PROG 2(a) COMPUTING DISTANCE USING EUCLIDEAN MEASURE

Distance:

By using modules:

```
import math as m
def dis(x1,x2,y1,y2):
        dist=m.sqrt((x2-x1)**2+(y2-y1)**2)
        return dist
dist=dis(2,4,6,8)
print(dist)
```

without module and function:

```
x1=int(input())
x2= int(input())
y1= int(input())
y2= int(input())
d=((x2-x1)**2+(y2-y1)**2)**0.5
print(d)
```

PROG 2(b)

AIM:
To read a CSV file and perform the following operations:
-display the description of the file
-display the file column wise
-display the first five entries of the file
-display the last five entries of the file
-display the size of particular column

code:

```
imort pandas as pd
df = pd.read_csv("./iris.csv")
print(f"discribing iris.csv\n{df.describe()}")
print(f"displaying coloumn 'sepal.length'\n{df['sepal.length']}")
print(f"first 5 entries are \n{df.head(5)}")
print(f"last 5 entries are \n{df.tail(5)}")
print(f"length of a particular column 'sepal.length' is len(df['sepal.length'])}")
```

2c : SCATTER PLOT

AIM:
Upload iris data set and create a scatter port using sepal length and petal width to separate the spices class

SOURCE CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("./iris.csv")
species=df["variety"].unique()
for i in species:
x = df.loc[df["variety"]==i]["sepal.length"]
y = df.loc[df["variety"]==i]["petal.width"]
plt.scatter(x,y)
plt.legend(species)
plt.xlabel("sepal.length cm")
plt.ylabel("petal.width cm")
plt.show()
```

PROG 3(a) CONFUSION MATRIX:

AIM: Evaluation matrix of ML algorithm using confusion matrix ; precision, accuracy, recall, specificity, sensitivity

PROGRAM:

```
import pandas as pd
import sklearn.linear_model as sk
import sklearn.model_selection as md
from sklearn import metrics
df = pd.read_csv("./iris.csv")
df = df.drop(df[df["variety"] == "Setosa"].index)
uni = df["variety"].unique()
df["variety"] = df["variety"].replace(uni, [0, 1])
```

```
x = df.iloc[:, :4]
y = df["variety"]
X_train, X_test, y_train, y_test = md.train_test_split(
x, y, test_size=0.4,random_state=5)
logreg = sk.LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(x)

tm, fm = metrics.confusion_matrix(y, y_pred)
tn = tm[0]
fn = tm[1]
fp = fm[0]
tp = fm[1}

accuracy = (tp + tn)/(tp + tn + fp + fn)
sensitivity = tp/(tn + fn)
sensitivity = "{:2f}".format(sensitivity)
precision = tp/(tp + fp)
precision = "{:2f}".format(precision)
specificity = tn/(tn + fp)
print({"Accuracy": accuracy,
"Precision": precision,
"Sensitivity": sensitivity,
"Specificity": specificity})
```

3B:

AIM:

To write a python function even digit(lower, upper)which will find all numbers between the lower and upper limit such that each digit of number is an even number.

PROGRAM:

```
L = []
def evendigit1(lower, upper):
for i in range(lower, upper+1):
L.append(i)
for j in str(i):
if int(j) % 2 != 0:
L.remove(i)
break
return L
a, b = map(int, input("Enter Lower and Upper Limit: ").split())
y = evendigit1(a, b)
print(y)
```

PROG 4: KNN CLASSIFICATION

AIM: To upload Iris.csv with three features sepal length, Sepal width and species, read the Test data from user to apply KNN Classification Algorithm and Predict the Test case (Assume K=5).without using predefined function.

CODE:

```
import pandas as pd

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

data = pd.read_csv('./iris.csv')

train_data,test_data =train_test_split(data,test_size=0.2,random_state=5)

x_train,y_train = train_data.loc[:,["sepal.length","petal.width"]],train_data['variety']

x_test,y_test = test_data.loc[:,["sepal.length","petal.width"]],test_data['variety']

model = KNeighborsClassifier(n_neighbors=5)

model.fit(x_train,y_train)

predicted_data = model.predict(x_test)

def replacing_catagory(l):

for i,e in enumerate(l) :

if e == 'Versicolor':

l[i]=1

if e == 'Setosa':

l[i]=0
```

```
if e == 'Virginica':
l[i]=2
return l
y_test = replacing_catagory(np.array(y_test))
predicted_data = replacing_catagory(predicted_data)
mse = mean_squared_error(y_test, predicted_data)
print('The mean squared error is ',mse)
```

PROG 5 NAIVE BAYES CLASSIFICATION

AIM: To upload Iris.csv dataset with sepal length, Sepal width, petal length, petal width. Employee Naive Bayes Classification methodology and predict the type of species for a given set of attributes.
Note : Do not use Predefined Function

SOURCE CODE:

```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
data = pd.read_csv("./iris.csv")
data = data.drop(data[data["variety"] == "Setosa"].index)
uni = data["variety"].unique()
data["variety"] = data["variety"].replace(uni, [0, 1])
def calculate_prior(df, Y):
classes = sorted(list(df[Y].unique()))
```

```
prior = []
z = len(df)
for i in classes:
prior.append(len(df[df[Y] == i])/z)
return prior
def calculate_likelihood_gaussian(df, feat_name, feat_val, Y, label):
feat = list(df.columns)
df = df[df[Y]==label]
mean, std = df[feat_name].mean(), df[feat_name].std()
p_x_given_y = (1 / (np.sqrt(2 * np.pi) * std)) * np.exp(-((feat_val-mean)*2 / (2 *
std*2 )))
return p_x_given_y
def naive_bayes_gaussian(df, X, Y):
# get feature names
features = list(df.columns)[:-1]
# calculate prior
prior = calculate_prior(df, Y)
Y_pred = []
# loop over every data sample
for x in X:
# calculate likelihood
labels = sorted(list(df[Y].unique()))
likelihood = [1]*len(labels)
for j in range(len(labels)):
for i in range(len(features)):
```

```
likelihood[j] *= calculate_likelihood_gaussian(df, features[i], x[i], Y, labels[j])
# calculate posterior probability (numerator only)
post_prob = [1]*len(labels)
for j in range(len(labels)):
post_prob[j] = likelihood[j] * prior[j]
Y_pred.append(np.argmax(post_prob))
return np.array(Y_pred)
train, test = train_test_split(data, test_size=.2, random_state=41)
X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_gaussian(train, X=X_test, Y="variety")
print("The accuracy of the model is",f1_score(Y_test, Y_pred))
```

PROG 6 :Decision tree classification methodologies Using iris dataset

Aim: Employing Decision tree Classification methodologies to the iris dataset and plotting the result.

Source code:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import tree
```

```
import matplotlib.pyplot as plt

df = pd.read_csv('C:/iris.csv')

x = df.drop('variety', axis=1)

y = df['variety']

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)

dt = DecisionTreeClassifier()

dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:",accuracy)

fig, ax = plt.subplots(figsize=(10,10))

tree.plot_tree(dt,feature_names=x.columns,class_names = np.unique(y),filled =

True, ax = ax)

plt.show()
```

PROG 7 ANALYSIS AND INTERPRETATION OF DATA

AIM: To manipulate the Twitter Data Set by removing the Punctuation, Numbers, Special Characters and word length<=3, tokenize the Words and Stem and generate a word cloud for the Twitter dataset and retrieve the top 15 positive and negative tags.

SOURCE CODE

```
import numpy as np

import pandas as pd

from nltk.corpus import stopwords
```

```
from nltk.tokenize import word_tokenize
import nltk
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
data=pd.read_csv('Twitter_Data.csv')
data=data.iloc[:50,:]
nltk.download('punkt')
nltk.download('stopwords')
print(data.head())
stop_words = set(stopwords.words('english'))
data['tokenized_sents'] = data.apply(lambda row:
nltk.word_tokenize(str(row['clean_text'])), axis=1)
for i in range(50):
words = data.loc[i,'tokenized_sents']
wordsFiltered = []
for w in words:
if w not in stop_words:
wordsFiltered.append(w)
data.at[i,'tokenized_sents']=wordsFiltered
print(data.head())
comment_words=''
for i in range(50):
tokens=data.iloc[i,2]
comment_words += " ".join(tokens)+" "
```

```
wordcloud = WordCloud(collocations = False, background_color =

'white').generate(comment_words)

# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight_layout(pad = 0)

plt.show()
```

PROG 8 LINEAR REGRESSION

AIM: To write a python program to compute linear regression curve between salary and experience from a csv dataset.

SOURCE CODE:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


dataset = pd.read_csv("./Salary_Data.csv")

print(dataset.head())

print(dataset.info())

X = dataset.iloc[:,:-1].values #independent variable array

y = dataset.iloc[:,:-1].values #dependent variable vector

# splitting the dataset

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=0)
#print('Training Data\n',X_train)
#print('Testing Data\n',X_test)
# fitting the regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train) #actually produces the linear eqn for the data
# Plotting the graph for the Training dataset

plt.scatter(X_train,y_train,color='red') # plotting the observation line
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line
plt.title("Salary vs Experience (Training set)") # stating the title of the graph

plt.xlabel("Years of experience") # adding the name of x-axis
plt.ylabel("Salaries") # adding the name of y-axis
plt.show() # specifies end of graph
# Plotting the graph for the Testing dataset

plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line
plt.title("Salary vs Experience (Testing set)")

plt.xlabel("Years of experience")
plt.ylabel("Salaries")
plt.show()
```

```
plt.show()
```