

**Institute of Distance and Open Learning**

Vidya Nagari, Kalina, Santacruz East – 400098.

***CERTIFICATE***

This is to certify that **Mr. Sawant Hrushikesh Sunil Sandhya** of **Master in Computer Application** (MCA) Semester I has completed the specified term work in the subject of **Artificial Intelligence & Machine Learning** satisfactorily within this institute as laid down by University of Mumbai during the academic year 2023 to 2024.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

Subject In-charge External Examiner Coordinator – M.C.A

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Title** | **Date** | **Signature** |
| 1. | **INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘NumPy’ LIBRARIES.** | 11-June-23 |  |
| 2. | **INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Pandas’ LIBRARIES.** | 11-June-23 |  |
| 3. | **INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Scipy’ LIBRARIES.** | 11-June-23 |  |
| 4. | **INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Matplot’ LIBRARIES.** | 11-June-23 |  |
| 5. | **INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Scikit-Learn’ LIBRARIES.** | 11-June-23 |  |
| 6. | **MULTIPLE REGRESSION ANALYSIS** | 17-June-23 |  |
| 6a 'Experience' and 'Education' to predict the 'Salary'. Print  the regression summary, including coefficients, p-values,  and R-squared. Create a scatter plot. |  |  |
| 7. | **LOGISTIC REGRESSION** | 17-June-23 |  |
| 8. | **WATER JUG PROBLEM** | 17-June-23 |  |
| 9. | **LOGIC PROGRAMMING WITH PROLOG** | 17-June-23 |  |
| 9a Write a program to derive the predicate  for e.g.: Sachin is batsman, batsman is cricketer |  |  |
| 10. | **TIC-TAC-TOE PROBLEM** | 11-July-23 |  |
| 11. | **CORRELATION MATRIX OF IRIS FEATURE** | 11-July-23 |  |
| 11a Use Iris dataset to identify and visualize features that  are strongly correlated. Implement correlation analysis  and use matplotlib to create a heatmap to visually  represent the correlation matrix. |  |  |
| 12. | **DECISION TREE** | 11-July-23 |  |
| 13. | **CLASSIFYING DATA USING SUPPORT VECTOR MACHINES (SVMS)** | 11-July-23 |  |

Practical 1: INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘NumPy’ LIBRARIES

import numpy as np

a=np.array([1,2,3,4,5,6])

print(a[0])

1

----------------------------------

a=np.array([9,4,6,2,1,7,3])

np.sort(a)

array([1, 2, 3, 4, 6, 7, 9])

-----------------------------------

a=np.array([3,4,9,16,25])

np.sqrt(a)

array([1.73205081, 2. , 3. , 4. , 5. ])

-----------------------------------------------

a=np.arange(51)

print (a)

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

48 49 50]

---------------------------------------------------

a=np.linspace(0,30,num=6)

print(a)

[ 0. 6. 12. 18. 24. 30.]

-------------------------------------------

a=np.array([1,2,3,4])

b=np.array([5,6,7,4])

print(np.concatenate((a,b)))

[1 2 3 4 5 6 7 4]

---------------------------------------------------

Practical 2: INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Pandas’ LIBRARIES

import pandas as pd

import numpy as np

info=np.array(['p','a','n','d','a','s'])

a=pd.Series(info)

print(a)

0 p

1 a

2 n

3 d

4 a

5 s

dtype: object

import pandas as pd

import numpy as np

info=np.array(['pandas','python','java'])

df=pd.DataFrame(info)

print(df)

0

0 pandas

1 python

2 java

from pandas.core.arrays.categorical import simplefilter

import numpy as np

from scipy.spatial import Delaunay

import matplotlib.pyplot as plt

points=np.array([

    [1,2],

    [2,4],

    [6,4],

    [4,5],

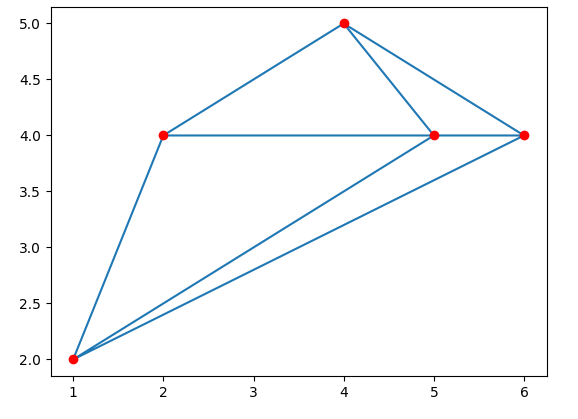
     [5,4]])

simplefilter= Delaunay(points).simplices

plt.triplot(points[:,0],points[:,1],simplefilter)

plt.scatter(points[:,0],points[:,1],color='r')

plt.show()



Practical 3: INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Scipy’ LIBRARIES

import scipy as sc

a=sc.cbrt(27)

print(a)

3.0000000000000004

import scipy

print(scipy.\_\_version\_\_)

1.10.1

Practical 4: INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Matplot’ LIBRARIES

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

x=np.array([2.4,5.0,1.5,3.8,8.7,3.6,1.2,8.1,2.5,5,1.6,1.6,2.4,3.9,5.4])

y=np.array([2.1,4.7,1.7,3.6,8.7,3.2,1.0,8.0,2.4,6,1.1,1.3,2.4,3.9,4.8])

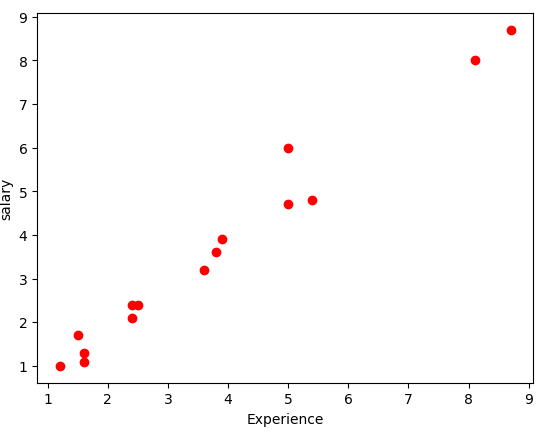
n=np.size(x)

plt.scatter(x,y,color='red')

plt.xlabel("Experience")

plt.ylabel("salary")

plt.show()



Practical 5: INTRODUCTION TO PYTHON PROGRAMMING: LEARN ‘Scikit-Learn’ LIBRARIES

from sklearn.datasets import load\_iris

iris = load\_iris()

A= iris.data

y = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

print("Feature names:", feature\_names)

print("Target names:", target\_names)

print("\nFirst 10 rows of A:\n", A[:10])

Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of A:

[[5.1 3.5 1.4 0.2]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]

[5. 3.6 1.4 0.2]

[5.4 3.9 1.7 0.4]

[4.6 3.4 1.4 0.3]

[5. 3.4 1.5 0.2]

[4.4 2.9 1.4 0.2]

[4.9 3.1 1.5 0.1]]

**Practical 6: MULTIPLE REGRESSION ANALYSIS**

**6a: 'Experience' and 'Education' to predict the 'Salary'. Print the regression summary, including coefficients, p-values, and R-squared. Create a scatter plot.**

**Code:**

import numpy as np

import pandas as pd

import statsmodels.api as sm

import matplotlib.pyplot as plt

# Create a sample dataset

np.random.seed(0)

n = 100 # number of samples

experience = np.random.rand(n) \* 10 # generate random experience values

education = np.random.randint(1, 5, size=n) # generate random education levels (assuming integer values)

salary = 5000 + 200 \* experience + 1000 \* education + np.random.normal(0, 1000, size=n) # generate salary based on experience and education

# Create a DataFrame

data = pd.DataFrame({'Experience': experience, 'Education': education, 'Salary': salary})

# Perform multiple regression analysis

X = sm.add\_constant(data[['Experience', 'Education']])

y = data['Salary']

model = sm.OLS(y, X).fit()

# Print regression summary

print(model.summary())

# Create a scatter plot

fig, ax = plt.subplots(figsize=(8, 6))

ax.scatter(data['Experience'], data['Salary'], label='Experience', alpha=0.7)

ax.scatter(data['Education'], data['Salary'], label='Education', alpha=0.7)

ax.set\_xlabel('Experience/Education')

ax.set\_ylabel('Salary')

ax.legend()

plt.show()

**Output:**

OLS Regression Results

==============================================================================

Dep. Variable: Salary R-squared: 0.700

Model: OLS Adj. R-squared: 0.693

Method: Least Squares F-statistic: 113.0

Date: Mon, 22 Jan 2024 Prob (F-statistic): 4.59e-26

Time: 14:42:19 Log-Likelihood: -827.15

No. Observations: 100 AIC: 1660.

Df Residuals: 97 BIC: 1668.

Df Model: 2

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const 4877.9452 255.390 19.100 0.000 4371.068 5384.823

Experience 195.1715 34.004 5.740 0.000 127.682 262.661

Education 1061.7062 85.087 12.478 0.000 892.831 1230.581

==============================================================================

Omnibus: 1.569 Durbin-Watson: 1.926

Prob(Omnibus): 0.456 Jarque-Bera (JB): 1.608

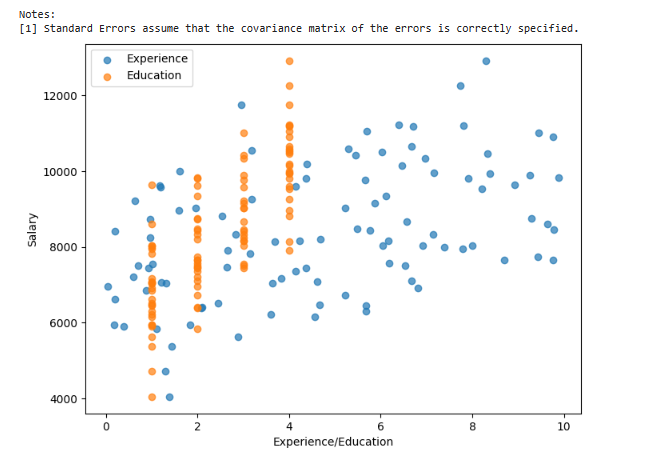
Skew: 0.248 Prob(JB): 0.447

Kurtosis: 2.626 Cond. No. 16.6

==============================================================================

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



**Practical 7: LOGISTIC REGRESION**

**Code:**

# Install necessary libraries

!pip install scikit-learn

!pip install matplotlib

# Import libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_classification

from sklearn.linear\_model import LogisticRegression

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

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

# Create a synthetic dataset

X, y = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=42)

# Split the dataset 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)

# Create and train a logistic regression model

model = LogisticRegression()

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

# Make predictions on the test set

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

# Evaluate the model

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

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

classification\_rep = classification\_report(y\_test, y\_pred)

# Print evaluation metrics

print(f"Accuracy: {accuracy:.2f}")

print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_rep)

# Plot decision boundary

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

h = 0.02

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k')

plt.title('Logistic Regression Decision Boundary')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

**Output:**

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)

Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)

Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)

Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.2)

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)

Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)

Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)

Accuracy: 0.90

Confusion Matrix:

[[97 7]

[13 83]]

Classification Report:

precision recall f1-score support

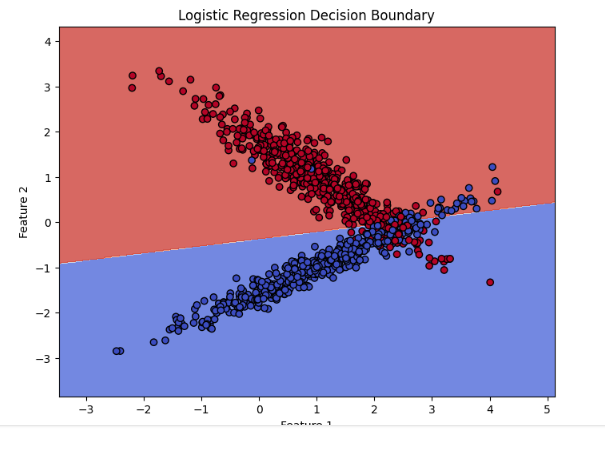
0 0.88 0.93 0.91 104

1 0.92 0.86 0.89 96

accuracy 0.90 200

macro avg 0.90 0.90 0.90 200

weighted avg 0.90 0.90 0.90 200



**Practical 8: WATER JUG PROBLEM**

**Code:**

jug(0,0):-write("\nFill 3g jug."),jug(0,3).

jug(0,3):-write("\nPour water from 3g jug to 4g jug."),jug(3,0).

jug(3,0):-write("\nFill 3g jug."),jug(3,3).

jug(3,3):-write("\nPour water from 3g jug to 4g jug until 4g jug is full."),jug(4,2).

jug(4,2):-write("\nEmpty 4g jug."),jug(0,2).

jug(0,2):-write("Pour water from 3g jug to 4g jug."),jug(2,0).

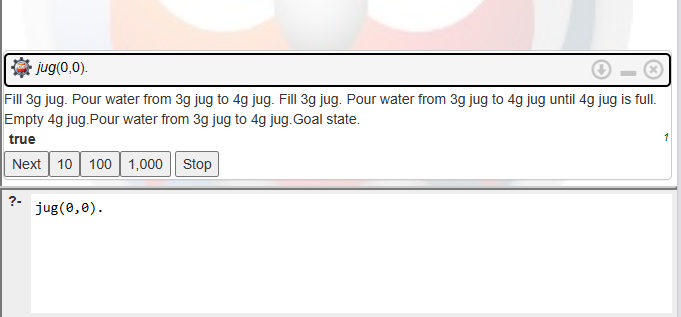
jug(2,0):-write("Goal state.").

jug(X,Y):-X>4,not(Y>3),write("\n 4G jug overflow."),not(jug(2,0)).

jug(X,Y):-not(X>4),Y>3,write("\n 3G jug overflow."),not(jug(0,2)).

jug(X,Y):-X>4,Y>3,write("\n BOTH jug overflow."),not(jug(2,0)).

**Output:**



**Practical 9: LOGIC PROGRAMMING WITH PROLOG**

**9a: Write a program to derive the predicate.**

**For e.g.: Sachin is batsman, batsman is cricketer**

**Code:**

% Facts - defining male and female

male(sachin).

male(john).

male(david).

male(george).

female(mary).

female(lisa).

female(emily).

female(rose).

% Facts - defining parent relations

parent(sachin, john). % Sachin is the parent of John

parent(sachin, lisa). % Sachin is the parent of Lisa

parent(john, mary). % John is the parent of Mary

parent(john, david). % John is the parent of David

parent(lisa, emily). % Lisa is the parent of Emily

parent(lisa, george). % Lisa is the parent of George

% Rules for family relations

father(Father, Child) :- male(Father), parent(Father, Child).

mother(Mother, Child) :- female(Mother), parent(Mother, Child).

grandfather(Grandfather, Grandchild) :- father(Grandfather, Parent), parent(Parent, Grandchild).

grandmother(Grandmother, Grandchild) :- mother(Grandmother, Parent), parent(Parent, Grandchild).

sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.

brother(Brother, Person) :- male(Brother), sibling(Brother, Person).

sister(Sister, Person) :- female(Sister), sibling(Sister, Person).

uncle(Uncle, NieceNephew) :- brother(Uncle, Parent), parent(Parent, NieceNephew).

aunt(Aunt, NieceNephew) :- sister(Aunt, Parent), parent(Parent, NieceNephew).

nephew(Nephew, UncleAunt) :- male(Nephew), (uncle(UncleAunt, Nephew); aunt(UncleAunt, Nephew)).

niece(Niece, UncleAunt) :- female(Niece), (uncle(UncleAunt, Niece); aunt(UncleAunt, Niece)).

cousin(Person1, Person2) :- parent(Parent1, Person1), parent(Parent2, Person2), sibling(Parent1, Parent2), Person1 \= Person2.

**Output:**



**Practical 10: TIC-TAC-TOE PROBLEM**

**Code:**

def print\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 9)

def check\_winner(board):

# Check rows

for row in board:

if row[0] == row[1] == row[2] and row[0] != ' ':

return True

# Check columns

for col in range(3):

if board[0][col] == board[1][col] == board[2][col] and board[0][col] != ' ':

return True

# Check diagonals

if board[0][0] == board[1][1] == board[2][2] and board[0][0] != ' ':

return True

if board[0][2] == board[1][1] == board[2][0] and board[0][2] != ' ':

return True

return False

def is\_board\_full(board):

for row in board:

if ' ' in row:

return False

return True

def play\_game():

board = [[' ' for \_ in range(3)] for \_ in range(3)]

current\_player = 'X'

while True:

print\_board(board)

# Get player move

row = int(input("Enter row (0, 1, or 2): "))

col = int(input("Enter column (0, 1, or 2): "))

# Check if the chosen cell is already occupied

if board[row][col] != ' ':

print("Cell already occupied. Try again.")

continue

# Make the move

board[row][col] = current\_player

# Check for a winner

if check\_winner(board):

print\_board(board)

print(f"Player {current\_player} wins!")

break

# Check for a tie

if is\_board\_full(board):

print\_board(board)

print("It's a tie!")

break

# Switch players

current\_player = 'O' if current\_player == 'X' else 'X'

if \_\_name\_\_ == "\_\_main\_\_":

play\_game()

**Practical 11: CORRELATION MATRIX OF IRIS FEATURE**

**6a: Use Iris dataset to identify and visualize features that are strongly correlated. Implement correlation analysis and use matplotlib to create a heatmap to visually represent the correlation matrix.**

**Code:**

# Install necessary libraries (if not already installed)

!pip install seaborn

!pip install scikit-learn

# Import libraries

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

import pandas as pd

# Load Iris dataset

iris = load\_iris()

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

iris\_df['target'] = iris.target

# Calculate correlation matrix

correlation\_matrix = iris\_df.corr()

# Create a heatmap using seaborn

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Iris Dataset - Feature Correlation Heatmap')

plt.show()

**Output:**

Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)

Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)

Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)

Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.47.2)

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2)

Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.1)

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.3.post1)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)

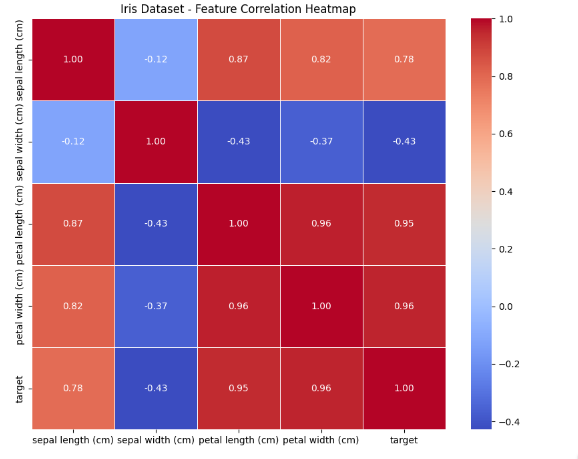
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)

Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)

Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)



**Practical 12: DECISION TREE**

**Code:**

from sklearn import tree

# Features: Weight and Texture (1 for smooth, 0 for bumpy)

# Labels: 0 for Orange, 1 for Apple

features = [[140, 1], [130, 1], [150, 0], [170, 0]]

labels = [0, 0, 1, 1]

# Creating Decision Tree Classifier

classifier = tree.DecisionTreeClassifier()

classifier = classifier.fit(features, labels)

# Predicting fruit based on weight and texture

weight = int(input("Enter fruit weight (in grams): "))

texture = int(input("Enter fruit texture (1 for smooth, 0 for bumpy): "))

prediction = classifier.predict([[weight, texture]])

if prediction == 0:

print("The fruit is an Orange.")

else:

print("The fruit is an Apple.")

**Output:**



**Practical 13: CLASSIFYING DATA USING SUPPORT VECTOR MACHINES (SVMS)**

**Code:**

from sklearn import svm

# Simulated data for binary classification

data = [

[7, 8], # Sweetness, Creaminess

[5, 4],

[3, 6],

[9, 7],

[8, 5],

[4, 3]

]

# Labels (0: Dislike, 1: Like)

labels = [1, 0, 0, 1, 1, 0]

# Create a Support Vector Machine (SVM) model with linear kernel

model\_svm = svm.SVC(kernel='linear')

# Fit the model

model\_svm.fit(data, labels)

# Input from user

sweetness = int(input("Enter sweetness level (1-10): "))

creaminess = int(input("Enter creaminess level (1-10): "))

# Predict for the input data

prediction = model\_svm.predict([[sweetness, creaminess]])

# Output prediction

if prediction[0] == 1:

print("Prediction (SVM): You might like this ice cream flavor!")

else:

print("Prediction (SVM): You might not like this ice cream flavor.")

**Output:**

