SCHOOL OF COMPUTER SCIENCE, ENGINEERING AND APPLICATIONS

BHARATHIDASAN UNIVERSITY

TIRUCHIRAPPALLI 620 023

MASTERS OF DATA SCIENCE
PREDICTIVE ANALYTICS LAB
SEMESTER-III

REGISTER NUMBER								
2	3	M	S	C	D	S		

DEC – 2024

SCHOOL OF COMPUTER SCIENCE, ENGINEERING AND APPLICATIONS

BHARATHIDASAN UNIVERSITY

KHAJAMALAI CAMPUS

TIRUCHIRAPPALLI 620 023

CERTIFICATE

This is to certify that the bonafide record of work done in the School of Computer Science, Engineering and Applications, Khajamalai Campus, Bharathidasan University, Tiruchirappalli 620023, during the 3rd semester of the year 2024-2025.

SUBMITTED BY

NAME	
REG.NO	23MSCDS
CLASS	II MSC DS
SUBJECT	Predictive Analytics Lab
SUB.CODE	MDS23037P
SEMESTER	III

For the Practical Examination held on _____

STAFF IN-CHARGE SIGNATURE: HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER SIGNATURE: INTERNAL EXAMINER SIGNATURE:

EX. NO.01	**Write a python program to compute** **1) Central tendency : mean, median, mode** **2) Measures of dispersion : variance and standard deviation**
DATE:	

.

AIM:

To write a python program to compute central tendency and measure of dispersion

PROCEDURE:

1. Import necessary libraries such ass math library
2. Create separate function for central tendency and measures of dispersion
3. Define data within the program
4. Use a scrpt to call these functions and display the results
5. Print the calculated measures

PROGRAM:

1 MEAN:

```
def cal_mean(data):
        total_sum = sum(data)
        count = len(data)
        mean = total_sum / count
        return mean

data = [10,20,30,40,50]
mean_value = cal_mean(data)
print(f"The mean of the dataset is :{mean_value}")
```

OUTPUT:

The mean of the dataset is:30.3

2. MEDIAN

```
def cal_median(data):
  sorted_data=sorted(data)
  count=len(sorted_data)
  if count%2==1:
     median=sorted_data[count//2]
     return median
  else:
```

```
    mid1=sorted_data[count//2-1]
    mid2=sorted_data[count//2]
    median=((mid1+mid2)/2)
    return median
data=[10,20,30,40,50]
print(f"the median of the data is: {cal_median(data)}")
```

OUTPUT:

The median of the dataset is:30

3. MODE

```
def find_mode(data):
        return max(set(data), key = data.count)

data = [1,2,4,6,7,8,8,8,5,3,8]
mode = find_mode(data)
print(f"The mode of the dataset is :{mode}")
```

OUTPUT:

The mode of the dataset is: 8

4. VARIANCE

```
def cal_variance(data):
        mean = sum(data)/len(data)
```

```
    squared_diffs = [(x-mean)**2 for x in data]
    variance = sum(squared_diffs) / len(data)
    return variance
data = [1,2,3,4,5]
result = cal_variance(data)
print(f"The variance of the dataset is : {result}")
```

OUTPUT:

The variance of the datast is : 2.0

5.STANDARD DEVIATION

```
import math
def cal_standard_deviation (data):
   mean=sum (data)/len (data)
   variance=sum((x-mean)**2 for x in data)/len (data)
   standard_deviation=math.sqrt(variance)
   return standard_deviation
data=[10,12,23,23,16,23,21,16]
std_dev=cal_standard_deviation(data)
print (f"the standard deviation of the dataset is: {std_dev}")
```

OUTPUT:

Standard Deviation :4.898979485566356

RESULT:

The python program to compute central tendency and measure of dispersion was successfully executed

EX. NO. 02 DATE:	**Study of python basic libraries such as numpy, statistics, math and scipy**

AIM:

To write a python program for study of python basic libraries such as ststistics matech numpy & scipy**.**

PROCEDURE:

1. Import libraries
2. Use np.array to create a numpy array
3. Np.mean() to calculate the average of elements.
4. Statistics.median() calculate the middle value.
5. Stats.mode() calculate most feequency
6. Print the mean, median and mode of the data

PROGRAM

```
import numpy as np
from scipy import stats
import statistics
data = np.array([1,2,3,2,4,4,4,5])
mean = np.mean(data)
median = statistics.median(data)
mode = stats.mode(data)
print("mean:",mean)
print("median.", median)
print("mode:",mode)
```

OUTPUT:

Mean: 3.125

Median: 3.5

Mode: modeResult(mode=4,count=3)

RESULT:

The python program for numpy,scipy,statistics libraries are successfully executed.

EX. NO. 03	Study of python libraries for machine learning application such as pandas and matplotlib
DATE:	

AIM;

To write a python program for machine learning applications such as pandas and matplotlib

PROCEDURE:

1. Import pandas and matplotlib libraries
2. Create a dictionary named as data
3. Use pd,DataFrame(data) to convert data into two dimensiona table format
4. Perfom filter operation based on age
5. Perform sort operation using Sort_values()
6. Create a bar chart for the score of students

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
data ={
    'Name': ['John', 'Mary', 'Jane', 'Bob'],
    'Age': [25, 31, 22, 35],
    'Score': [90, 85, 88, 92]
      }
df = pd.DataFrame(data)
print("Original DataFrame:", df)
df_filtered = df[df['Age'] > 30]
print("\n Filtered DataFrame (Age > 30):", df_filtered)
df_sorted = df.sort_values(by='Score', ascending = False)
print("\n Sorted DataFrame(Score in descending order):", df_sorted)
plt.figure(figsize = (10,5))
plt.bar(df['Name'],df['Score'])
plt.xlabel('Student Name')
plt.ylabel('Score')
plt.title('Student Scores')
plt.xticks(rotation = 45)
plt.grid(axis = 'y')
plt.show()
```

OUTPUT:

original dataframe:Name Age Score

0 john 25 90

1 mary 31 85

2 jane 22 88

3 bob 35 92

Filtered DataFrame (age>30): Name Age Score

1 mary 31 85

3 bob 35 92

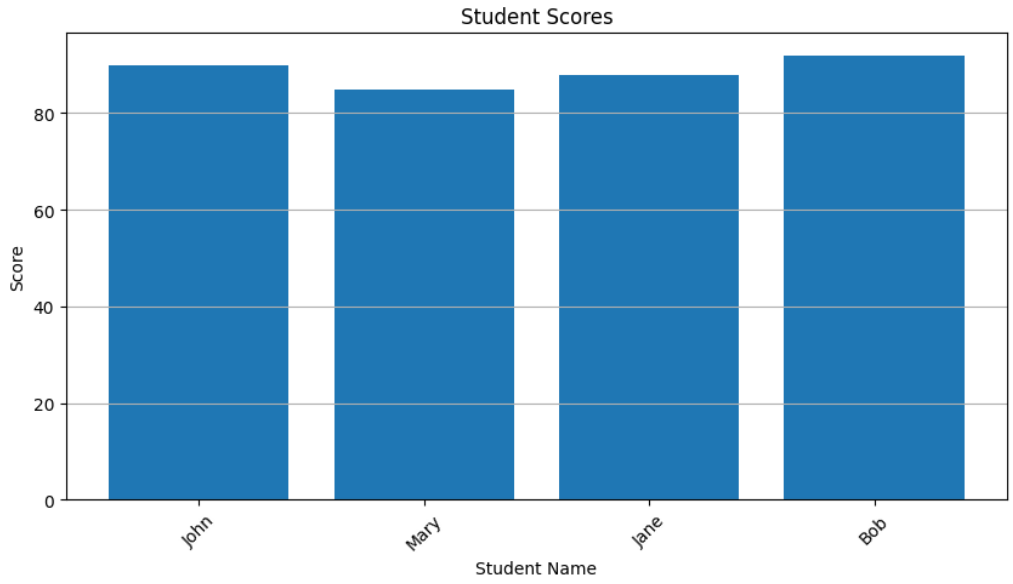
Sorted DataFrame(score in Descending order): Name Age Score

3 bob 35 92

0 john 25 90

2 jane 22 88

1 mary 31 85



RESULT:

The python program for pandas and matplot libraries are successfully executed.

EX. NO. 04	Write a python program to implement a simple linear regression
DATE:	

AIM:

To write a python program to implement a simple linear regression

PROCEDURE:

1. Import numpy, matplotlib and linear regression from sklearn.linear_model
2. Create two array xand y using np.array
3. Define model and fit the model
4. Predict the model using model.predict(0
5. Create scatter plot for year Vs experience

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
x = np.array([1,2,3,4,5,6,7,8,9,10]).reshape(-1,1)
y =
np.array([30000,35000,40000,45000,50000,55000,60000,65000,70000,75000])
model = LinearRegression()
model.fit(x,y)
y_pred = model.predict(x)
plt.scatter(x,y,color = 'g', label = 'Actual data')
plt.plot(x,y_pred,color = 'r', label = 'Regression')
plt.title('Experience vs Salary')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```

OUTPUT:

RESULT:

The python program tp implement Linear Regression was successfully executed.

EX. NO. 05 DATE:	**Implementation of multiple linear regression for house price prediction using sklearn**

AIM;

To write a python program for implementation of multiple linear regression for house price prediction using sklearn

PROCEDURE:

1. Import pandas, numpy, train_test_split from sklearm.model_selection, Linearregression from klearn.linear_model and accuracy_scor from sklearnmetrics
2. Create dictionary named as data
3. Use pd.DataFrame() to convert dictionary into two dimensional format
4. Convert dataframe into csv file using df.to_csv(0
5. Load the file using pd.read_csv()
6. Define two variable x and y
7. Split train and test data using train-test-split(0
8. Define LinearRegression and fit the model
9. Predict the model using model.predict(0
10. Measure accuracy score

PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
data={
    'size':[2100,1600,1500,2300,1400,9000,1100,1900,2200,1800],
    'bedroom':[3,2,4,3,2,1,4,3,5,2],
    'bathroom':[2,3,1,3,2,3,1,3,2,1],
'price':[200000,210000,320000,430000,100000,180000,210000,150000,900000,700000]
    }
df=pd.DataFrame(data)
df.to_csv("house_price.csv",index=False)
data=pd.read_csv("house_price.csv")
x=data[['size','bedroom','bathroom']]
y=data['price']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print(y_pred)
mse=mean_squared_error(y_test,y_pred)
r2_scor=(y_test,y_pred)
print("mean square error:",mse)
```

OUTPUT:

126921.29892019 269176.61785941]

mean square error: 300576275082.2588

Result:

The python program to implement Multiple linear regression was successfully executed.

EX. NO. 06	Implementation of a decision tree using sklearn and its parameter tuning
DATE:	

AIM;

To write a python program for implementation of a decision tree using sklearn and its parameters tuning

PROCEDURE:

1. Import pandas, train-test_split, Gridsearch from sklearn.model_selection
2. Load_ iris dataset from sklearn.datasets
3. Import decisiontreeClassifier from sklearn,tree and import accuracy-score
4. Define two variable x and y
5. Split the dataset into train and test data
6. Define parameter grid
7. Perform grid search and train decision tree

PROGRAM:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
clf = DecisionTreeClassifier(
    criterion='gini',
    max_depth=3,
    min_samples_split=2,
    min_samples_leaf=1,
    random_state=42
)
clf.fit(X_train, y_train)
train_accuracy = clf.score(X_train, y_train)
test_accuracy = clf.score(X_test, y_test)
print(f"Training Accuracy: {train_accuracy:.2f}")
print(f"Testing Accuracy: {test_accuracy:.2f}")
tree_rules = export_text(clf, feature_names=iris.feature_names)
print("\nDecision Tree Rules:\n")
print(tree_rules)
```

```
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.title("Decision Tree Visualization")
plt.show()
```

OUTPUT:

Training Accuracy: 0.96

Testing Accuracy: 1.00

Decision Tree Rules:

|--- petal length (cm) <= 2.45

| |--- class: 0

|--- petal length (cm) > 2.45

| |--- petal length (cm) <= 4.75

| | |--- petal width (cm) <= 1.65

| | | |--- class: 1

| | |--- petal width (cm) > 1.65

| | | |--- class: 2

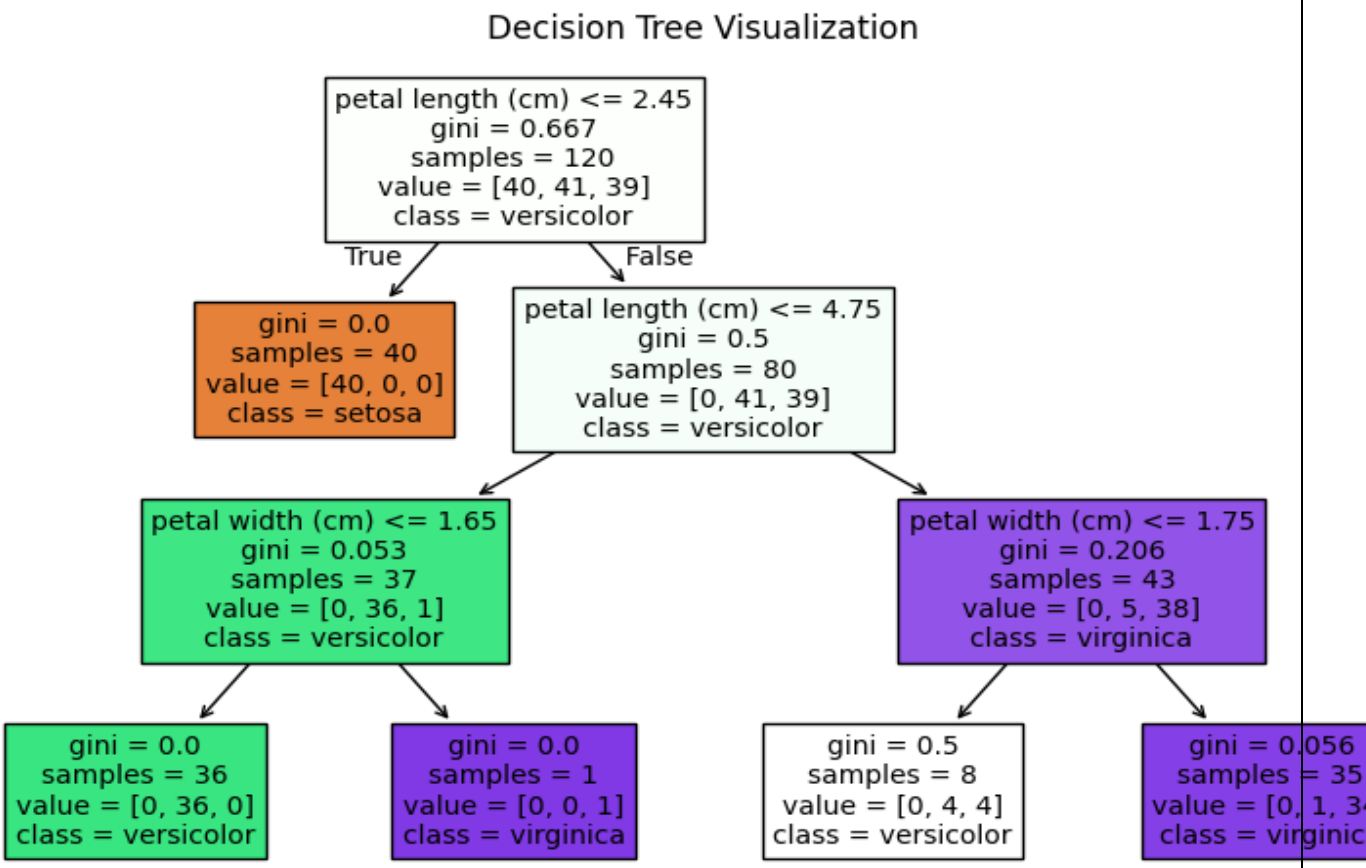
| |--- petal length (cm) > 4.75

| | |--- petal width (cm) <= 1.75

| | | |--- class: 1

| | |--- petal width (cm) > 1.75

| | | |--- class: 2



RESULT:

The python program to implement decision tree was successfully executed

EX. NO. 07	Implementation of KNN using sklearn
DATE:	

:

AIM:

To write a python program for implementation of KNN using sklearn

PROCEDURE:

1. Import dataset, train_test_split from model_selection, KNeighborsClassifier from sklearn.neighbors and accuracy_score
2. Load iris dataset
3. Split dataset into train and test data
4. Define k value is 3
5. Define KNeighborsClassifier with k
6. Fit the model and predict the model
7. Measeure the accuracy score

PROGRAM:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
data = load_iris()
x = data.data
y = data.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state=42)
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
print(f"Accuracy of KNN with k ={k}: {accuracy:.2f}")
```

OUTPUT:

Accuracy of KNN with k =3: 1.00

RESULT:

The python program to implement kneighbour classifier was successfully executed.

EX. NO. 08	
DATE:	**Implementation of Logistic Regression using sklearn**

AIM:

To write a python program toimplementation of logstic regression using sklearn

PROCEDURE:

1. Import pandas, train-test-split, LogisticRegression and accuracy-score
2. Create dictionary named as data and convert to dataframe
3. Convert it into csv file and read the csv file using pd.read_csv()
4. Split the dataset into rain and test data
5. Define LogisticRegression model and fit the model
6. Define threshold value is 0.5
7. Measure probability using model.predict_proba()
8. Predict the model and measure accuracy score

PROGRAM:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
data = pd.DataFrame({
    'age': [25, 45, 30, 50, 60, 35, 40, 55, 70, 28],
    'resting_heart_rate': [72, 85, 78, 90, 88, 70, 75, 80, 95, 68],
    'max_heart_rate': [180, 150, 170, 160, 155, 175, 165, 158, 140, 190],
    'cholesterol': [200, 250, 220, 240, 260, 210, 230, 245, 270, 190],
    'target': [0, 1, 0, 1, 1, 0, 0, 1, 1, 0]})
data.to_csv('heartrate.csv', index=False)
data = pd.read_csv('heartrate.csv')
x = data.drop('target', axis = 1)
y = data['target']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=42)
model = LogisticRegression()
model.fit(x_train,y_train)
threshold = 0.5
print('If the value lies between 0 to 0.5, the output is 0')
print('If the value lies between 0.5 to 1, the output is 1')
y_prob = model.predict_proba(x_test)[:,1]
y_pred = (y_prob >= threshold).astype(int)
accuracy = accuracy_score(y_test,y_pred)
print(f"Accuracy of Logistic Regression with threshold: {accuracy:.2f}")
```

OUTPUT:

If the value lies between 0 to 0.5, the output is 0

If the value lies between 0.5 to 1, the output is 1

Accuracy of Logistic Regression with threshold: 1.00

RESULT:

The python program to implement logistic regression was successfully executed.

EX. NO. 09 DATE:	Implementation of K-Means Clustering

AIM:

To write a python program to implanting k-means clustering

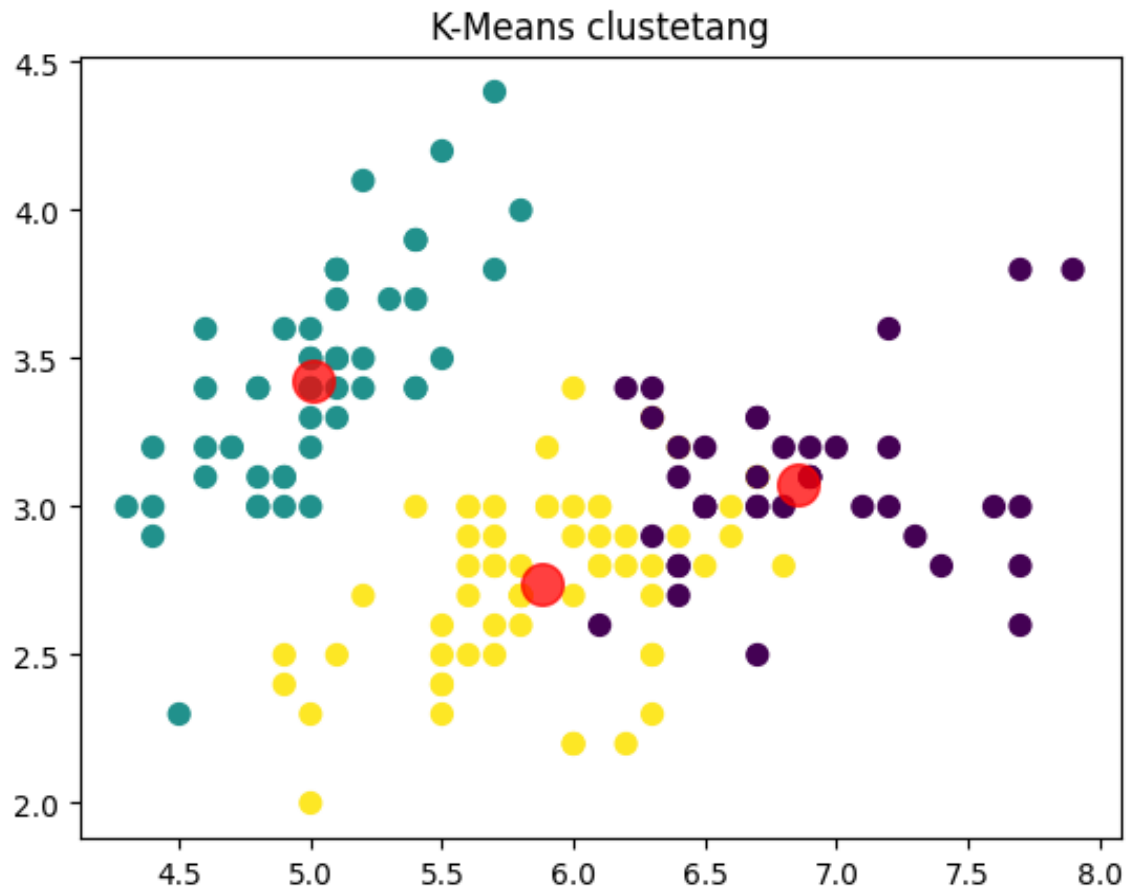
PROCEDURE:

1. Import numpy, kmeans, iris dataset from sklearn and matplotlib
2. Load iris dataset
3. Define kmeans model with cluster value
4. Fit the model and predict the model
5. Define cluster center using kmeans.cluster_centers_
6. Create scatterplot for iris dataset

PROGRAM:

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
data = load_iris()
X = data.data
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit (X)
y_kmeans = kmeans.predict(X)
plt.scatter (X[:, 0], X[:, 1], c=y_kmeans, cmap='viridis', s=50)
centers = kmeans.cluster_centers_
plt.scatter (centers[:,0],centers[:,1],c='red',s=200, alpha=0.75)
plt.title("K-Means clustetang")
plt.show()
```

OUTPUT:



RESULT:

The python program to implement K means clustering was successfully executed.