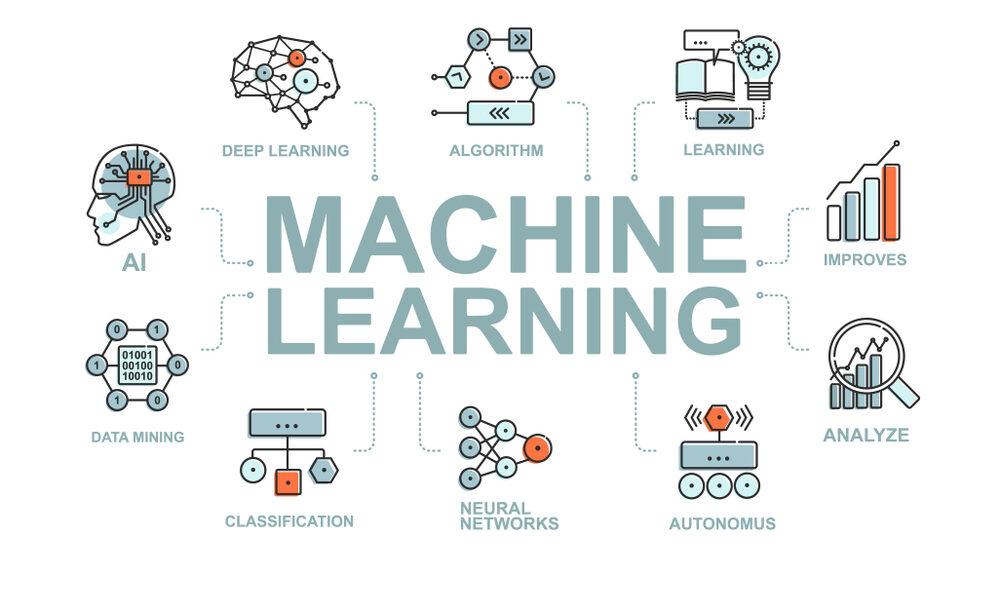
**Practical File**

**“Fundamentals of Machine Learning Lab”**

**(AIDS-258)**



**Submitted By:**

**Student Name:** Kirty Gupta

**Enrolment no:** 03317711922

**Branch & Section:** AI-DS (A)

**Submitted To:**

* Prof. Deepali Virmani

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Experiment Title | Page No. | Date | Grade/  Evaluation | T.Sign |
| 1 | Study and Implement Linear Regression. | 3 | 09.02.24 |  |  |
| 2 | Study and Implement Logistic Regression. | 9 | 16.02.24 |  |  |
| 3 | Study and Implement K Nearest Neighbour (KNN). | 15 | 23.02.24 |  |  |
| 4 | Study and Implement classification using SVM. | 22 | 01.03.24 |  |  |
| 5 | Study and Implement Bagging using Random Forests. | 31 | 08.03.24 |  |  |
| 6 | Study and Implement Naive Bayes. | 37 | 15.03.24 |  |  |
| 7 | Study and Implement Decision Trees. | 46 | 22.03.24 |  |  |
| 8 | Study and Implement K-means Clustering to Find Natural Patterns in Data. | 51 | 12.04.24 |  |  |
| 9 | Study and Implement Gaussian Mixture Model Using the Expectation Maximization. | 58 | 19.04.24 |  |  |
| 10 | Study and Implement Classification based on association rules. | 64 | 26.04.24 |  |  |

**Experiment 1**

|  |
| --- |
| **Title:** Study and Implement Linear Regression.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

import numpy as np

import pandas as pd

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 import metrics

# To ignore warnings

import warnings

warnings.filterwarnings("ignore") # To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('USA\_housing.csv')

df.head()

df.info()

df.describe()

sns.pairplot(df, palette='Set2')

# Checking the number of missing values in each column

df.isnull().sum()

# Dropping the missing values

df = df.dropna()

# Checking the number of missing values in each column

df.isnull().sum()

y = df['Price']

print(y)

X = df.drop('Price', axis = 1)

print(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3)

reg = LinearRegression()

reg.fit(X\_train, y\_train)

print(reg.intercept\_)

prediction = reg.predict(X\_test)

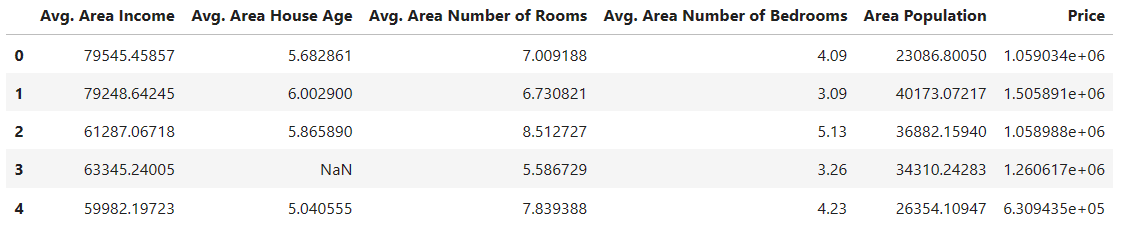
plt.scatter(y\_test, prediction)

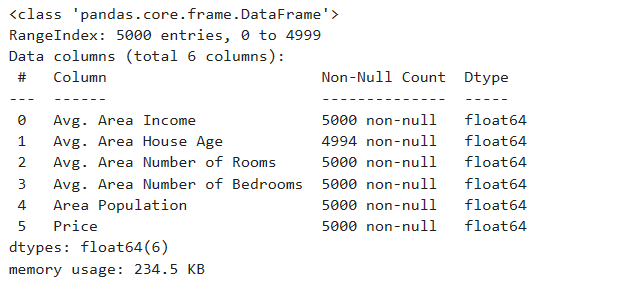
print('MAE: ', metrics.mean\_absolute\_error(y\_test, prediction))

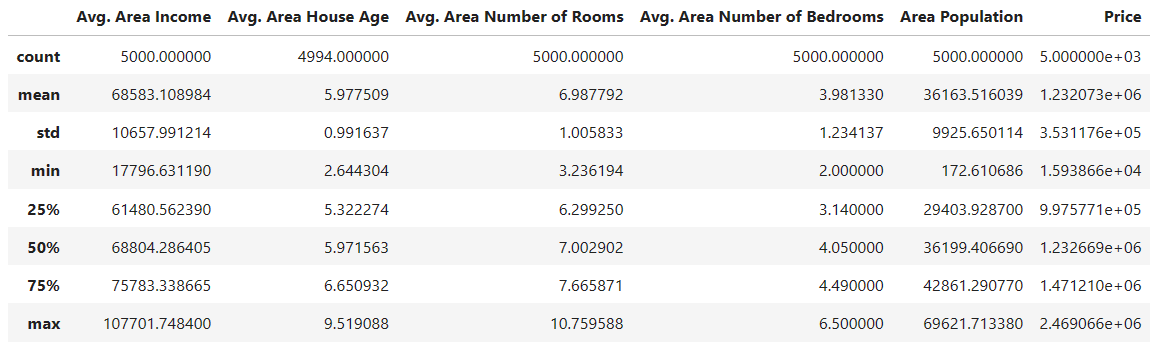
print('MSE: ', metrics.mean\_squared\_error(y\_test, prediction))

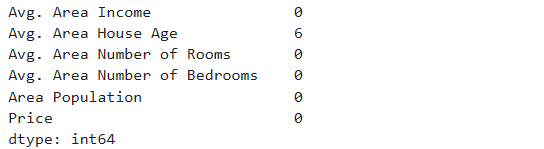
print('RMSE: ', np.sqrt(metrics.mean\_absolute\_error(y\_test, prediction)))

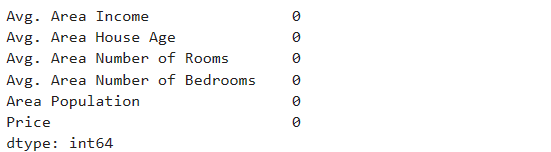
**Output:**

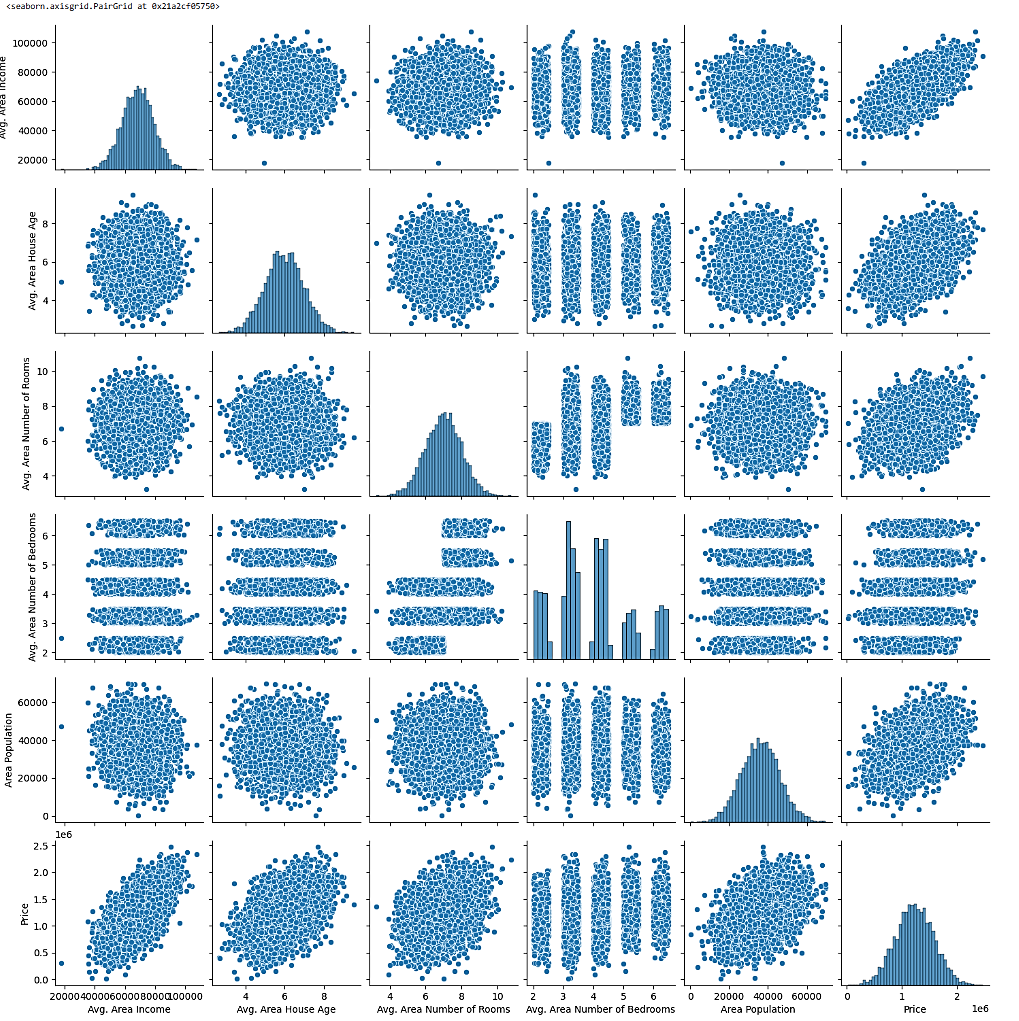
****

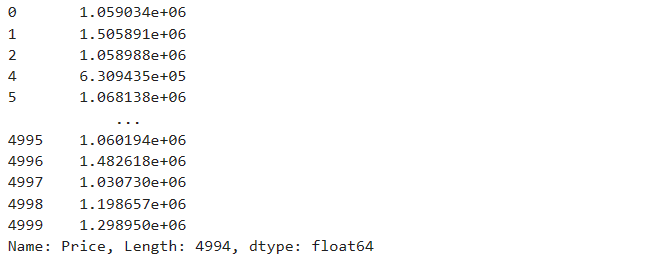
****

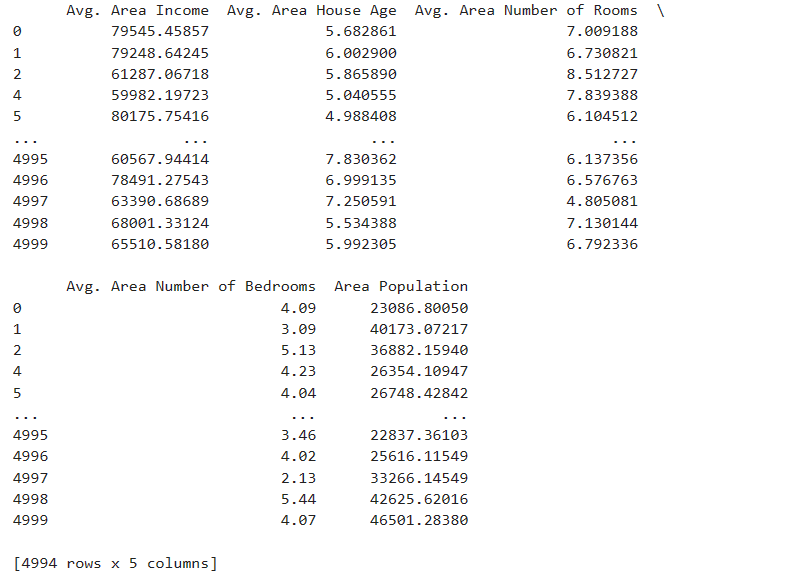


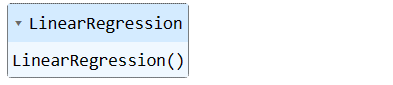




****

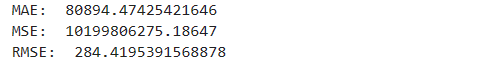
****

****



****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 2**

|  |
| --- |
| **Title:** Study and Implement Logistic Regression.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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 LogisticRegression

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('titanic\_train.csv')

df.head()

df.info()

sns.heatmap(df.isnull())

df.drop(['Cabin'], axis = 1, inplace = True)

df.head()

mean\_value = int(df['Age'].mean())

df['Age'].fillna(mean\_value, inplace = True)

df.info()

sns.heatmap(df.isnull())

sns.countplot(x = 'Survived', data = df)

male = pd.get\_dummies(df['Sex'], drop\_first = True)

embark = pd.get\_dummies(df['Embarked'], drop\_first = True)

df1 = pd.concat([df, embark, male], axis = 1)

df1.head()

df2 = df1.drop(['PassengerId', 'Sex', 'Name', 'Ticket', 'Embarked'], axis=1)

df2.head()

y = df2['Survived']

X = df2.drop(['Survived'], axis = 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

model = LogisticRegression()

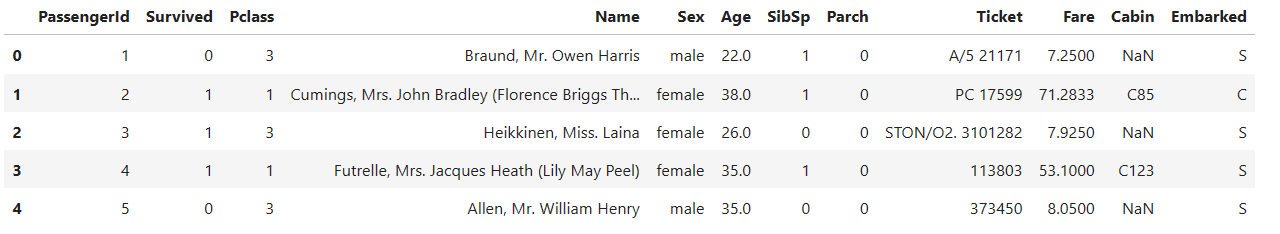
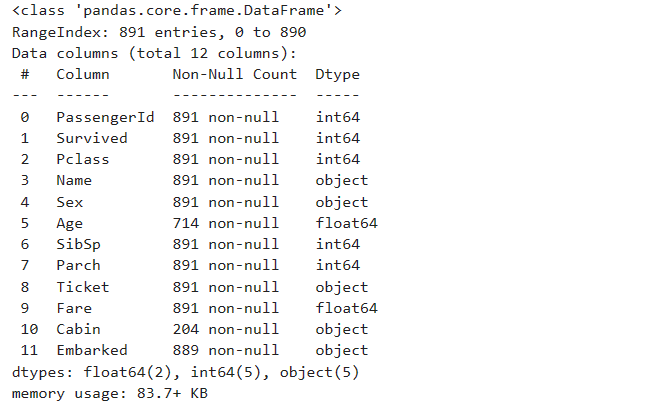
model.fit(X\_train, y\_train)

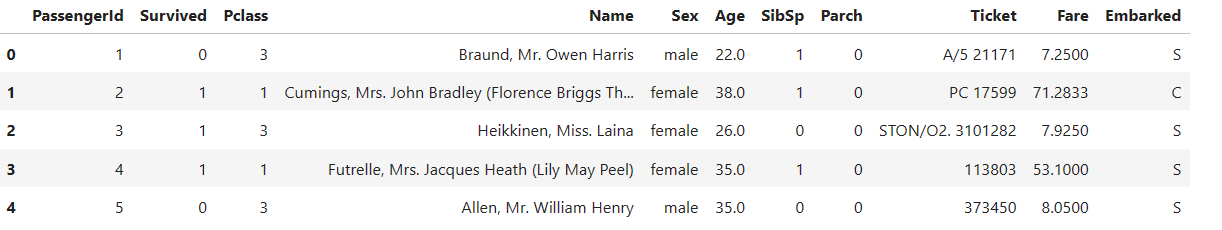
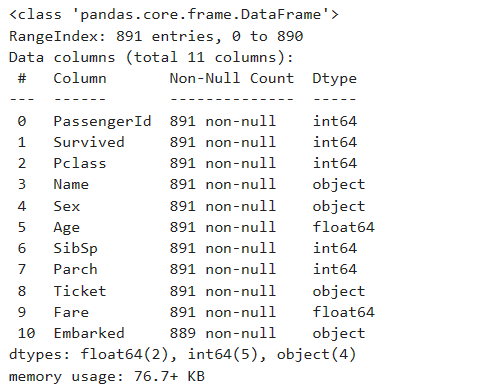
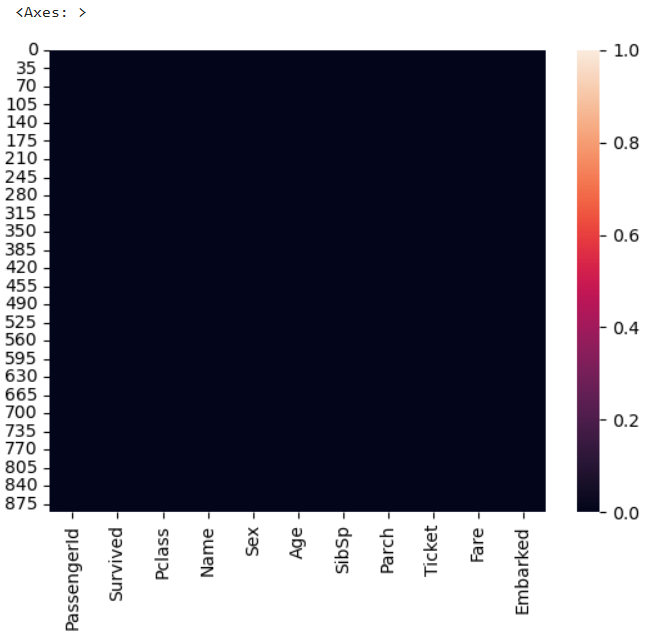
prediction = model.predict(X\_test)

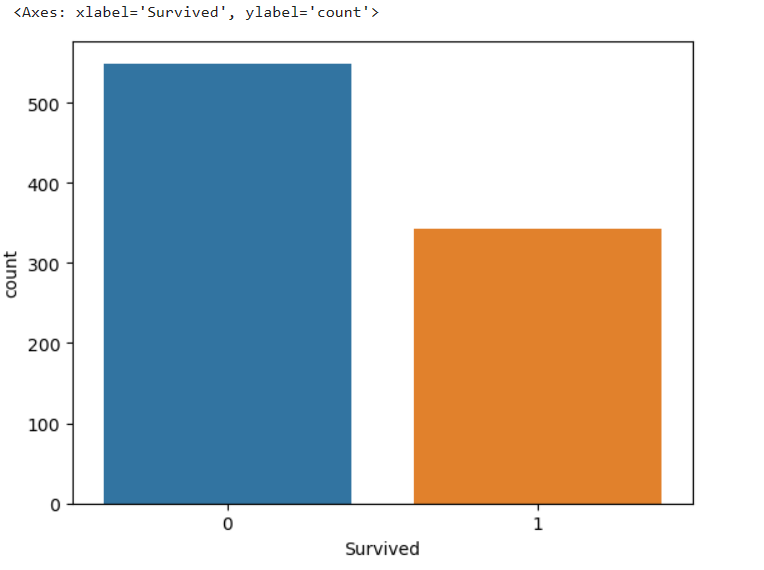
confusion\_matrix(y\_test, prediction)

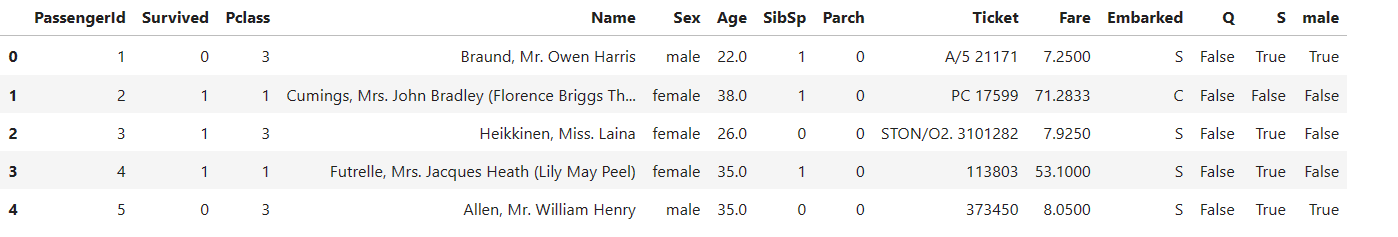
print(classification\_report(y\_test, prediction))

**Output:**

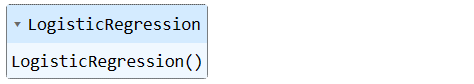
**** **** 

**** **** 

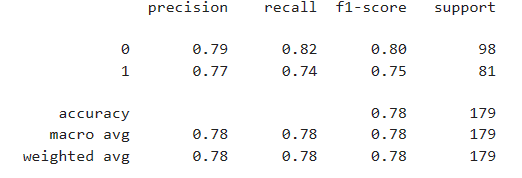
****

****





****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 3**

|  |
| --- |
| **Title:** Study and Implement K Nearest Neighbour (KNN).  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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.neighbors import KNeighborsClassifier

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('KNN\_Project\_Data.csv')

df.head()

print(df.info())

print(df.describe())

# Checking for missing values

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

# Check for class balance

print(df['TARGET CLASS'].value\_counts())

# Correlation matrix

corr\_matrix = df.corr()

plt.figure(figsize=(12, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

# Box plot for each feature

plt.figure(figsize=(16, 8))

sns.boxplot(data=df.drop(columns='TARGET CLASS'))

plt.title('Boxplot of Features')

plt.xticks(rotation=45)

plt.show()

X = df.drop('TARGET CLASS', axis = 1)

print(X)

y = df['TARGET CLASS']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 1)

knn = KNeighborsClassifier(n\_neighbors = 5, metric = 'euclidean')

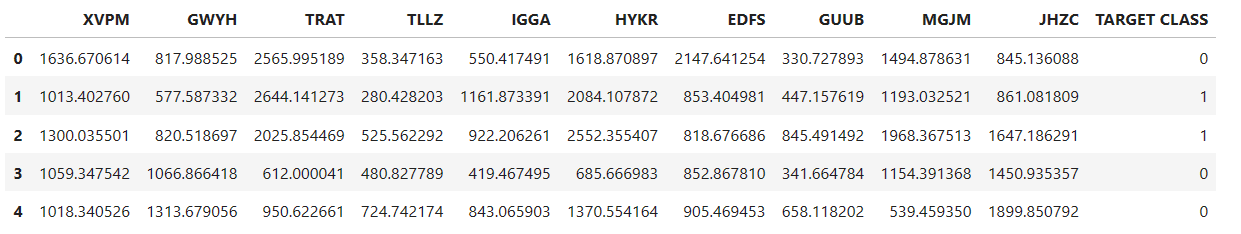
knn.fit(X\_train,y\_train)

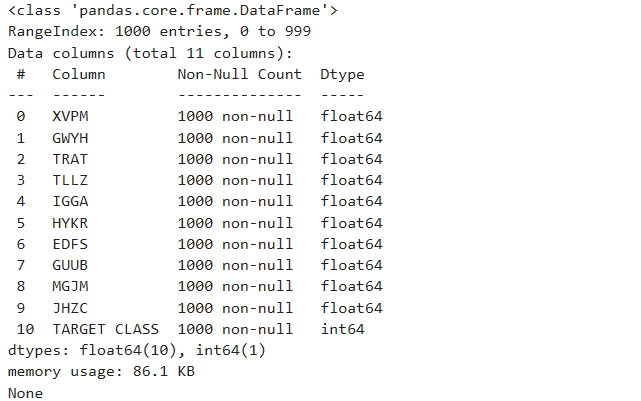
y\_pred = knn.predict(X\_test)

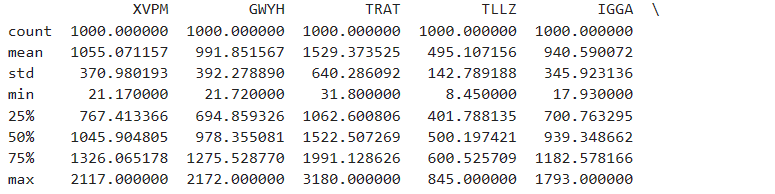
confusion\_matrix(y\_test, y\_pred)

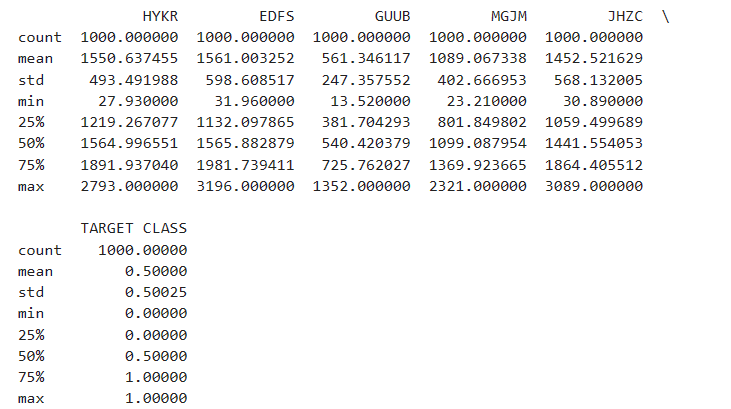
print(classification\_report(y\_test, y\_pred))

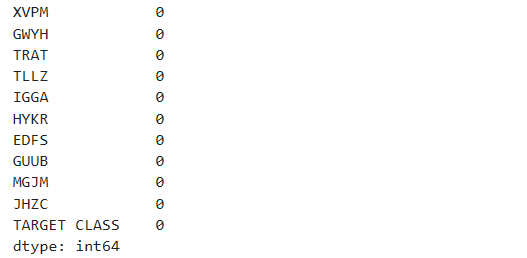
**Output:**

****

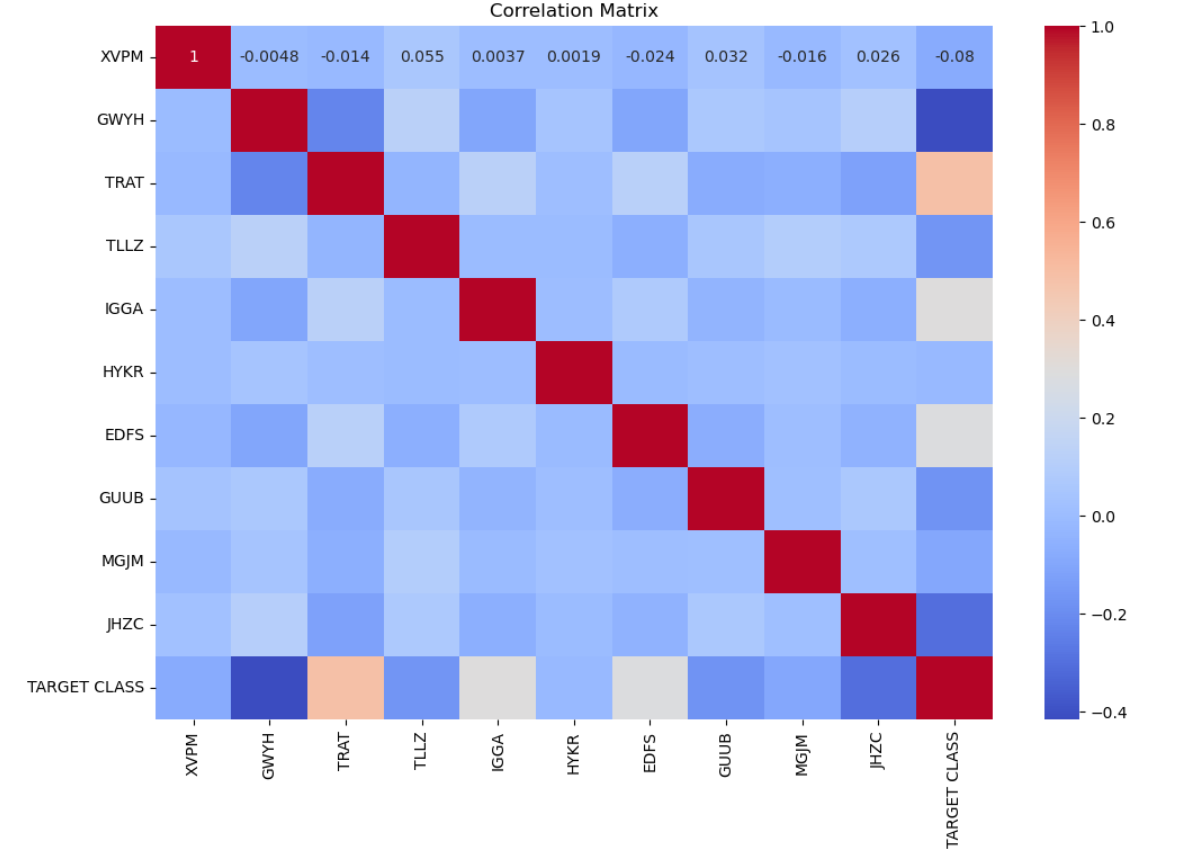
****

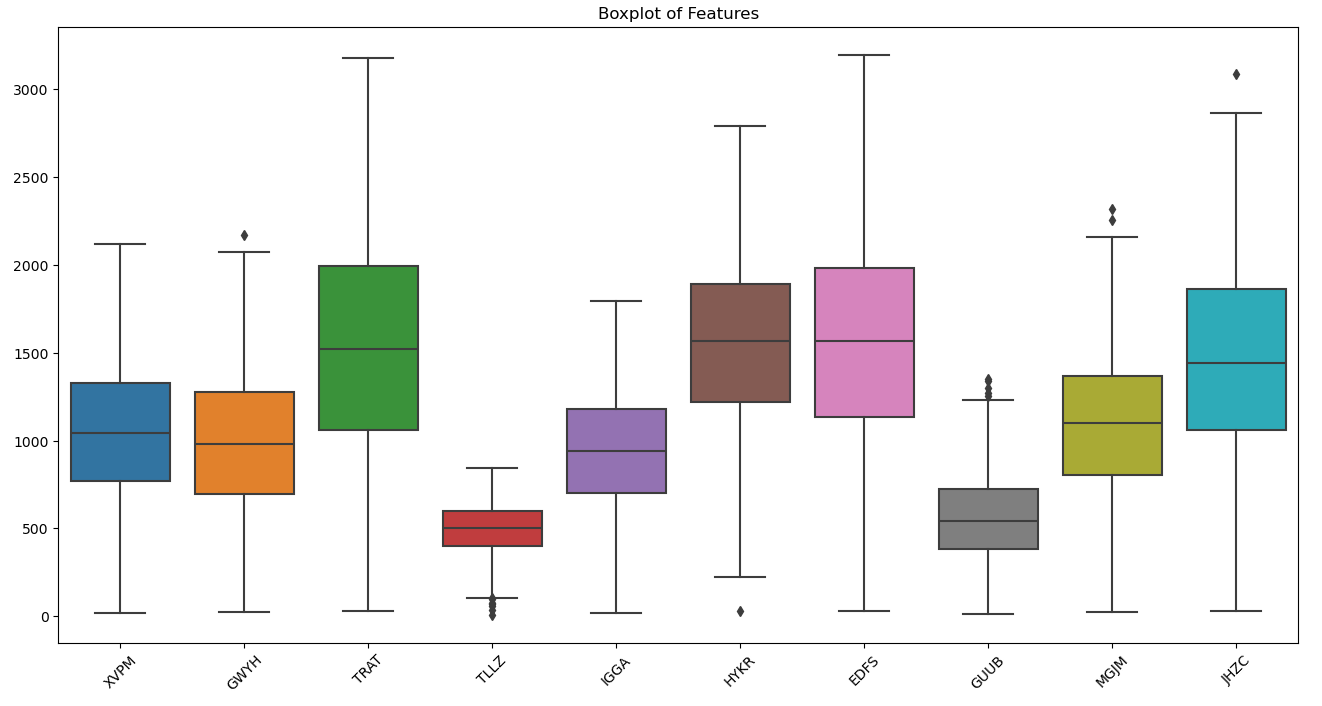
****

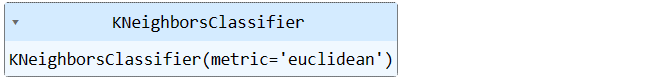
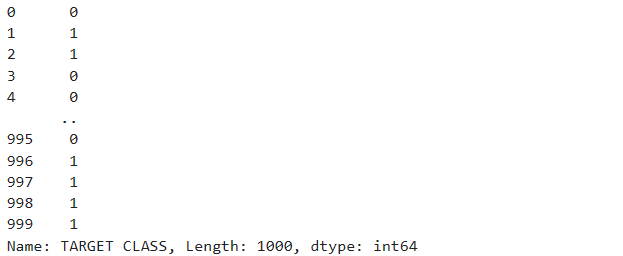
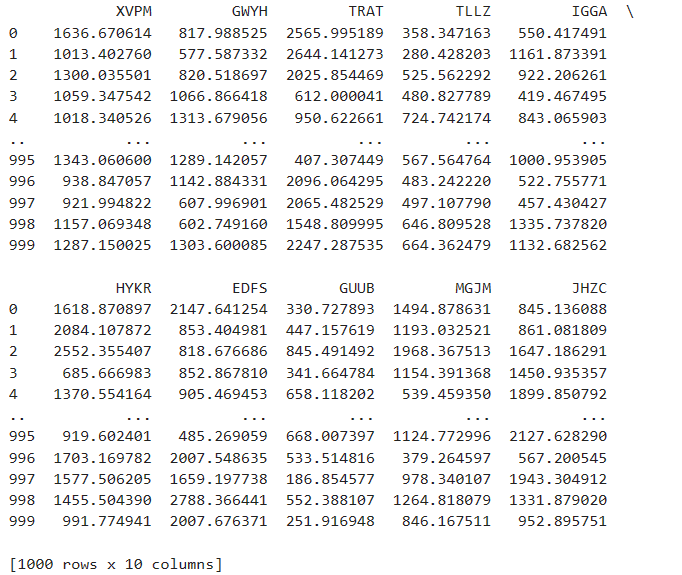
****

****

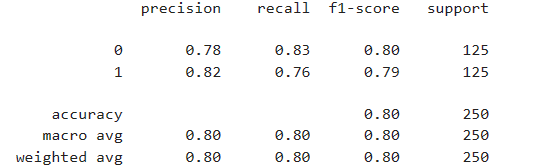
****

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 4**

|  |
| --- |
| **Title:** Study and Implement classification using SVM.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

cancer\_df = pd.read\_csv('breast-cancer.csv')

cancer\_df.head()

cancer\_df.info()

cancer\_df.describe()

cancer\_df.isnull().sum()

plt.figure(figsize=(6, 4))

sns.countplot(data=cancer\_df, x='diagnosis')

plt.title('Class Distribution')

plt.xlabel('Diagnosis')

plt.ylabel('Count')

plt.show()

# Determine the number of features

num\_features = len(cancer\_df.columns[:-1])

# Calculate the number of rows and columns for subplots

num\_rows = (num\_features - 1) // 6 + 1 # Max 6 plots per row

num\_cols = min(6, num\_features) # Maximum 6 columns

# Visualize the distribution of features

plt.figure(figsize=(14, 10))

for i, feature in enumerate(cancer\_df.columns[:-1]):

plt.subplot(num\_rows, num\_cols, i+1)

sns.histplot(data=cancer\_df, x=feature, kde=True, bins=20)

plt.title(feature)

plt.tight\_layout()

plt.show()

\*\*To predict whether the cancer type is Malignant(M)[1] or Benign(B)[0]\*\*

# diagnosis column values

print("Class distribution:")

cancer\_df['diagnosis'].value\_counts()

# Encoding diagnosis Column

cancer\_df.replace({"diagnosis":{"M":1,"B":0}}, inplace=True)

# Visualize the correlation matrix

plt.figure(figsize=(12, 8))

sns.heatmap(cancer\_df.corr(), cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

X = cancer\_df.drop(['diagnosis','id'], axis = 1)

print(X)

y = cancer\_df['diagnosis']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,stratify=y,random\_state=2)

clf = svm.SVC(kernel='linear')

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

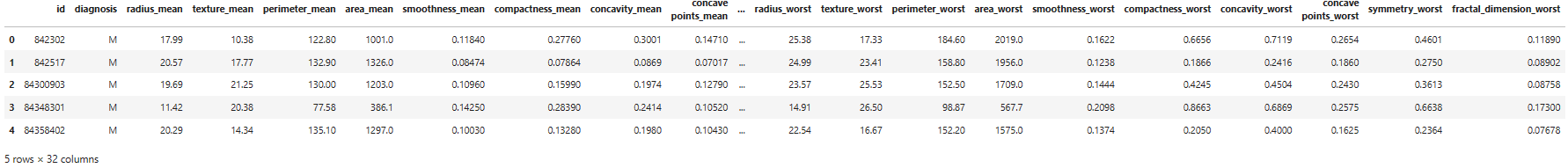
print('Accuracy: ', metrics.accuracy\_score(y\_test, y\_pred))

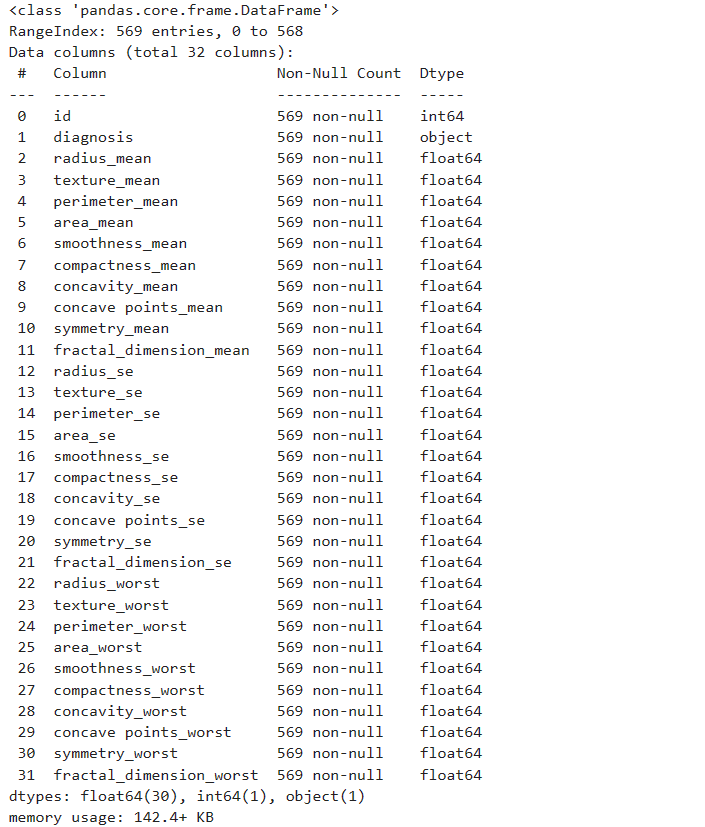
print('Precision: ', metrics.precision\_score(y\_test, y\_pred))

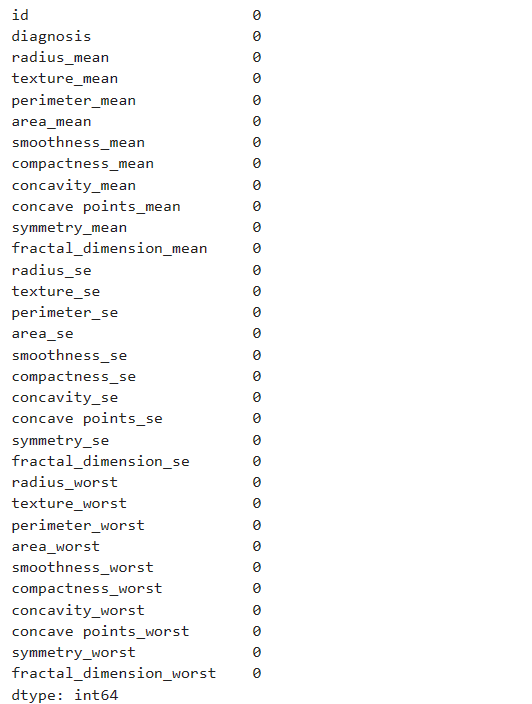
print(classification\_report(y\_test, y\_pred))

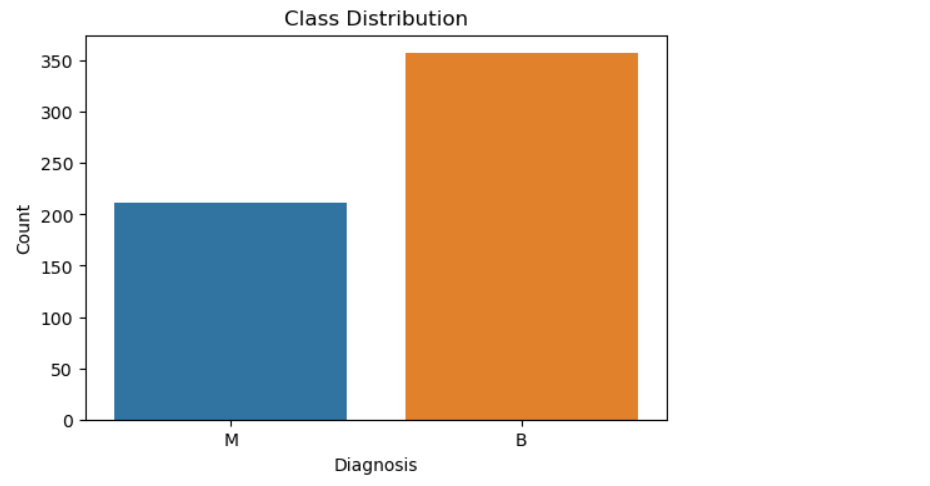
print(confusion\_matrix(y\_test, y\_pred))

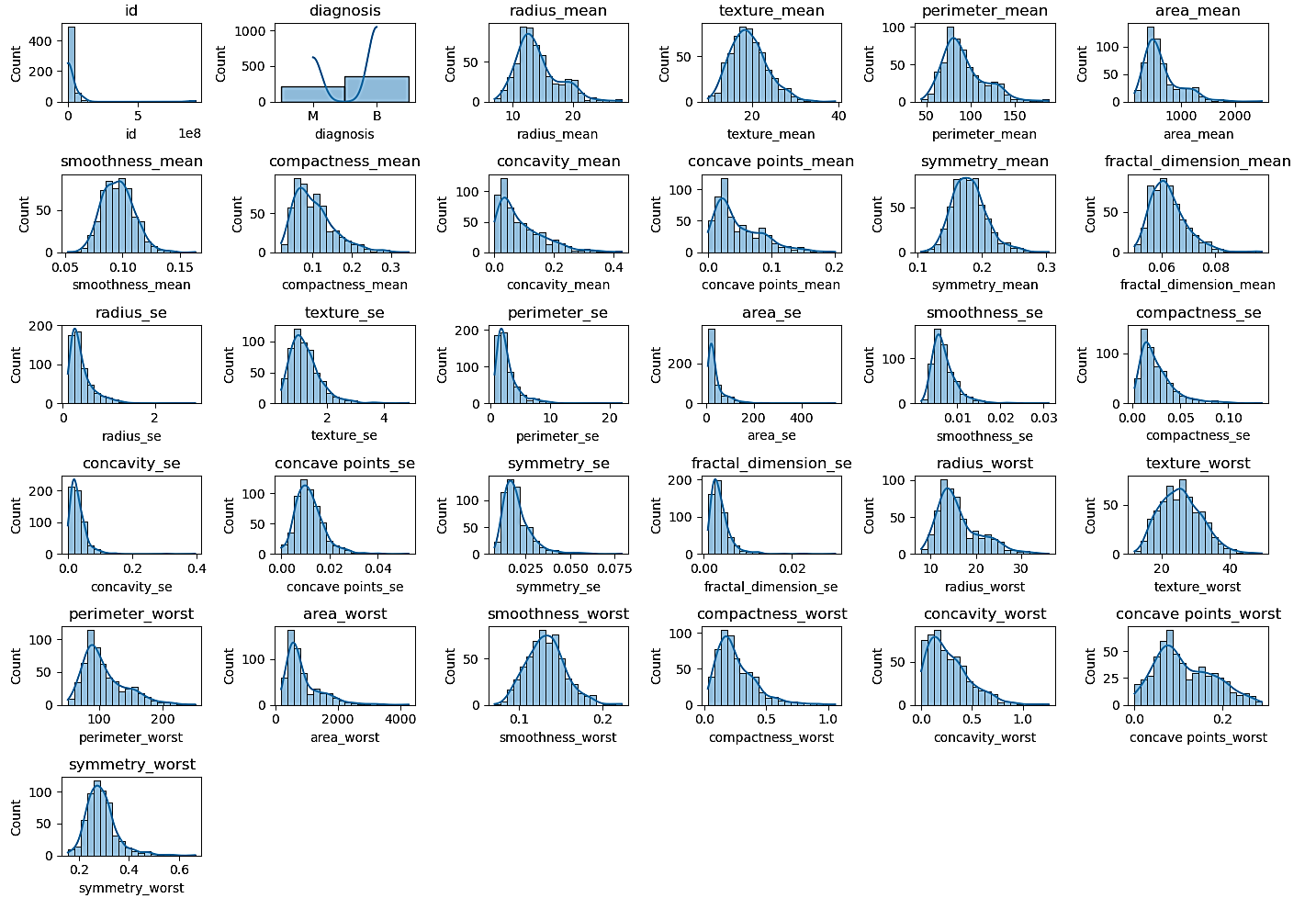
**Output:**

****

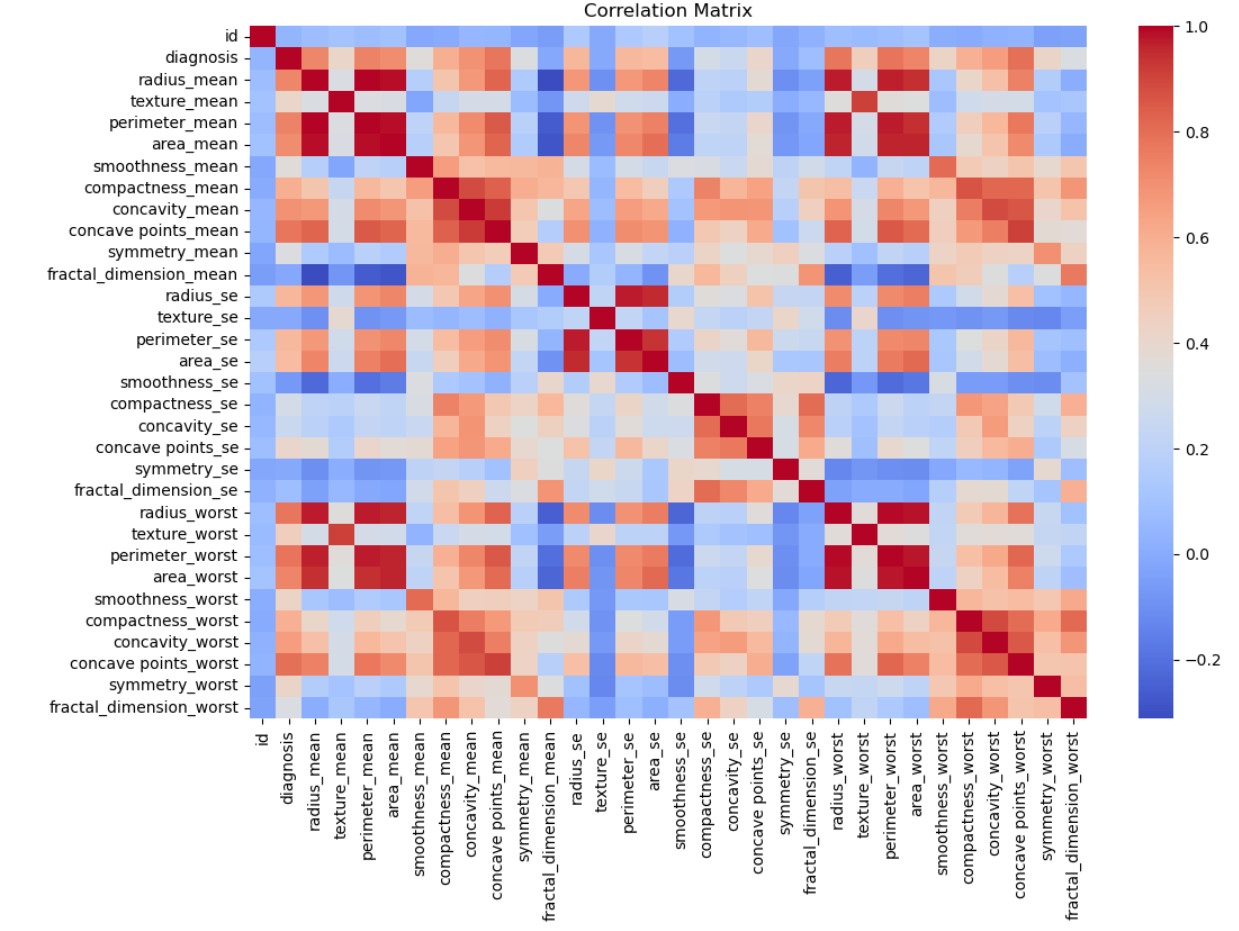
****

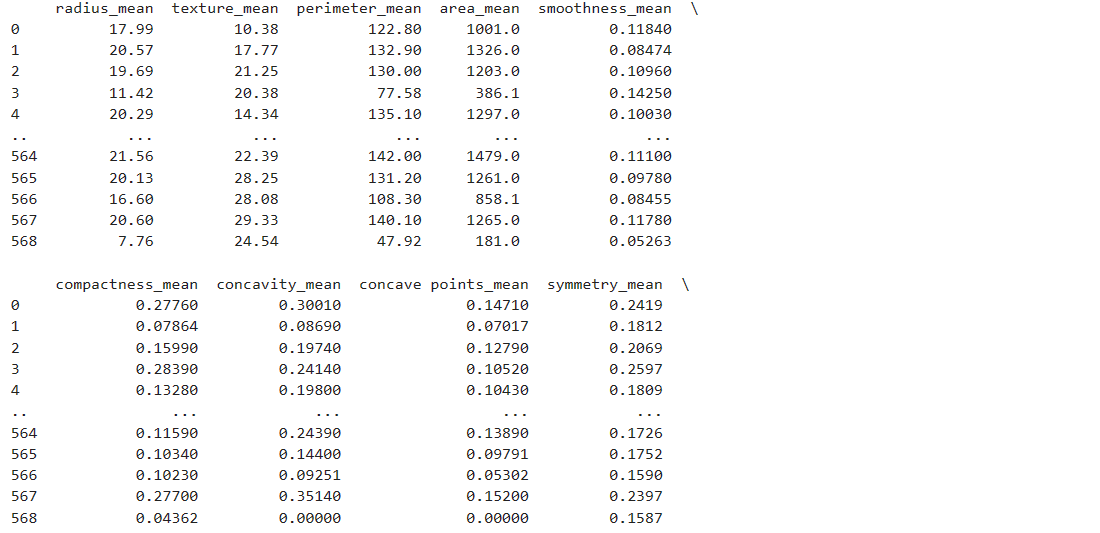
****

****

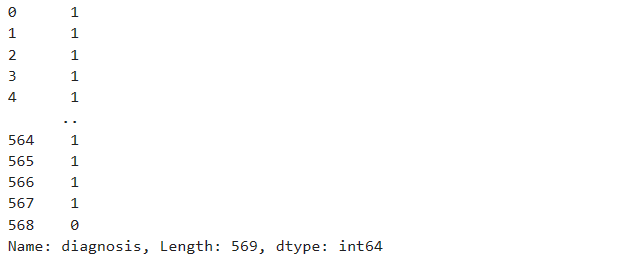
****

****

****

****

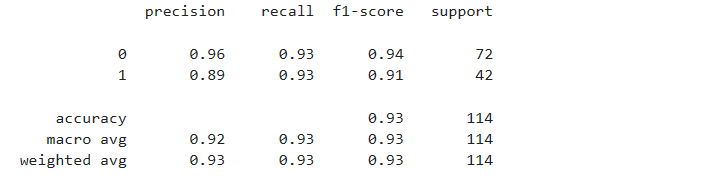
****

****

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 5**

|  |
| --- |
| **Title:** Study and Implement Bagging using Random Forests.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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.ensemble import RandomForestClassifier

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import LabelEncoder

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('Kyphosis.csv')

df.head()

df.info()

df.describe()

df.isnull().sum()

# Visualizing the distribution of the target variable

plt.figure(figsize=(6, 4))

sns.countplot(data=df, x='Kyphosis')

plt.title('Kyphosis Distribution')

plt.xlabel('Kyphosis')

plt.ylabel('Count')

plt.show()

# Encode the 'Kyphosis' column into numeric values

label\_encoder = LabelEncoder()

df['Kyphosis'] = label\_encoder.fit\_transform(df['Kyphosis'])

# Visualizing the correlation between features

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

df['Kyphosis'].value\_counts()

- \*\*0 : Absent\*\*

- \*\*1 : Present\*\*

X = df[['Age', 'Number', 'Start']]

print(X)

y = df['Kyphosis']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 2)

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

y\_pred = rf.predict(X\_test)

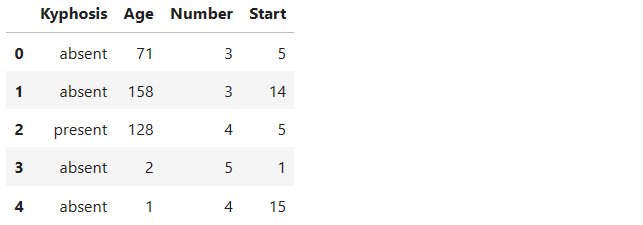
accuracy = accuracy\_score(y\_test, y\_pred)

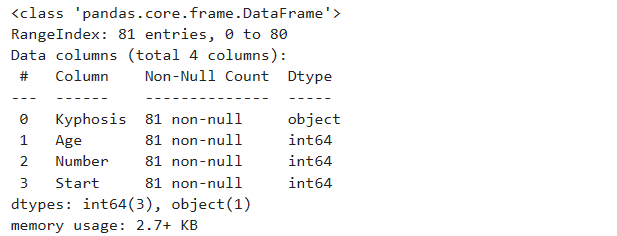
print('Accuracy is :', accuracy)

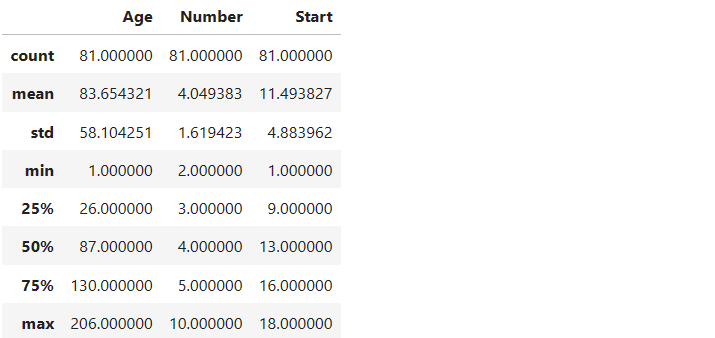
print(classification\_report(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred))

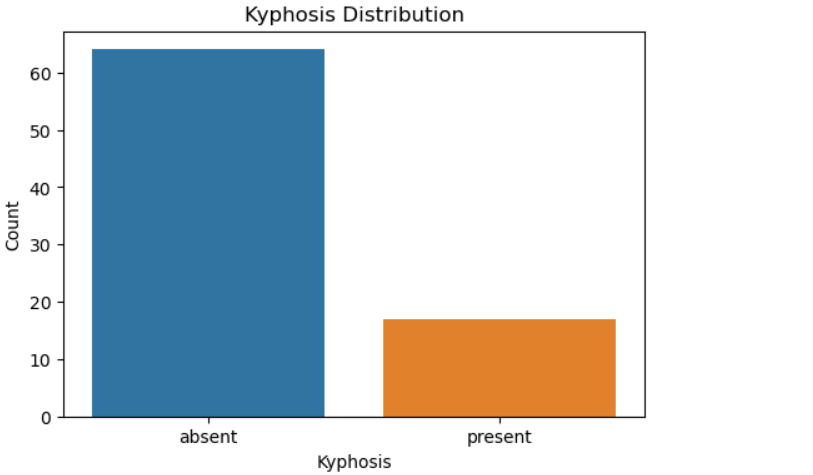
**Output:**

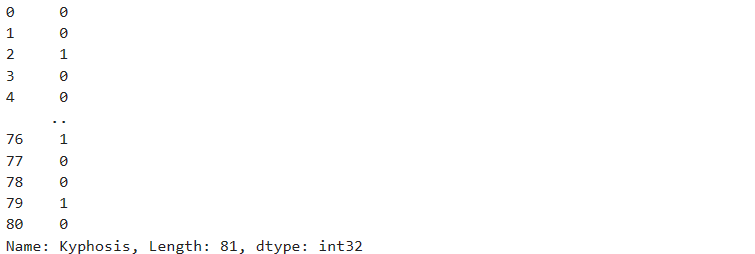
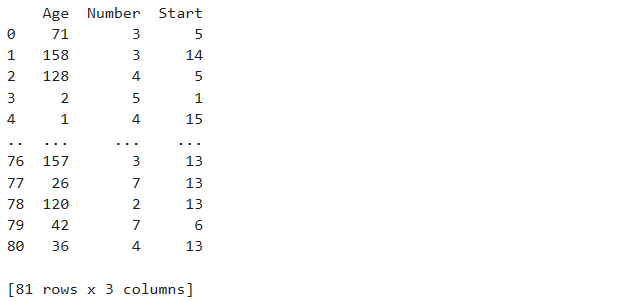
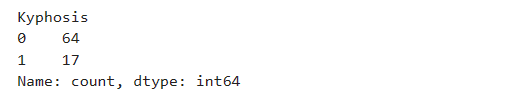
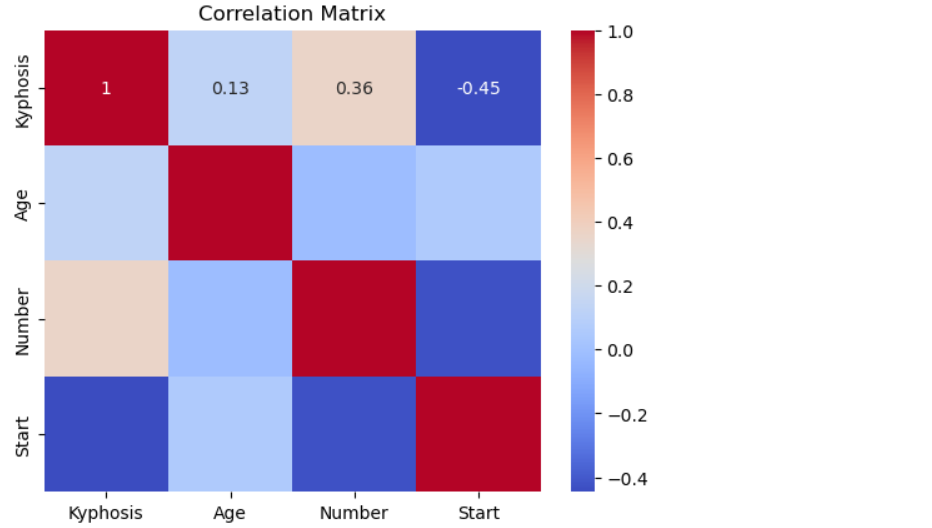
****

****

****

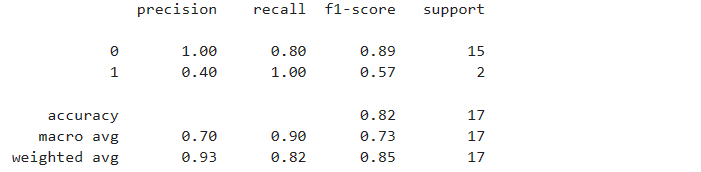
****

****

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 6**

|  |
| --- |
| **Title:** Study and Implement Naive Bayes.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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.naive\_bayes import GaussianNB

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

cancer\_df = pd.read\_csv('breast-cancer.csv')

cancer\_df.head()

cancer\_df.info()

cancer\_df.describe()

cancer\_df.isnull().sum()

\*\*To predict whether the cancer type is Malignant(M)[1] or Benign(B)[0]\*\*

# diagnosis column values

cancer\_df['diagnosis'].value\_counts()

# Visualizing the distribution of the target variable

plt.figure(figsize=(6, 4))

sns.countplot(data=cancer\_df, x='diagnosis')

plt.title('Diagnosis Distribution')

plt.xlabel('Diagnosis')

plt.ylabel('Count')

plt.show()

# Visualizing the distribution of features

plt.figure(figsize=(16, 10))

for i, feature in enumerate(cancer\_df.columns[2:12]): # Excluding the first two non-numeric columns

plt.subplot(3, 4, i+1)

sns.histplot(data=cancer\_df, x=feature, kde=True, bins=20)

plt.title(feature)

plt.tight\_layout()

plt.show()

# Encoding diagnosis Column

cancer\_df.replace({"diagnosis":{"M":1,"B":0}}, inplace=True)

# Visualizing the correlation between features

sns.heatmap(cancer\_df.corr(), cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

X = cancer\_df.drop(['diagnosis','id'], axis = 1)

print(X)

y = cancer\_df['diagnosis']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,stratify=y,random\_state=2)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

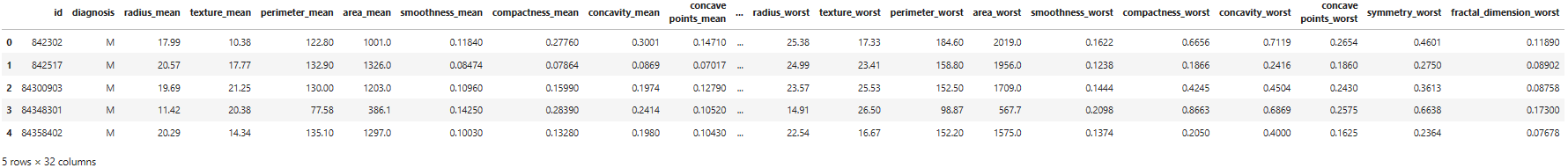
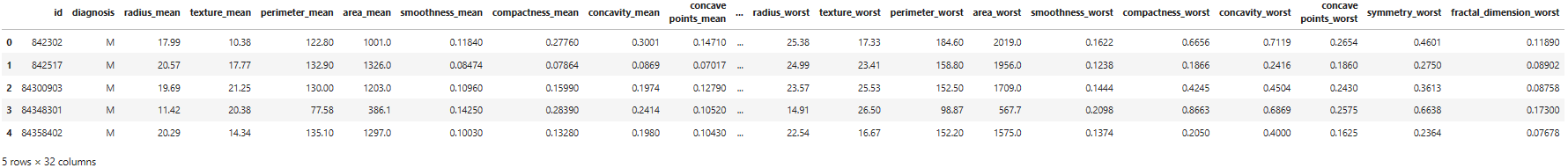
accuracy = accuracy\_score(y\_test, y\_pred)

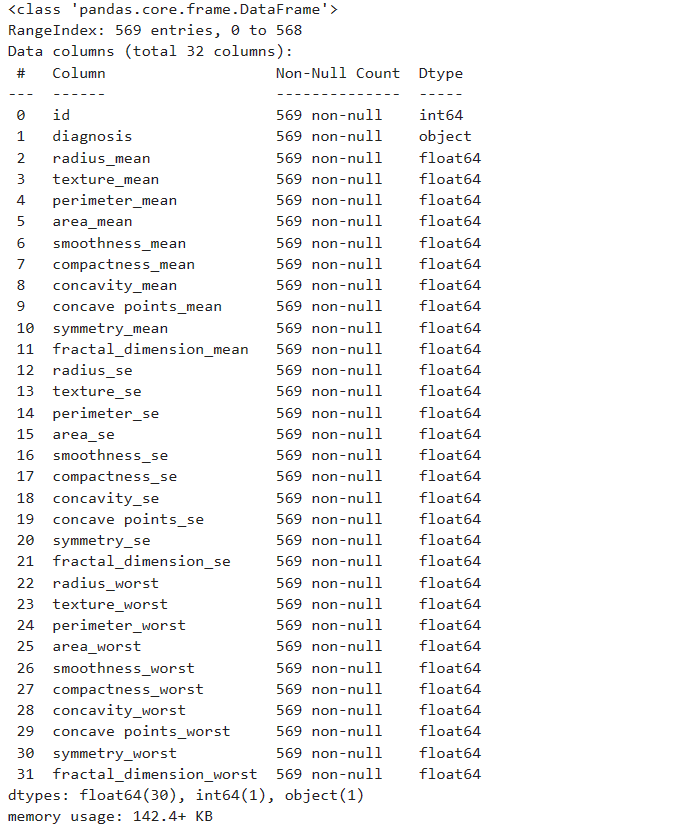
print('Accuracy: ', accuracy)

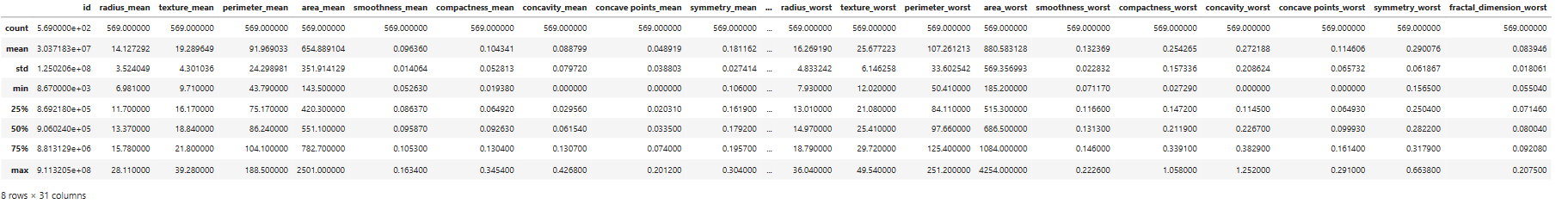
print(classification\_report(y\_test, y\_pred))

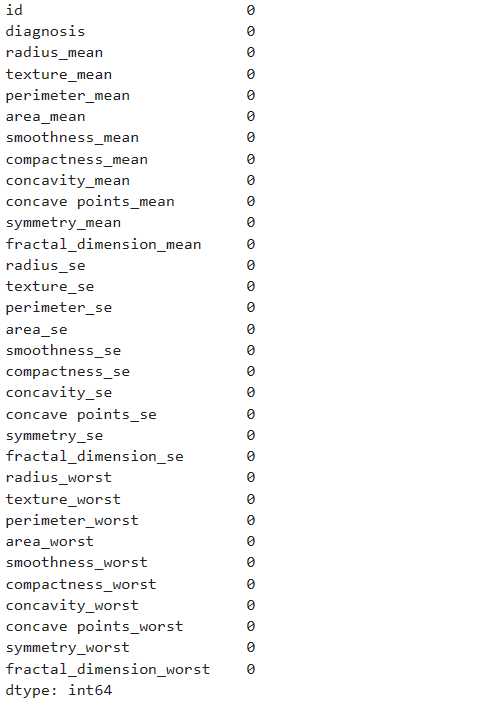
print(confusion\_matrix(y\_test, y\_pred))

**Output:**

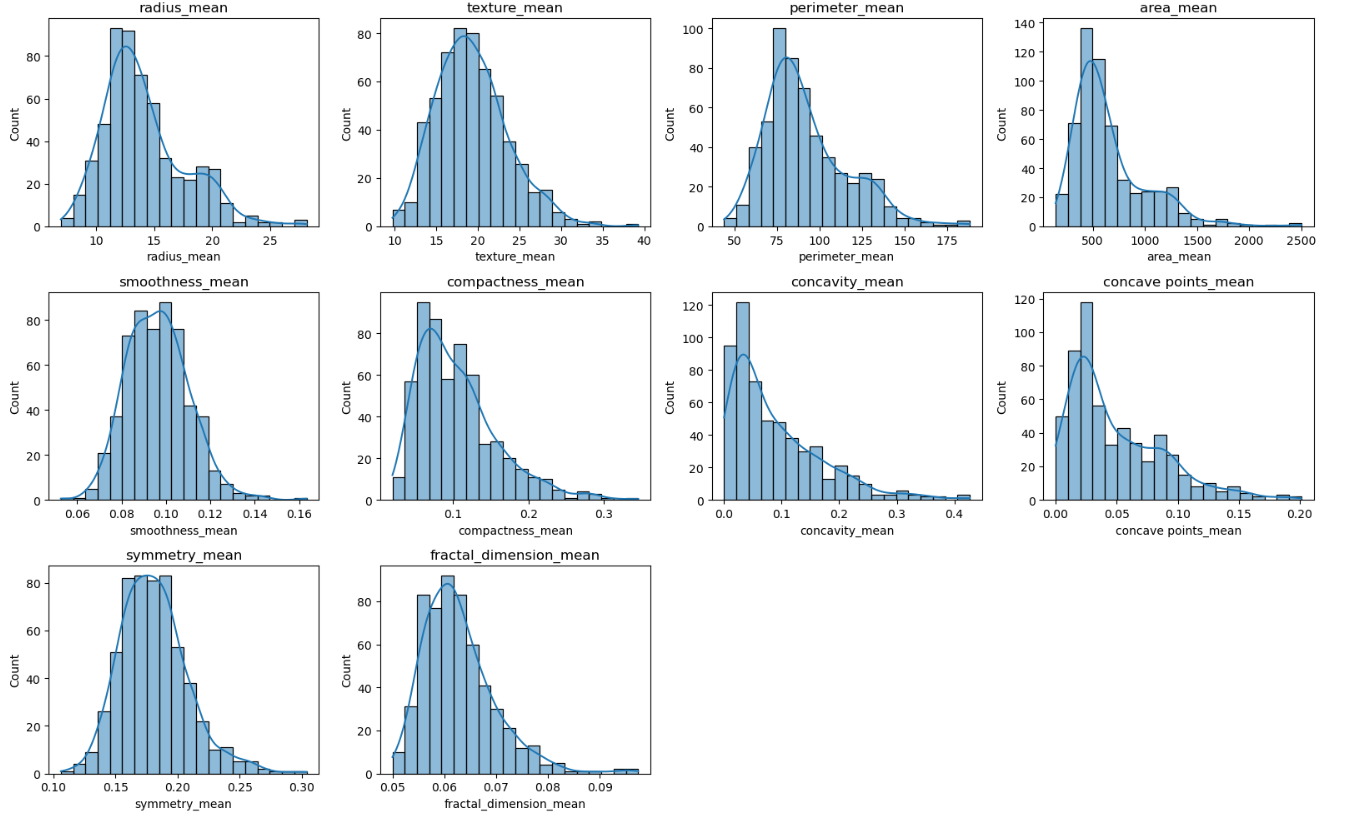
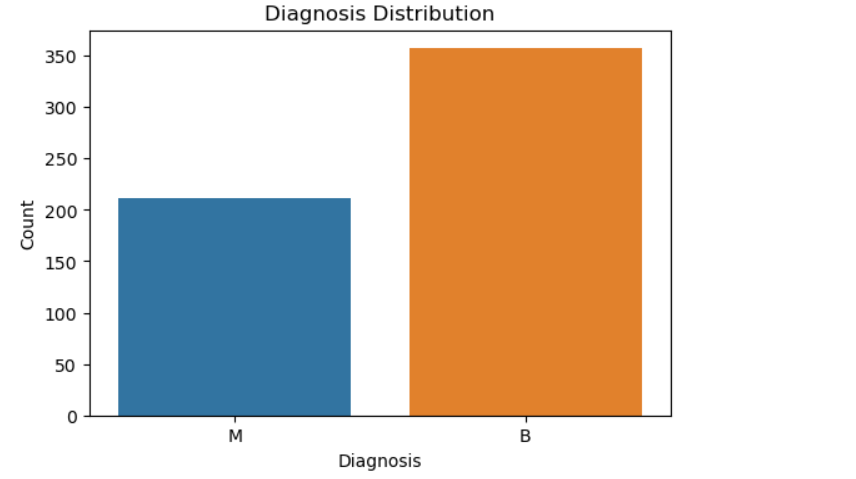
****

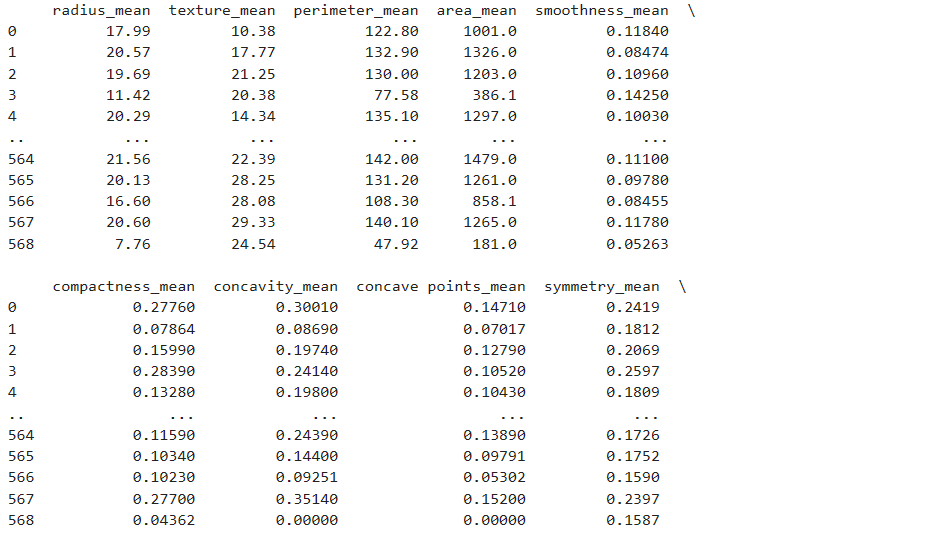
****

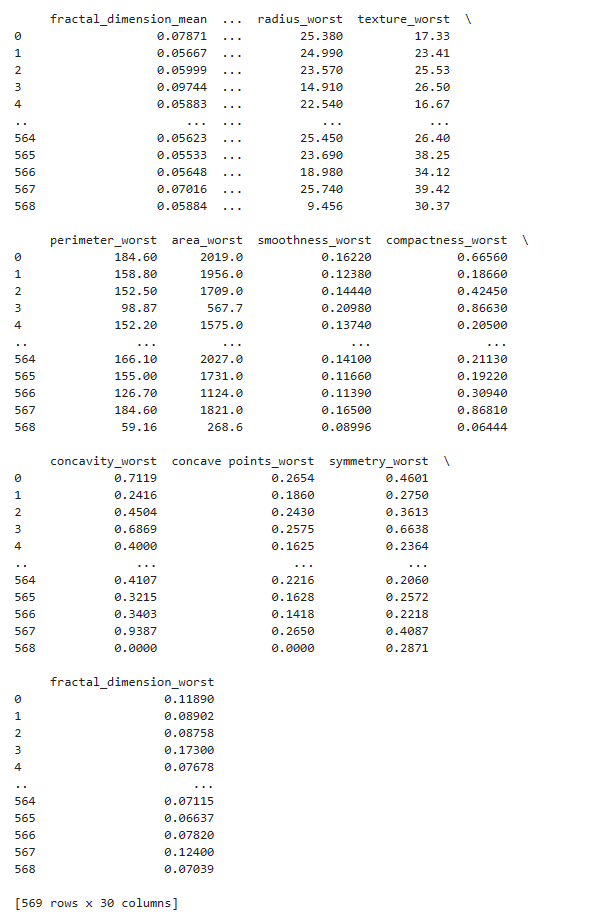
****

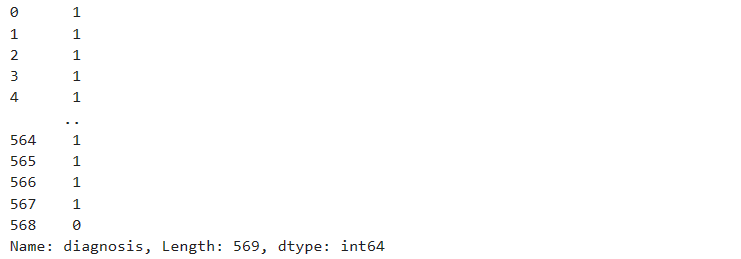
****

****

****

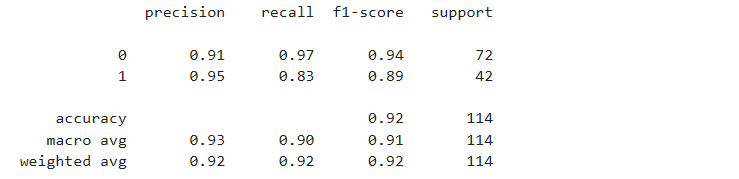
****

****

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 7**

|  |
| --- |
| **Title:** Study and Implement Decision Trees.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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.tree import DecisionTreeClassifier

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import LabelEncoder

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('Kyphosis.csv')

df.head()

\*Kyphosis Distribution: A count plot showing the distribution of the target variable 'Kyphosis'.\*

# Visualize the class distribution of the target variable

plt.figure(figsize=(6, 4))

sns.countplot(data=df, x='Kyphosis')

plt.title('Kyphosis Distribution')

plt.xlabel('Kyphosis')

plt.ylabel('Count')

plt.show()

# Encode the 'Kyphosis' column into numeric values

label\_encoder = LabelEncoder()

df['Kyphosis'] = label\_encoder.fit\_transform(df['Kyphosis'])

# Visualize the correlation between features

sns.heatmap(df.corr(), cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

X = df[['Age', 'Number', 'Start']]

print(X)

y = df['Kyphosis']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 2)

clf\_entropy = DecisionTreeClassifier(criterion = 'entropy', random\_state = 100, max\_depth = 5, min\_samples\_leaf = 5)

clf\_entropy.fit(X\_train, y\_train)

y\_pred = clf\_entropy.predict(X\_test)

y\_pred

accuracy = accuracy\_score(y\_test, y\_pred)

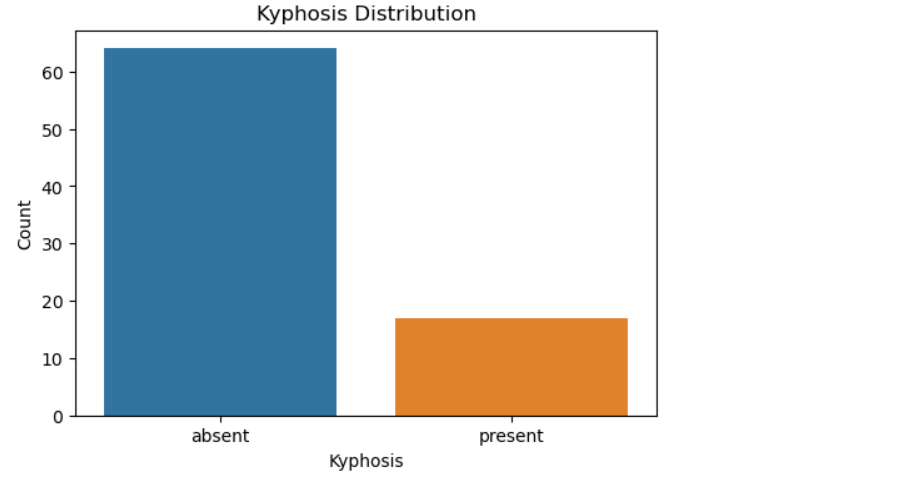
print('Accuracy is :', accuracy)

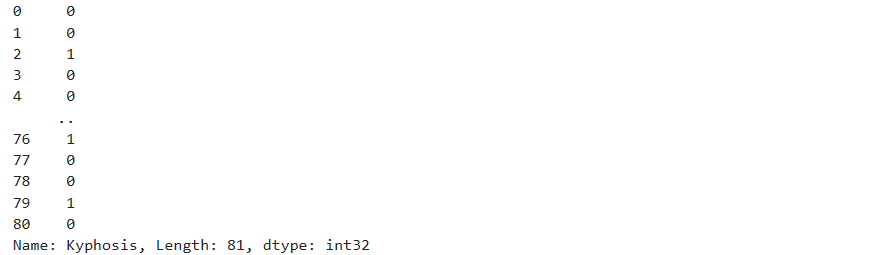
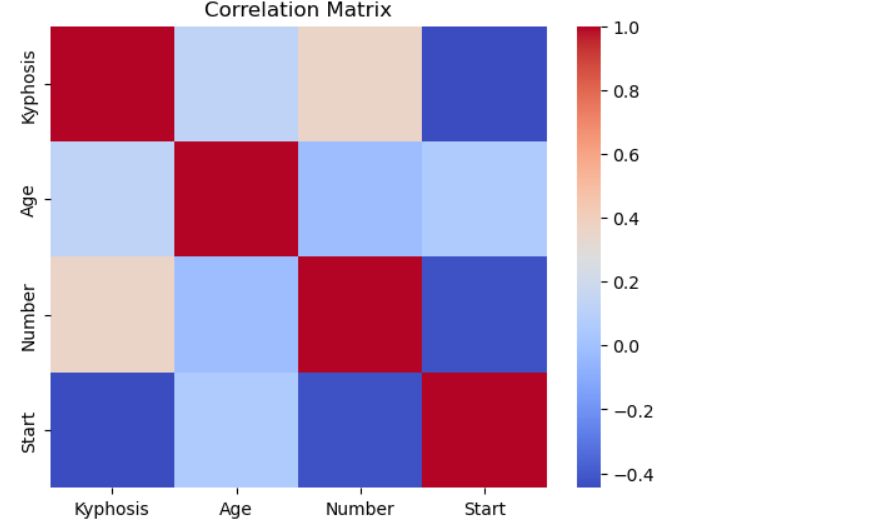
print(classification\_report(y\_test, y\_pred))

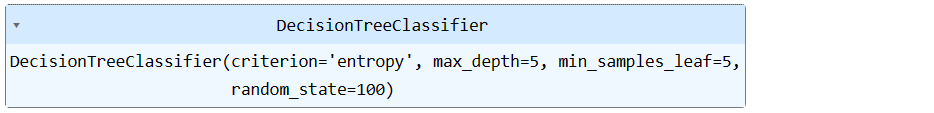
print(confusion\_matrix(y\_test, y\_pred))

**Output:**

****

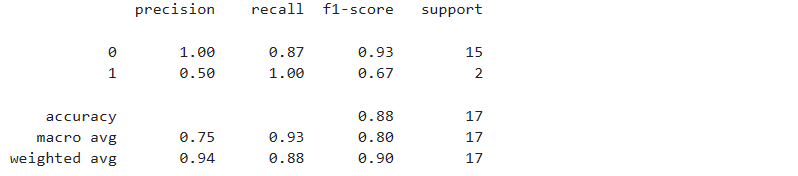
****

****

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 8**

|  |
| --- |
| **Title:** Study and Implement K-means Clustering to Find Natural Patterns in Data.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

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.cluster import KMeans

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('Mall\_Customers.csv')

df.head()

df.info()

df.describe()

df.isnull().sum()

# Check for duplicate rows

print("Number of duplicate rows:", df.duplicated().sum())

# Visualize the distribution of numerical features

plt.figure(figsize=(12, 12))

sns.pairplot(df, diag\_kind='kde')

plt.suptitle('Pairplot of Numerical Features', y=1.02) # Placing the title at the top

plt.show()

X = df.iloc[:, [3,4]].values

print(X)

# Finding optimal number of clusters using the elbow method

wcss\_list = [] # Initializing the list for the values of WCSS

# Using for loop for iterations from 1 to 10

for i in range(1,11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=2)

kmeans.fit(X)

wcss\_list.append(kmeans.inertia\_)

plt.plot(range(1,11), wcss\_list)

plt.title('The Elbow Method Graph')

plt.xlabel('Number of clusters(k)')

plt.ylabel('wcss\_list')

plt.show()

# Training the K-Means Model on a dataset

kmeans = KMeans(n\_clusters=5, init='k-means++', random\_state=2)

y\_predict = kmeans.fit\_predict(X)

y\_predict

from collections import Counter

print(Counter(y\_predict))

### Visualizing the clusters

plt.figure(figsize = (9, 7))

# Scatter plot for customers in cluster 0 (Medium Income - Medium Score)

plt.scatter(X[y\_predict == 0, 0], X[y\_predict == 0, 1], s=100, c='blue', label='Medium Income - Medium Score')

# Scatter plot for customers in cluster 1 (High Income - Low Score)

plt.scatter(X[y\_predict == 1, 0], X[y\_predict == 1, 1], s=100, c='green', label='High Income - Low Score')

# Scatter plot for customers in cluster 2 (Low Income - Low Score)

plt.scatter(X[y\_predict == 2, 0], X[y\_predict == 2, 1], s=100, c='red', label='Low Income - Low Score')

# Scatter plot for customers in cluster 3 (Low Income - High Score)

plt.scatter(X[y\_predict == 3, 0], X[y\_predict == 3, 1], s=100, c='cyan', label='Low Income - High Score')

Scatter plot for customers in cluster 4 (High Income - High Score)

plt.scatter(X[y\_predict == 4, 0], X[y\_predict == 4, 1], s=100, c='magenta', label='High Income - High Score')

# Scatter plot for centroids (cluster centers)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='yellow', label='Centroids')

plt.title('Clusters of customers')

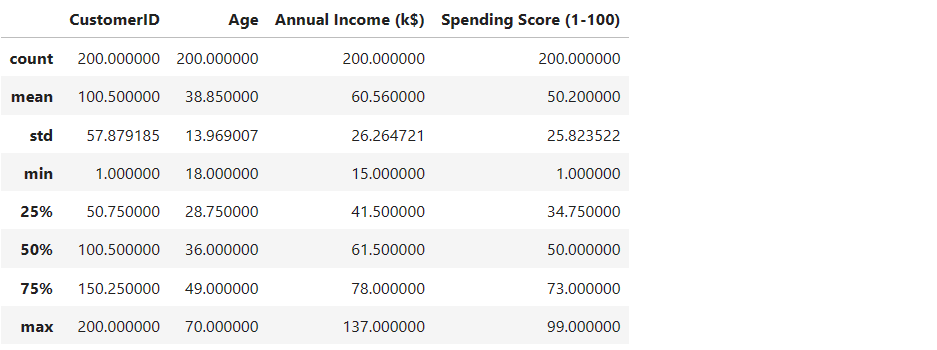
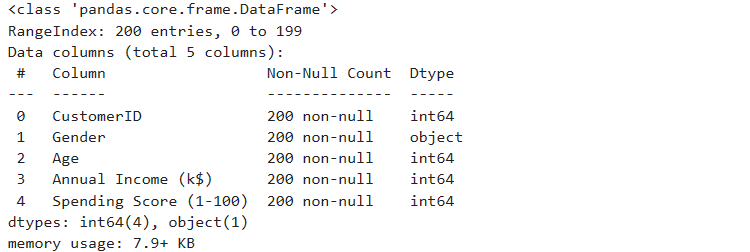
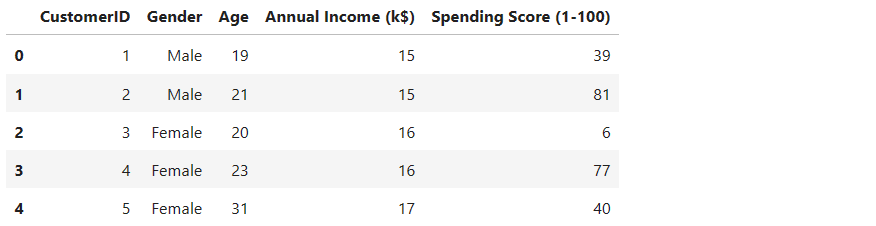
plt.xlabel('Annual Income (k$)')

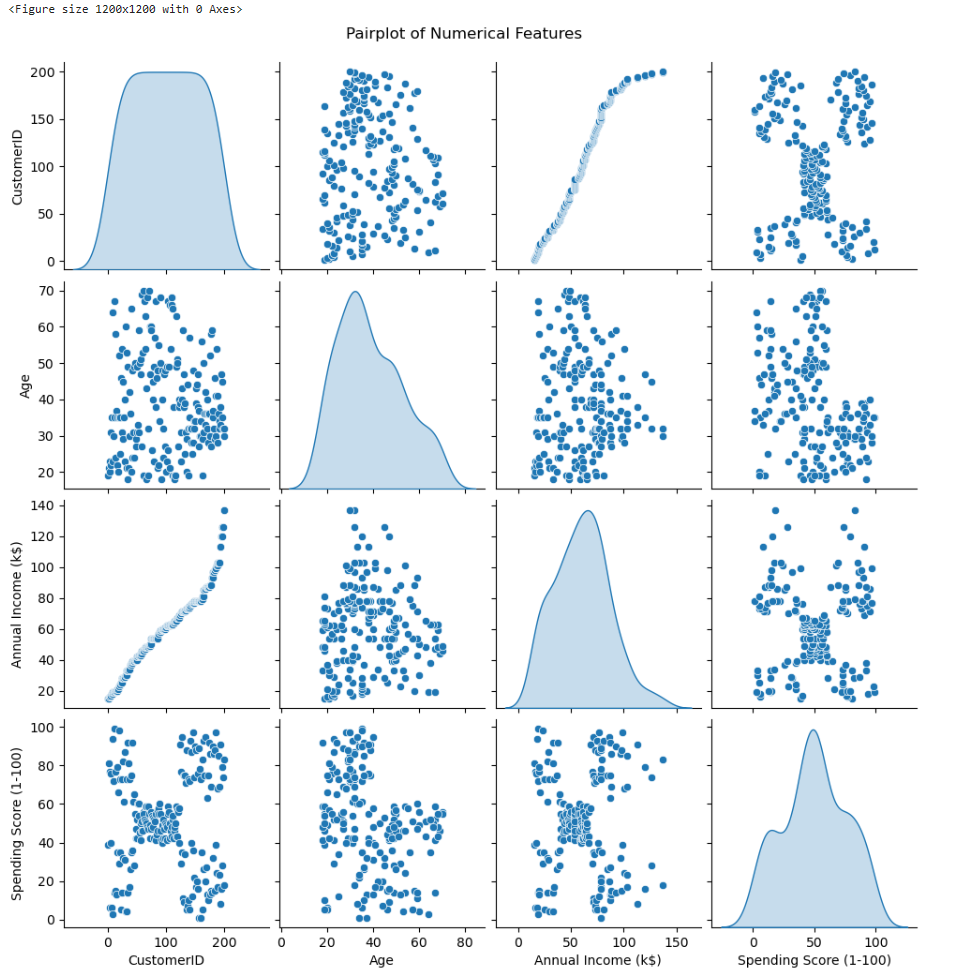
plt.ylabel('Spending Score (1 - 100)')

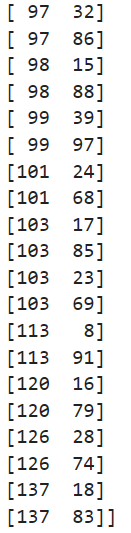
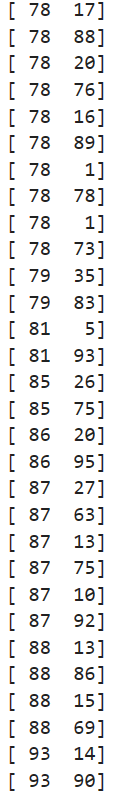
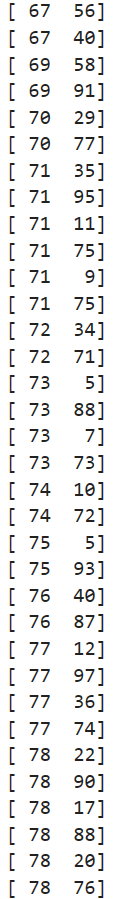
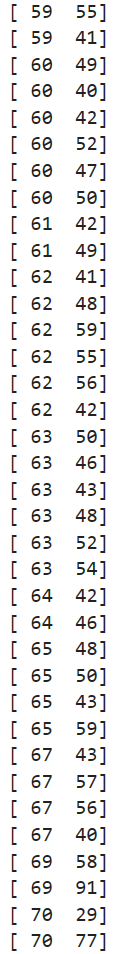
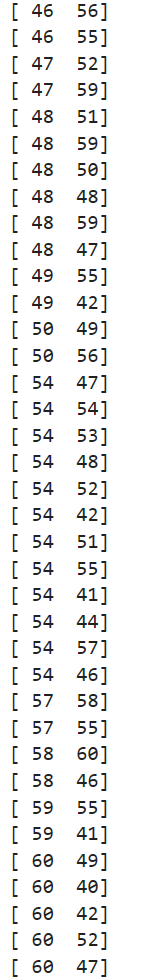
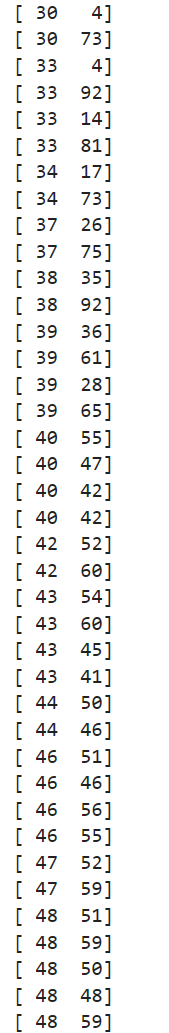
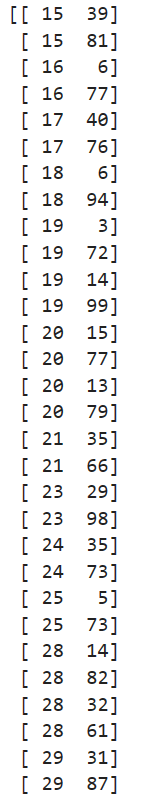
plt.legend()

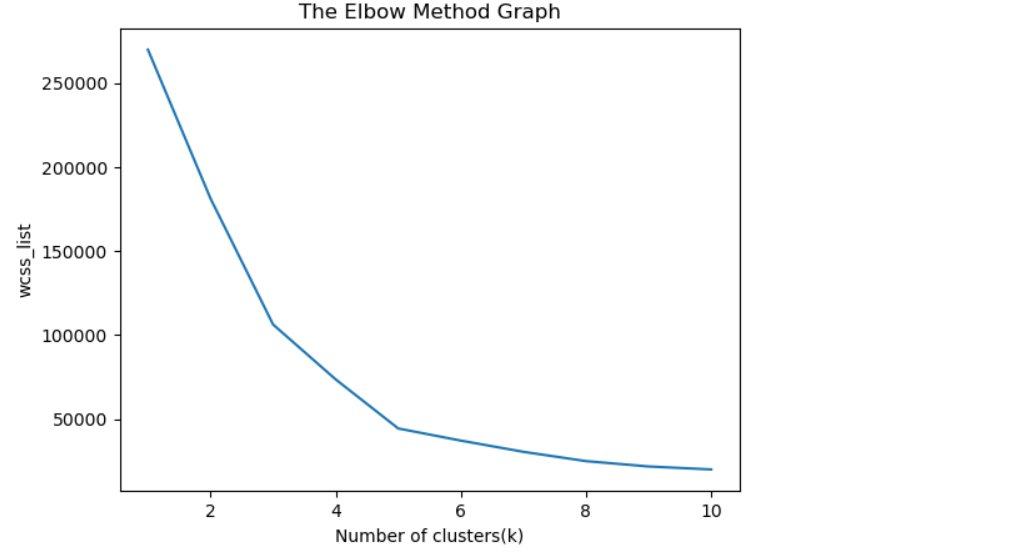
plt.show()

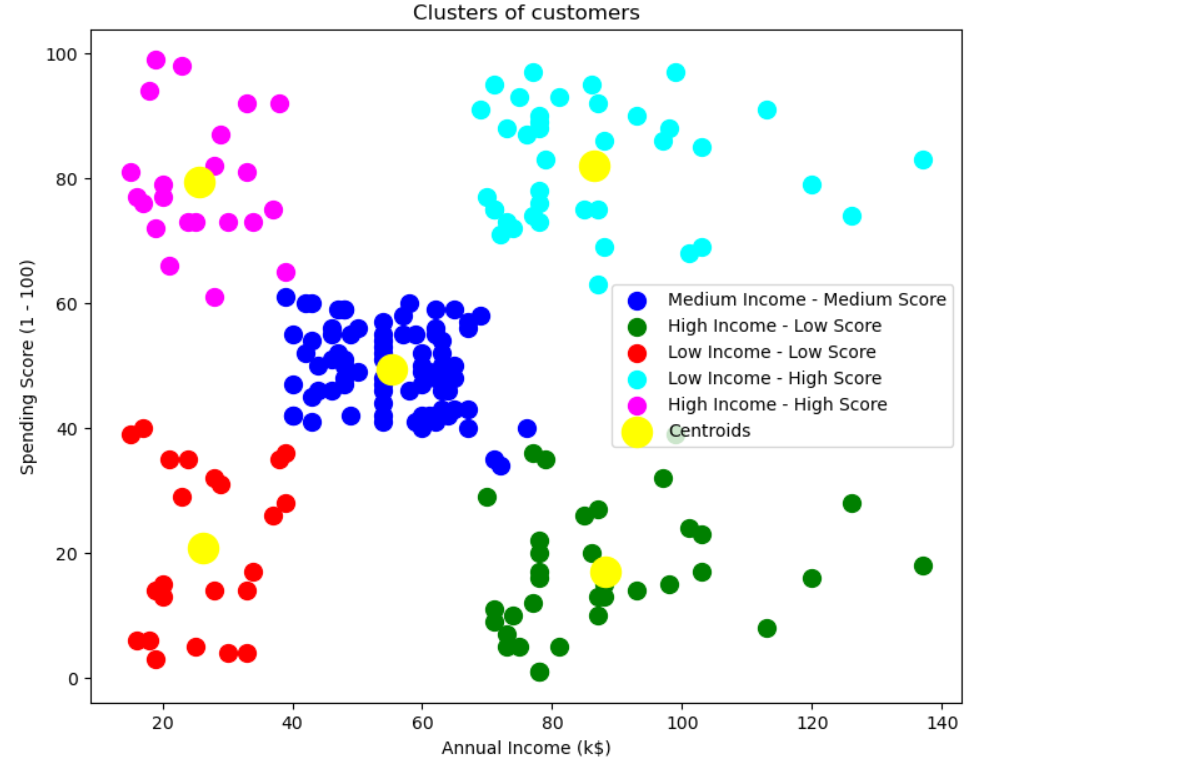
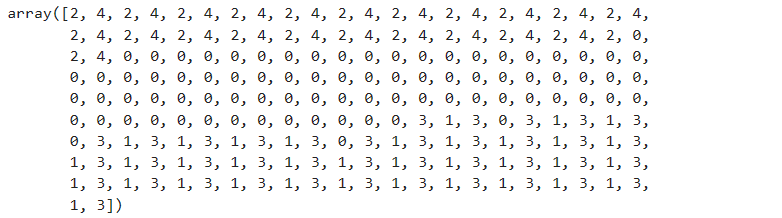
**Output:**

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 9**

|  |
| --- |
| **Title:** Study and Implement Gaussian Mixture Model Using the Expectation Maximization.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.mixture import GaussianMixture

from sklearn.cluster import KMeans

from sklearn import metrics

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

# Load the dataset

iris\_df = pd.read\_csv('iris.csv')

iris\_df.head()

iris\_df.info()

iris\_df.describe()

# Visualize the distribution of each feature

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

sns.pairplot(iris\_df)

plt.suptitle('Pairplot of Iris Dataset', y=1.02)

plt.show()

# Select the first two columns

X = iris\_df.iloc[:, [0, 1]]

# Plot the data

plt.scatter(X.iloc[:, 0], X.iloc[:, 1])

plt.title('Scatter plot of Iris dataset')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

# Implement Gaussian Mixture Model

gmm = GaussianMixture(n\_components=3)

# Fit the GMM model to the dataset

gmm.fit(X)

# Assign a label to each sample

labels = gmm.predict(X)

iris\_df['labels'] = labels

# Visualize the clusters

colors = ['r', 'g', 'b']

for label, color in zip(range(3), colors):

cluster = iris\_df[iris\_df['labels'] == label]

plt.scatter(cluster.iloc[:, 0], cluster.iloc[:, 1], c=color, label=f'Cluster {label}')

plt.title('Clusters using Gaussian Mixture Model')

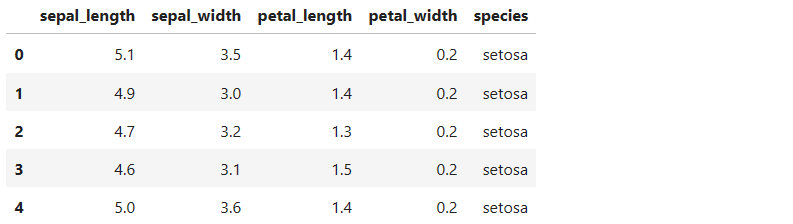
plt.xlabel('Feature 1')

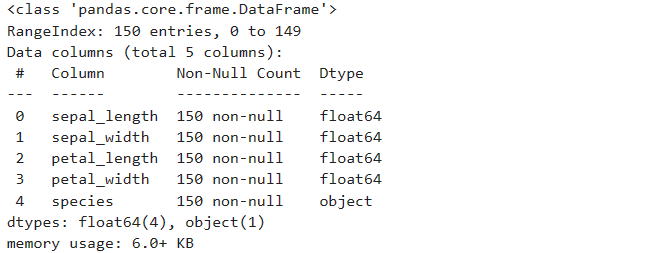
plt.ylabel('Feature 2')

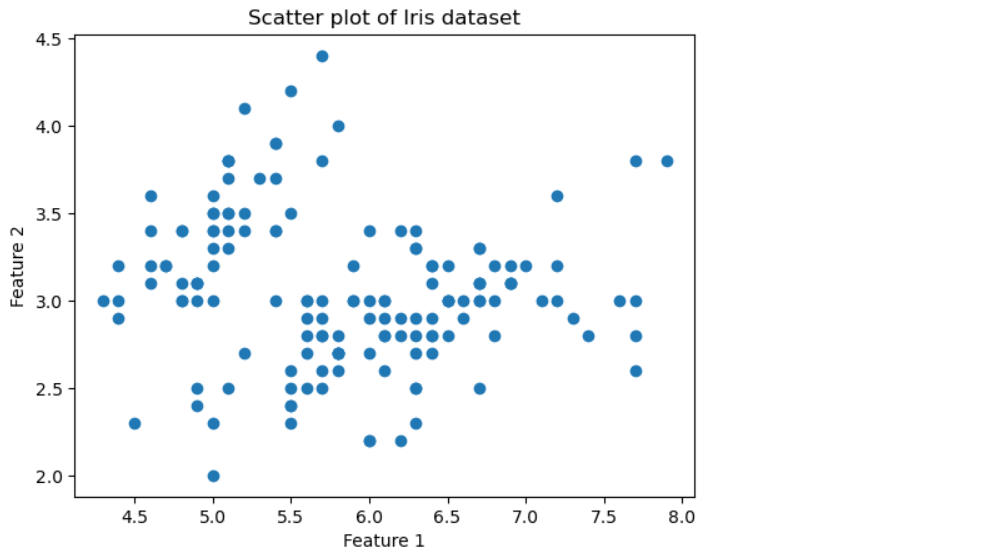
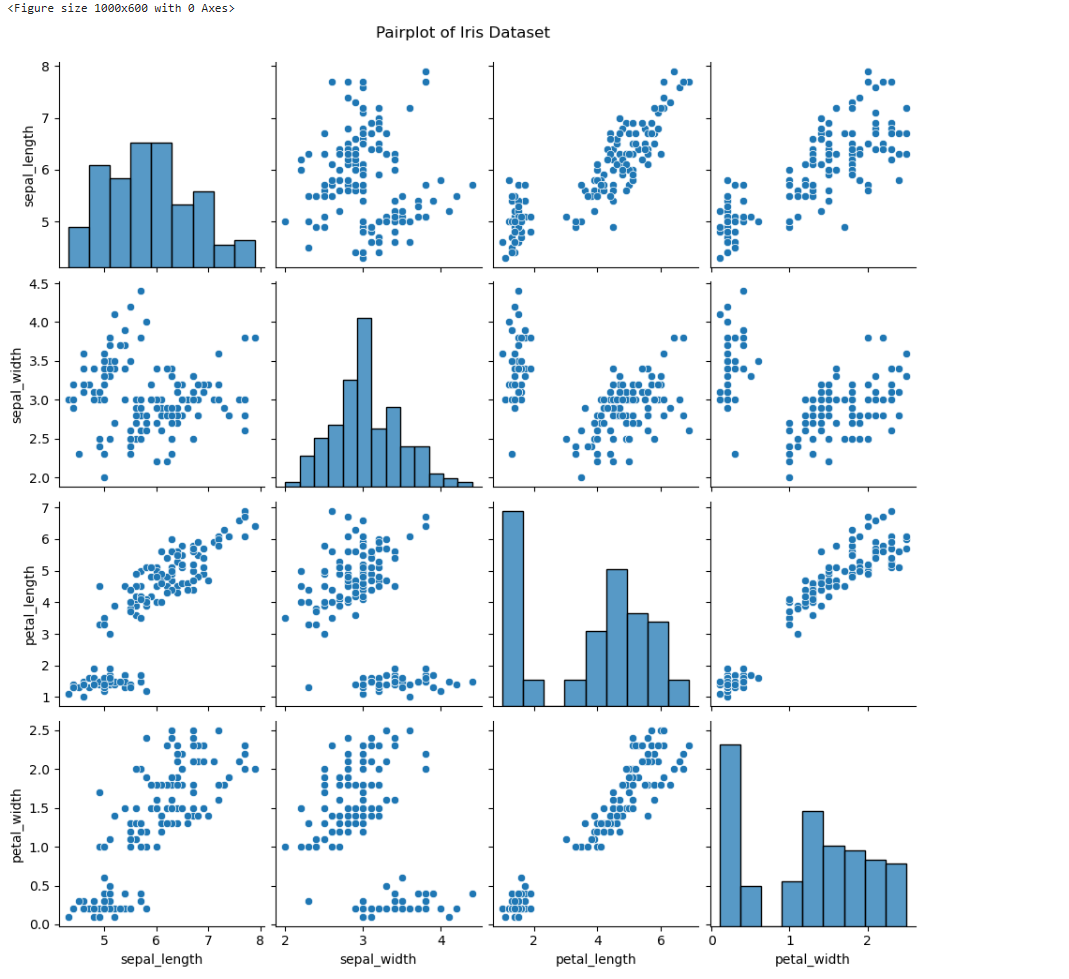
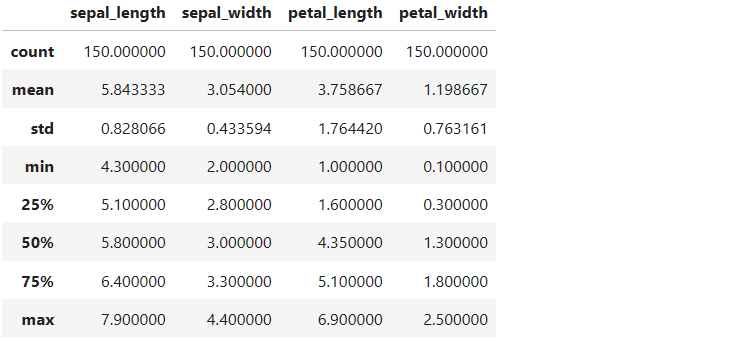
plt.legend()

plt.show()

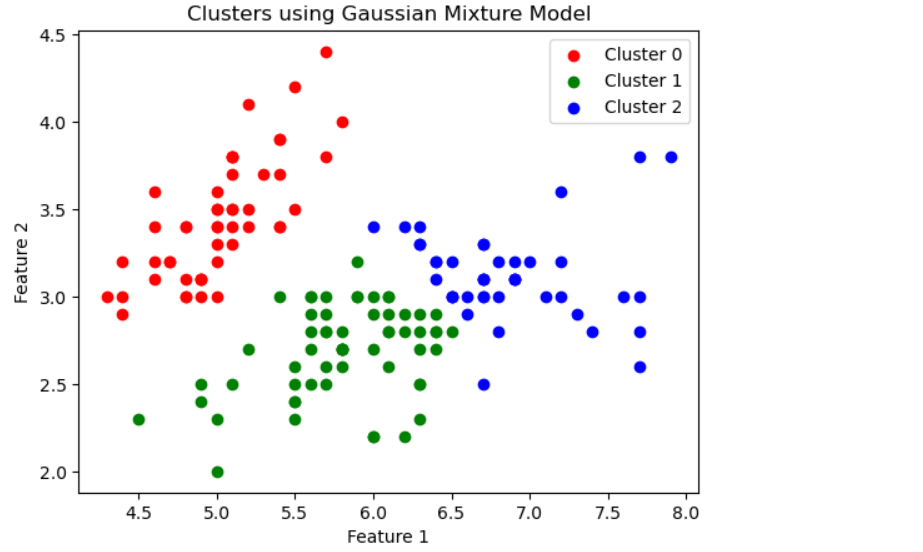
**Output:**

****

****

****

****

****

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Experiment 10**

|  |
| --- |
| **Title:** Study and Implement Classification based on association rules.  **Abstract:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn import metrics

# To ignore warnings

import warnings

warnings.filterwarnings("ignore")

# Load the dataset

df = pd.read\_csv('stocks.csv')

df.head()

# Preprocess the dataset

df = df.astype(bool).astype(int)

# This preprocessing step effectively converts all non-zero values in the DataFrame to 1 and all zero values to 0, effectively binarizing the data.

# Apply Apriori algorithm to find frequent itemsets

frequent\_itemsets = apriori(df, min\_support=0.5, use\_colnames=True)

# Generate association rules

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

# Display the generated association rules

print(rules)

# Use association rules to generate features

# For each transaction, count the number of antecedents that are present

def count\_antecedents(row):

antecedents = set(rules['antecedents'])

count = sum(1 for item in row.index if item in antecedents and row[item] == 1)

return count

df['antecedent\_count'] = df.apply(count\_antecedents, axis=1)

# Define features (X) and target variable (y)

X = df[['antecedent\_count']]

y = df['Volume']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a classifier (Random Forest as an example)

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# Predict on the testing set

y\_pred = clf.predict(X\_test)

# Evaluate classifier's performance

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

classification\_report = metrics.classification\_report(y\_test, y\_pred)

print("Classification Report:\n", classification\_report)

confusion\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", confusion\_matrix)

# Visualize confusion matrix

sns.heatmap(confusion\_matrix, annot=True, cmap='Blues', fmt='g')

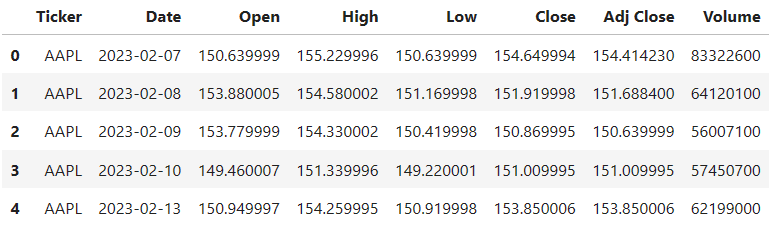
plt.xlabel('Predicted labels')

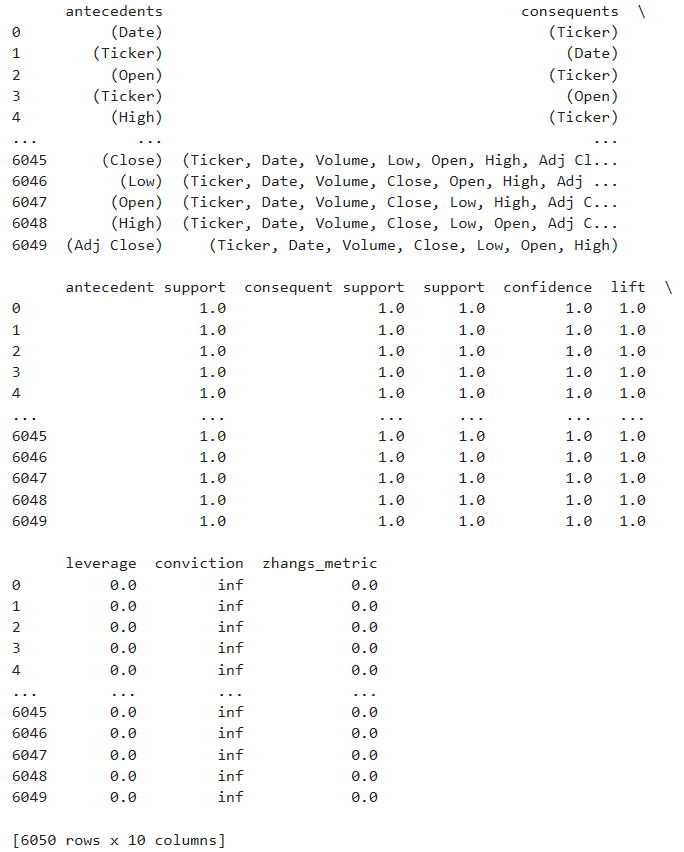
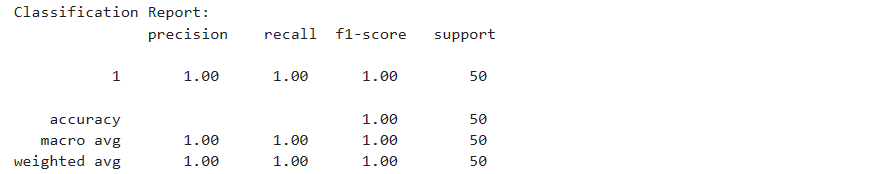
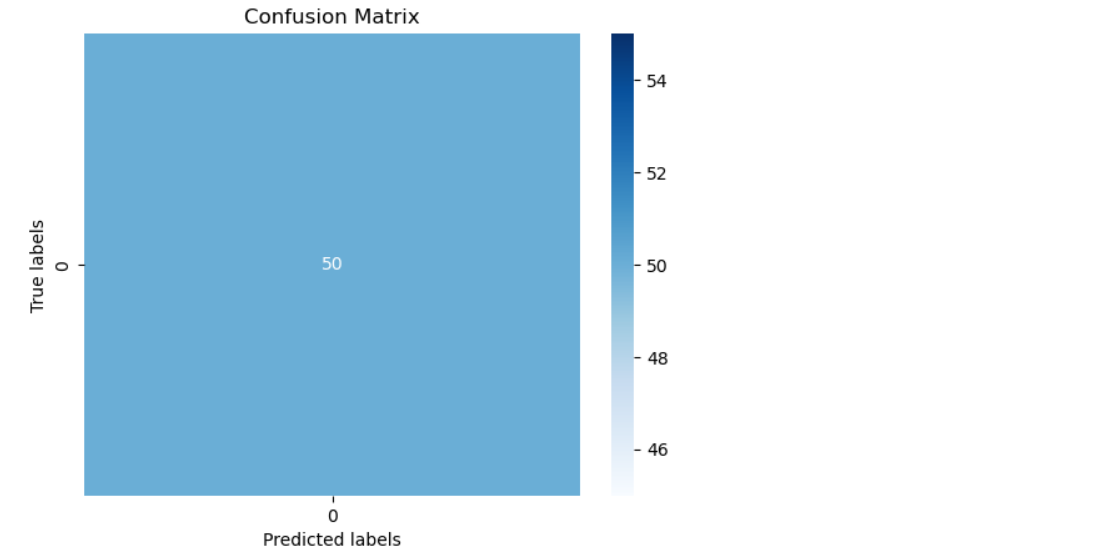
plt.ylabel('True labels')

plt.title('Confusion Matrix')

plt.show()

**Output:**

****

****    

**Learning Outcomes:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |