10, 6))

sns.boxplot(x=df[variable])

plt.title('Box Plot of Price After Outlier Treatment')

plt.xlabel('Price')

plt.show()



* + 1. **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'])

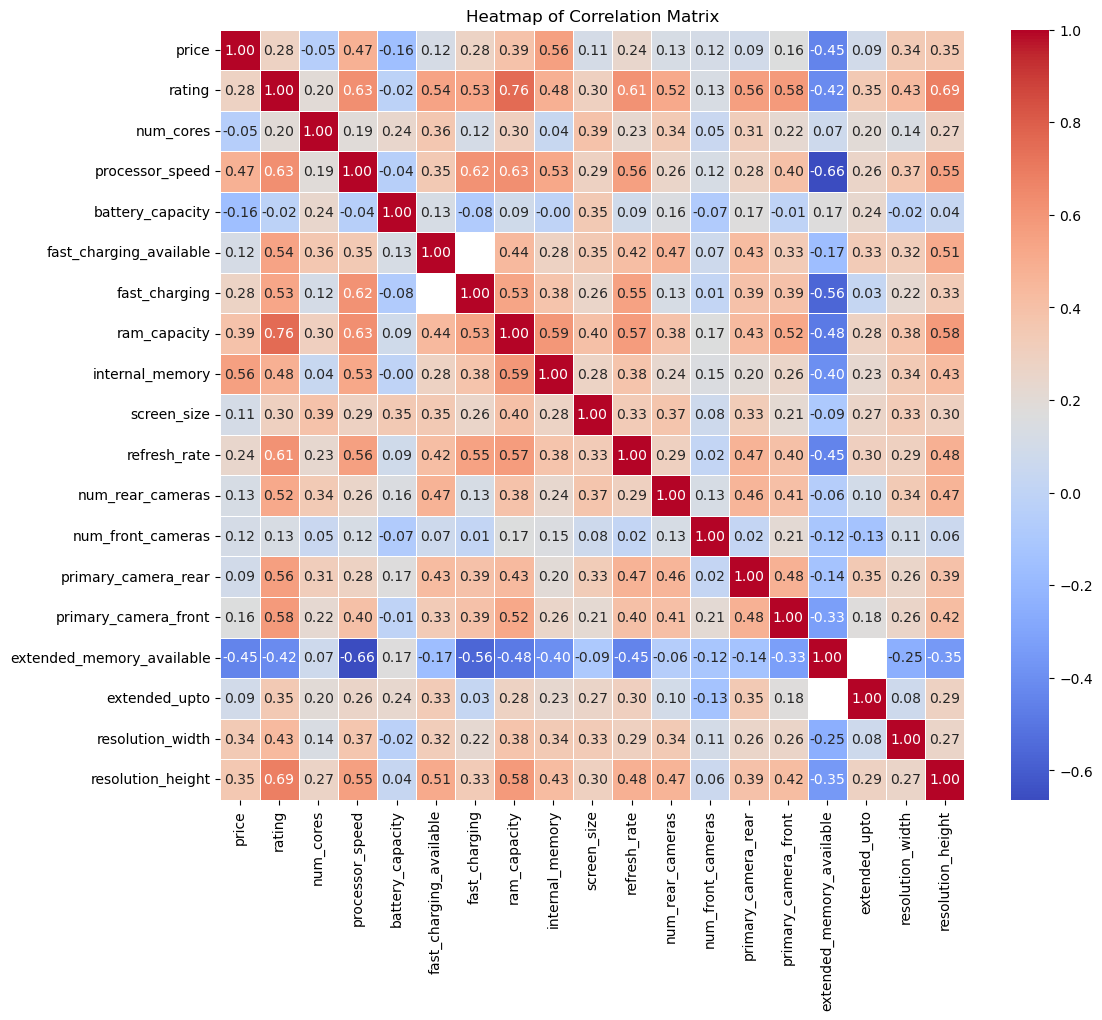
correlation\_matrix = numerical\_df.corr()

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()



* + 1. **Standardize the continuous variable (if any)**

from sklearn.preprocessing import StandardScaler

numerical\_df = df.select\_dtypes(include=['float64', 'int64'])

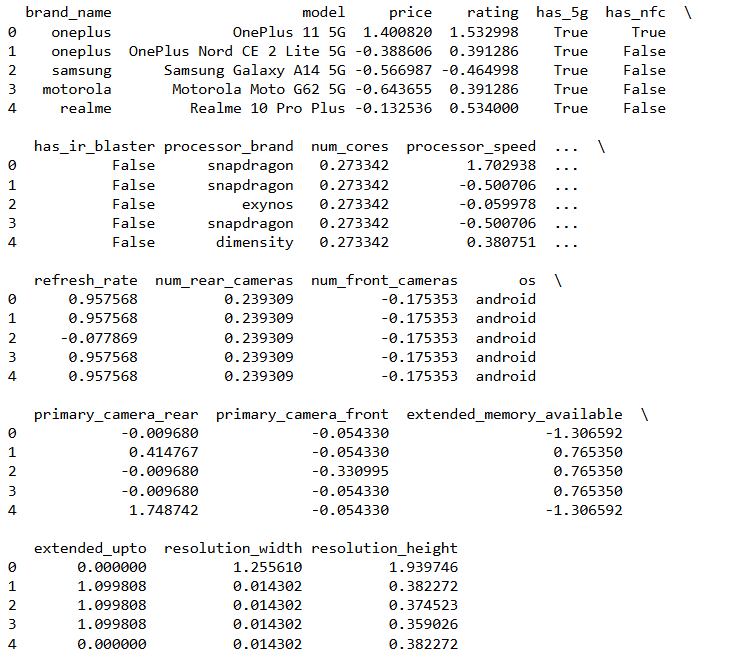
scaler = StandardScaler()

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())

1. **For the data set in Q1,**
2. **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):

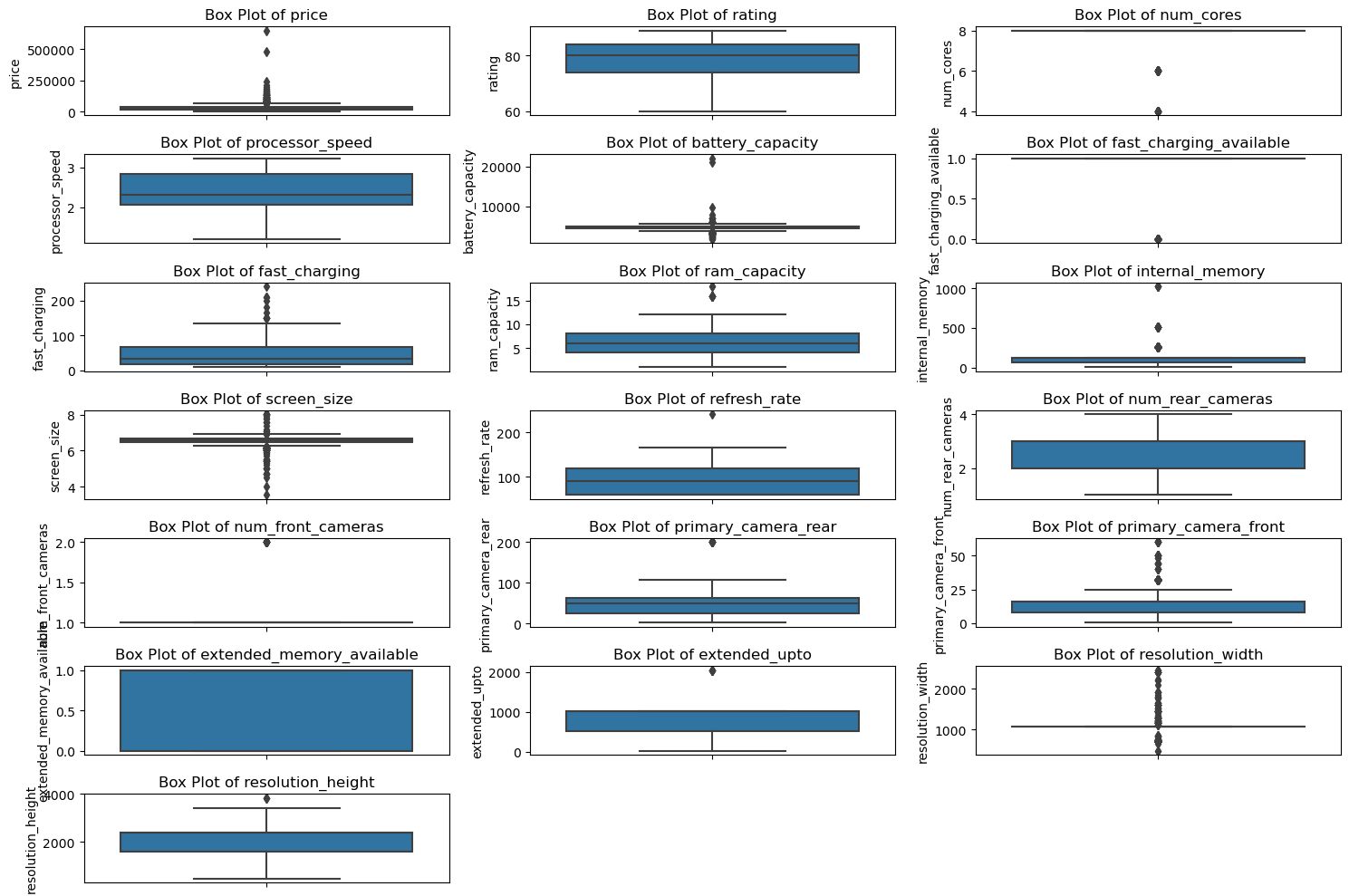
plt.subplot(len(numerical\_df.columns) // 3 + 1, 3, i + 1)

sns.boxplot(y=df[col])

plt.title(f'Box Plot of {col}')

plt.tight\_layout()

plt.show()



1. **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)

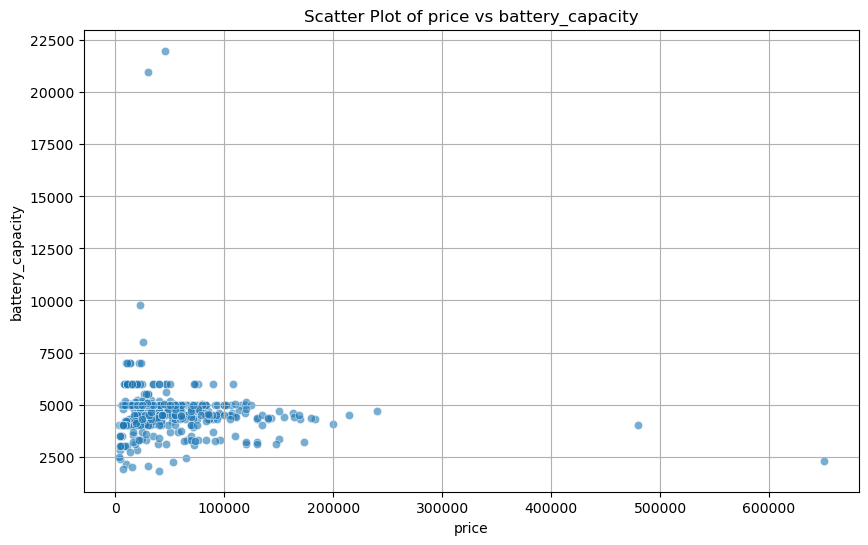
plt.title(f'Scatter Plot of {x\_var} vs {y\_var}')

plt.xlabel(x\_var)

plt.ylabel(y\_var)

plt.grid(True)

plt.show()



1. **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)

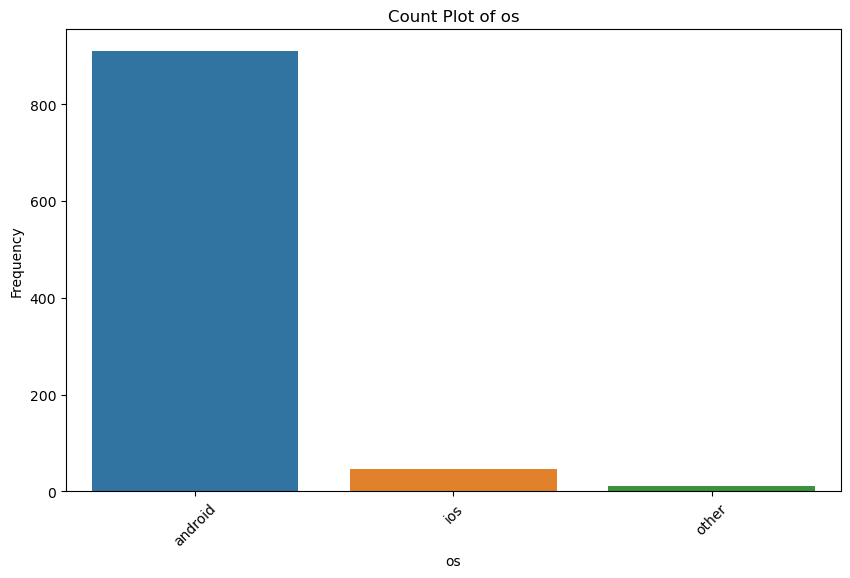
plt.title(f'Count Plot of {categorical\_var}')

plt.xlabel(categorical\_var)

plt.ylabel('Frequency')

plt.xticks(rotation=45) # Rotate labels if necessary for better readability

plt.show()



1. **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='-')

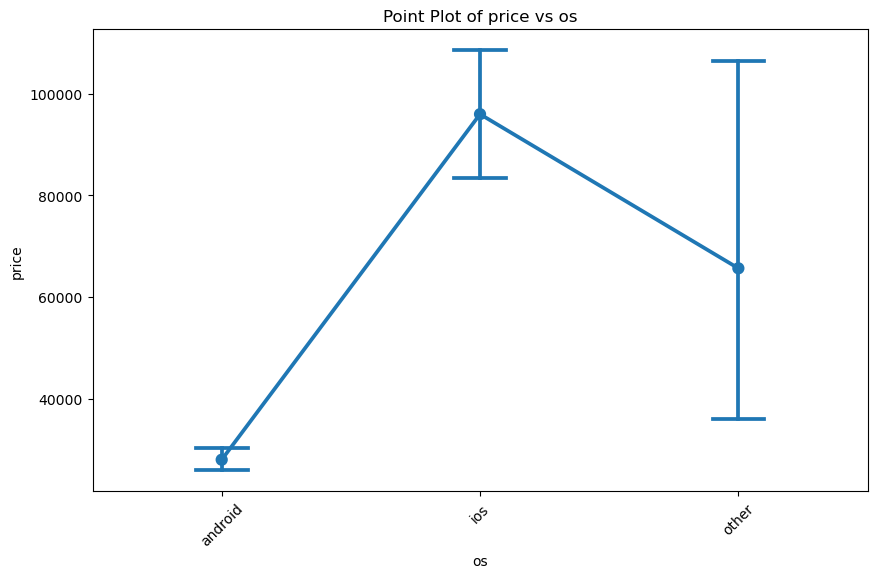
plt.title(f'Point Plot of {continuous\_var} vs {categorical\_var}')

plt.xlabel(categorical\_var)

plt.ylabel(continuous\_var)

plt.xticks(rotation=45) # Rotate labels if necessary for better readability

plt.show()



1. **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)

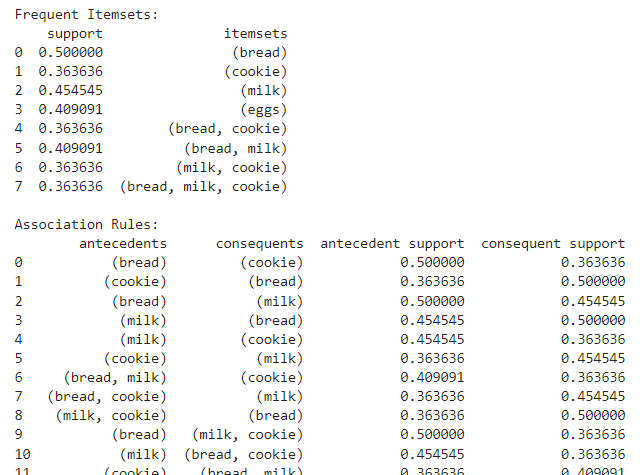
print("\nAssociation Rules:")

print(rules)

**sample dataset:**



**Output**



1. **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)

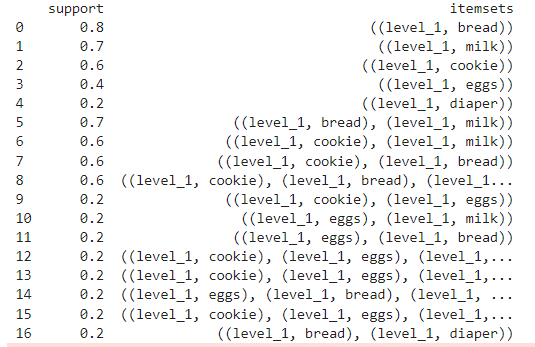
df = df.stack().reset\_index().pivot\_table(index='level\_0', columns=0, aggfunc=lambda x: 1, fill\_value=0)

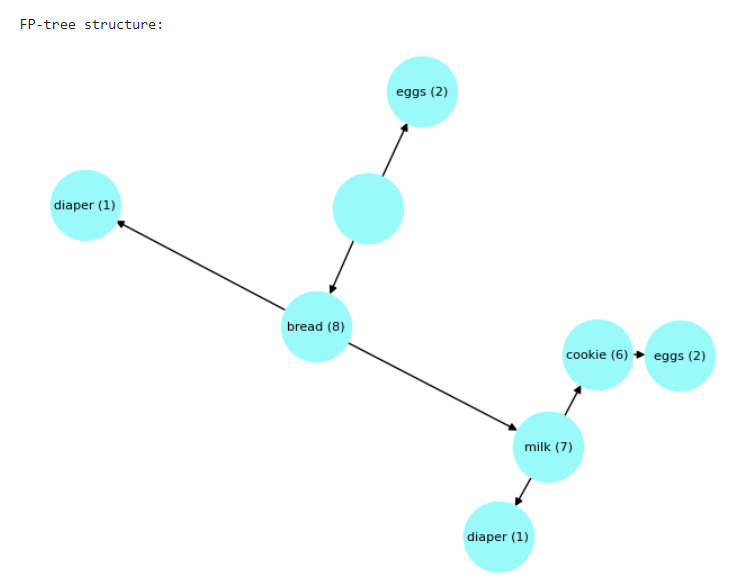
# Apply the FP-growth algorithm

frequent\_itemsets = fpgrowth(df, min\_support=0.2, use\_colnames=True)

# Display frequent itemsets

print(frequent\_itemsets)





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

import pandas as pd

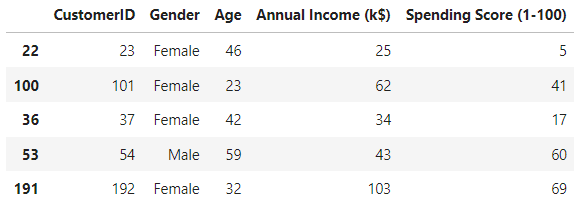
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import LabelEncoder

data=pd.read\_csv("Mall\_Customers.csv")

#data.sample(5)



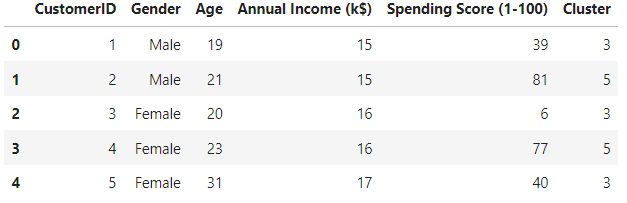
X=data[['Annual Income (k$)','Spending Score (1-100)']]

kmean=KMeans(n\_clusters=7,random\_state=0)

y\_kmeans=kmean.fit\_predict(X)

data['Cluster']=y\_kmeans

#data.head(5)



plt.figure(figsize=(8,6))

plt.scatter(X.iloc[:,0],X.iloc[:,1],c=y\_kmeans,s=50)

centroids = kmean.cluster\_centers\_

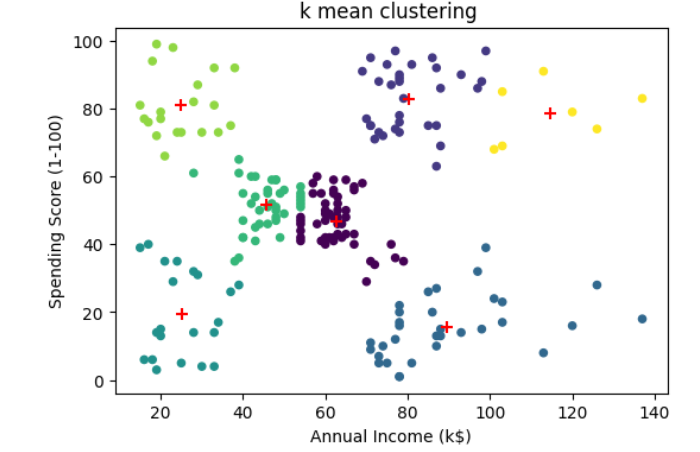
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, marker='+', label='Centroids')

plt.title('k mean clustering ')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.show()



1. **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))

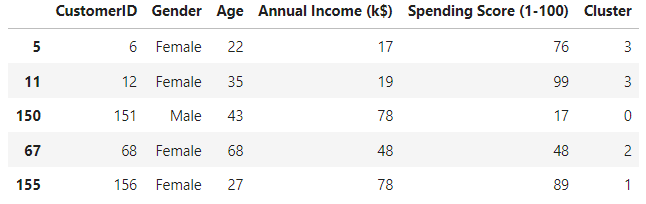
dendrogram(linkage\_matrix)

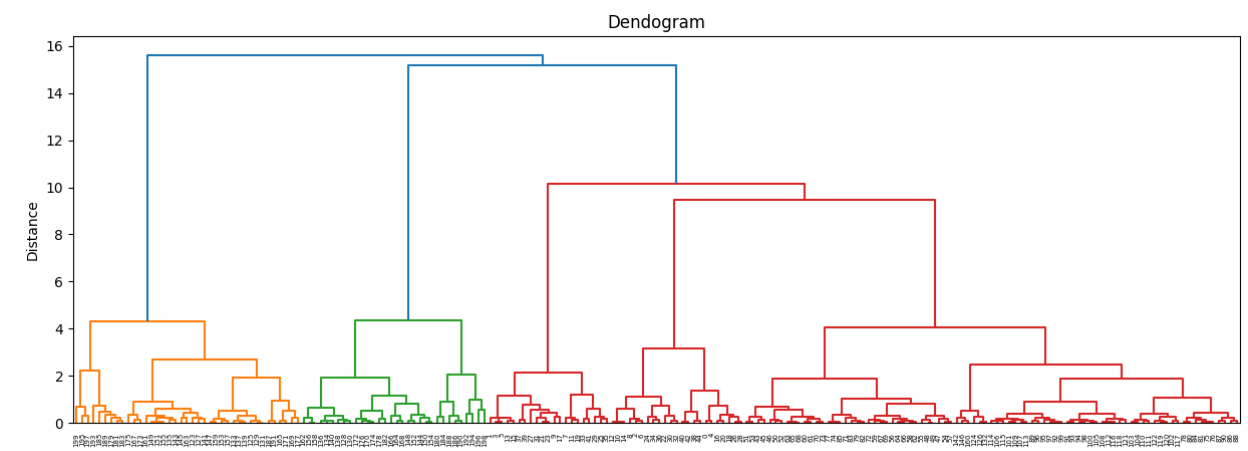
plt.title('Dendogram')

plt.xlabel('levels')

plt.ylabel('Distance')

plt.show()





1. **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)

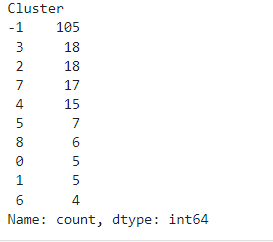
plt.title("DBSCAN Clustering of mall customers")

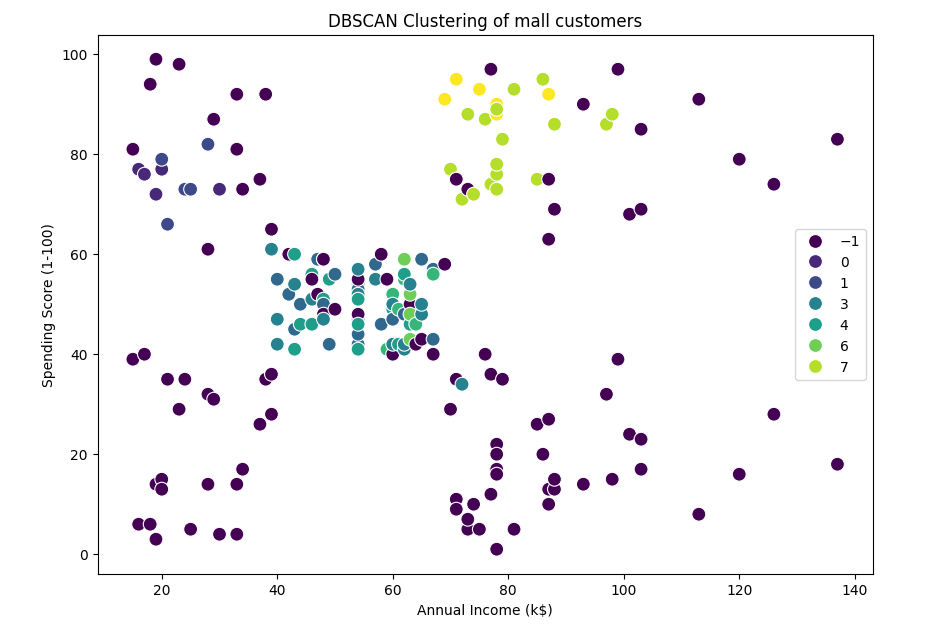
plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()





1. **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')

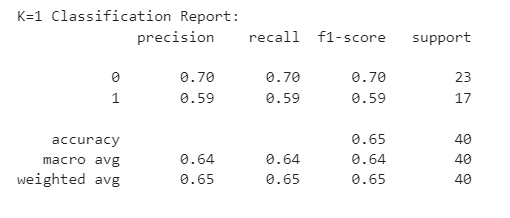
plt.title('KNN Accuracy for Different K Values')

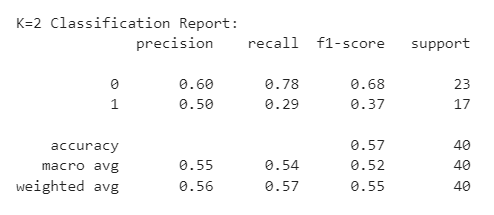
plt.xlabel('K (Number of Neighbors)')

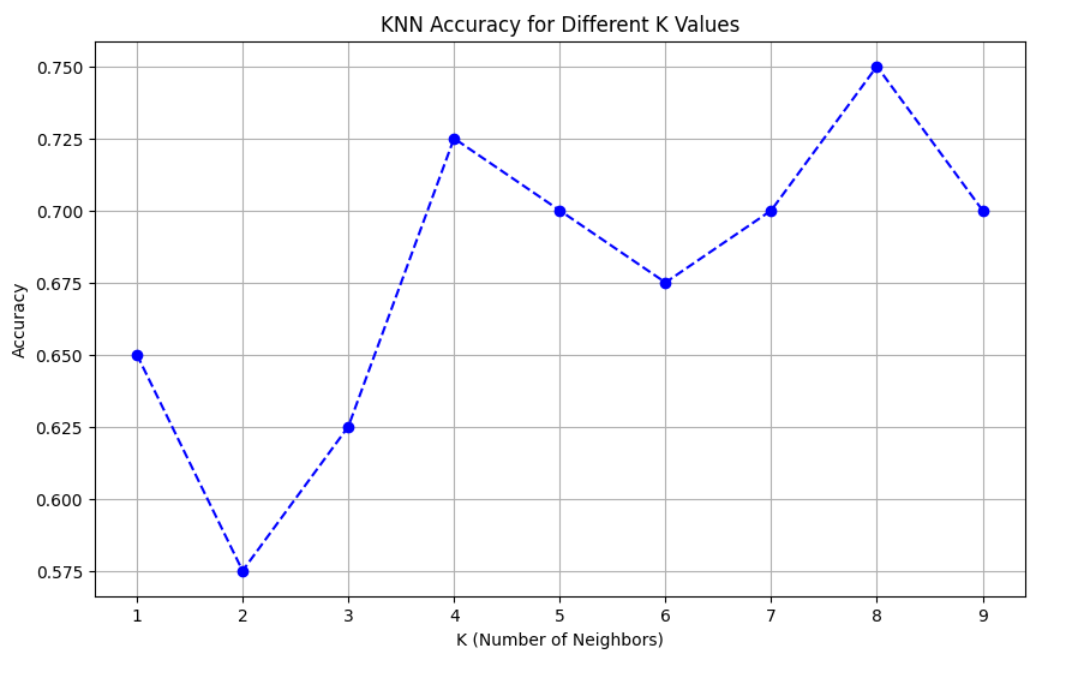
plt.ylabel('Accuracy')

plt.grid(True)

plt.show()







# 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**

1. **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()

