Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

DEPARTMENT OF COMPUTER SCIENCE [PG]

Practical Record on

MCC2L2C31 – MACHINE LEARNING PRACTICAL

Submitted by

Mr. ALLURI ABISHEK KUMAR

24MCAB07

KRISTU JAYANTI COLLEGE (Autonomous)
(Reaccredited 'A++' Grade by NAAC)
K.Narayanapura, Kothanur Post, Bengaluru– 560 077
2025 – 2026

Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

DEPARTMENT OF COMPUTER SCIENCE [PG]

MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to Certify that Mr. Alluri Abishek Kumar bearing Registration No. ***24MCAB07*** *of III Semester MCA has successfully completed the Practical exercises for the course* ***MCC2L2C31- Machine Learning Practical*** *during the academic year 2025 – 2026.*

Faculty In-Charge   Head of the Department

Valued by the Examiners   **Center : Kristu Jayanti College**

1. ________________________   **Date :**

2. ________________________

Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

TABLE OF CONTENTS

Sl. No	Date	Program Title	Page No.
1.	02/08/2025	Program to perform basic Matrix operation	1-3
2.	11/08/2025	Program to implement basic Operation on set and Dictionary	4-6
3.	18/08/2025	Program to implement Regular Expression	7-8
4.	25/08/2025	Program to Perform File Handling Operation	9-12
5.	30/08/2025	Program to store details of student using Data Frame and perform primary Operation	13-14
6.	01/09/2025	Program to implement Linear Regression problem	15-16
7.	08/09/2025	Program to implement Pre-Processing Technique	17-18
8.	15/09/2025	Program to implement Decision tree algorithm using given dataset	19-22
9.	20/09/2025	Program to implement Naive Bayse Algorithm	23-25
10.	04/10/2025	Program to implement Artificial Neural Network	26-28
11.	06/10/2025	Program to implement K-Means clustering algorithm	29-30
12.	13/10/2025	Program to implement Dimensionality reduction techniques.	31-32
13	27/10/2025	Program to visualize the data using 2D and 3D plots.	33-34

1. Program to perform basic matrix operation.

Aim:

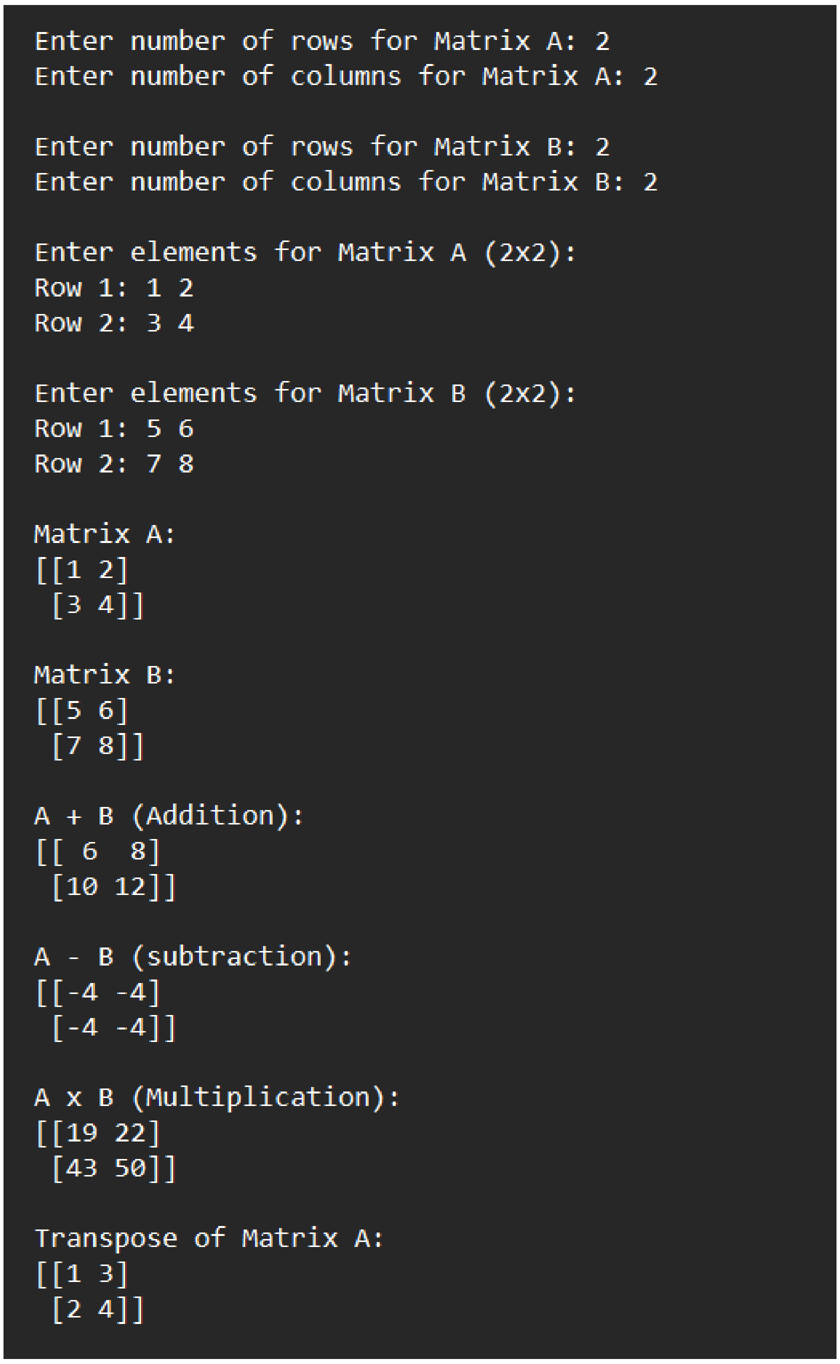
To develop a Program to perform basic matrix operation.

Source Code:

```
import numpy as np

def input_matrix(rows, cols, name):
    print(f"\nEnter elements for Matrix {name} ({rows}x{cols}):")
    matrix = []
    for i in range(rows):
        row = list(map(int, input(f"Row {i+1}: ").split()))
        while len(row) != cols:
            print(f" Please enter exactly {cols} elements.")
            row = list(map(int, input(f"Row {i+1}: ").split()))
        matrix.append(row)
    return np.array(matrix)

r1 = int(input("Enter number of rows for Matrix A: "))
c1 = int(input("Enter number of columns for Matrix A: "))
r2 = int(input("\nEnter number of rows for Matrix B: "))
c2 = int(input("Enter number of columns for Matrix B: "))
A = input_matrix(r1, c1, 'A')
B = input_matrix(r2, c2, 'B')
print("\nMatrix A:")
print(A)
print("\nMatrix B:")
print(B)
if A.shape == B.shape:
    print("\nA + B (Addition):")
    print(A + B)
```

```
else:
    print("\n Addition not possible (different dimensions)")
if A.shape == B.shape:
    print("\nA - B (subtraction):")
    print(A - B)
else:
    print("\n subtraction not possible (different dimensions)")
if c1 == r2:
    product = np.dot(A, B)
    print("\nA x B (Multiplication):")
    print(product)
else:
    print("\n Multiplication not possible (A's columns != B's rows)")
print("\nTranspose of Matrix A:")
print(np.transpose(A))
```

Output:



```
Enter number of rows for Matrix A: 2
Enter number of columns for Matrix A: 2

Enter number of rows for Matrix B: 2
Enter number of columns for Matrix B: 2

Enter elements for Matrix A (2x2):
Row 1: 1 2
Row 2: 3 4

Enter elements for Matrix B (2x2):
Row 1: 5 6
Row 2: 7 8

Matrix A:
[[1 2]
 [3 4]]

Matrix B:
[[5 6]
 [7 8]]

A + B (Addition):
[[ 6  8]
 [10 12]]

A - B (subtraction):
[[-4 -4]
 [-4 -4]]

A x B (Multiplication):
[[19 22]
 [43 50]]

Transpose of Matrix A:
[[1 3]
 [2 4]]
```

2. Program to perform basic operation with dictionary and set data types.

Aim:

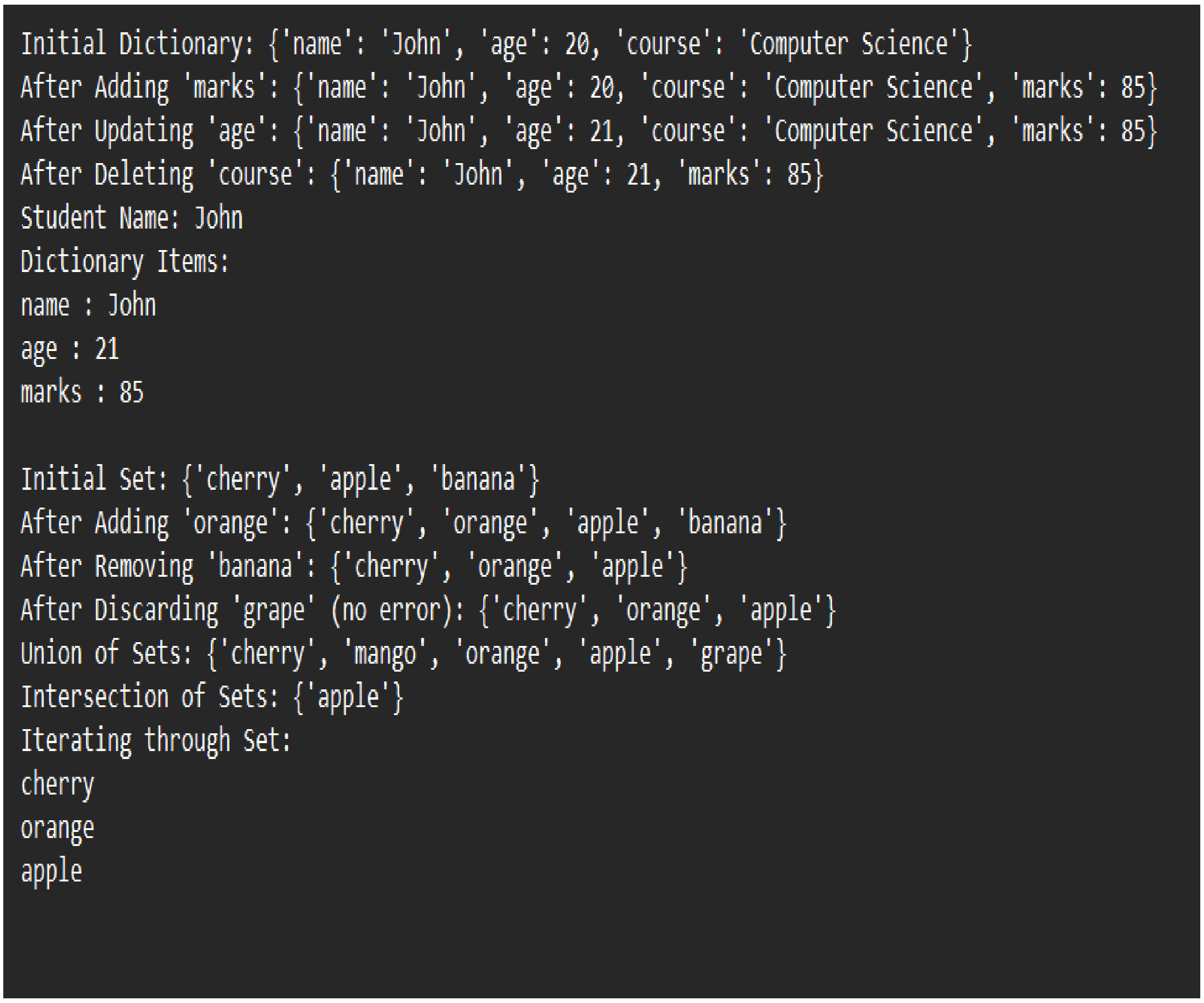
To develop a Program to perform basic operation with dictionary and set data types.

Source Code:

```
student = {
    "name": "John",
    "age": 20,
    "course": "Computer Science"
}
print("Initial Dictionary:", student)
student["marks"] = 85
print("After Adding 'marks':", student)
student["age"] = 21
print("After Updating 'age':", student)
del student["course"]
print("After Deleting 'course':", student)
print("Student Name:", student["name"])
print("Dictionary Items:")
for key, value in student.items():
    print(f"{key} : {value}")

# Set Operations
fruits = {"apple", "banana", "cherry"}
print("\nInitial Set:", fruits)
fruits.add("orange")
print("After Adding 'orange':", fruits)
fruits.remove("banana")
print("After Removing 'banana':", fruits)
fruits.discard("grape")
```

```
print("After Discarding 'grape' (no error):", fruits)
more_fruits = {"mango", "apple", "grape"}
union_set = fruits.union(more_fruits)
print("Union of Sets:", union_set)
intersection_set = fruits.intersection(more_fruits)
print("Intersection of Sets:", intersection_set)
print("Iterating through Set:")
for fruit in fruits:
    print(fruit)
```

Output:



```
Initial Dictionary: {'name': 'John', 'age': 20, 'course': 'Computer Science'}
After Adding 'marks': {'name': 'John', 'age': 20, 'course': 'Computer Science', 'marks': 85}
After Updating 'age': {'name': 'John', 'age': 21, 'course': 'Computer Science', 'marks': 85}
After Deleting 'course': {'name': 'John', 'age': 21, 'marks': 85}
Student Name: John
Dictionary Items:
name : John
age : 21
marks : 85

Initial Set: {'cherry', 'apple', 'banana'}
After Adding 'orange': {'cherry', 'orange', 'apple', 'banana'}
After Removing 'banana': {'cherry', 'orange', 'apple'}
After Discarding 'grape' (no error): {'cherry', 'orange', 'apple'}
Union of Sets: {'cherry', 'mango', 'orange', 'apple', 'grape'}
Intersection of Sets: {'apple'}
Iterating through Set:
cherry
orange
apple
```

3. Program to implement regular expressions in python.

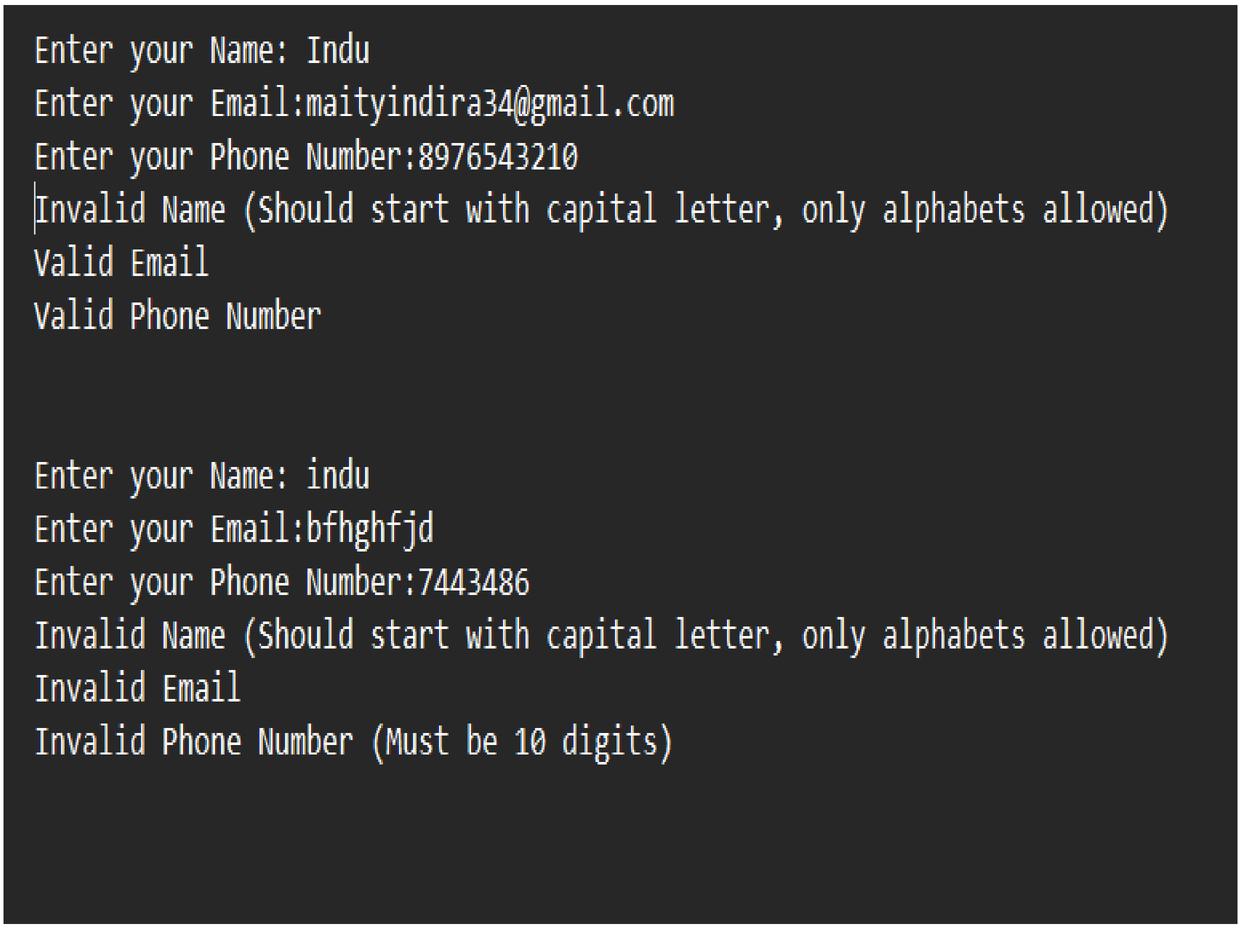
Aim:

To develop a Program to implement Regular Expression in Python.

Source Code:

```
import re

name_pattern = r"^[A-Z][a-zA-Z\s]+"
email_pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"
phone_pattern = r"^\d{10}$"
name = input("Enter your Name: ")
email = input("Enter your Email:")
phone = input("Enter your Phone Number:")
if re.match(name_pattern, name):
    print(" Valid Name")
else:
    print("Invalid Name (Should start with capital letter, only alphabets allowed)")
if re.match(email_pattern, email):
    print("Valid Email")
else:
    print("Invalid Email")
if re.match(phone_pattern, phone):
    print(" Valid Phone Number")
else:
    print("Invalid Phone Number (Must be 10 digits)")
```

Output:



```
Enter your Name: Indu
Enter your Email:maityindira34@gmail.com
Enter your Phone Number:8976543210
Invalid Name (Should start with capital letter, only alphabets allowed)
Valid Email
Valid Phone Number


Enter your Name: indu
Enter your Email:bfhghfjd
Enter your Phone Number:7443486
Invalid Name (Should start with capital letter, only alphabets allowed)
Invalid Email
Invalid Phone Number (Must be 10 digits)
```

4. Program to perform file handling operation.

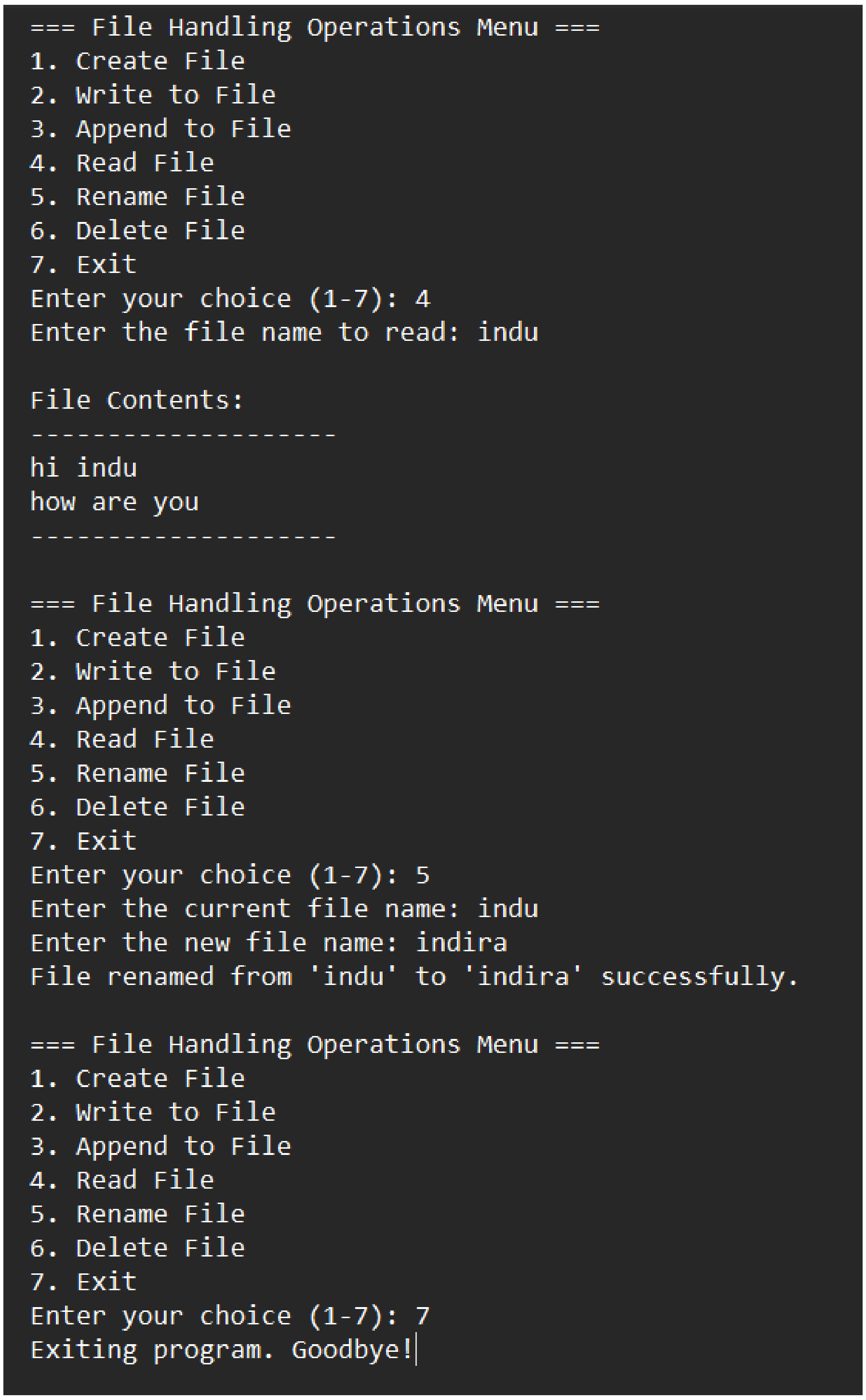
Aim:

To develop a Program to perform file handling operation.

Source Code:

```
import os

def create_file():
    filename = input("Enter the file name to create: ")
    try:
        with open(filename, 'x') as file:
            print(f"File '{filename}' created successfully!")
    except FileExistsError:
        print(f"File '{filename}' already exists.")

def write_to_file():
    filename = input("Enter the file name to write into: ")
    content = input("Enter the content to write: ")
    with open(filename, 'w') as file:
        file.write(content)
    print(f"Content written to '{filename}' successfully.")

def append_to_file():
    filename = input("Enter the file name to append into: ")
    content = input("Enter the content to append: ")
    with open(filename, 'a') as file:
        file.write("\n" + content)
    print(f"Content appended to '{filename}' successfully.")
def read_file():
    filename = input("Enter the file name to read: ")
    try:
        with open(filename, 'r') as file:
            print("\nFile Contents:")
            print("--------------------")
            print(file.read())
            print("--------------------")
    except FileNotFoundError:
        print(f"File '{filename}' does not exist.")

def rename_file():
    old_name = input("Enter the current file name: ")
    new_name = input("Enter the new file name: ")
    try:
        os.rename(old_name, new_name)
        print(f"File renamed from '{old_name}' to '{new_name}' successfully.")
    except FileNotFoundError:
        print(f"File '{old_name}' does not exist.")

def delete_file():
    filename = input("Enter the file name to delete: ")
    try:
        os.remove(filename)
        print(f"File '{filename}' deleted successfully.")
    except FileNotFoundError:
        print(f"File '{filename}' does not exist.")

# Main Menu
while True:
    print("\n=== File Handling Operations Menu ===")
    print("1. Create File")
    print("2. Write to File")
    print("3. Append to File")
    print("4. Read File")
    print("5. Rename File")
    print("6. Delete File")
    print("7. Exit")
    choice = input("Enter your choice (1-7): ")
    if choice == '1':
        create_file()
    elif choice == '2':
        write_to_file()
    elif choice == '3':
        append_to_file()
    elif choice == '4':
        read_file()
    elif choice == '5':
        rename_file()
    elif choice == '6':
        delete_file()
    elif choice == '7':
        print("Exiting program. Goodbye!")
        break
    else:
        print("Invalid choice! Please enter a number between 1 and 7.")
```

Output:



```
=== File Handling Operations Menu ===
1. Create File
2. Write to File
3. Append to File
4. Read File
5. Rename File
6. Delete File
7. Exit
Enter your choice (1-7): 4
Enter the file name to read: indu

File Contents:
--------------------
hi indu
how are you
--------------------

=== File Handling Operations Menu ===
1. Create File
2. Write to File
3. Append to File
4. Read File
5. Rename File
6. Delete File
7. Exit
Enter your choice (1-7): 5
Enter the current file name: indu
Enter the new file name: indira
File renamed from 'indu' to 'indira' successfully.

=== File Handling Operations Menu ===
1. Create File
2. Write to File
3. Append to File
4. Read File
5. Rename File
6. Delete File
7. Exit
Enter your choice (1-7): 7
Exiting program. Goodbye!
```

5. Program to store the details of the student using data frame and perform primary operation.

Aim:

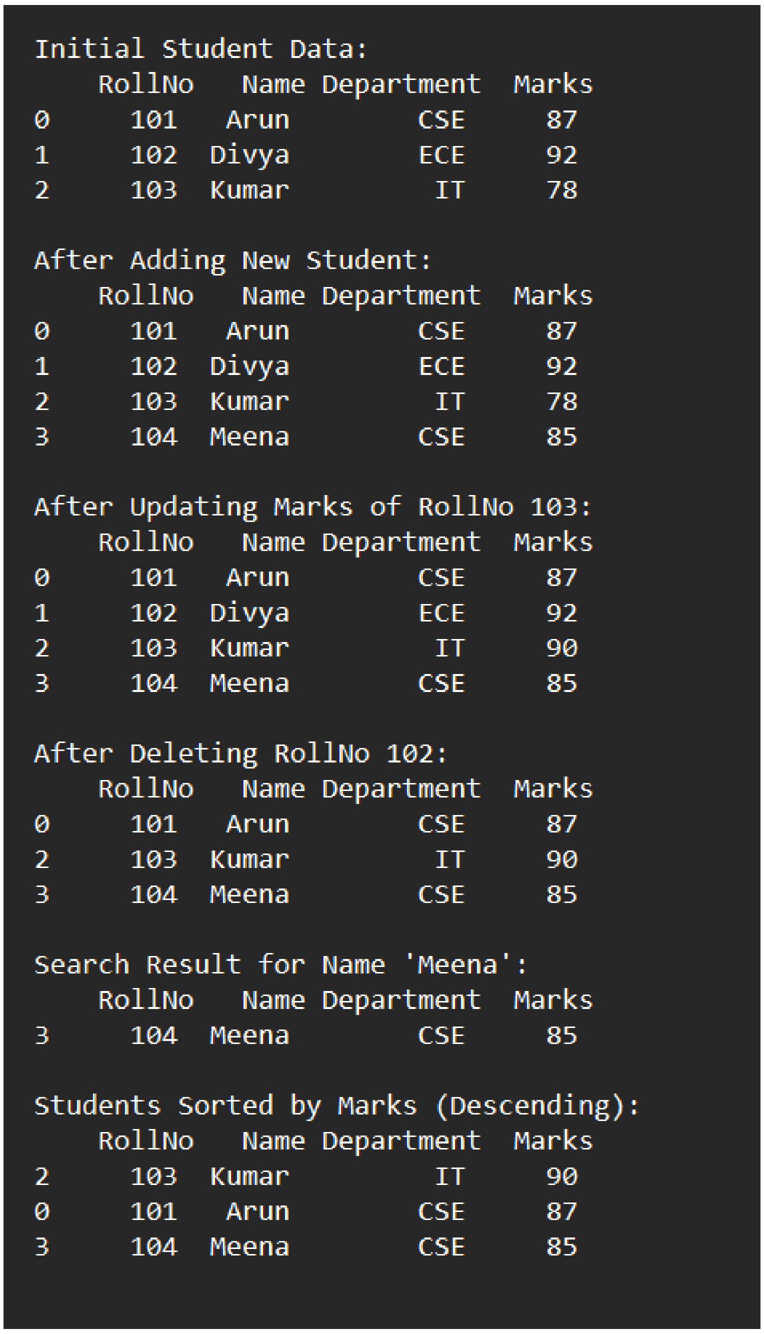
To develop a Program to store the details of the student using data frame and perform primary operation.

Source Code:

```
import pandas as pd
data = {
    "RollNo": [101, 102, 103],
    "Name": ["Arun", "Divya", "Kumar"],
    "Department": ["CSE", "ECE", "IT"],
    "Marks": [87, 92, 78]
}
students = pd.DataFrame(data)
print("Initial Student Data:\n", students)

new_student = {"RollNo": 104, "Name": "Meena", "Department": "CSE", "Marks": 85}
students = pd.concat([students, pd.DataFrame([new_student])], ignore_index=True)
print("\nAfter Adding New Student:\n", students)

students.loc[students["RollNo"] == 103, "Marks"] = 90
print("\nAfter Updating Marks of RollNo 103:\n", students)

students = students[students["RollNo"] != 102]
print("\nAfter Deleting RollNo 102:\n", students)

search_name = "Meena"
result = students[students["Name"] == search_name]
print(f"\nSearch Result for Name '{search_name}':\n", result)

sorted_students = students.sort_values(by="Marks", ascending=False)
```

```
print("\nStudents Sorted by Marks (Descending):\n", sorted_students)
```

Output:



```
Initial Student Data:
    RollNo   Name Department  Marks
0      101   Arun        CSE     87
1      102  Divya        ECE     92
2      103  Kumar         IT     78

After Adding New Student:
    RollNo   Name Department  Marks
0      101   Arun        CSE     87
1      102  Divya        ECE     92
2      103  Kumar         IT     78
3      104  Meena        CSE     85

After Updating Marks of RollNo 103:
    RollNo   Name Department  Marks
0      101   Arun        CSE     87
1      102  Divya        ECE     92
2      103  Kumar         IT     90
3      104  Meena        CSE     85

After Deleting RollNo 102:
    RollNo   Name Department  Marks
0      101   Arun        CSE     87
2      103  Kumar         IT     90
3      104  Meena        CSE     85

Search Result for Name 'Meena':
    RollNo   Name Department  Marks
3      104  Meena        CSE     85

Students Sorted by Marks (Descending):
    RollNo   Name Department  Marks
2      103  Kumar         IT     90
0      101   Arun        CSE     87
3      104  Meena        CSE     85
```

6. Program to implement linear regression algorithm.

Aim:

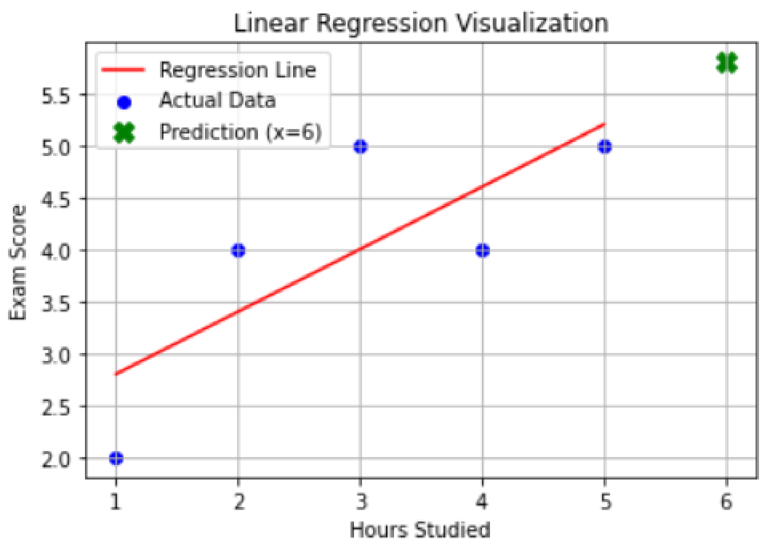
To develop a Program to implement linear regression algorithm.

Source Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Dataset
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5])
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
x_new = np.array([[6]])
y_new_pred = model.predict(x_new)

print("Coefficient (slope):", model.coef_[0])
print("Intercept:", model.intercept_)
print("Predicted values for training data:", y_pred)
print(f"Prediction for x = 6: {y_new_pred[0]:.2f}")

plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.scatter(x_new, y_new_pred, color='green', s=100, marker='X', label='Prediction (x=6)')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.title('Linear Regression Visualization')
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
Coefficient (slope): 0.6000000000000002
Intercept: 2.1999999999999993
Predicted values for training data: [2.8 3.4 4.  4.6 5.2]
Prediction for x = 6: 5.80
```



7. Program to implement preprocessing techniques.

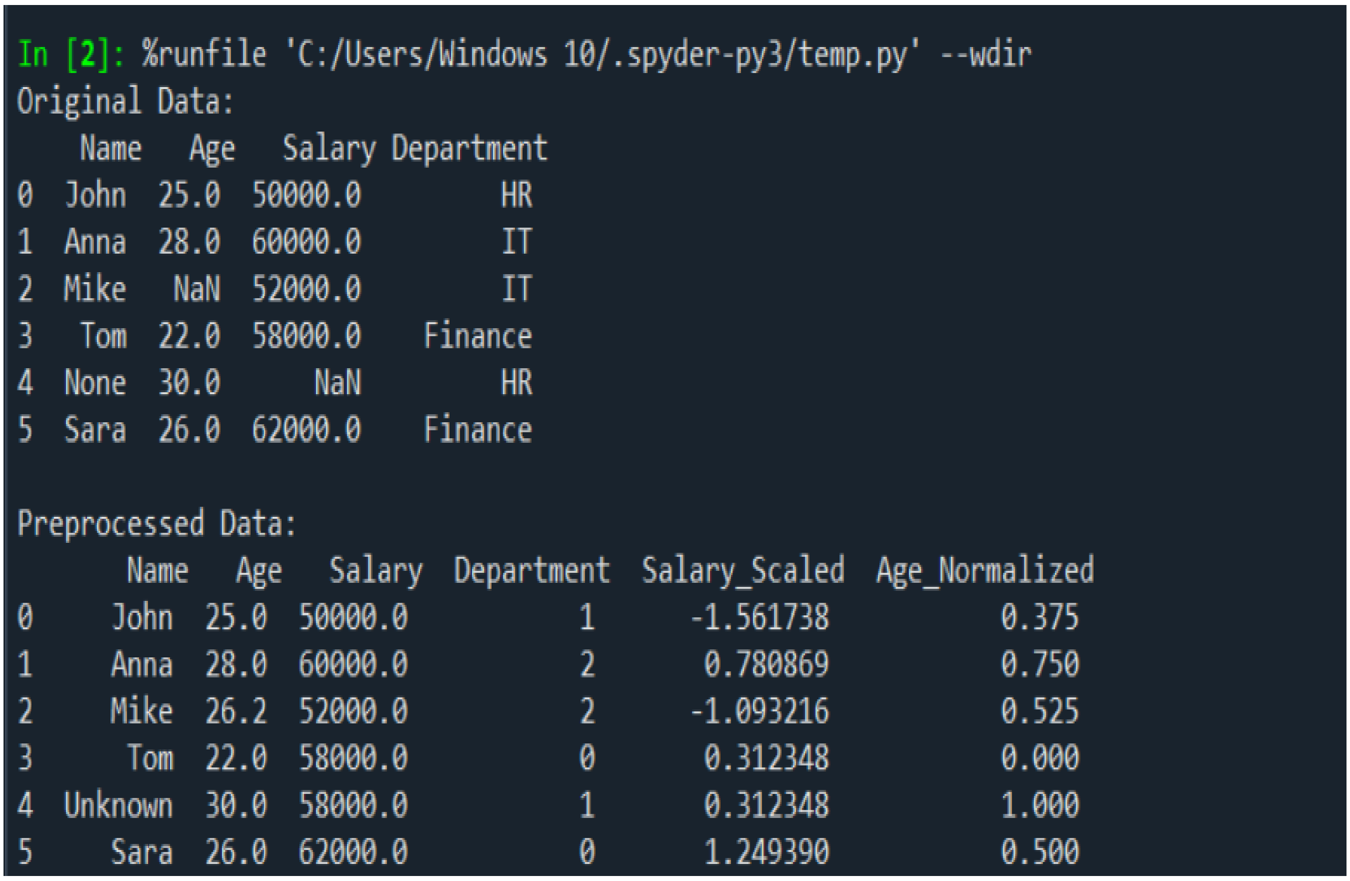
Aim:

To develop a Program to implement preprocessing techniques.

Source Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler
data = {
    "Student": ["Arun", "Divya", "Kumar", "Meena", "Raj"],
    "Department": ["CSE", "ECE", "IT", "CSE", "ECE"],
    "Marks": [87, None, 78, 85, 90],
    "Attendance": [75, 82, 60, 95, 88]
}
df = pd.DataFrame(data)
print("Original Data:\n", df)
df["Marks"].fillna(df["Marks"].mean(), inplace=True)
print("\nAfter Handling Missing Values:\n", df)
le = LabelEncoder()
df["Department_Encoded"] = le.fit_transform(df["Department"])
print("\nAfter Encoding Department:\n", df)
scaler1 = StandardScaler()
df["Marks_Standardized"] = scaler1.fit_transform(df[["Marks"]])
scaler2 = MinMaxScaler()
df["Attendance_Normalized"] = scaler2.fit_transform(df[["Attendance"]])
print("\nAfter Scaling & Normalization:\n", df)
features = df[["Marks_Standardized", "Attendance_Normalized", "Department_Encoded"]]
print("\nFinal Feature Set (for ML model):\n", features)
```

Output



```
In [2]: %runfile 'C:/Users/Windows 10/.spyder-py3/temp.py' --wdir
Original Data:
    Name   Age   Salary Department
0  John  25.0  50000.0         HR
1  Anna  28.0  60000.0         IT
2  Mike   NaN  52000.0         IT
3   Tom  22.0  58000.0    Finance
4  None  30.0      NaN         HR
5  Sara  26.0  62000.0    Finance

Preprocessed Data:
       Name   Age   Salary  Department  Salary_Scaled  Age_Normalized
0      John  25.0  50000.0           1      -1.561738           0.375
1      Anna  28.0  60000.0           2       0.780869           0.750
2      Mike  26.2  52000.0           2      -1.093216           0.525
3       Tom  22.0  58000.0           0       0.312348           0.000
4   Unknown  30.0  58000.0           1       0.312348           1.000
5      Sara  26.0  62000.0           0       1.249390           0.500
```

8. Program to implement Decision Tree Algorithm.

Aim:

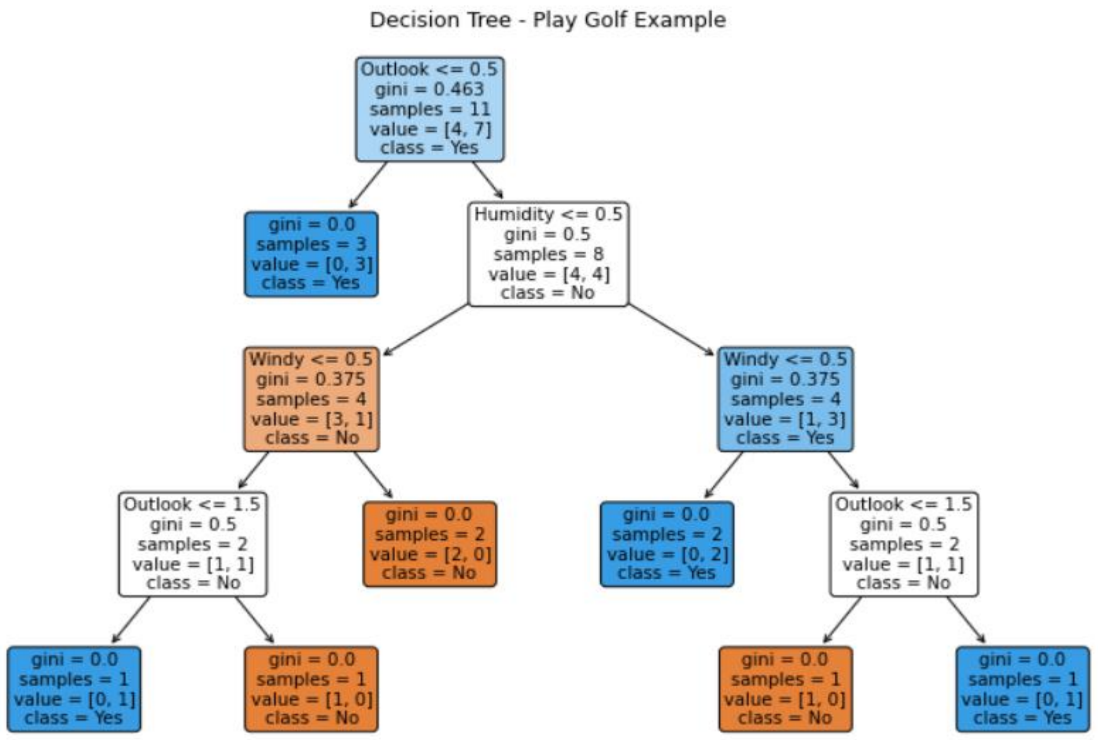
To develop a Program to implement Decision Tree Algorithm.

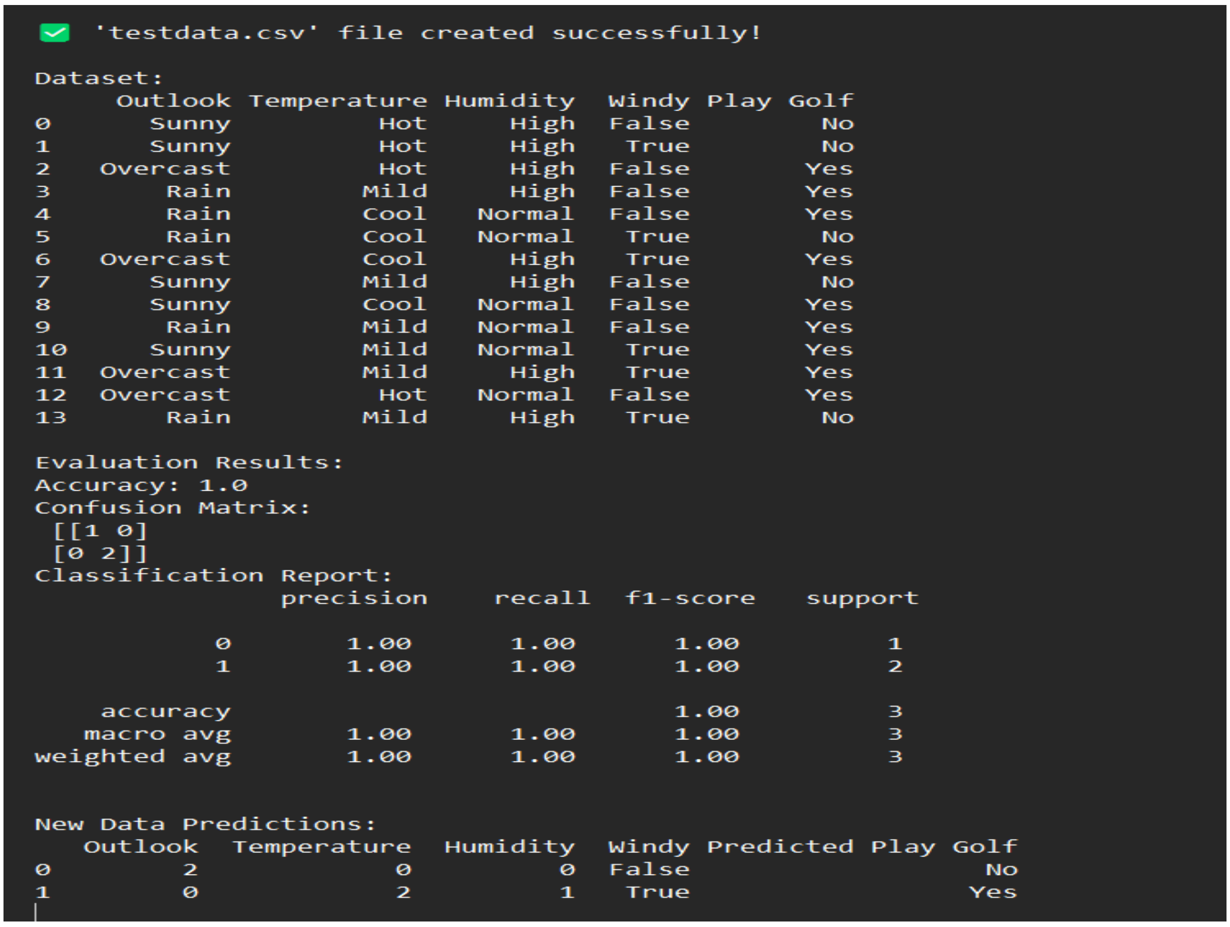
Source Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

data = {
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain',
                'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast',
                'Overcast', 'Rain'],
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool',
                    'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild',
                    'Hot', 'Mild'],
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal',
                 'High', 'High', 'Normal', 'Normal', 'Normal', 'High',
                 'Normal', 'High'],
    'Windy': [False, True, False, False, False, True,
              True, False, False, False, True, True, False, True],
    'Play Golf': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No',
                  'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
}

df = pd.DataFrame(data)
df.to_csv('testdata.csv', index=False)
print("✅ 'testdata.csv' file created successfully!\n")
data = pd.read_csv('testdata.csv')
```

```
df = pd.DataFrame(data)
print("Dataset:")
print(df)

label_encoders = {}
for column in ['Outlook', 'Temperature', 'Humidity', 'Play Golf']:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

X = df[['Outlook', 'Temperature', 'Humidity', 'Windy']]
y = df['Play Golf']

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

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)



y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("\nEvaluation Results:")
print("Accuracy:", round(accuracy, 2))
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)



new_data = pd.DataFrame({
    'Outlook': ['Sunny', 'Overcast'],
    'Temperature': ['Cool', 'Mild'],
    'Humidity': ['High', 'Normal'],
    'Windy': [False, True]
})

for column in ['Outlook', 'Temperature', 'Humidity']:
    new_data[column] = label_encoders[column].transform(new_data[column])

new_predictions = clf.predict(new_data)
new_data['Predicted Play Golf'] = label_encoders['Play
Golf'].inverse_transform(new_predictions)

print("\nNew Data Predictions:")
print(new_data)

plt.figure(figsize=(12, 8))
plot_tree(
    clf,
    feature_names=X.columns,
    class_names=label_encoders['Play Golf'].classes_,
    filled=True,
    rounded=True,
    fontsize=10
)
plt.title("Decision Tree - Play Golf Example")
plt.show()
```

Output:





```
✅ 'testdata.csv' file created successfully!

Dataset:
     Outlook Temperature Humidity  Windy Play Golf
0      Sunny         Hot     High  False        No
1      Sunny         Hot     High   True        No
2   Overcast         Hot     High  False       Yes
3       Rain        Mild     High  False       Yes
4       Rain        Cool   Normal  False       Yes
5       Rain        Cool   Normal   True        No
6   Overcast        Cool     High   True       Yes
7      Sunny        Mild     High  False        No
8      Sunny        Cool   Normal  False       Yes
9       Rain        Mild   Normal  False       Yes
10     Sunny        Mild   Normal   True       Yes
11  Overcast        Mild     High   True       Yes
12  Overcast         Hot   Normal  False       Yes
13      Rain        Mild     High   True        No

Evaluation Results:
Accuracy: 1.0
Confusion Matrix:
 [[1 0]
 [0 2]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3


New Data Predictions:
   Outlook  Temperature  Humidity  Windy Predicted Play Golf
0        2            0         0  False                  No
1        0            2         1   True                 Yes
```

9. Program to implement Naïve Bayes Algorithm.

Aim:

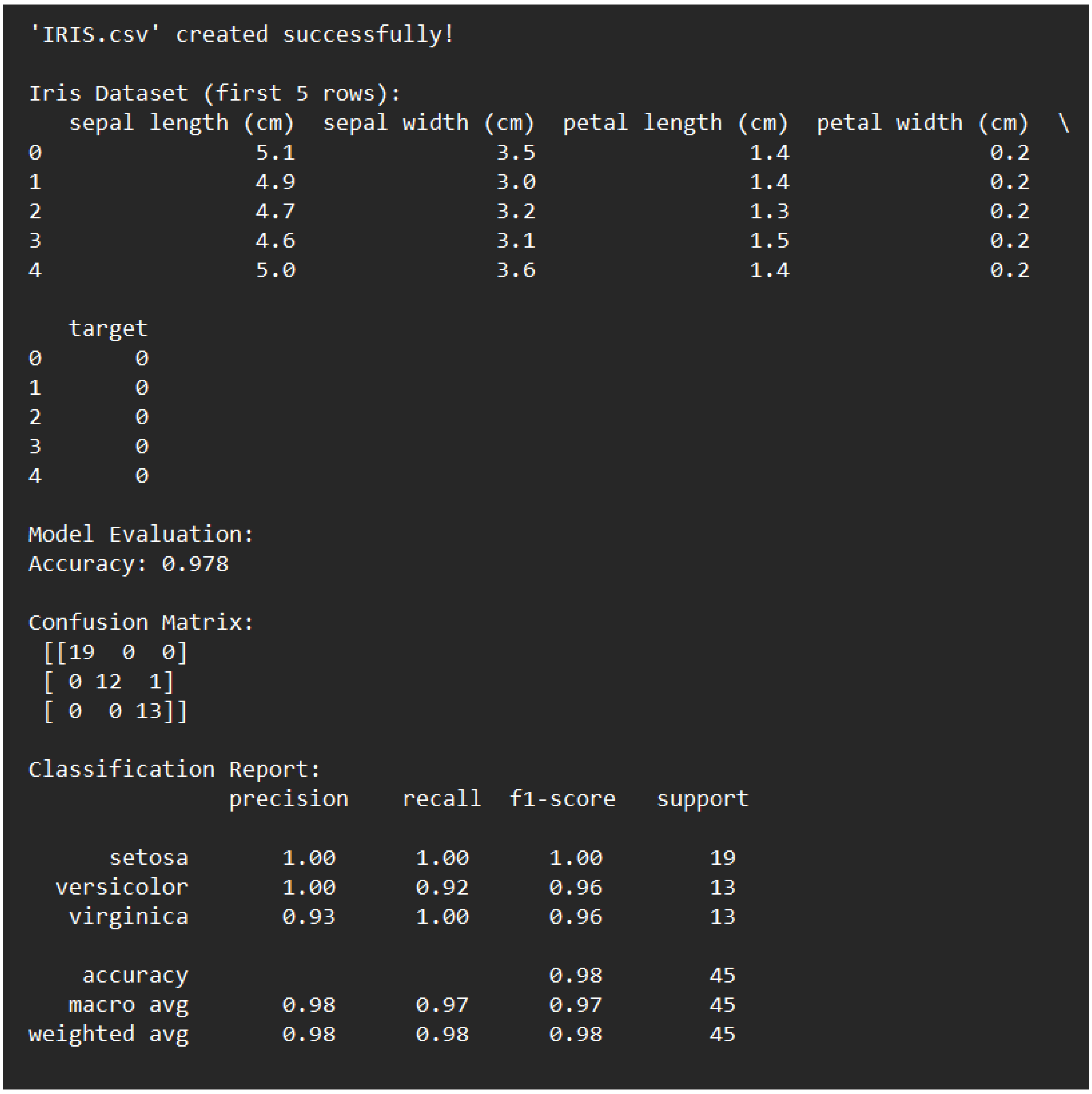
To develop a Program to implement Naïve Bayes Algorithm.

Source Code:

```
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
from sklearn.datasets import load_iris



iris = load_iris()

# Create a DataFrame
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

# Save it as a CSV file (optional)
iris_df.to_csv('IRIS.csv', index=False)
print("✅ 'IRIS.csv' created successfully!\n")

print("Iris Dataset (first 5 rows):")
print(iris_df.head())

# STEP 2: Split features and target
X = iris_df.iloc[:, :-1].values
y = iris_df['target'].values

# STEP 3: Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

```
# STEP 4: Train the Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# STEP 5: Make Predictions
y_pred = model.predict(X_test)

# STEP 6: Evaluate Model
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=iris.target_names)

print("\nModel Evaluation:")
print("Accuracy:", round(accuracy, 3))
print("\nConfusion Matrix:\n", confusion)
print("\nClassification Report:\n", report)
```

Output:



```
'IRIS.csv' created successfully!

Iris Dataset (first 5 rows):
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0

Model Evaluation:
Accuracy: 0.978

Confusion Matrix:
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]

Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      0.92      0.96        13
   virginica       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

10.Program to implement Artificial Neutral Network.

Aim:

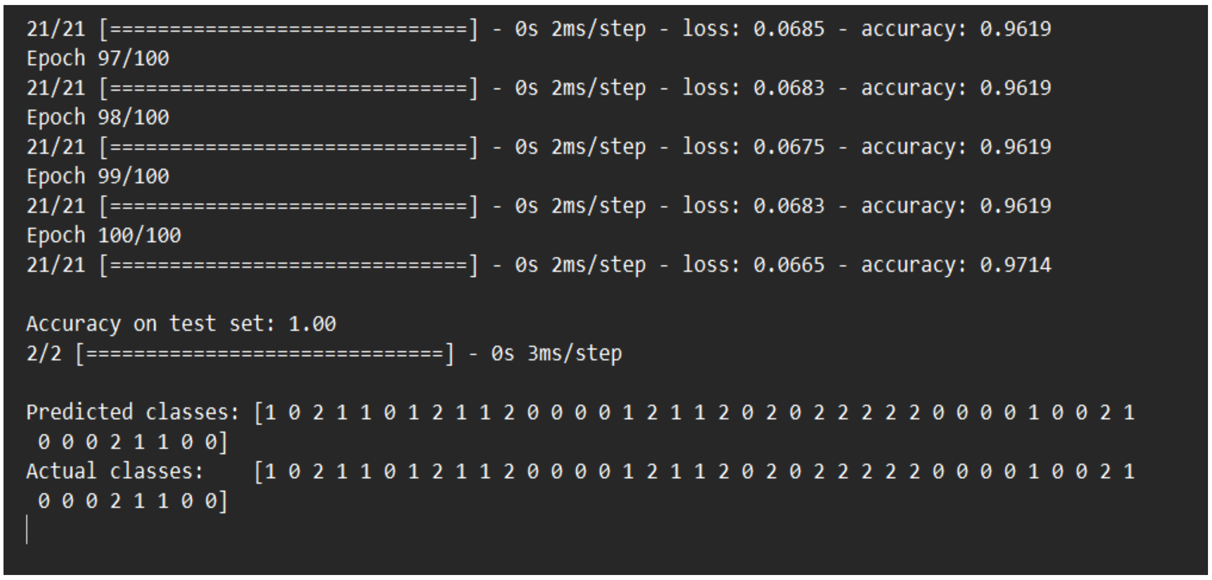
To develop a Program to implement Artificial Neutral Network.

Source Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load the dataset
iris_df = pd.read_csv('IRIS.csv')

# Features and target
X = iris_df.iloc[:, :-1].values
y = iris_df['target'].values

# Encode target labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
y = to_categorical(y)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

# Build the neural network model
model = Sequential()
model.add(Dense(10, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(y.shape[1], activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=5, verbose=1)

# Evaluate the model
accuracy = model.evaluate(X_test, y_test, verbose=0)[1]
print(f"\nAccuracy on test set: {accuracy:.2f}")

# Make predictions
predictions = model.predict(X_test)
predicted_classes = np.argmax(predictions, axis=1)
actual_classes = np.argmax(y_test, axis=1)

print("\nPredicted classes:", predicted_classes)
print("Actual classes:   ", actual_classes)
```

Output:



```
21/21 [==============================] - 0s 2ms/step - loss: 0.0685 - accuracy: 0.9619
Epoch 97/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0683 - accuracy: 0.9619
Epoch 98/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0675 - accuracy: 0.9619
Epoch 99/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0683 - accuracy: 0.9619
Epoch 100/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0665 - accuracy: 0.9714

Accuracy on test set: 1.00
2/2 [==============================] - 0s 3ms/step

Predicted classes: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0]
Actual classes:    [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0]
```

11.Program to implement K-means clustering algorithm.

Aim:

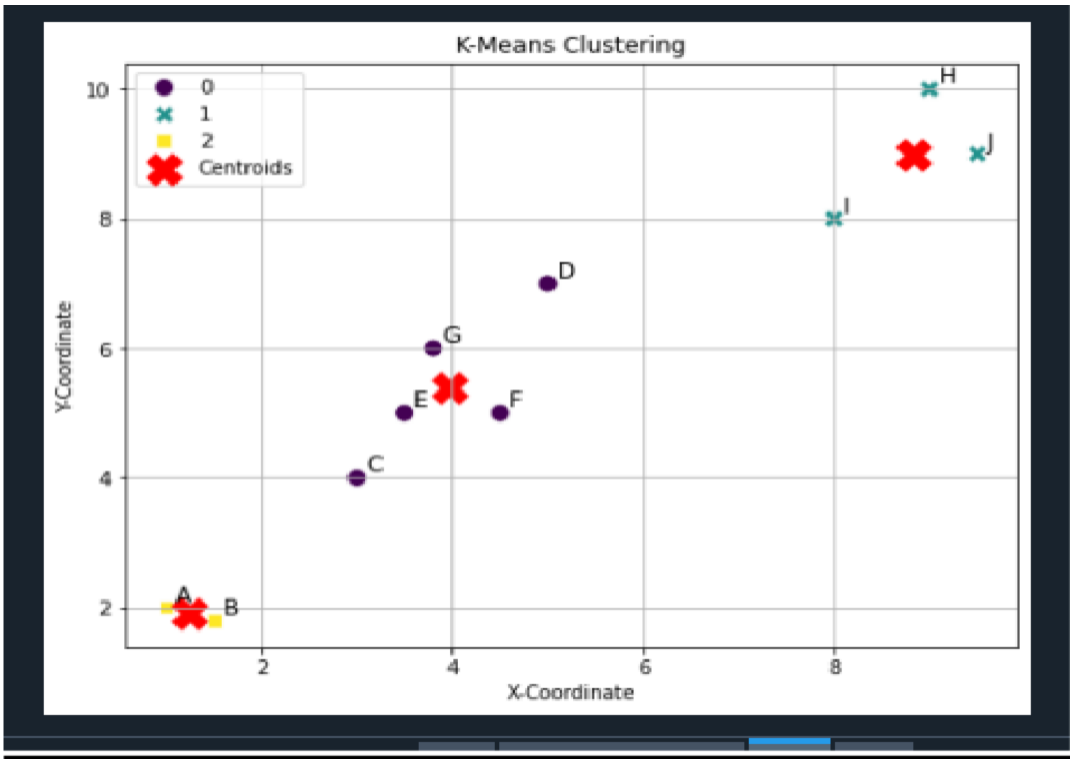
To develop a Program to implement K-means clustering algorithm.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Sample data
data = {
    'Point': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
    'X': [1, 1.5, 3, 5, 3.5, 4.5, 3.8, 9, 8, 9.5],
    'Y': [2, 1.8, 4, 7, 5, 5, 6, 10, 8, 9]
}

# Create DataFrame
df = pd.DataFrame(data)
X = df[['X', 'Y']]

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=0)
df['Cluster'] = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_

# Plot the clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x='X', y='Y', hue='Cluster', data=df, palette='viridis', s=100, style='Cluster')

# Plot centroids
```

```
plt.scatter(centroids[:, 0], centroids[:, 1], s=300, c='red', label='Centroids', marker='X')

# Annotate points
for i in range(len(df)):
    plt.text(df['X'][i]+0.1, df['Y'][i]+0.1, df['Point'][i], fontsize=12)

plt.title('K-Means Clustering')
plt.xlabel('X-Coordinate')
plt.ylabel('Y-Coordinate')
plt.legend()
plt.grid(True)
        plt.show()
```

Output:



12.Program to implement Dimensionality Reduction Techniques.

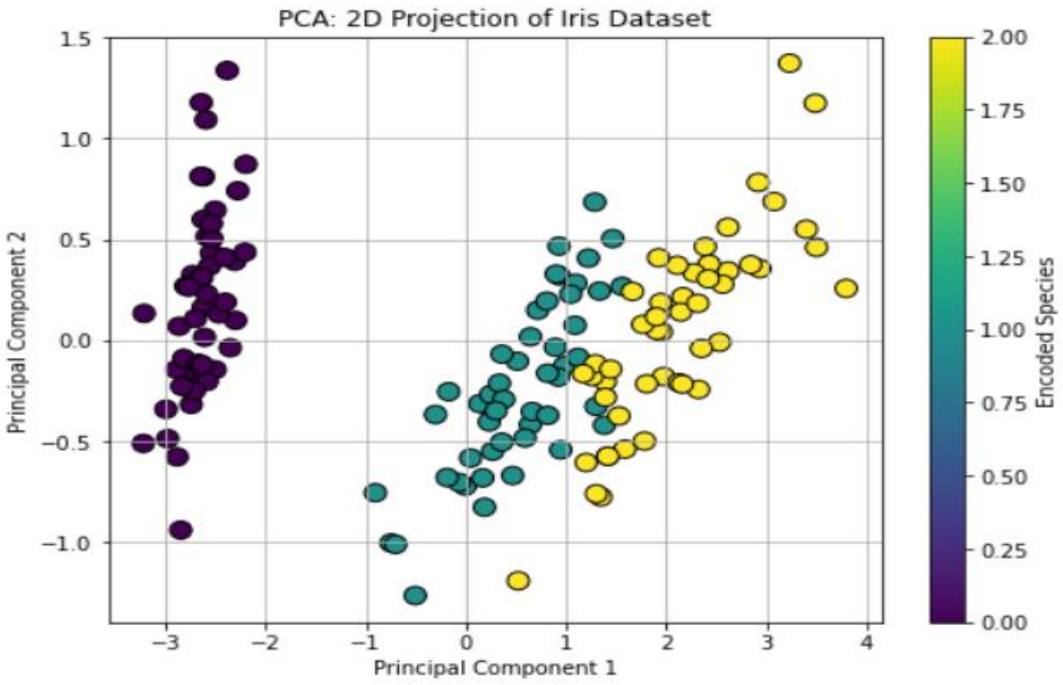
Aim:

To develop a Program to implement Dimensionality Reduction Techniques.

Source Code:

```
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Load dataset
df = pd.read_csv("IRIS.csv")

# Features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Encode target labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Apply PCA to reduce to 2 dimensions
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
# Plot the 2D PCA projection
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_encoded, cmap='viridis', edgecolor='k', s=100)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA: 2D Projection of Iris Dataset')
plt.colorbar(label='Encoded Species')
plt.grid(True)
plt.show()
```

Output:


PCA: 2D Projection of Iris Dataset
Principal Component 1
Principal Component 2
Encoded Species
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

13. Program to Visualize data using 2D and 3D plot.

Aim:

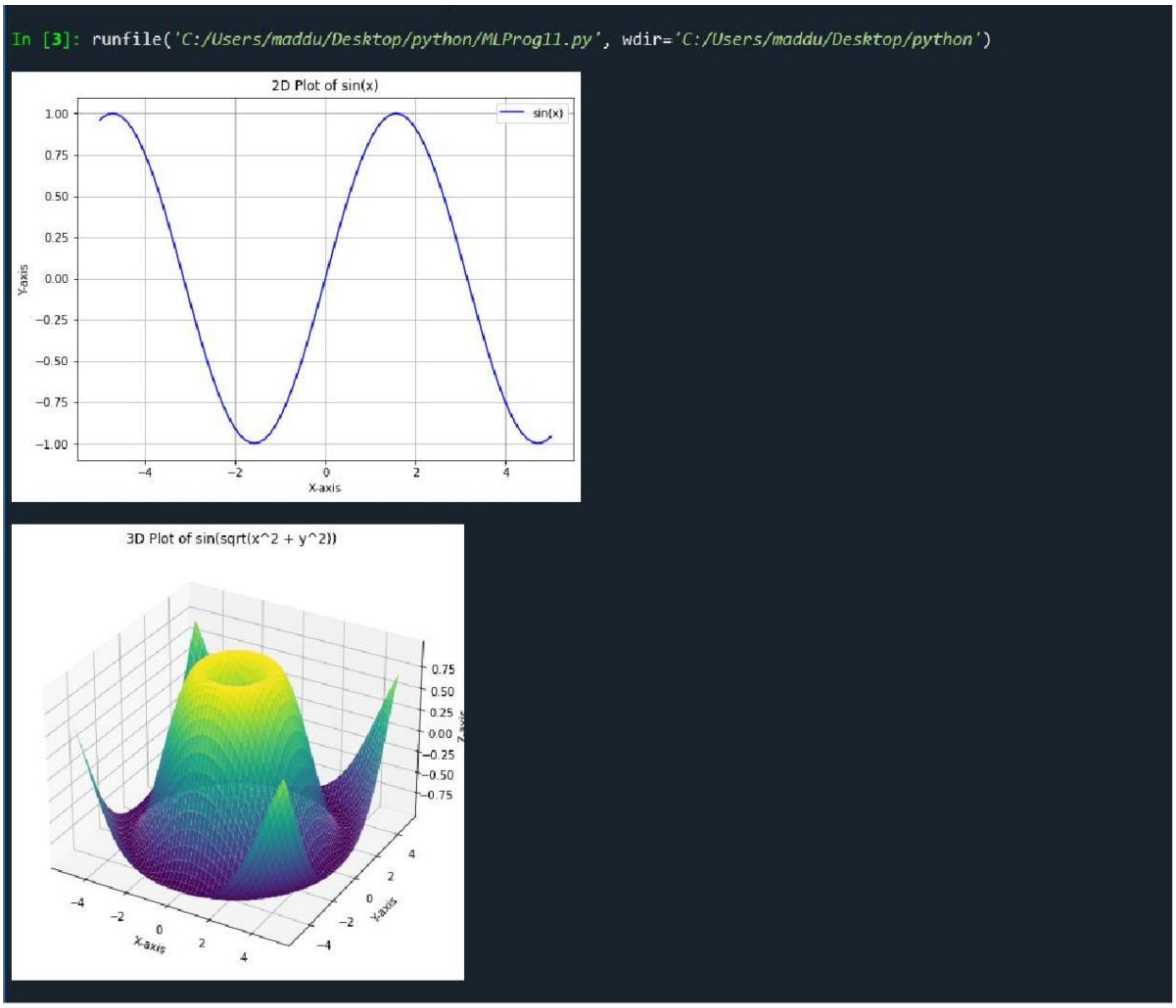
To develop a Python program to visualize data using 2D and 3D plot.

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# ----------------------
# 2D Plot of sin(x)
# ----------------------
x = np.linspace(-5, 5, 100)
y = np.sin(x)

plt.figure(figsize=(8, 6))
plt.plot(x, y, label="sin(x)", color='b')
plt.title("2D Plot of sin(x)")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()

# ----------------------
# 3D Plot of sin(sqrt(x^2 + y^2))
# ----------------------
x_3d = np.linspace(-5, 5, 100)
y_3d = np.linspace(-5, 5, 100)
x_3d, y_3d = np.meshgrid(x_3d, y_3d)
z_3d = np.sin(np.sqrt(x_3d**2 + y_3d**2))
fig = plt.figure(figsize=(10, 7))
```

```
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x_3d, y_3d, z_3d, cmap='viridis')
  ax.set_title("3D Plot of sin(sqrt(x^2 + y^2))")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.show()
```

Output:



```
In [3]: runfile('C:/Users/maddu/Desktop/python/MLProg11.py', wdir='C:/Users/maddu/Desktop/python')
```