7 Lab Manual for Machine Learning

Practical : 1
AIM : Numerical Computing with Python (NumPy, Matplotlib)

Practical : 2
AIM : Introduction to Pandas for data import and export (Excel, CSV etc.)

Practical : 3
AIM : Basic Introduction to Scikit learn

Practical : 4
AIM : Implement the Find-S concept learning algorithm that finds the most specific hypothesis that is consistent with the given training data.

Practical : 5
AIM : Import Pima Indian diabetes data. Apply select K best and chi2 for feature selection. Identify the best features

Practical : 6
AIM : Write a program to learn a decision tree and use it to predict class labels of test data
Training and test data will be explicitly provided by instructor. Tree pruning should not be performed.

Practical : 7
AIM : ML Project. Use the following dataset as music.csv

Practical : 8
AIM : Write a program to use a K-nearest neighbor it to predict class labels of test data. Training and test data must be provided explicitly.

Practical : 9
AIM : Import vgsales.csv from kaggle platform.
a. Find rows and columns in dataset
b. Find basic information regarding dataset using describe command.
c. Find values using values command.

Practical : 10
AIM : Project on regression
a. Import home_data.csv on kaggle using pandas
b. Understand data by running head .info and describe command
c. Plot the price of house with respect to area using matplotlib library
d. Apply linear regression model to predict the price of house

Practical : 11
AIM : Write a program to cluster a set of points using K-means. Training and test data must be provided explicitly.

Practical : 12
AIM : Import Iris dataset
a. Find rows and columns using shape command
b. Print first 30 instances using head command
c. Find out the data instances in each class.(use groupby and size)
e. Plot the univariategraphs(box plot and histogr[illegible]s)
f. Plot the multivariate plot(scatter matrix)
g. Split data to train model by 80% data values
h. Apply K-NN and k means clustering to check accuracy and decide which is better.

PRACTICAL : 1

AIM: Numerical Computing with Python (NumPy, Matplotlib)

Introduction:

Numerical computing with Python refers to the use of Python programming language for performing mathematical and scientific computations. Python is a high-level programming language that is easy to learn, has a clear and concise syntax, and is widely used in scientific computing.

Python provides several libraries that are useful for numerical computing, including:

1. NumPy: NumPy is a library for numerical computing in Python. It provides an array object, which is faster and more efficient than traditional Python lists for performing mathematical operations. NumPy also provides functions for linear algebra, Fourier transforms, and random number generation.
2. SciPy: SciPy is a library for scientific computing in Python. It provides functions for optimization, signal processing, and statistics, among other things.
3. Matplotlib: Matplotlib is a library for creating data visualizations in Python. It provides a variety of plots, including line plots, scatter plots, and histograms.
4. Pandas: Pandas is a library for data analysis in Python. It provides data structures for working with structured data, such as tables and time series.
5. SymPy: SymPy is a library for symbolic mathematics in Python. It provides tools for working with mathematical expressions and equations symbolically, rather than numerically.

Numerical computing with Python is used in a wide range of scientific and engineering applications, including data analysis, image processing, and simulation. It is also used in machine learning and deep learning applications, where numerical computations are performed on large datasets.

Numpy

→ **Installing Numpy**

pip install numpy

→ **Importing Numpy Library**

importnumpy as np

→ **Creating arrays : NumPy provides several functions for creating arrays, including np.array, np.zeros, np.ones, and np.random.rand. For example, to create an array of zeros with shape (3, 3), you can use:**

Code:

```
importnumpy as np
a = np.zeros((3, 3))
```

Output:

```
In [2]: import numpy as np
        a = np.zeros((3, 3))
        a

Out[2]: array([[0., 0., 0.],
               [0., 0., 0.],
               [0., 0., 0.]])
```

→ **Array indexing and slicing: NumPy arrays can be indexed and sliced like regular Python lists, but also support more advanced indexing and slicing operations. For example, to extract the first row of a 2-dimensional array a, you can use:**

Code:

```
row = a[0, :]
```

Output:

```
In [4]: row = a[0, :]
        row

Out[4]: array([0., 0., 0.])
```

→ **Mathematical functions: NumPy provides a range of mathematical functions for performing operations on arrays, such as np.add, np.subtract, np.multiply, np.divide, np.power, and np.sqrt. For example, to compute the element-wise square root of an array a, you can use:**

Code:

```
x=np.array([4,9,16,25])
b = np.sqrt(x)
```

Output:

```
In [ ]: x=np.array([4,9,16,25])
        b = np.sqrt(x)
        b

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

→ **Aggregation functions: NumPy provides functions for aggregating arrays, such as np.sum, np.mean, np.median, np.std, and np.max. For example, to compute the sum of all elements in an array a, you can use:**

```
x=np.array([4,9,16,25])
total = np.sum(x)
```

Output:

```
In [9]: x=np.array([4,9,16,25])
        total = np.sum(x)
        total

Out[9]: 54
```

→ **Linear algebra functions: NumPy provides functions for performing linear algebra operations, such as matrix multiplication (np.dot), matrix inversion (np.linalg.inv), and eigendecomposition (np.linalg.eig). For example, to compute the dot product of two matrices A and B, you can use:**

Code:

```
x=np.array([4,9,16,25])
y=np.array([1,2,3,4])
C = np.dot(x,y)
```

Output

```
In [11]: x=np.array([4,9,16,25])
         y=np.array([1,2,3,4])
         C = np.dot(x,y)
         C

Out[11]: 170
```

Matplotlib

→ **Installing Library**

pip install matplotlib

→ **Importing Library**

importmatplotlib.pyplot as plt

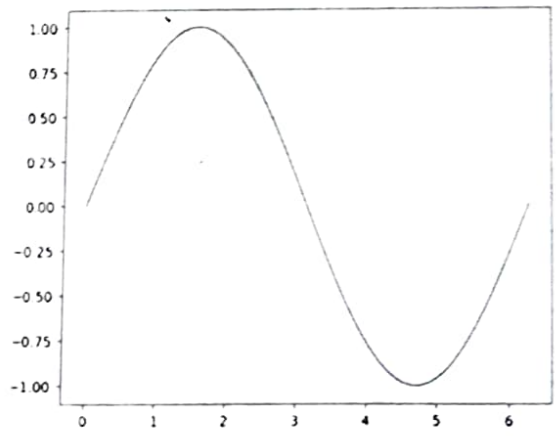
→ **Creating a line plot: Line plots are a simple way to visualize a sequence of data points. To create a line plot in Matplotlib, you can use the plot function. For example, to plot a sine wave from 0 to 2ð, you can use:**

Code:

```
importnumpy as np
importmatplotlib.pyplot as plt
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

Output:

```
In [14]: import numpy as np
         import matplotlib.pyplot as plt
         x = np.linspace(0, 2*np.pi, 100)
         y = np.sin(x)
         plt.plot(x, y)
         plt.show()
```

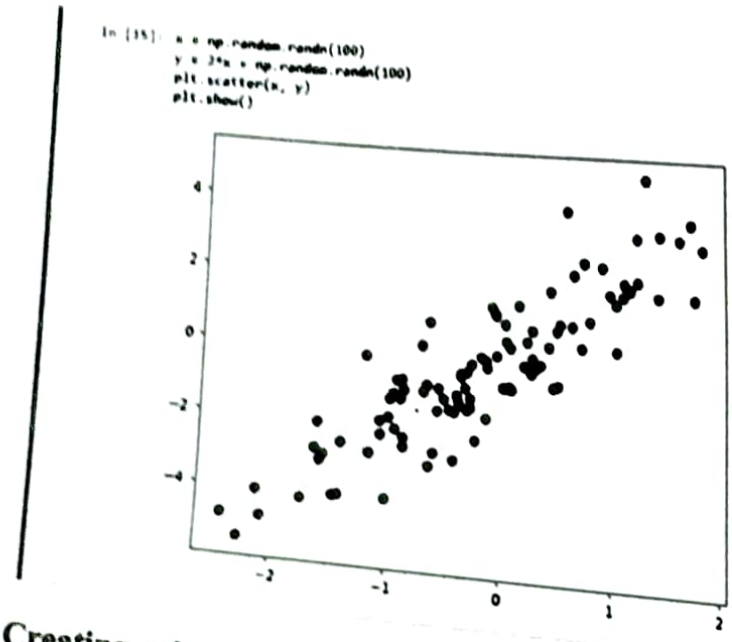


→ **Creating a scatter plot: Scatter plots are a useful way to visualize the relationship between two variables. To create a scatter plot in Matplotlib, you can use the scatter function. For example, to plot the relationship between two variables x and y, you can use:**

Code:

```
x = np.random.randn(100)
y = 2*x + np.random.randn(100)
plt.scatter(x, y)
plt.show()
```

Output:



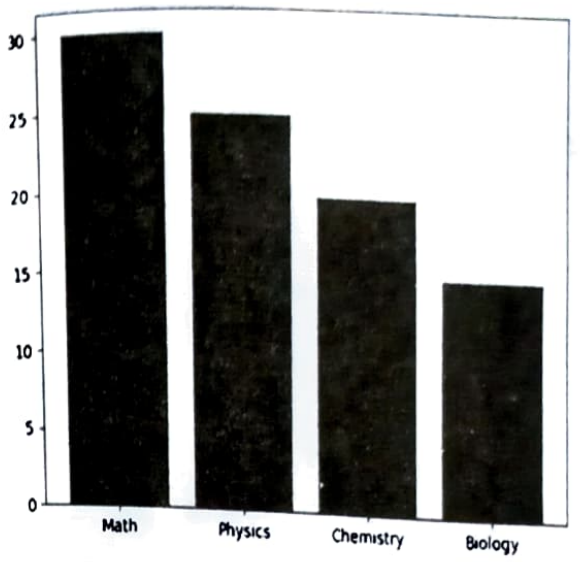
→ **Creating a bar plot: Bar plots are a useful way to compare the values of different categories. To create a bar plot in Matplotlib, you can use the bar or barh function. For example, to plot the number of students who prefer different subjects, you can use:**

Code:

```
subjects = ['Math', 'Physics', 'Chemistry', 'Biology']
students = [30, 25, 20, 15]
plt.bar(subjects, students)
plt.show()
```

Output:

```
In [16]: subjects = ['Math', 'Physics', 'Chemistry', 'Biology']
         students = [30, 25, 20, 15]
         plt.bar(subjects, students)
         plt.show()
```



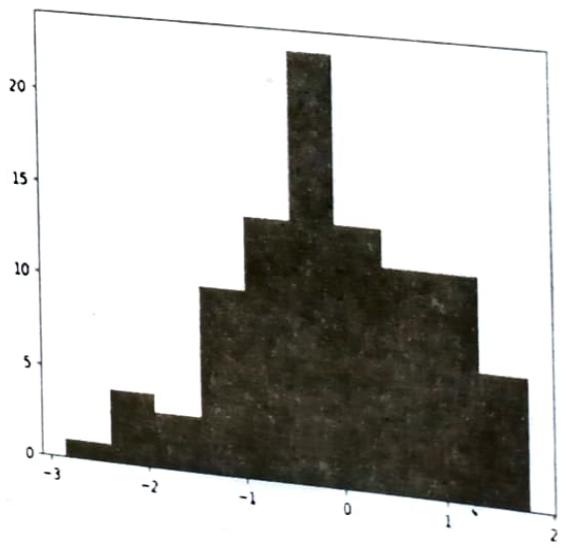
→ **Creating a histogram: Histograms are a useful way to visualize the distribution of a dataset. To create a histogram in Matplotlib, you can use the hist function. For example, to plot the distribution of exam scores, you can use:**

Code:

```
scores = np.random.randn(100)
plt.hist(scores, bins=10)
plt.show()
```

Output:

```
In [17]: scores = np.random.randn(100)
         plt.hist(scores, bins=10)
         plt.show()
```



PRACTICAL : 2

AIM: Introduction to Pandas for data import and export (Excel, CSV etc.)

Pandas is a popular library in Python for data analysis and manipulation. It provides a range of powerful tools for working with structured data, including the ability to import and export data in various formats.

Pandas

→ **Installing Library**

pip install pandas

→ **Importing Library**

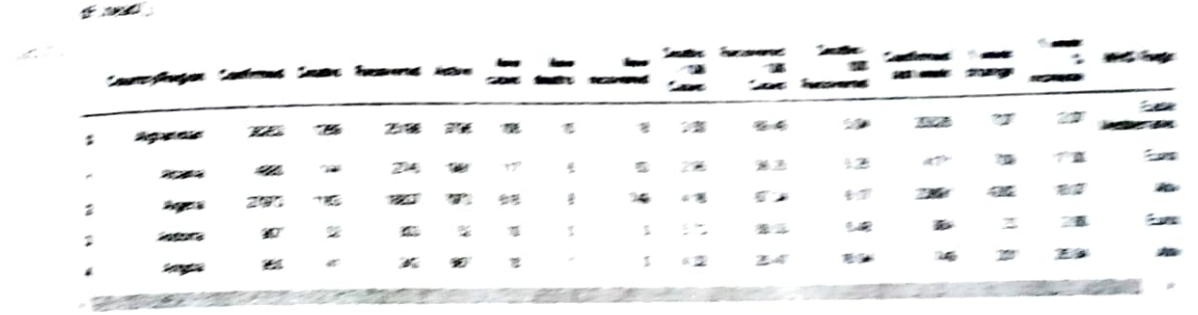
import pandas as pd

→ **Reading a CSV file:**

Code:

```
import pandas as pd
df=pd.read_csv("D:\photos\Covid_dataset\country_wise_latest.csv")
df.head()
```

Output:



	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase	WHO Region
0	Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69.49	5.04	35526	737	2.07	Eastern Mediterranean
1	Albania	4880	144	2745	1991	117	6	63	2.95	56.25	5.25	4171	709	17.00	Europe
2	Algeria	27973	1163	18837	7973	616	8	749	4.16	67.34	6.17	23691	4282	18.07	Africa
3	Andorra	907	52	803	52	10	1	1	5.73	88.53	6.48	884	23	2.60	Europe
4	Angola	950	41	242	667	18	1	16	4.32	25.47	16.94	749	201	26.84	Africa

→ **Reading an Excel file:**

Code:

```
import pandas as pd
df = pd.read_excel("C:\Users\DELL\Desktop\Historicalinvestemp.xlsx")
df.head()
```

Output:

```
In [17]: import pandas as pd
         df = pd.read_excel("C:\Users\DELL\Desktop\Historicalinvestemp.xlsx");
         df.head()

Out[17]:
```

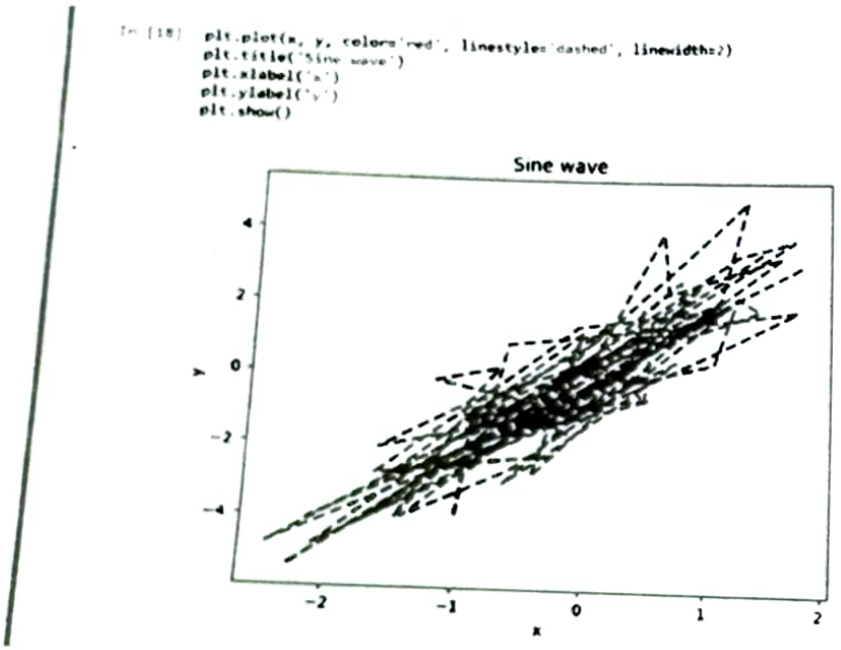
	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	Annual Returns on investments in	NaN	NaN

→ **Customizing plots: Matplotlib provides a range of options for customizing plots, such as changing the colors and styles of lines, adding labels and titles, and changing the size and aspect ratio of the plot. For example, to add a title and axis labels to a plot, you can use:**

Code:

```
plt.plot(x, y, color='red', linestyle='dashed', linewidth=2)
plt.title('Sine wave')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Output:



○ ○ ○

PRACTICAL : 2

AIM: Introduction to Pandas for data import and export (Excel, CSV etc.)

Pandas is a popular library in Python for data analysis and manipulation. It provides a range of powerful tools for working with structured data, including the ability to import and export data in various formats.

Pandas

→ **Installing Library**

pip install pandas

→ **Importing Library**

import pandas as pd

→ **Reading a CSV file:**

Code:

```
import pandas as pd
df=pd.read_csv("D:\photos\Covid_dataset\country_wise_latest.csv")
df.head()
```

Output:

```
In [7]: import pandas as pd
        df = pd.read_csv("D:\photos\Covid_dataset\country_wise_latest.csv")
        df.head()
```

Out[7]:

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase	WHO Regic
0	Afghanistan	36263	1269	25198	9796	106	10	18	3 50	69 49	5 04	35526	737	2 07	Easte Mediterrane
1	Albania	4880	144	2745	1991	117	6	63	2 95	56 25	5 25	4171	709	17 00	Euro
2	Algeria	27973	1163	18837	7973	616	8	749	4 16	67 34	6 17	23691	4282	18 07	Afr
3	Andorra	907	52	803	52	10	0	0	5 73	88 53	6 48	884	23	2 60	Euro
4	Angola	950	41	242	667	18	1	0	4 32	25 47	16 94	749	201	26 84	Afr

→ **Reading an Excel file:**

Code:

```
import pandas as pd
df = pd.read_excel("C:/Users/DELL/Desktop/Historicalinvesttemp.xlsx");
df.head()
```

Output:

```
In [17]: import pandas as pd
         df = pd.read_excel("C:/Users/DELL/Desktop/Historicalinvesttemp.xlsx");
         df.head()
```

Out[17]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	Annual Returns on Investments in	NaN	NaN

Required Dependency

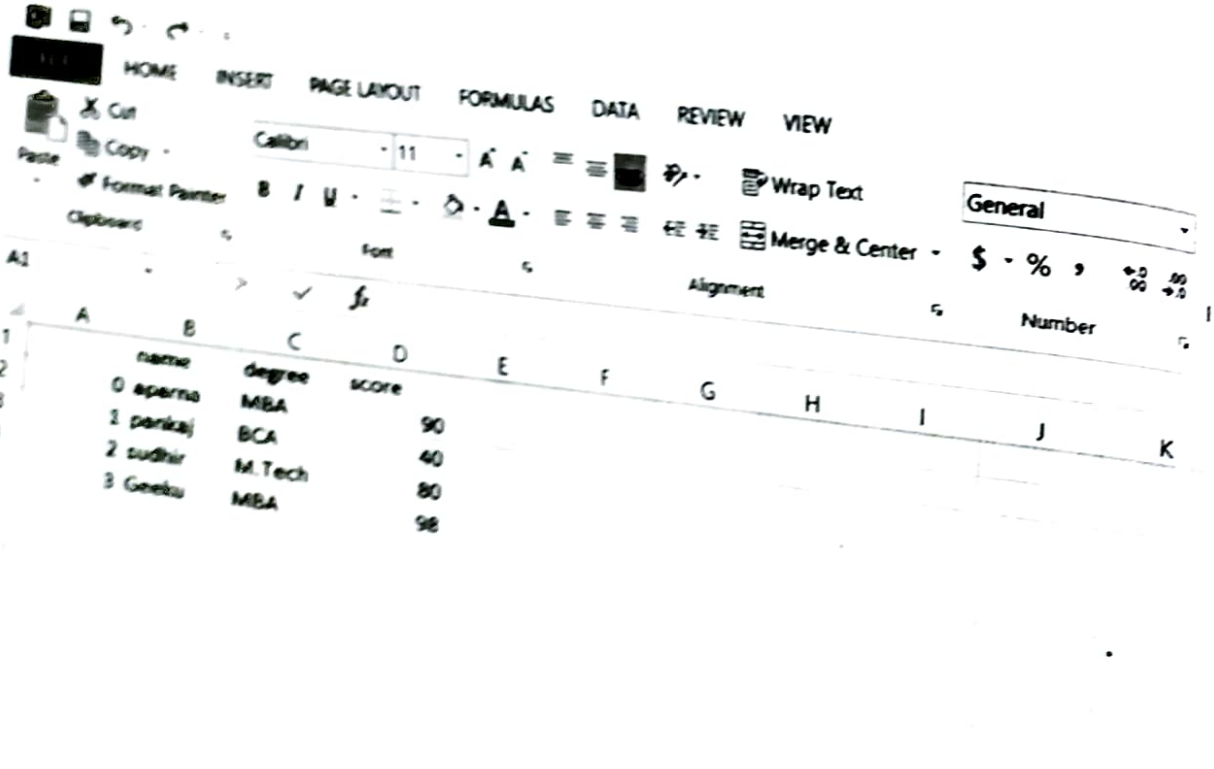
pip install openpyxl

→ **Writing a CSV file:**

Code:

```
import pandas as pd
# list of name, degree, score
nme = ["aparna", "pankaj", "sudhir", "Geeku"]
deg = ["MBA", "BCA", "M.Tech", "MBA"]
scr = [90, 40, 80, 98]
# dictionary of lists
dict = {'name': nme, 'degree': deg, 'score': scr}
df = pd.DataFrame(dict)
df.to_csv('C:/Users/DELL/Desktop/file3.csv')
```

Output:

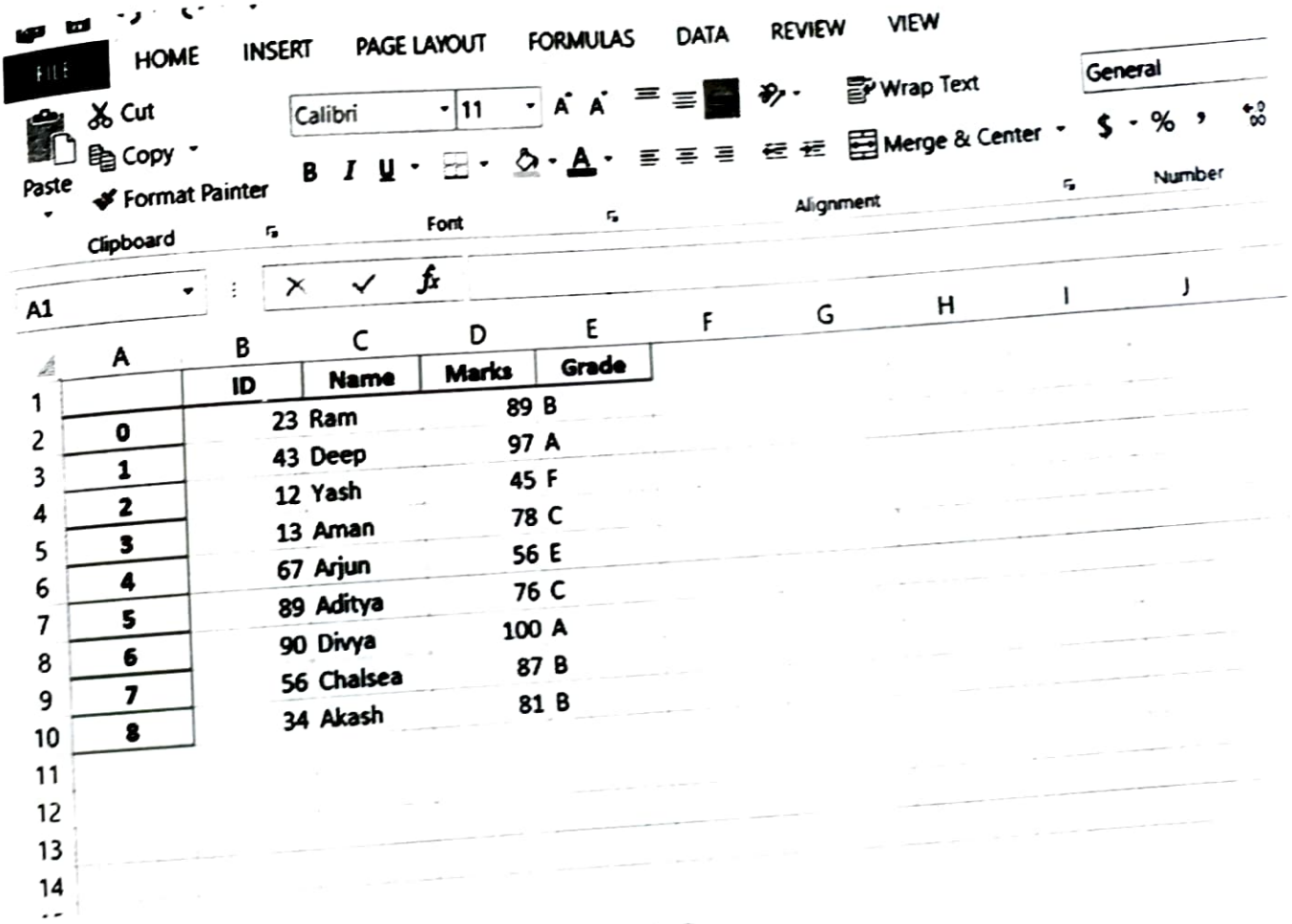


→ **Writing an Excel file:**

Code:

```
# importing packages
import pandas as pd
# dictionary of data
dct = {'ID': {0: 23, 1: 43, 2: 12,
        3: 13, 4: 67, 5: 89,
        6: 90, 7: 56, 8: 34},
    'Name': {0: 'Ram', 1: 'Deep',
        2: 'Yash', 3: 'Aman',
        4: 'Arjun', 5: 'Aditya',
        6: 'Divya', 7: 'Chalsea',
        8: 'Akash' },
    'Marks': {0: 89, 1: 97, 2: 45, 3: 78,
        4: 56, 5: 76, 6: 100, 7: 87,
8: 81},
    'Grade': {0: 'B', 1: 'A', 2: 'F', 3: 'C',
        4: 'E', 5: 'C', 6: 'A', 7: 'B',
        8: 'B'}
    }
# forming dataframe
data = pd.DataFrame(dct)
# storing into the excel file
data.to_excel("C:/Users/DELL/Desktop/
output.xlsx")
```

Output :



○○○

PRACTICAL : 3

AIM: Basic Introduction to Scikit learn

Scikit-learn, also known as sklearn, is a popular Python library used for machine learning and statistical modeling. It provides a range of tools for data preprocessing, feature extraction, supervised and unsupervised learning, and model evaluation.

Scikit-learn is built on top of other popular Python libraries such as NumPy, SciPy, and matplotlib, and provides a consistent interface for various machine learning algorithms.

Some of the key features of Scikit-learn include:

1. Data preprocessing: Scikit-learn provides a range of tools for data preprocessing, such as scaling, normalization, and feature selection.
2. Supervised learning: Scikit-learn provides a range of algorithms for supervised learning, including linear regression, logistic regression, decision trees, and support vector machines.
3. Unsupervised learning: Scikit-learn provides a range of algorithms for unsupervised learning, including clustering, dimensionality reduction, and anomaly detection.
4. Model selection and evaluation: Scikit-learn provides tools for model selection and evaluation, including cross-validation, hyperparameter tuning, and metrics for evaluating model performance.

To use Scikit-learn, you typically follow the following workflow:

1. Prepare the data: Load and preprocess the data using Scikit-learn's data preprocessing tools.
2. Choose a model: Select a model that is appropriate for your problem, based on the type of data and the task at hand.
3. Train the model: Fit the model to the training data using Scikit-learn's fit function.
4. Evaluate the model: Evaluate the performance of the model using Scikit-learn's evaluation metrics.
5. Tune the model: Fine-tune the model by adjusting its hyperparameters using Scikit-learn'shyperparameter tuning tools.
6. Use the model: Use the trained model to make predictions on new data.

Here are some examples of basic functions in Scikit-learn:

→ **Loading a dataset**

Scikit-learn provides a number of built-in datasets that you can use for testing and practicing. For example, to load the iris dataset, you can use:

Code:

```
fromsklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target
```

Output :

```
In [31]: from sklearn.datasets import load_iris
         iris = load_iris()
         X, y = iris.data, iris.target
         X

Out[31]: array([[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],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
                [5.7, 3.8, 1.7, 0.3],
```

→ **Splitting data into training and test sets**

Before training a model, it's important to split the data into training and test sets to evaluate the model's performance on new, unseen data. Scikit-learn provides a function for doing this, called train_test_split. For example, to split the iris dataset into training and test sets with 80% of the data used for training, you can use

Code:

```
fromsklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Output :

```
In [34]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         X_train

Out[34]: array([[4.6, 3.6, 1. , 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [6.7, 3.1, 4.4, 1.4],
                [4.8, 3.4, 1.6, 0.2],
                [4.4, 3.2, 1.3, 0.2],
                [6.3, 2.5, 5. , 1.9],
                [6.4, 3.2, 4.5, 1.5],
                [5.2, 3.5, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.2, 4.1, 1.5, 0.1],
                [5.8, 2.7, 5.1, 1.9],
                [6. , 3.4, 4.5, 1.6],
                [6.7, 3.1, 4.7, 1.5],
                [5.4, 3.9, 1.3, 0.4],
                [5.4, 3.7, 1.5, 0.2],
                [5.5, 2.4, 3.7, 1. ],
                [6.3, 2.8, 5.1, 1.5],
                [6.4, 3.1, 5.5, 1.8],
                [6.6, 3. , 4.4, 1.4],
```

→ **Training a model**

Scikit-learn provides a wide range of machine learning models that you can use for different tasks. For example, to train a logistic regression model on the iris dataset, you can use

Code:

```
fromsklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

Output:

```
In [35]: from sklearn.linear_model import LogisticRegression
         model = LogisticRegression()
         model.fit(X_train, y_train)

         C:\Users\DELL\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\linear_model\_logistic
         g: lbfgs failed to converge (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(
Out[35]: LogisticRegression
         LogisticRegression()
```

→ **Evaluating a model**

Once you've trained a model, you can evaluate its performance on the test set using a variety of metrics. Scikit-learn provides functions for computing many of these metrics, such as accuracy, precision, recall, and F1 score. For example, to compute the accuracy of the logistic regression model on the test set, you can use:

Code:

```
fromsklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Output:

```
In [37]: from sklearn.metrics import accuracy_score
         y_pred = model.predict(X_test)
         accuracy = accuracy_score(y_test, y_pred)
         accuracy

Out[37]: 1.0
```

→ Cross-validation

Cross-validation is a technique for evaluating a model's performance by splitting the data into multiple subsets and training and testing the model on different combinations of the subsets. Scikit-learn provides a function for performing k-fold cross-validation, called KFold. For example, to perform 5-fold cross-validation on the logistic regression model, you can use:

Code:

```
fromsklearn.model_selection import KFold
kf = KFold(n_splits=5)
scores = []
fortrain_index, test_index in kf.split(X):
X_train,X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
score = accuracy_score(y_test, y_pred)
scores.append(score)
mean_score = np.mean(scores)
```

Output :

```
In [39]: from sklearn.model_selection import KFold
         kf = KFold(n_splits=5)
         scores = []
         for train_index, test_index in kf.split(X):
             X_train, X_test = X[train_index], X[test_index]
             y_train, y_test = y[train_index], y[test_index]
             model.fit(X_train, y_train)
             y_pred = model.predict(X_test)
             score = accuracy_score(y_test, y_pred)
             scores.append(score)
         mean_score = np.mean(scores)
         mean_score

         C:\Users\DELL\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\linear_model\_logi
         g: lbfgs failed to converge (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(
         C:\Users\DELL\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\linear_model\_logi
         g: lbfgs failed to converge (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(

Out[39]: 0.9266666666666665
```

○○○

PRACTICAL : 4

AIM: Implement the Find-S concept learning algorithm that finds the most specific hypothesis that is consistent with the given training data.

Conditions:Hypothesis can only be conjunction (AND) of literals. Literals are either attributes or their negations.

Steps Involved In Find-S :

1. start with the most specific hypothesis.
   h = {Ø, Ø, Ø, Ø, Ø, Ø}
2. Take the next example and if it is negative, then no changes occur to the hypothesis.
3. If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to a general condition.
4. Keep repeating the above steps till all the training examples are complete.
5. After we have completed all the training examples we will have the final hypothesis when can use to classify the new examples.

Algorithm :

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute constraint a, in h
   If the constraint a, is satisfied by x
   Then do nothing
   Else replace a, in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Code:

```
import pandas as pd
importnumpy as np
data = pd.read_csv('C:/Users/DELL/Desktop/data.csv')
    concepts = np.array(data)[:,:-1]
   target = np.array(data)[:,-1]
   def train(con, tar):
   fori, val in enumerate(tar):
   ifval == 'yes':
  specific_h = con[i].copy()
  break

 fori, val in enumerate(con):
 if tar[i] == 'yes':
for x in range(len(specific_h)):
```

```
ifval[x] != specific_h[x]:
specific_h[x] = '?'
else:
pass
returnspecific_h
print(train(concepts, target))
```

Output

```
In [55]: import pandas as pd
         import numpy as np
         data = pd.read_csv('C:/Users/DELL/Desktop/data.csv')
         concepts = np.array(data)[:,:-1]
         target = np.array(data)[:,-1]
         def train(con, tar):
             for i, val in enumerate(tar):
                 if val == 'yes':
                     specific_h = con[i].copy()
                     break

             for i, val in enumerate(con):
                 if tar[i] == 'yes':
                     for x in range(len(specific_h)):
                         if val[x] != specific_h[x]:
                             specific_h[x] = '?'
                         else:
                             pass
             return specific_h
         print(train(concepts, target))

         ['sunny' 'warm' '?' 'strong' '?' '?']
```

The find_s function takes in a dataset data as input, where each row represents an instance and the last column represents the class label (0 or 1). The function initializes the most specific hypothesis to a string of '0's with the same length as the number of attributes in the dataset.

The function then loops through each instance in the dataset. If the instance is positive (i.e., the class label is 1), the function loops through each attribute in the instance. If the attribute is not already part of the hypothesis and the current value of the attribute in the hypothesis is '0', the function adds the attribute to the hypothesis.

Finally, the function returns the final hypothesis.

PRACTICAL : 5

AIM: Import Pima Indian diabetes data. Apply select K best and chi2 for feature selection. Identify the best features

Code:

```
import pandas as pd
importnumpy as np
fromsklearn.feature_selection import SelectKBest, chi2

# Load the dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-
diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataset = pd.read_csv(url, names=names)

# Split dataset into features (X) and target variable (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Apply feature selection using chi2 and SelectKBest
kbest = SelectKBest(score_func=chi2, k=4)
X_new = kbest.fit_transform(X, y)

# Print the names of the best features
mask = kbest.get_support()
best_features = X.columns[mask]
print(best_features)
```

Output:

```
In [20]: import pandas as pd
         import numpy as np
         from sklearn.feature_selection import SelectKBest, chi2

         # Load the dataset
         url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
         names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
         dataset = pd.read_csv(url, names=names)

         # Split dataset into features (X) and target variable (y)
         X = dataset.iloc[:, :-1]
         y = dataset.iloc[:, -1]

         # Apply feature selection using chi2 and SelectKBest
         kbest = SelectKBest(score_func=chi2, k=4)
         X_new = kbest.fit_transform(X, y)

         # Print the names of the best features
         mask = kbest.get_support()
         best_features = X.columns[mask]
         print(best_features)

         Index(['plas', 'test', 'mass', 'age'], dtype='object')
```

○ ○ ○

PRACTICAL : 6

AIM: Write a program to learn a decision tree and use it to predict class labels of test data

Training and test data will be explicitly provided by instructor. Tree pruning should not be performed.

Code:

```
fromsklearn.tree import DecisionTreeClassifier
fromsklearn.metrics import accuracy_score

# Load the training and test data
X_train = [[1, 2], [2, 3], [3, 1], [4, 2], [3, 3], [2, 1], [1, 3], [4, 1]]
y_train = [0, 0, 1, 1, 0, 1, 1, 0]
X_test = [[1, 1], [2, 2], [3, 4], [4, 3]]
y_test = [1, 0, 1, 0]

# Create a decision tree classifier and fit it to the training data
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
In [21]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import accuracy_score

         # Load the training and test data
         X_train = [[1, 2], [2, 3], [3, 1], [4, 2], [3, 3], [2, 1], [1, 3], [4, 1]]
         y_train = [0, 0, 1, 1, 0, 1, 1, 0]
         X_test = [[1, 1], [2, 2], [3, 4], [4, 3]]
         y_test = [1, 0, 1, 0]

         # Create a decision tree classifier and fit it to the training data
         clf = DecisionTreeClassifier(random_state=42)
         clf.fit(X_train, y_train)

         # Make predictions on the test data
         y_pred = clf.predict(X_test)

         # Calculate the accuracy of the classifier
         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy)


         Accuracy: 0.5
```

In this example, we are using a small synthetic dataset for demonstration purposes. You can replace X_train, y_train, X_test, and y_test with your own training and test data.

The DecisionTreeClassifier class is used to create a decision tree classifier. We set random_state to 42 for reproducibility. The fit() method is used to train the classifier on the training data. We then use the predict() method to make predictions on the test data, and calculate the accuracy of the classifier using the accuracy_score() function from scikit-learn's metrics module.

Note that we are not performing any tree pruning in this example. If you want to perform tree pruning, you can set the max_depth parameter or other pruning parameters in the DecisionTreeClassifier class.

❍ ❍ ❍

PRACTICAL : 7

AIM: ML Project. Use the following dataset as music.csv

a. **Store file as musie.csv and import it to python using pandas**
b. **Prepare the data by splitting data in input(age ,gender) and output(genre) data set**
c. **Use decision tree model from sklearn to predict the genre of various age group people.(Ex A male of age 21 likes hiphop whereas female of age 22 likes dance)**
d. **Calculate the accuracy of the model. vary training and test size to check different accuracy values model achieves.**

Code:

```
#Storing data into CSV
fromsklearn.model_selection import train_test_split
import pandas as pd
fromsklearn.tree import DecisionTreeClassifier
fromsklearn.metrics import accuracy_score
# list of name, degree, score
age = [20,23,25,26,29,30,31,33,37,20,21,25,26,27,30,31,34,34]
gender = [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
genre=["HipHop","HipHop","HipHop","Jazz","Jazz","Jazz","Classical","Classical","Classical","
Dance","Dance","Dance","Acoustic","Acoustic","Acoustic","Classical","Classical","Classical"]
# dictionary of lists
dict = {'Age': age, 'gender': gender, 'genre': genre}
df = pd.DataFrame(dict)
df.to_csv('C:/Users/DELL/Desktop/music.csv')

# Load the music data
data = pd.read_csv('C:/Users/DELL/Desktop/music.csv')

# Split the data into input and output sets
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Convert gender to binary values
X[:, 1] = pd.factorize(X[:, 1])[0]

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

```
# Train a decision tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
In [4]: #Storing data into CSV
        from sklearn.model_selection import train_test_split
        import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score
        # list of name, degree, score
        age = [20,23,25,26,29,30,31,33,37,20,21,25,26,27,30,31,34,34]
        gender = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
        genre=["HipHop","HipHop","HipHop","Jazz","Jazz","Jazz","Classical","Classical","Classical","Dance","Dance","Dance","Acoustic","Ac
        # dictionary of lists
        dict = {'Age': age, 'gender': gender, 'genre': genre}
        df = pd.DataFrame(dict)
        df.to_csv('C:/Users/DELL/Desktop/music.csv')

        # Load the music data
        data = pd.read_csv('C:/Users/DELL/Desktop/music.csv')

        # Split the data into input and output sets
        X = data.iloc[:, :-1].values
        y = data.iloc[:, -1].values

        # Convert gender to binary values
        X[:, 1] = pd.factorize(X[:, 1])[0]

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

        # Train a decision tree classifier
        clf = DecisionTreeClassifier(random_state=42)
        clf.fit(X_train, y_train)

        # Make predictions on the test data
        y_pred = clf.predict(X_test)

        # Calculate the accuracy of the classifier
        accuracy = accuracy_score(y_test, y_pred)
        print("Accuracy:", accuracy)

Accuracy: 0.6666666666666666
```

In this example, we first load the music data from the 'music.csv' file using pandas and split it into input (age and gender) and output (genre) sets. We convert the gender values to binary using the factorize() function from pandas. We then split the data into training and test sets using the train_test_split() function from scikit-learn.

Next, we train a decision tree classifier on the training data and make predictions on the test data using the predict() method. We calculate the accuracy of the classifier using the accuracy_score() function from scikit-learn's metrics module.

Finally, we can vary the test_size parameter in train_test_split() to check the different accuracy values that the model achieves. For example, we can set test_size=0.2 to use 20% of the data for testing, or test_size=0.5 to use 50% of the data for testing.

○ ○ ○

PRACTICAL : 8

AIM: Write a program to use a K-nearest neighbor it to predict class labels of test data. Training and test data must be provided explicitly.

Code :

```
fromsklearn.neighbors import KNeighborsClassifier
fromsklearn.metrics import accuracy_score

# Load the training and test data
X_train = [[1, 2], [2, 3], [3, 1], [4, 2], [3, 3], [2, 1], [1, 3], [4, 1]]
y_train = [0, 0, 1, 1, 0, 1, 1, 0]
X_test = [[1, 1], [2, 2], [3, 4], [4, 3]]
y_test = [1, 0, 1, 0]

# Create a K-nearest neighbor classifier and fit it to the training data
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
In [5]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score

        # Load the training and test data
        X_train = [[1, 2], [2, 3], [3, 1], [4, 2], [3, 3], [2, 1], [1, 3], [4, 1]]
        y_train = [0, 0, 1, 1, 0, 1, 1, 0]
        X_test = [[1, 1], [2, 2], [3, 4], [4, 3]]
        y_test = [1, 0, 1, 0]

        # Create a K-nearest neighbor classifier and fit it to the training data
        clf = KNeighborsClassifier(n_neighbors=3)
        clf.fit(X_train, y_train)

        # Make predictions on the test data
        y_pred = clf.predict(X_test)

        # Calculate the accuracy of the classifier
        accuracy = accuracy_score(y_test, y_pred)
        print("Accuracy:", accuracy)

        Accuracy: 0.75
```

In this example, we are using a small synthetic dataset for demonstration purposes. You can replace X_train, y_train, X_test, and y_test with your own training and test data.

The KNeighborsClassifier class is used to create a K-nearest neighbor classifier. We set the n_neighbors parameter to 3, which specifies the number of neighbors to use in the classification. The fit() method is used to train the classifier on the training data. We then use the predict() method to make predictions on the test data, and calculate the accuracy of the classifier using the accuracy_score() function from scikit-learn's metrics module.

Note that the K-nearest neighbor algorithm is a lazy learner, which means it does not actually learn a model from the training data, but instead memorizes the training data and uses it to make predictions on new data. This can make the algorithm slow when dealing with large datasets, but it can also be more flexible and adaptive than other algorithms. ○○○

PRACTICAL : 9

AIM: Import vgsales.csv from kaggle platform.

a. Find rows and columns in dataset

b. Find basic information regarding dataset using describe command.

c. Find values using values command.

- **Code:**

```
import pandas as pd
df = pd.read_csv("C:/Users/HP/Downloads/vgsales.csv")
df
```

Output:

```
import pandas as pd
df = pd.read_csv("C:/Users/HP/Downloads/vgsales.csv")
df
```

✓ 0.4s

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	259	Asteroids	2600	1980	Shooter	Atari	4.00	0.26	0.00	0.05	4.31
1	545	Missile Command	2600	1980	Shooter	Atari	2.56	0.17	0.00	0.03	2.76
2	1768	Kaboom!	2600	1980	Misc	Activision	1.07	0.07	0.00	0.01	1.15
3	1971	Defender	2600	1980	Misc	Atari	0.99	0.05	0.00	0.01	1.05
4	2671	Boxing	2600	1980	Fighting	Activision	0.72	0.04	0.00	0.01	0.77
...	...	...	...	...	...	...	...	...	...	...	...
16319	16565	Mighty No. 9	XOne	2016	Platform	Deep Silver	0.01	0.00	0.00	0.00	0.01
16320	16572	Resident Evil 4 HD	XOne	2016	Shooter	Capcom	0.01	0.00	0.00	0.00	0.01
16321	16673	Farming 2017 - The Simulation	PS4	2016	Simulation	UIG Entertainment	0.00	0.01	0.00	0.00	0.01
16322	16679	Rugby Challenge 3	XOne	2016	Sports	Alternative Software	0.00	0.01	0.00	0.00	0.01
16323	16692	Chou Ezaru wo Akuo Mono: Kai wo Touki ni Shiru...	PSV	2016	Action	dramatic create	0.00	0.00	0.01	0.00	0.01

16324 rows × 11 columns

a. Find rows and columns in dataset

Code:

```
print("Number of rows and columns in the dataset:", df.shape)
```

Output:

```
# Find the number of rows and columns in the dataset
print("Number of rows and columns in the dataset:", df.shape)
```

✓ 0.6s

```
Number of rows and columns in the dataset: (16324, 11)
```

b. Find basic information regarding dataset using describe command.

Code: print("Basic information about the dataset:\n", df.describe())

Output:

```
print("Basic information about the dataset:\n", df.describe())
✓ 0.9s
Basic information about the dataset:
              Rank          Year      NA_Sales      EU_Sales      JP_Sales  \
count  16324.000000  16324.000000  16324.000000  16324.000000  16324.000000
mean    8291.588270   2006.404251      0.265464      0.147581      0.078673
std     4792.043734      5.826744      0.821658      0.508809      0.311584
min        1.000000   1980.000000      0.000000      0.000000      0.000000
25%     4135.750000   2003.000000      0.000000      0.000000      0.000000
50%     8293.500000   2007.000000      0.080000      0.020000      0.000000
75%    12439.250000   2010.000000      0.240000      0.110000      0.040000
max    16600.000000   2016.000000     41.490000     29.020000     10.220000

       Other_Sales  Global_Sales
count  16324.000000  16324.000000
mean       0.048334      0.540328
std        0.189902      1.565860
min        0.000000      0.010000
25%        0.000000      0.060000
50%        0.010000      0.170000
75%        0.040000      0.480000
max       10.570000     82.740000
```

c. Find values using values command.

Code:

print("Values in the dataset:\n", df.values)

Output:

```
print("Values in the dataset:\n", df.values)
✓ 0.8s
Values in the dataset:
[[259 'Asteroids' '2600' ... 0.0 0.05 4.31]
[545 'Missile Command' '2600' ... 0.0 0.03 2.76]
[1768 'Kaboom!' '2600' ... 0.0 0.01 1.15]
...
[16573 'Farming 2017 - The Simulation' 'PS4' ... 0.0 0.0 0.01]
[16579 'Rugby Challenge 3' 'XOne' ... 0.0 0.0 0.01]
[16592 'Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shirube Kareru' 'PSV'
 ... 0.01 0.0 0.01]]
```

○ ○ ○

PRACTICAL : 10

AIM: Project on regression

a. Import home_data.csv on kaggle using pandas

b. Understand data by running head ,info and describe command

c. Plot the price of house with respect to area using matplotlib library

d. Apply linear regression model to predict the price of house

Solution:

a. Import home_data.csv on Kaggle using pandas:

To import the "home_data.csv" file, you first need to download it from Kaggle to your local machine. Then, you can use the following code to import it into a pandasdataframe:

Python

Code:

```
import pandas as pd
df = pd.read_csv("C:/Users/DELL/Desktop/home_data.csv")
df
```

Output:

```
In [1]: import pandas as pd
        df = pd.read_csv("C:/Users/DELL/Desktop/home_data.csv")
        df
```

Out[1]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bu
0	7129300520	20141013T000000	221900	3	1.00	1180	5650	1.0	0	0		7	1180	0	19
1	6414100192	20141209T000000	538000	3	2.25	2570	7242	2.0	0	0		7	2170	400	19
2	5631500400	20150225T000000	180000	2	1.00	770	10000	1.0	0	0		6	770	0	19
3	2487200875	20141209T000000	604000	4	3.00	1960	5000	1.0	0	0		7	1050	910	19
4	1954400510	20150218T000000	510000	3	2.00	1680	8080	1.0	0	0		8	1680	0	19
...															
21608	263000018	20140521T000000	360000	3	2.50	1530	1131	3.0	0	0		8	1530	0	20
21609	6600060120	20150223T000000	400000	4	2.50	2310	5813	2.0	0	0		8	2310	0	20
21610	1523300141	20140623T000000	402101	2	0.75	1020	1350	2.0	0	0		7	1020	0	20
21611	291310100	20150116T000000	400000	3	2.50	1600	2388	2.0	0	0		8	1600	0	20
21612	1523300157	20141015T000000	325000	2	0.75	1020	1076	2.0	0	0		7	1020	0	20

21613 rows × 21 columns

b. Understand data by running head, info, and describe command:

You can use the following commands to get a better understanding of the data:

Code:

```
# View the first 5 rows of the dataframe
df.head()
# Get information about the dataframe
df.info()
```

```
# Get descriptive statistics of the dataframe
df.describe()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  int64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(16), object(1)
memory usage: 3.5+ MB
```

Out[7]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	g
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.00
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.65
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.17
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.00
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.00
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.00
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.00
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.00

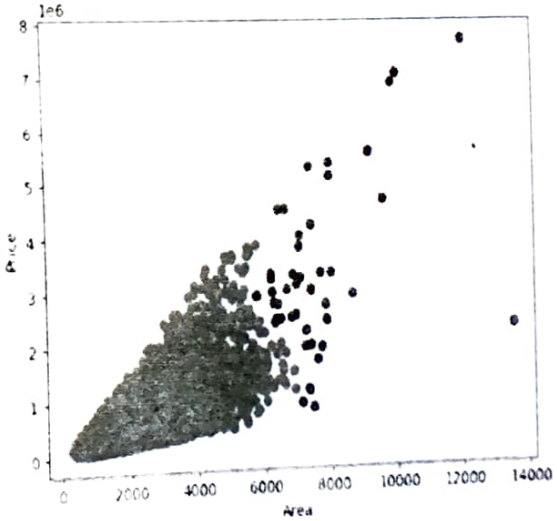
The head() method displays the first 5 rows of the dataframe, while info() provides information about the dataframe's columns and data types. The describe() method gives you descriptive statistics of the numeric columns in the dataframe.

c. Plot the price of house with respect to area using matplotlib library

Code:

```
importmatplotlib.pyplot as plt
plt.scatter(df['sqft_living'], df['price'])
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```

Output:



This code creates a scatter plot of the sqft_living column (representing the area of the house) on the x-axis and the price column on the y-axis. The xlabel() and ylabel() functions are used to label the x and y-axes, respectively.

d. Apply linear regression model to predict the price of house

Code:

```
fromsklearn.linear_model import LinearRegression
# Create a linear regression object
model = LinearRegression()
# Train the model using the sqft_living and price columns
model.fit(df[['sqft_living']], df['price'])
# Predict the price of a house with an area of 2500 sqft
predicted_price = model.predict([[2500]])
print(predicted_price)
```

Output:

```
In [13]: import pandas as pd
         df = pd.read_csv("C:/Users/DELL/Desktop/home_data.csv")
         # View the first 5 rows of the dataframe
         from sklearn.linear_model import LinearRegression
         # Create a linear regression object
         model = LinearRegression()
         # Train the model using the sqft_living and price columns
         model.fit(df[['sqft_living']], df['price'])
          # Predict the price of a house with an area of 2500 sqft
         predicted_price = model.predict([[2500]])
         print(predicted_price)

         [657978.17625703]
```

This code imports the LinearRegression class from the sklearn.linear_modelmodule and creates a new LinearRegression object. It then trains the model using the sqft_living and price columns of the dataframe. Finally, it uses the trained model to predict the price of a house with an area of 2500 sqft and prints the predicted price to the console.

❍ ❍ ❍

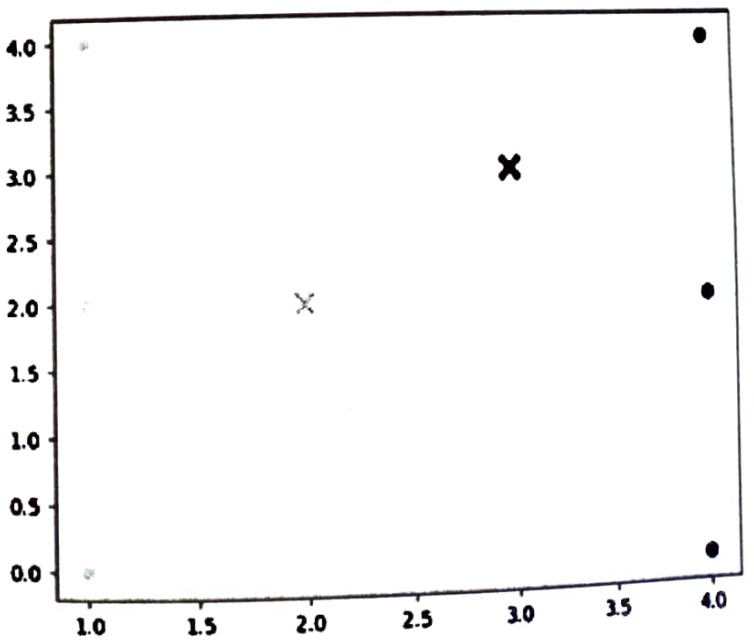
PRACTICAL : 11

AIM: Write a program to cluster a set of points using K-means. Training and test data must be provided explicitly.

Code:

```
importnumpy as np
importmatplotlib.pyplot as plt
fromsklearn.cluster import KMeans

# Generate some sample data
X_train = np.array([[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]])
X_test = np.array([[2, 2], [3, 3]])

# Create a KMeans object with 2 clusters
kmeans = KMeans(n_clusters=2)

# Fit the KMeans model to the training data
kmeans.fit(X_train)

# Predict the cluster labels for the test data
y_pred = kmeans.predict(X_test)

# Visualize the clusters and the test data
plt.scatter(X_train[:, 0], X_train[:, 1], c=kmeans.labels_)
plt.scatter(X_test[:, 0], X_test[:, 1], marker='x', s=100, linewidths=3, c=y_pred)
plt.show()
```

Output:



In this program, we first generate some sample training and test data. The X_train array contains 6 points with 2 features each, and the X_test array contains 2 points with the same 2 features.

We then create a KMeans object with 2 clusters and fit it to the training data using the fit() method. This trains the KMeans model on the training data and determines the centroids of the clusters.

Next, we predict the cluster labels for the test data using the predict() method. This assigns each test point to one of the two clusters.

Finally, we visualize the clusters and the test data using a scatter plot. The training data points are colored according to their assigned cluster label, while the test data points are marked with an 'x' and colored according to their predicted cluster label. The resulting plot shows the two clusters and how the test data points are assigned to them.

○○○

PRACTICAL : 12

AIM: Import Iris dataset

a. **Find rows and columns using shape command**
b. **Print first 30 instances using head command**
c. **Find out the data instances in each class.(use groupby and size)**
d. **Plot the univariategraphs(box plot and histograms)**
e. **Plot the multivariate plot(scatter matrix)**
f. **Split data to train model by 80% data values**
g. **Apply K-NN and k means clustering to check accuracy and decide which is better.**

Dataset link: https://www.kaggle.com/datasets/jillanisofttech/iris-dataset-uci

Code:

```
import pandas as pd
df = pd.read_csv("C:/Users/HP/Downloads/iris.csv")
df
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

a. Find rows and columns using shape command

Code:

```
print("Number of rows and columns in the dataset:", df.shape)
```

Output:

```
# Find the number of rows and columns in the dataset
print("Number of rows and columns in the dataset:", df.shape)
```

✓ 0.7s

```
Number of rows and columns in the dataset: (150, 6)
```

Print first 30 instances using head command

Code:

```
print("First 30 instances in the dataset:\n", df.head(30))
```

Output:

First 30 instances in the dataset:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.4	Iris-setosa
5	6	5.4	3.9	1.7	0.3	Iris-setosa
6	7	4.6	3.4	1.4	0.2	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.1	Iris-setosa
9	10	4.9	3.1	1.5	0.2	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.1	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.2	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa
21	22	5.1	3.7	1.5	0.4	Iris-setosa
22	23	4.6	3.6	1.0	0.2	Iris-setosa
23	24	5.1	3.3	1.7	0.5	Iris-setosa
24	25	4.8	3.4	1.9	0.2	Iris-setosa
25	26	5.0	3.0	1.6	0.2	Iris-setosa
26	27	5.0	3.4	1.6	0.4	Iris-setosa
27	28	5.2	3.5	1.5	0.2	Iris-setosa
28	29	5.2	3.4	1.4	0.2	Iris-setosa
29	30	4.7	3.2	1.6	0.2	Iris-setosa

Find out the data instances in each class.(use groupby and size)

Code:

```
class_counts = df.groupby('Species').size()
print("Data instances in each class:\n", class_counts)
```

Output:

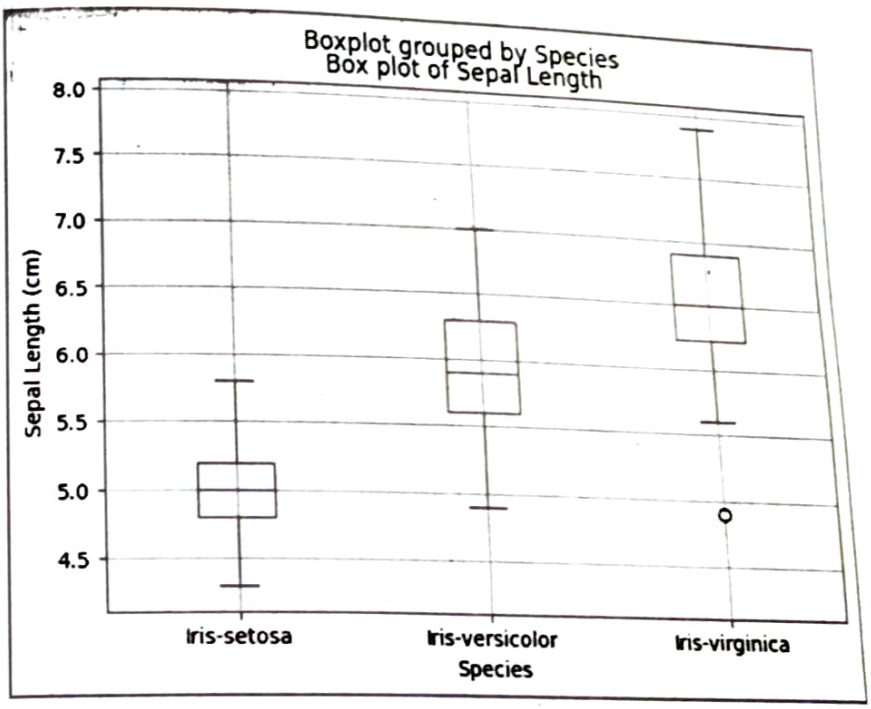
```
Data instances in each class:
 Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

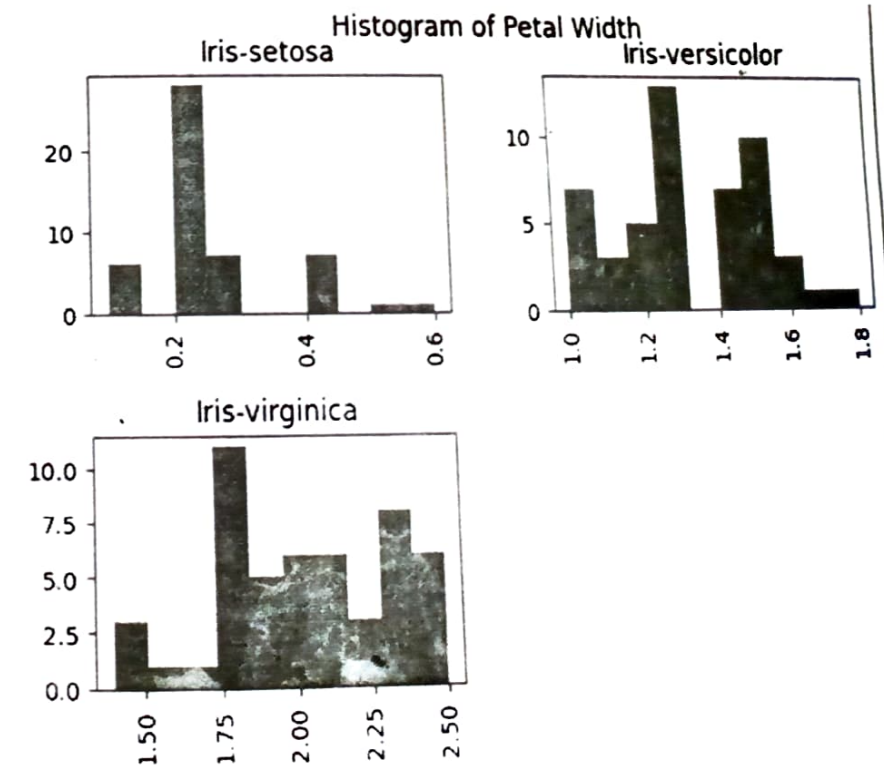
d. Plot the univariategraphs(box plot and histograms)

Code:

```
importmatplotlib.pyplot as plt

# Box plot of sepal length
df.boxplot(column='SepalLengthCm', by='Species')
plt.title("Box plot of Sepal Length")
plt.xlabel("Species")
plt.ylabel("Sepal Length (cm)")
plt.show()

# Histogram of petal width
df.hist(column='PetalWidthCm', by='Species')
plt.suptitle("Histogram of Petal Width")
plt.xlabel("Petal Width (cm)")
plt.ylabel("Count")
plt.show()
```

Output:



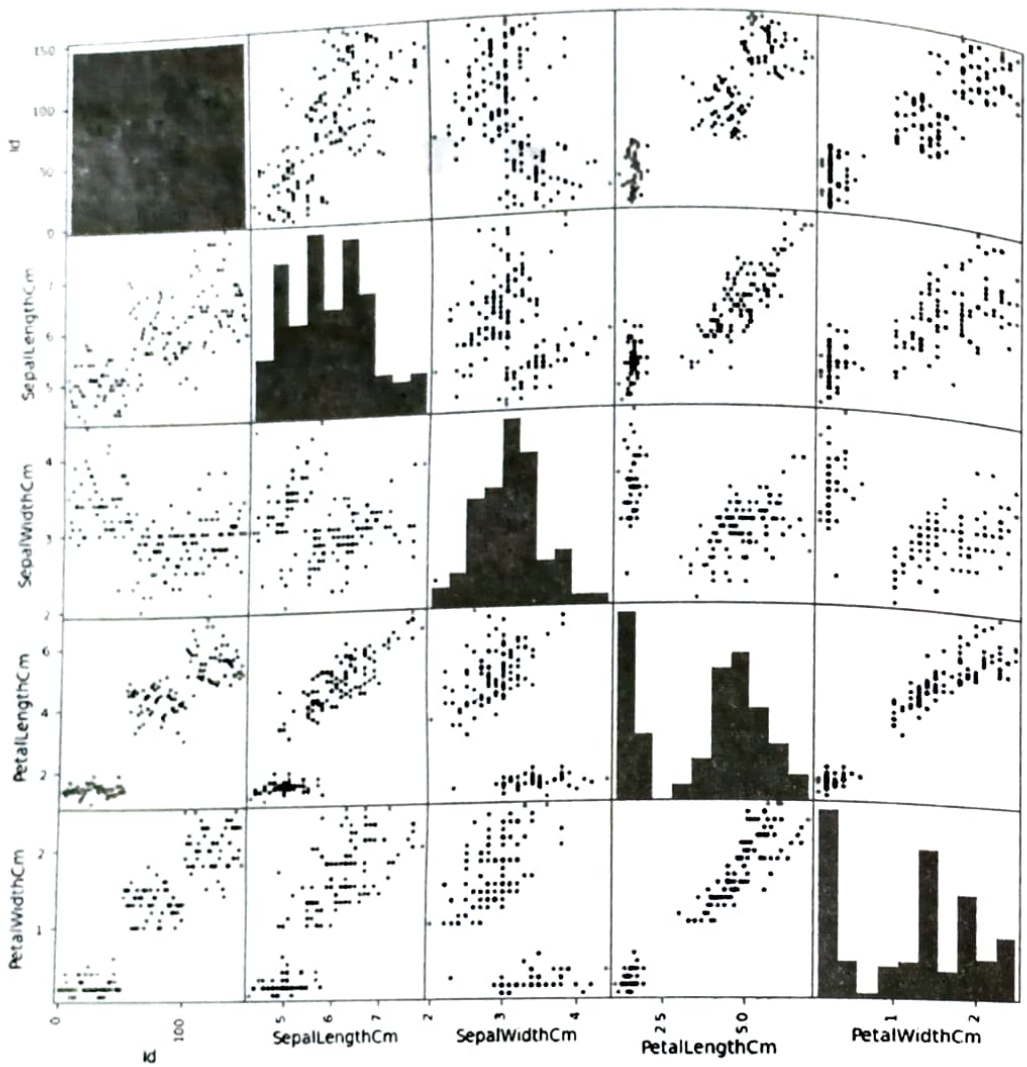


e. Plot the multivariate plot(scatter matrix)

Code:

```
frompandas.plotting import scatter_matrix
scatter_matrix(df, alpha=0.8, figsize=(10,10), diagonal='hist')
plt.show()
```

Output:



f. Split data to train model by 80% data values

Code:

```
fromsklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:, -1], test_size=0.2,
random_state=42)

# Print the shapes of the resulting datasets
print("X_train shape:", X_train.shape)
```

```
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

Output:

```
X_train shape: (120, 5)
X_test shape: (30, 5)
y_train shape: (120,)
y_test shape: (30,)
```

h. Apply K-NN and k means clustering to check accuracy and decide which is better.

Code:

```
fromsklearn.neighbors import KNeighborsClassifier
fromsklearn.cluster import KMeans
fromsklearn.metrics import accuracy_score

# K-NN classification
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
knn_acc = accuracy_score(y_test, knn_pred)

# K-means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_train)
kmeans_pred = kmeans.predict(X_test)
kmeans_acc = accuracy_score(y_test, kmeans_pred)

# Compare accuracy of K-NN and K-means clustering
print("K-NN accuracy:", knn_acc)
print("K-means accuracy:", kmeans_acc)
```

Output:

```
K-NN accuracy: 1.0
K-means accuracy: 0.0
```

○ ○ ○

Abbreviation

AI	:	Artificial Intelligence
ANN	:	Artificial Neural Networks
F1	:	F1 Score
GPU	:	Graphics Processing Unit
kNN	:	k-Nearest Neighbour
LDA	:	Linear Discriminant Analysis
LOOCV	:	Leave-One-Out Cross Validation
ML	:	Machine Learning
NLP	:	Natural Language Processing
NN	:	Neural Network
PCA	:	Principal Component Analysis
RF	:	Random Forest
RL	:	Reinforcement Learning
SAS	:	Statistical Analysis System
SVD	:	Single Value Decomposition
SVM	:	Support Vector Machine

○○○

MODEL QUESTION PAPER-1 FOR MID SEMESTER EXAM

(UNIT 1, 2, 3, 6)

Marks : 30

Q.1 a. Give the difference between supervised learning and unsupervised learning. 3

b. Explain well-posed learning problem with suitable example. 7

Q.2 a. Explain types of data with example. 3

b. Write a short note on data quality and remediation. 7

OR

Q.2 a. Explain structure of box-plot. 3

b. Write a short note on feature subset selection method. 7

Q.3 a. Describe k-fold cross validation method with example. 7

b. Compare and contrast NumPy with Pandas. 3

OR

Q.3 a. Consider the following confusion matrix of the win/loss prediction of cricket match. Calculate the accuracy, error rate, sensitivity, specificity, precision, recall and F-measure of the model. 7

	Actual Win	Actual Loss
Predicted Win	80	6
Predicted Loss	5	9

b. State features of Matplotlib. 3

■

MODEL QUESTION PAPER-2 FOR MID SEMESTER EXAM

(UNIT 1, 2, 3, 6)

Marks : 30

Q.1 a. Give the difference between qualitative data and quantitative data. 3

b. Write a short note on dimensionality reduction. 7

Q.2 a. Describe application of Machine Learning in the field of healthcare. 3

b. Explain different tools and technology used in Machine Learning. 7

OR

Q.2 a. Identify main three features for following well-posed problem 3

Spam Detection Sytem

b. Explain types of machine learning in detail. 7

Q.3 a. Explain Holdout method in detail. 7

b. How to load dataset using Numpy? Explain. 7

OR

Q.3 a. Explain overfitting and underfitting with suitable example. 3

b. How to plot a vertical line and a horizontal line using Matplotlib? 7

■

MODEL QUESTION PAPER-3 FOR MID SEMESTER EXAM

(UNIT 1, 2, 3, 6)

Marks : 30 3

Q.1 a. Give the applications of Scikit-Learn. 7

b. Describe basic concept of Machine Learning and its application. 3

Q.2 a. State various strategies to handle missing values. 7

b. Describe measures of central tendency with appropriate examples. 3

OR

Q.2 a. What is IQR? How it is meseaured? 7

b. Describe Machine Learning Activities in detail. 3

Q.3 a. Describe model parameter tuning in detail. 7

b. Give the difference between predictive model and descriptive model. 3

OR

Q.3 a. Consider the following confusion matrix of the win/loss prediction of cricket match. Calculate the accuracy, error rate, sensitivity, specificity, precision, recall and F-measure of the model. 7

	Actual Win	Actual Loss
Predicted Win	70	10
Predicted Loss	15	5

b. Give the difference between Bagging and Boosting. 3

■

MODEL QUESTION PAPER 1

DE – SEMESTER IV (NEW SYLLABUS) EXAMINATION

Subject: Fundamentals of Machine Learning (4341603)

Time: 2 hours 30 min **Total Marks: 70**

Instructions:

1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary.**
3. **Figures to the right indicate full marks.**

			Marks	CO (Course Outcome)	Cognitive Level (As per Revised Bloom's Taxonomy)
Q.1	(a)	State the features of NumPy.	03	CO 5	R
	(b)	Define Human Learning. Give the types of human learning.	04	CO 1	R
	(c)	Describe different types of Machine Learning activities.	07	CO 2	U
		OR			
	(c)	Describe types of data in Machine Learning with suitable examples.	07	CO 2	U
Q.2	(a)	Give the difference between predictive model and descriptive model.	03	CO 2	U
	(b)	Explain the concept of penalty and reward in reinforcement learning.	04	CO 1	U
	(c)	While predicting malignancy of tumour of a set of patients using a classification model, following are the data recorded: 1. Correct predictions – 15 malignant, 75 benign 2. Incorrect predictions – 4 malignant, 6 benign Create confusion matrix. Calculate the sensitivity, specificity, precision, Recall and F-measure of the model.	07	CO 3	A
		OR			
Q.2	(a)	Explain model underfitting in brief.	03	CO 2	U
	(b)	Explain application of ML in healthcare sector.	04	CO 1	U
	(c)	Can the performance of a learning model be improved? If yes, explain how.	07	CO 2	A
Q. 3	(a)	Out of 200 emails, a classification model correctly predicted 150 spam emails and 30 ham emails. What is the error rate of the model?	03	CO 2	A
	(b)	Write a short note on PCA.	04	CO 2	U
	(c)	Discuss the kNN model in detail.	07	CO 3	U
		OR			

	(b)	Find standard deviation for the following data: 5, 10, 15, 20, 25, 29	04	CO 2	A
	(c)	Draw and describe classification model in detail.	07	CO 3	U
Q. 4	(a)	Give the weakness of logistic regression.	03	CO 3	R
	(b)	Explain test data and training data in brief.	04	CO 3	U
	(c)	Generate large itemsets and association rules using Aprion algorithm on the following data set with minimum support value and minimum confidence value set as 50% and 75% respectively.	07	CO 4	A

TID	Item Purchased
T1	Cheese, Milk, Cookies
T2	Butter, Milk, Bread
T3	Cheese, Milk, Butter, Brread
T4	Butter, Bread

OR

Q. 4	(a)	Explain the applications of simple linear regression.	03	CO 3	R
	(b)	State the strengths and weakness of kNN algorithm.	04	CO 3	U
	(c)	During a research work, you found 7 observations as described with the data points below. You want to create 3 clusters from these observations using K-means algorithm. After first iteration, the clusters C1, C2, C3 has following observations: C1: {(2,2), (4,4), (6,6)} C2: {(0,4), (4,0)} C3: {(5,5), (9,9)} If you want to run a second iteration then what will be the cluster centroids?	07	CO 4	A
Q.5	(a)	How can we create 2D arrays in NumPy?	03	CO 5	A
	(b)	Explain features of Scikit-Learn.	04	CO 5	U
	(c)	Describe the concept of clustering using appropriate real-world example.	07	CO 4	U
		OR			
Q.5	(a)	How can we plot a horizontal line in matplotlib?	03	CO 5	A
	(b)	Explain the steps to import csv file in Pandas.	04	CO 5	U
	(c)	Compare and contrast supervised learning and unsupervised learning.	07	CO 4	U

Q. 3	(a)	Out of 200 emails, a classification model correctly predicted 150 spam emails and 30 ham emails. What is the accuracy of the model?	03	CO 2	A
	(b)	Write a short note on Histogram.	04	CO 2	
	(c)	Explain classification steps in detail.	07	CO 3	U
Q. 4	(a)	Give any three examples of supervised learning.	03	CO 3	U
	(b)	Explain the use of unsupervised learning in expression identification.	04	CO 4	R
	(c)	Explain simple linear regression using a graph explaining slope and intercept.	07	CO 3	U
		OR			U
Q. 4	(a)	Give any three applications of multiple linear regression.	03	CO 3	
	(b)	Give the difference between supervised learning and unsupervised learning.	04	CO 4	R
	(c)	Explain logistic regression in detail.	07	CO 3	U
Q.5	(a)	Explain data types used in Scikit-Learn.	03	CO 5	U
	(b)	Explain use of Pandas in Machine Learning.	04	CO 5	U
	(c)	Explain how the Market Basket Analysis uses the concepts of association analysis.	07	CO 4	U
		OR			U
Q.5	(a)	Give the application of Matplotlib.	03	CO 5	
	(b)	Give the difference between Pandas and NumPy.	04	CO 5	U
	(c)	Explain the k-means clustering method with a step-by-step algorithm.	07	CO 4	U

MODEL QUESTION PAPER 2

DE – SEMESTER IV (NEW SYLLABUS) EXAMINATION

Subject: Fundamentals of Machine Learning (4341603)

Time: 2 hours 30 min **Total Marks: 70**

Instructions:

1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary.**
3. **Figures to the right indicate full marks.**

			Marks	CO (Course Outcome)	Cognitive Level (As per Revised Bloom's Taxonomy)
Q.1	(a)	State the appropriate data type of following examples. a) Height of students b) Blood Group of students c) Grade of students	03	CO 2	U
	(b)	Explain well-posed problem with suitable example.	04	CO 1	U
	(c)	Define box plot. Explain detailed interpretation of a box plot	07	CO 2	R
		OR			
	(c)	Explain, in details, the different strategies of addressing missing data values.	07	CO 2	R
Q.2	(a)	Give the difference between bagging and boosting.	03	CO 2	U
	(b)	Explain the use of Machine Learning approach in Robotics.	04	CO 1	U
	(c)	Explain, in details, the process of evaluating the performance of a classification model.	07	CO 3	U
		OR			
Q.2	(a)	Explain the main purpose of a descriptive model.	03	CO 2	U
	(b)	Explain the use of Python in the field of Machine Learning.	04	CO 1	U
	(c)	Explain, in details, Holdout method.	07	CO 2	U
Q. 3	(a)	Define outliers. Give one example.	03	CO 2	R
	(b)	Find mean and median for the following data: 4, 5, 7, 8, 8, 9, 11, 12, 14	04	CO 2	A
	(c)	Write a short note on Support Vector Machine.	07	CO 3	U
		OR			
Q. 3	(a)	Define predictive models. Give one example.	03	CO 2	R

	(b)	Find standard deviation for the following data: 5, 10, 15, 20, 25, 29	04	CO 2	A
	(c)	Draw and describe classification model in detail.	07	CO 3	U
Q. 4	(a)	Give the weakness of logistic regression.	03	CO 3	R
	(b)	Explain test data and training data in brief.	04	CO 3	U
	(c)	Generate large itemsets and association rules using Apriori algorithm on the following data set with minimum support value and minimum confidence value set as 50% and 75% respectively.	07	CO 4	A

TID	Item Purchased
T1	Cheese, Milk, Cookies
T2	Butter, Milk, Bread
T3	Cheese, Milk, Butter, Brread
T4	Butter, Bread

OR

Q. 4	(a)	Explain the applications of simple linear regression.	03	CO 3	R
	(b)	State the strengths and weakness of kNN algorithm.	04	CO 3	U
	(c)	During a research work, you found 7 observations as described with the data points below. You want to create 3 clusters from these observations using K-means algorithm. After first iteration, the clusters C1, C2, C3 has following observations: C1: {(2,2), (4,4), (6,6)} C2: {(0,4), (4,0)} C3: {(5,5), (9,9)} If you want to run a second iteration then what will be the cluster centroids?	07	CO 4	A
Q.5	(a)	How can we create 2D arrays in NumPy?	03	CO 5	A
	(b)	Explain features of Scikit-Learn.	04	CO 5	U
	(c)	Describe the concept of clustering using appropriate real-world example.	07	CO 4	U
		OR			
Q.5	(a)	How can we plot a horizontal line in matplotlib?	03	CO 5	A
	(b)	Explain the steps to import csv file in Pandas.	04	CO 5	U
	(c)	Compare and contrast supervised learning and unsupervised learning.	07	CO 4	U

❖ ❖ ❖

MODEL QUESTION PAPER 3

DE – SEMESTER IV (NEW SYLLABUS) EXAMINATION

Subject: Fundamentals of Machine Learning (4341603)

Time: 2 hours 30 min **Total Marks: 70**

Instructions:

1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary.**
3. **Figures to the right indicate full marks.**

			Marks	CO (Course Outcome)	Cognitive Level (As per Revised Bloom's Taxonomy)
Q.1	(a)	Give the difference between qualitative data and quantitative data.	03	CO 2	R
	(b)	Describe use of R in Machine Learning.	04	CO 1	R
	(c)	Give the difference between Supervised Learning, Unsupervised Learning and Reinforcement Learning.	07	CO 1	U
		OR			
	(c)	Define Machine Learning. Explain any four applications of Machine Learning in brief.	07	CO 1	U
Q.2	(a)	Write basic steps for feature subset selection.	03	CO 2	R
	(b)	Explain data pre-processing in brief.	04	CO 2	U
	(c)	Describe K-fold cross validation method in detail.	07	CO 2	U
		OR			
Q.2	(a)	Explain the factors which lead to the data quality issues.	03	CO 2	R
	(b)	Write a brief note on dimensionality reduction.	04	CO 2	U
	(c)	Describe ensemble learning approach in detail.	07	CO 2	U
Q. 3	(a)	Give the difference between classification and regression.	03	CO 3	U
	(b)	Define following terms: Support Vectors, Hyperplane, Margin	04	CO 3	R
	(c)	Define Classification. Explain classification learning steps in detail using flowchart.	07	CO 3	U
		OR			
Q. 3	(a)	Give any three examples of supervised learning in Industry 4.0.	03	CO 3	R
	(b)	Compare and contrast between Single linear regression and multiple linear regression.	04	CO 3	U
	(c)	Write and discuss kNN algorithm.	07	CO 3	U
Q. 4	(a)	Define: Support, Confidence	03	CO 4	R

	(b)	Draw the diagram which shows the clustering process.	04	CO 4	U
	(c)	Describe applications of unsupervised learning in detail.	07	CO 4	A
		OR			
Q. 4	(a)	State pros and cons of k-means clustering algorithm.	03	CO 4	R
	(b)	Give the difference between classification and clustering.	04	CO 4	U
	(c)	Write and explain Apriori algorithm with example.	07	CO 4	A
Q.5	(a)	Explain any three applications of Pandas.	03	CO 5	U
	(b)	How to plot a vertical line and a horizontal line using Matplotlib? Give the code.	04	CO 5	A
	(c)	Write a Python program to load the iris data from a given csv file into a dataframe and print the shape of the data, type of the data and first 3 rows using Scikit-Learn.	07	CO 5	A
		OR			
Q.5	(a)	Explain any three applications of Scikit-Learn.	03	CO 5	U
	(b)	How can you split data into training and test data using Pandas? Give the code.	04	CO 5	A
	(c)	Write a program to cluster a set of points using K-means. Training and test data must be provided explicitly.	07	CO 5	A