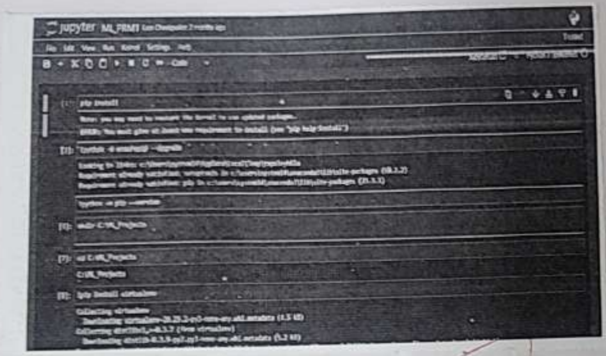
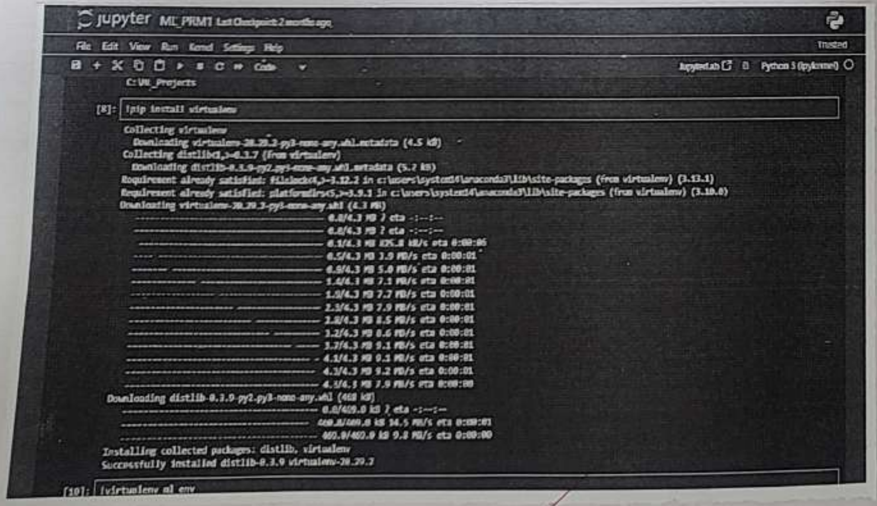


jupyter ML_PRM1 Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code

```
[1]: pip install
     Note: you may need to restart the kernel to use updated packages.
     ERROR: You must give at least one requirement to install (see "pip help install")

[3]: !python -m ensurepip --upgrade
     Looking in links: c:\Users\[illegible]\Temp\[illegible]
     Requirement already satisfied: setuptools in c:\users\[illegible]\anaconda3\lib\site-packages (68.2.2)
     Requirement already satisfied: pip in c:\users\[illegible]\anaconda3\lib\site-packages (23.3.1)

     !python -m pip --version

[6]: mkdir C:\ML_Projects

[7]: cd C:\ML_Projects
     C:\ML_Projects

[8]: !pip install virtualenv
     Collecting virtualenv
       Downloading virtualenv-20.29.2-py3-none-any.whl.metadata (4.5 kB)
     Collecting distlib<1,>=0.3.7 (from virtualenv)
       Downloading distlib-0.3.9-py2.py3-none-any.whl.metadata (5.2 kB)
```



jupyter ML_PRM1 Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
     C:\ML_Projects

[8]: !pip install virtualenv
     Collecting virtualenv
       Downloading virtualenv-20.29.2-py3-none-any.whl.metadata (4.5 kB)
     Collecting distlib<1,>=0.3.7 (from virtualenv)
       Downloading distlib-0.3.9-py2.py3-none-any.whl.metadata (5.2 kB)
     Requirement already satisfied: filelock<4,>=3.12.2 in c:\users\system14\anaconda3\lib\site-packages (from virtualenv) (3.13.1)
     Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\users\system14\anaconda3\lib\site-packages (from virtualenv) (3.10.0)
     Downloading virtualenv-20.29.2-py3-none-any.whl (4.3 MB)
        ---------------------------------------- 0.0/4.3 MB ? eta -:--:--
        ---------------------------------------- 0.0/4.3 MB ? eta -:--:--
        ---------------------------------------- 0.1/4.3 MB 825.8 kB/s eta 0:00:06
        ---- ----------------------------------- 0.5/4.3 MB 3.9 MB/s eta 0:00:01
        -------- ------------------------------- 0.9/4.3 MB 5.0 MB/s eta 0:00:01
        ------------- -------------------------- 1.4/4.3 MB 7.1 MB/s eta 0:00:01
        ----------------- ---------------------- 1.9/4.3 MB 7.7 MB/s eta 0:00:01
        --------------------- ------------------ 2.3/4.3 MB 7.9 MB/s eta 0:00:01
        -------------------------- ------------- 2.8/4.3 MB 8.5 MB/s eta 0:00:01
        ------------------------------ --------- 3.2/4.3 MB 8.6 MB/s eta 0:00:01
        ---------------------------------- ----- 3.7/4.3 MB 9.1 MB/s eta 0:00:01
        --------------------------------------- - 4.1/4.3 MB 9.1 MB/s eta 0:00:01
        ---------------------------------------- 4.3/4.3 MB 9.2 MB/s eta 0:00:01
        ---------------------------------------- 4.3/4.3 MB 7.9 MB/s eta 0:00:00
     Downloading distlib-0.3.9-py2.py3-none-any.whl (468 kB)
        ---------------------------------------- 0.0/469.0 kB ? eta -:--:--
        ---------------------------------------- 460.8/469.0 kB 14.5 MB/s eta 0:00:01
        ---------------------------------------- 469.0/469.0 kB 9.8 MB/s eta 0:00:00
     Installing collected packages: distlib, virtualenv
     Successfully installed distlib-0.3.9 virtualenv-20.29.2

[10]: !virtualenv ml_env
```

Date: 4/3/25.

Page No.: 1

Expt. No.: 1

Program1 Install and set up Python and essential libraries like Numpy and Pandas.

Setting up Python and essential libraries on a Windows system for machine learning involves a series of straight forward steps that prepare the environment for data analysis and algorithm development. By installing Python along with Numpy and Pandas, users can handle a wide array of data manipulation tasks efficiently.

Follow the below steps to set up Python and essential libraries such as Numpy and Pandas for machine learning on Windows.

Steps 1: Install Python

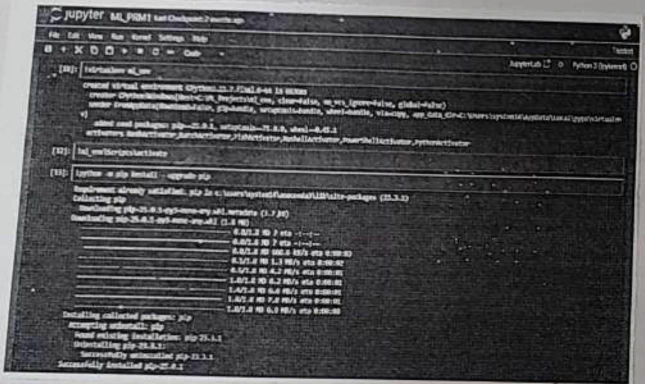
Download Python: Go to the official Python website at python.org, navigate to the "Downloads" section, and dowload the latest version for Windows. Choose the executable installer.

Install Python: Execute the downloaded file. It is crucial to check the box labeled "Add Python 3.x to PATH" at the start of the installation wizard. Select "Customize installation" and ensure all options, including "pip", are selected. In the "Advanced Options," choose "Install for all users" and set the installation path to C:\Python. Proceed by clicking "Install".

Step 2: Install PIP

PIP generally comes installed with python 3.4 and later. To confirm its installation, open Command Prompt and execute:

Teacher's Signature



jupyter ML_PRM1 Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Code
JupyterLab
Python 3 (ipykernel)

```
[10]: !virtualenv ml_env
      created virtual environment CPython3.12.7.final.0-64 in [illegible]ms
        creator CPython3Windows(dest=C:\ML_Projects\ml_env, clear=False, no_vcs_ignore=False, global=False)
        seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\[illegible]\AppData\Local\pypa\virtualen
      v)
          added seed packages: pip==25.0.1, setuptools==75.8.0, wheel==0.45.1
        activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
[12]: !ml_env\Scripts\activate
[13]: !python -m pip install --upgrade pip
      Requirement already satisfied: pip in c:\users\[illegible]\anaconda3\lib\site-packages (23.3.1)
      Collecting pip
        Downloading pip-25.0.1-py3-none-any.whl.metadata (3.7 kB)
      Downloading pip-25.0.1-py3-none-any.whl (1.8 MB)
         ---------------------------------------- 0.0/1.8 MB ? eta -:--:--
         ---------------------------------------- 0.0/1.8 MB ? eta -:--:--
         ---------------------------------------- 0.0/1.8 MB 660.6 kB/s eta 0:00:03
         ---------------------------------------- 0.1/1.8 MB 1.3 MB/s eta 0:00:02
         ---------------------------------------- 0.5/1.8 MB 4.2 MB/s eta 0:00:01
         ---------------------------------------- 1.0/1.8 MB 6.2 MB/s eta 0:00:01
         ---------------------------------------- 1.4/1.8 MB 6.6 MB/s eta 0:00:01
         ---------------------------------------- 1.8/1.8 MB 7.2 MB/s eta 0:00:01
         ---------------------------------------- 1.8/1.8 MB 6.3 MB/s eta 0:00:00
      Installing collected packages: pip
        Attempting uninstall: pip
          Found existing installation: pip 23.3.1
          Uninstalling pip-23.3.1:
            Successfully uninstalled pip-23.3.1
      Successfully installed pip-25.0.1
```

pip --version

If pip is not installed or if we need to udpate it, we can use the following command to install or upgrade pip:
Python -m ensurepip --upgrade

After installation, we can verify that pip is installed correctly by running:

python -m pip --version

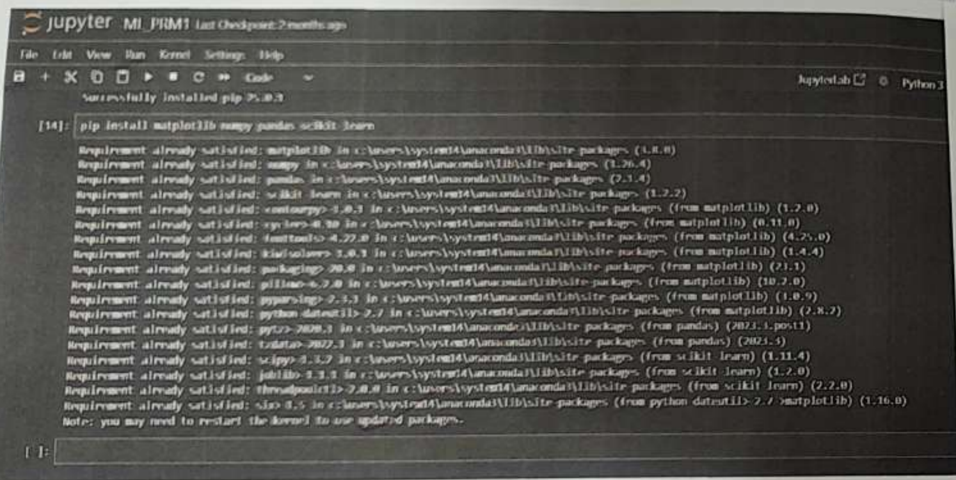
Step 3: Workspace Creation:
A dedicated directory for machine learning projects should be created for organizational clarity. This can be set up using Command Prompt:

```
C:\> mkdir C:\ML_Projects
C:\> cd C:\ML_Projects
```

Step 4: Creating a Virtual Environment
It is recommended to work in an isolated environment to manage dependencies more effectively and avoid conflicts between projects. The virtual environment tool should be installed and a new environment created:

```
C:\> pip install virtualenv # Install virtualenv
C:\> virtualenv ml_env # Create a new virtual environment
                          named ml_env
C:\> ml_env\Scripts\activate # Activate the virtual environment
```

While activated, any packages installed using pip will only


jupyter MI_PRM1 Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Code
JupyterLab
Python 3

```
Successfully installed pip 25.0.1
[14]: pip install matplotlib numpy pandas scikit-learn
Requirement already satisfied: matplotlib in c:\users\system14\anaconda3\lib\site-packages (3.8.0)
Requirement already satisfied: numpy in c:\users\system14\anaconda3\lib\site-packages (1.26.4)
Requirement already satisfied: pandas in c:\users\system14\anaconda3\lib\site-packages (2.1.4)
Requirement already satisfied: scikit-learn in c:\users\system14\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\system14\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\system14\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\system14\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.3.2 in c:\users\system14\anaconda3\lib\site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in c:\users\system14\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\system14\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\system14\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
[ ]:
```

affect this environment.
To exit the virtual environment, simply run : deactivate

Step 5 : Installing Necessary Tools :
Essential Libraries : Libraries such as Jupyter, Numpy, Pandas, Matplotlib, and scikit-learn should be installed if they are not already present. These can be installed using pip, which is Python's package manager. Open Command Prompt and enter the following commands :

```
(ml-env) C:\> python -m pip install --upgrade pip
(ml-env) C:\> pip instal matplotlib numpy pandas
              scikit - learn
```

Teacher's Signature ________

Output :

```
CSV File Data :
        Name     Age    Score
0    Srikanth    28     85
1    Snigdha     22     78
2    Mary        31     92

Excel File Data :
       Name    Course    Sem
0    Rajesh    BCA       1
1    Ramesh    BCA       2
2    Swati     BCOM      1
3    Florina   BCOM      3
4    Pooja     BBA       2
5    Raghu     BBA       4

Data Description :
CSV Data Description :
            Age          Score
count     3.000000       3.0
mean     27.000000      85.0
std       4.582576       7.0
min      22.000000      78.0
25%      25.000000      81.5
50%      28.000000      85.0
75%      29.500000      88.5
max      31.000000      92.0
```

Date: 11/3/25

Expt. No.: 2

Page No.: 4

2. Write a program to Load and explore the dataset of CSV and excel files using pandas.

Step 1: Creating CSV and Excel Files with Dummy Data

- Create CSV File: Open a text editor like Notepad or any other code editor. Enter the following data

```
Name , Age , Score
Srikanth, 28 , 85
Snigdha , 22 , 78
Mary , 31 , 92
```

Save this file as sample_data.csv in the C:\ML_Projects directory.

- Create Excel File: We can us Microsoft Excel or Google Sheets to create this file. Enter the below data:

Name	Course	Sem
Rajesh	BCA	1
Ramesh	BCA	2
Swati	BCOM	1
Florina	BCOM	3
Pooja	BBA	2
Raghu	BBA	4

Save this file as sample_data.xlsx in the C:\ML_Projects directory.

Teacher's Signature

```
Excel Data Description :
            Sem
count    6.000000
mean     2.166667
std      1.169045
min      1.000000
25%      1.250000
50%      2.000000
75%      2.750000
max      4.000000

Data Types in CSV File :
Name      object
Age       int64
Score     int 64
dtype : object

Data Types in Excel File :
Name      object
Course    object
Sem       int64
dtype : object
```

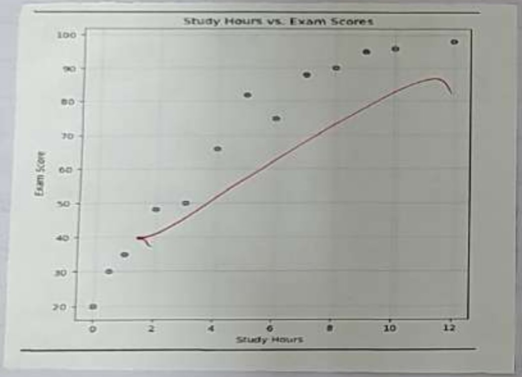
Date: ....

Expt. No.: ....

Step 2 : Python Code to Load and Explore the Data

```
import pandas as pd

# Define the file paths
csv_file_path = 'C:\\ML_Projects\\sample_data.csv'
excel_file_path = 'C:\\ML_Projects\\sample_data.xlsx'

# Load the CSV file
data_csv = pd.read_csv(csv_file_path)
print ("CSV File Data :")
print(data_csv)

# Load the Excel file
data_excel = pd.read_excel(excel_file_path)
print ("\nExcel File Data :")
print(data_excel)

# Basic Data Exploration)
print ("\nData Descriptions:")
print ("CSV Data Description :")
print (data_csv.describe())
print ("\n Excel Data Description :")
print (data_excel.describe())

# Displaying data types
print ("\n Data Types in CSV File :")
print (data_csv.dtypes)
print ("\n Data Types in Excel File :")
print (data_excel.dtypes)
```

Teacher's Signature



Date: 25/3/25

Expt. No.: 3 Page No.: 6

3. Write a program to Visualize the dataset to gain insights using Matplotlib by plotting scatter plots, bar charts.

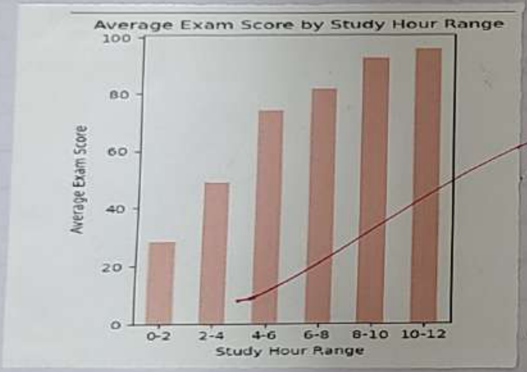
Step 1: Create the CSV File:
Create a CSV file with below data of student study hours and exam scores: Save this file as study_data.csv.

```
Student ID, Study Hours, Exam Score
1,5,82
2,2,48
3,8,90
4,1,35
5,3,50
6,4,66
7,9,95
8,6,75
9,7,88.
10,0.5,30
11,10,96
12,0,20
13,12,98
```

Step 2: Python Code:

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

#Load the data
data = pd.read_csv('C:\\ML_Projects\\study_data.csv')
```

Teacher's Signature



```
# Scatter plot of Study Hours vs. Exam Scores
plt. figure ( figsize = (14,7))
plt. subplot (1,2,1)
plt. scatter (data ['Study Hours'], data['Exam Score'],
              color = 'dodgerblue', edgecolor = 'k', alpha = 0.7)
plt. title ('Study Hours vs. Exam Scores)
plt. xlabel ('Study Hours')
plt. ylabel ('Exam Scores')
plt. grid (True)

# Bar chart of Average Exam Score by Study Hour Range
# Creating bins for study hour ranges
bins = [0,2,4,6,8,10,12]
labels = ['0-2','2-4','4-6','6-8','8-10','10-12']
data['Study Hour Range'] = pd.cut (data['Study Hours'],
                    bins = bins, labels = labels, right = False)
grouped_data = data.groupby ('Study Hour Range')['Exam Score'].
                    mean()
plt. subplot (1,2,2)
grouped_data.plot (kind = 'bar', color = 'salmon')
plt. title ('Average Exam Score by Study Hour Range')
plt. xlabel ('Study Hour Range')
plt. ylabel ('Average Exam Score')
plt. xticks (rotation = 0)
plt. tight_layout ()
plt. show()
```

Teacher's Signature

Output :

```
Data after handling missing values:
     Age   Gender    Income
0   25.0   Female   50000.0
1   30.0   Male     60000.0
2   29.5   Male     45000.0
3   28.0   Female   56250.0
4   35.0   Male     70000.0

Data after categorical encoding:
     Gender_Female   Gender_Male
0       1.0             0.0
1       0.0             1.0
2       0.0             1.0
3       1.0             0.0
4       0.0             1.0

Data after feature scaling:
     Scaled Age     Scaled Income
0    -1.382164      -0.727778
1     0.153574       0.436667
2     0.000000      -1.310001
3    -0.460721       0.000000
4     1.689312       1.601112
```

Date: 1/4/25

Expt. No.: 4 Page No.: 8

4. Write a program to Handle missing data, encode categorical variables, and perform feature scaling.

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler

# Create dummy data
data = {
        'Age' : [25, 30, None, 28, 35],
        'Gender' : ['Female', 'Male', 'Male', 'Female', 'Male'],
        'Income' : [50000, 60000, 45000, None, 70000]
}
df = pd.DataFrame(data)

# Handling missing data
imputer = SimpleImputer(strategy = 'mean')
df[['Age', 'Income']] = imputer.fit_transform(df[['Age',
                         'Income']])

#Print data after handling missing values
print("Data after handling missing values:")
print(df)

# Encoding categorical variables
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(df[['Gender']]).t
```

Teacher's Signature

```
#Print data after categorical encoding
encoded_df = pd.DataFrame(encoded_data, columns = encoder.
                get_feature_names_out([['Gender']])
print("\nData after categorical encoding :")
print(encoded_df)

#feature Scaling
scaler = Standardscaler()
scaled_data = scaler.fit_transform(df[['Age'], 'Income']])

#Print data after feature scaling
scaled_df = pd.DataFrame(scaled_data, columns =['Scaled
              Age', 'Scaled Income'])
print("\nData after feature scaling :")
print(scaled_df)
```

Output :

```
Output 1:
Accuracy on the test set : 1.00
Enter Exam Score 1 : 45
Enter Exam Score 2 : 50
Based on the exam scores provided, the student is predicted
to fail.

Output 2:
Accuracy on the test set : 1.00
Enter Exam Score1 : 75
Enter Exam Score2 : 89
Based on the exam scores provided, the student is
predicted to pass.
```

Date: 8/4/25

Expt. No.: 5

Page No.: 10

5. Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikitlearn and Train the classifier on the dataset and evaluate its performance.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Dummy student data: exam score1, exam score2,
                              pass/fail (features)
X = np.array([[80,75],[95,90],[60,50],[45,30],[30,40],
              [85,95],[70,60],[50,55],[40,45],[60,70]])
y = np.array([1,1,0,0,0,1,1,0,0,1])

# Split the 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)

# Initialize the k-NN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)

#Train the classifier on the training data
knn.fit(X_train, ytrain)

# Evaluate the classifier's performance
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy on the test set : {:.2f}".format(accurac
```

Teacher's Signature

Date: ....

Expt. No.: .... Page No.: ......11..........

```
#Take user input for exam scores
exam_score1 = float (input ("Enter Exam Score1 :" ))
exam_score2 = float (input ("Enter Exam Score2 :" ))

#Prepare the user input for prediction
user_input = np.array ([[exam_score1, exam_score2]])

# Use the trained k-NN classifier to predict the outcome
predicted_outcome = knn.predict (user_input)

if predicted_outcome [0] == 1 :
    print ("Based on the exam scores predicted , the
            student is predicted to pass .")
else :
    print ("Based on the exam scores provided , the student
            is predicted to fail.")
```

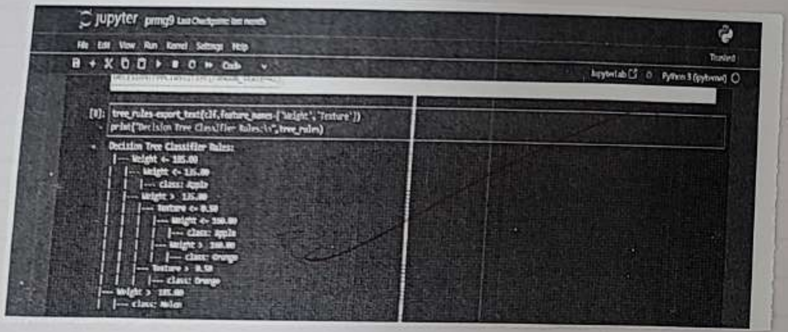
Output:

```
Enter the size of the house in sqft: 1600
Enter the number of bedrooms : 3
Predicted price for a house with size 1600.0 sqft and
3 bedrooms : Rs. 418163.93
```

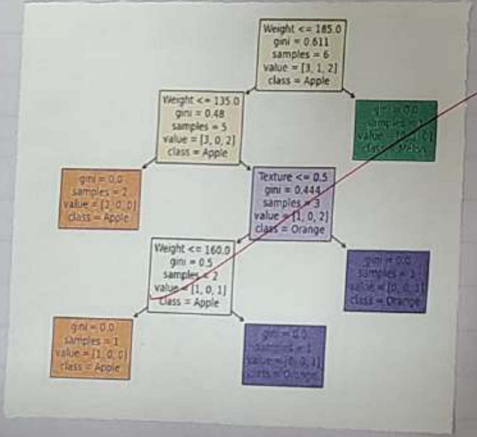
6. Write a program to implement a linear regression model for regression tasks and Train the model on a dataset with continuous target variables.

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Dummy house price prediction data: features (house size,
  number of bedrooms) and target variable (house price)
X = np.array([[1000,2],[1500,3],[1200,2],[1800,4],[900,2],
              [2000,3]])
y = np.array([300000,400000,350000,500000,280000,
              450000])

# Initialize the Linear Regression model
model = LinearRegression()

# Train the model on the dataset
model.fit(X,y)

# Take input from the user for new house data
size = float(input("Enter the size of the house in sqft:"))
bedrooms = int(input("Enter the number of bedrooms:"))
new data = np.array([[size,bedrooms]])

# Predict the price for the new house data
predicted_price = model.predict(new data)
```

```
# Print the predicted price for the new house data
print ("Predicted price for a house with size {} sqft
     and{} bedrooms : Rs. {:.2f}". format (size, bedrooms,
        predicted_price [0]))
```


jupyter prmg9 Last Checkpoint: last month
File Edit View Run Kernel Settings Help
Code
Trusted
JupyterLab
Python 3 (ipykernel)

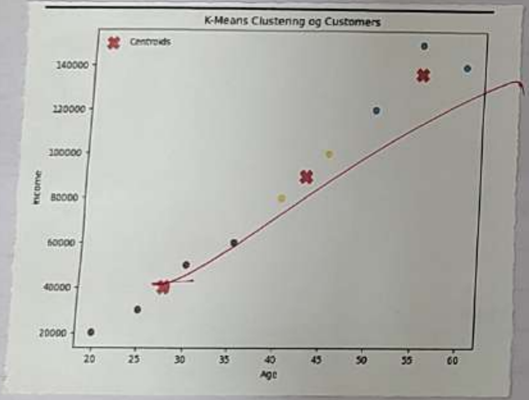
```
[8]: tree_rules=export_text(clf,feature_names=['Weight','Texture'])
     print("Decision Tree Classifier Rules:\n",tree_rules)

     Decision Tree Classifier Rules:
      |--- Weight <= 185.00
      |   |--- Weight <= 135.00
      |   |   |--- class: Apple
      |   |--- Weight >  135.00
      |   |   |--- Texture <= 0.50
      |   |   |   |--- Weight <= 160.00
      |   |   |   |   |--- class: Apple
      |   |   |   |--- Weight >  160.00
      |   |   |   |   |--- class: Orange
      |   |   |--- Texture >  0.50
      |   |   |   |--- class: Orange
      |--- Weight >  185.00
      |   |--- class: Melon
```



7 Write a program to implement a decision tree classifier using scikit-learn and visualize the decision tree and understand its splits.

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot tree
from sklearn.tree import export text
import matplotlib.pyplot as plt

# Custom dummy data for fruit classification
# Features : [Weight, Texture] -> Target : [Fruit Type]
X = np.array [[150,0], [170,1], [120,0], [140,1], [200,1],
                    [130,0]])
y= np.array(['Apple','Orange','Apple','Orange','Melon',
                    'Apple'])

# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier (random_state = 42)
clf.fit (X,y)

# Visualize the Decision Tree splits
tree_rules = export_text (clf, feature_name = ['Weight',
                    'Texture'])
print ("Decision Tree Classifier Rules :\n", tree rules)

#Plot the Decision Tree
plt. figure (figsize =(10,6))
plot tree (clf, filled = True, feature-names = ['Weight', 'Texture'],
                    class names = np.unique(y))
plt.show()
```

Teacher's Signature



8. Write a program to Implement K-Means clustering and Visualize clusters.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Generate dummy customer data (Age, Income)
X = np.array([[30,50000],[35,60000],[40,80000],[25,30000],
              [45,100000],[20,20000],[50,120000],[55,150000],
              [60,140000],[28,40000]])

# Initialize K-Means with 2 Clusters
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)

# Get cluster labels and cluster centers
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# Visualize the clusters
plt.figure(figsize=(8,6))
plt.scatter(X[:,0] X[:,1], c=labels, cmap='viridis', s=50,
            alpha=0.8)
plt.scatter(centers[:,0], centers[:,1], c='red', s=200,
            marker='X', label='Centroids')
plt.xlabel('Age')
plt.ylabel('Income')
plt.title('K-Means Clustering of Customers')
```

Teacher's Signature

```
plt.legend()
plt.show()
```

9 Introduce scikit-learn as a machine learning library.

Scikit-learn is a popular open-source machine learning library in Python that offers a comprehensive set of tools and algorithms for data analysis, modeling, and machine learning tasks. It is built on foundational libraries like NumPy, SciPy and Matplotlib. Scikit-learn provides a user-friendly and efficient framework for both beginners and experts in the field of data science.

Some key points to introduce scikit-learn as a machine learning library:

1. Comprehensive Machine Learning Library: Scikit-learn offers a wide range of machine learning algorithms and tools for various tasks such as classification, regression, clustering, dimensionality reduction, and more.

2. User-Friendly and Easy to Use: It is designed with a user-friendly interface and simple syntax, making it accessible for both beginners and experienced machine learning practitioners.

3. Integration with Scientific Computing Libraries: Scikit-learn integrates well with other scientific computing libraries in Python such as NumPy, SciPy, and Matplotlib, providing a powerful environment for machine learning tasks.

4. Extensive Documentation and Community Support: The library comes with comprehensive documentation, tutorials, and examples to help users understand and implement machine learning algorithms effectively. Additionally, there is a vibrant community

around scikit-learn that provides support and contributions

5. Efficient Implementation of Algorithms: Scikit-learn is built on top of NumPy, SciPy, and Python, which allows for efficient implementation of machine learning algorithms and scalability to large datasets.

6. Support for Model Evaluation and Validation: The library provides tools for model evaluation, hyperparameter tuning, cross-validation, and performance metrics, enabling users to assess and improve the quality of their machine learning models.

7. Flexibility and Customization: Scikit-learn offers flexibility for customization and parameter tuning, allowing users to adapt algorithms to their specific requirements and ~~d~~ datasets.

8. Wide Adoption and Industry Usage: Due to its ease of use, performance, and versatility, scikit-learn is widely adopted in academia, research, and industry for various machine learning applications.

Overall, scikit-learn is a powerful and versatile machine learning library in Python that empowers users to build and deploy machine learning models efficiently for a wide range of tasks and applications.

Expt. No.: 10 Date: 20/5/25

10 Install and set up scikit-learn and other necessary tools.

Step 1 : Install Python
Go to the official Python website at Python.org navigate to the "Download" section and download the latest version for windows. Choose the executable installer.

Step 2 : PIP generally comes installed with Python 3.4 and later.

To confirm use : pip --version
If not installed or to upgrade it :

```
python -m ensure pip --upgrade
```

After installation, verify using

```
python -m pip --version
```

Step 3 : Workspace Creation :

```
c:\> mkdir  c:\ML_Projects
c:\> cd     c:\ML_Projects
```

Step 4 : Creating a Virtual Environment

```
c:\> pip install virtualenv
c:\> virtualenv ml-env
c:\> ml_env\scripts\activate
```

To exit the virtual environment, simply run:
deactivate

Teacher's Signature ________________

step 5 : Installing necessary tools

```
(ml_env) C:\> python -m pip install --upgrade pip

(ml_env) C:\> pip install matplotlib numpy pandas
              scikit-learn
```