**DEPARTMENT OF COMPUTER APPLICATIONS**

**SNGIST GROUP OF INSTITUTIONS**

**North Paravur- 683520**

****

LABORATORY RECORD

*Certified that this is a Bonafide Record of work done by Mr./Ms.……………………………………………………………………………….....Reg.No:……………………………in the …………………………………………… Laboratory during the semester …………….. year ……..……… to …….………. at SNGIST,Thekkethazham,North Paravur,Ernakulam.*

*Faculty in Charge*

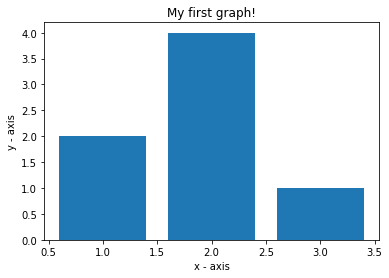
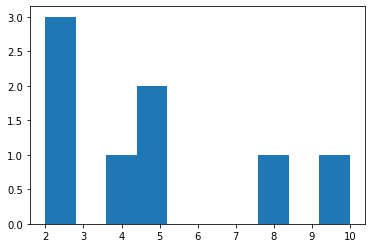
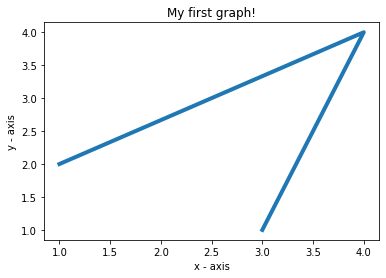
Submitted for the practical examination conducted on ……………………………………...........

Internal Examiner External Examiner

|  |  |  |  |
| --- | --- | --- | --- |
| SL. NO. | PROGRAM | DATE | PAGE NO |
| 1 | Data Visualisation | 07-12-2021 | 5 |
| 2 | Predict class label of a given data point using KNN | 07-01-2022 | 13 |
| 3 | Calculate accuracy of KNN using iris dataset | 07-01-2022 | 15 |
| 4 | Predict the class label of an unseen observation using Naive-bayes | 13-01-2022 | 17 |
| 5 | Predict the accuracy of standard dataset using naive bayes | 13-01-2022 | 19 |
| 6 | Generate classification report and confusion matrix of diabetes dataset using naive bayes algorithm | 14-01-2022 | 21 |
| 7 | Generate classification report and confusion matrix of diabetes dataset using KNN algorithm | 14-01-2022 | 23 |
| 8 | Implement Decision tree algorithm, find accuracy in pima Indian diabetes dataset | 20-01-2022 | 25 |
| 9 | Implement simple linear regression using salary dataset | 27-01-2022 | 27 |
| 10 | Implement Multiple Linear Regression using Advertising dataset. | 28-01-2022 | 31 |
| 11 | Implement SVM Classification using diabetes dataset. | 03-02-2022 | 33 |
| 12 | Implement SVM Regression using Advertising dataset. | 04-02-2022 | 35 |
| 13 | Implement KMeans using mall customer dataset. | 08-02-2022 | 37 |
| 14 | Implement Web Scrapping using python | 23-02-2022 | 41 |
| 15 | Implement problem on Natural Language Processing – part of speech, tagging Ngram using NLTK | 24-02-2022 | 43 |
| 16 | Program on CNN to classify images from any standard dataset in the public domain using keras framework | 24-02-2022 | 45 |

**INDEX**

**OUTPUT:**



**PROGRAM NO : 1 DATE : 07 – 12 – 2021**

**AIM : DATA VISUALIZATION**

**SOURCE CODE :**

import matplotlib.pyplot as plt

x = [1,4,3]

y = [2,4,1]

plt.plot(x, y, linewidth=4)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('My first graph!')

plt.show()

import matplotlib.pyplot as plt

x = [1,2,3]

y = [2,4,1]

plt.bar(x, y)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('My first graph!')

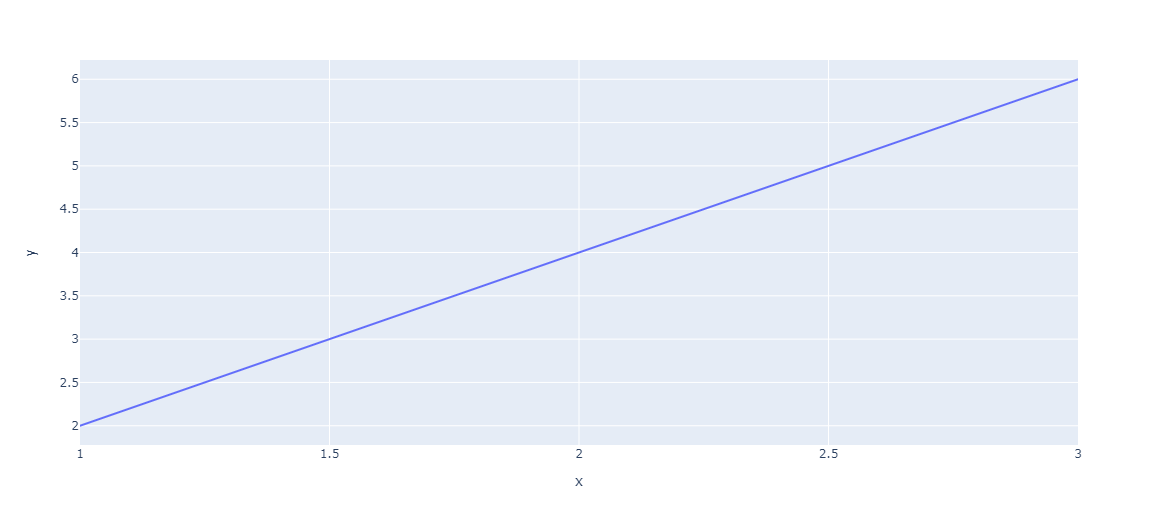
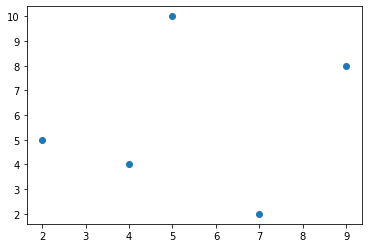
plt.show()

from matplotlib import pyplot as plt

y = [10, 5, 8, 4, 2,2,2,5]

plt.hist(y)

plt.show()

****

from matplotlib import pyplot as plt

x = [5, 2, 9, 4, 7]

y = [10, 5, 8, 4, 2]

plt.scatter(x, y)

plt.show()

**Data visualization using Ploty**

import plotly.express as px

fig = px.line(x=[1, 2, 3], y=[2, 4, 6])

fig.show()

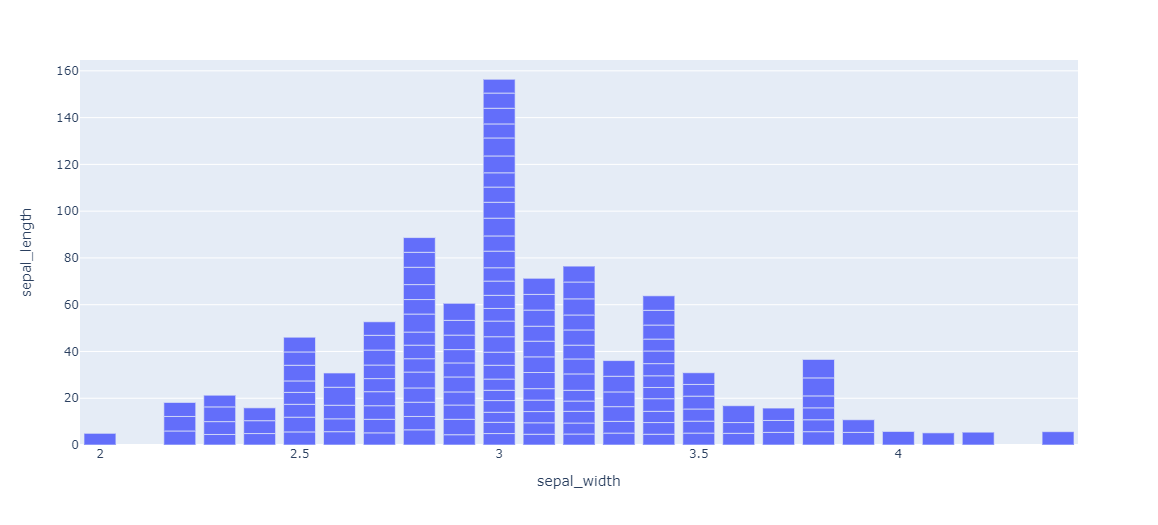
import plotly.express as px

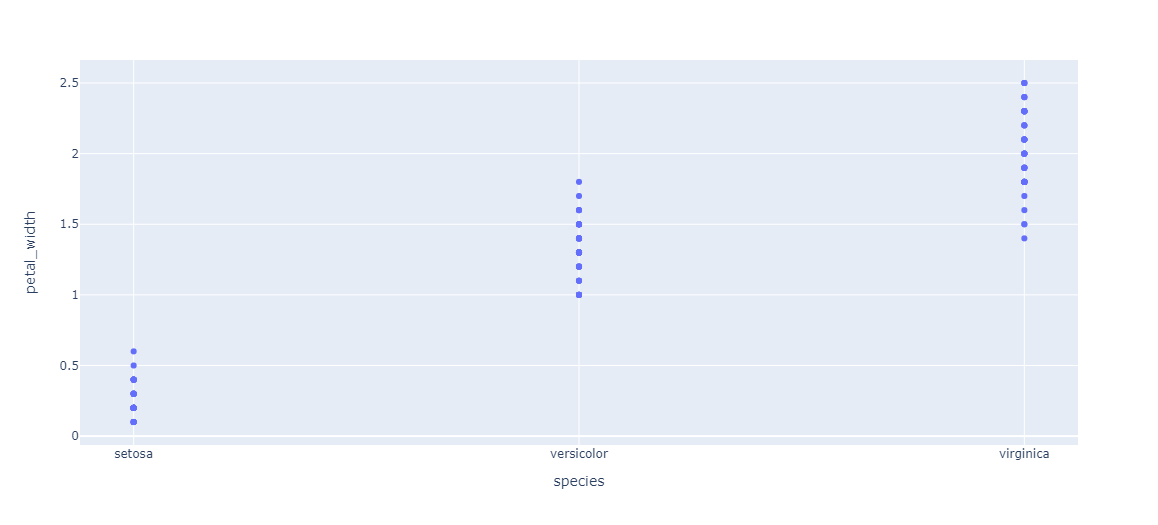
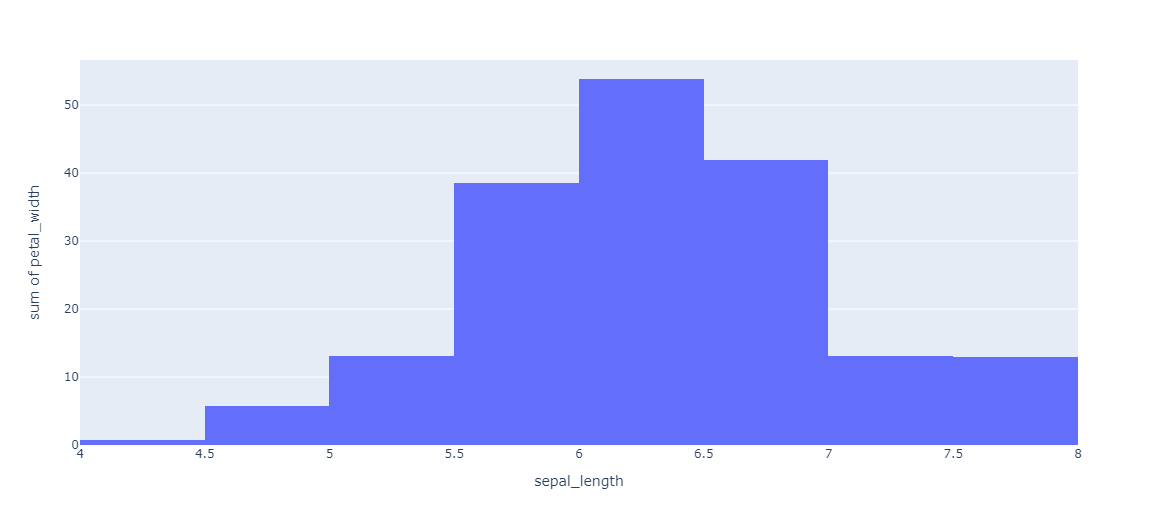
df = px.data.iris()

print(df)

fig = px.line(df, x="species", y="petal\_width")

fig.show()





import plotly.express as px

df = px.data.iris()

fig = px.bar(df, x="sepal\_width", y="sepal\_length")

fig.show()

import plotly.express as px

df = px.data.iris()

fig = px.histogram(df, x="sepal\_length", y="petal\_width")

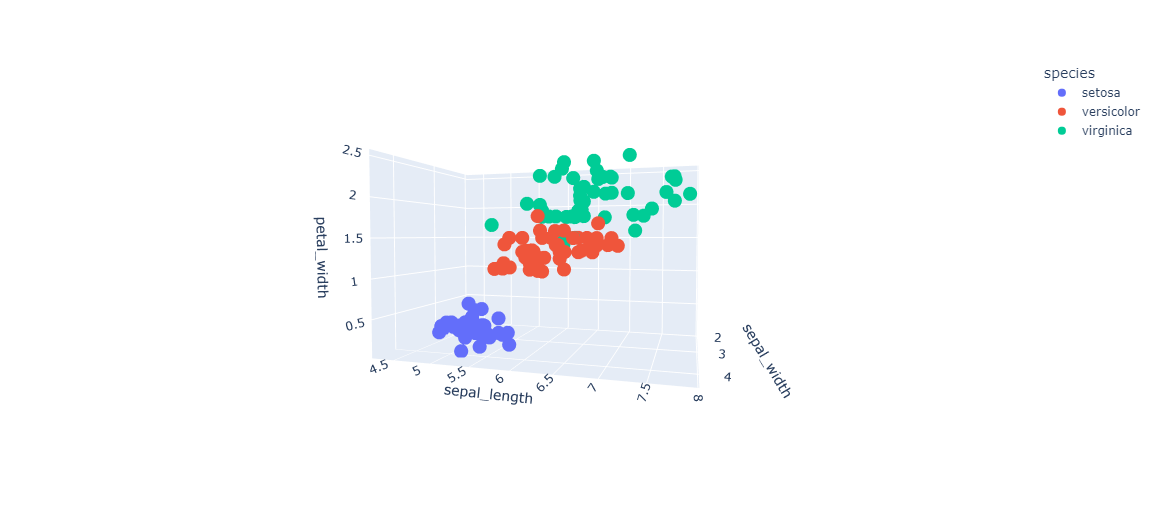
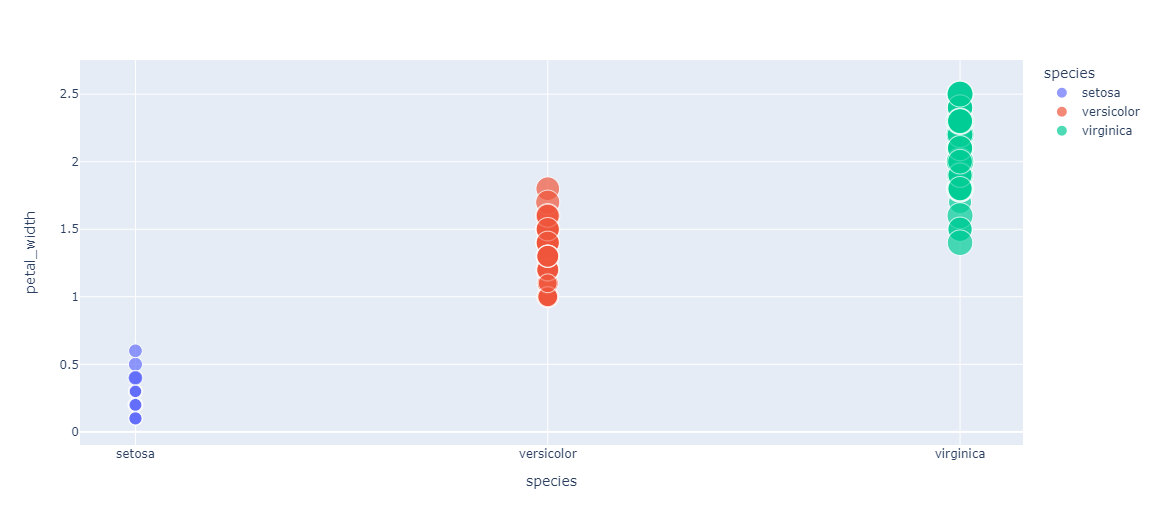
fig.show()

import plotly.express as px

df = px.data.iris()

fig = px.scatter(df, x="species", y="petal\_width")

fig.show()



import plotly.express as px

df = px.data.iris()

fig = px.scatter(df, x="species", y="petal\_width",

                 size="petal\_length", color="species")

fig.show()

import plotly.express as px

df = px.data.iris()

fig = px.scatter\_3d(df, x = 'sepal\_width',

                    y = 'sepal\_length',

                    z = 'petal\_width',

                    color = 'species')

fig.show()

**OUTPUT:**

[1 1 0 0]

[(7, 7), (7, 4), (3, 4), (1, 4)]

['Good']

**PROGRAM NO : 2 DATE : 07 – 01 – 2022**

**AIM : Predict class label of a given data point using KNN.**

**SOURCE CODE :**

from sklearn.neighbors import KNeighborsClassifier

x1=[7,7,3,1]

x2=[7,4,4,4]

target=['bad','bad','Good','Good']

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

target\_encoded=le.fit\_transform(target)

print(target\_encoded)

features=zip(x1,x2)

features = list(features)

features

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(features,target)

print(knn.predict([[3,7]]))

**OUTPUT:**

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

Labels: ['setosa' 'versicolor' 'virginica']

[2]

[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

0.9666666666666667

**PROGRAM NO : 3 DATE : 07 – 01 – 2022**

**AIM : Calculate accuracy of KNN using iris dataset.**

**SOURCE CODE :**

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.datasets import load\_iris

irisData = load\_iris()

print("Features: ", irisData.feature\_names)

print("Labels: ", irisData.target\_names)

X = irisData.data

y = irisData.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=7)

knn.fit(X\_train, y\_train)

print(knn.predict([[7.7,2.6,6.9,2.3]]))

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

print(y\_pred)

from sklearn.metrics import accuracy\_score

ac = accuracy\_score(y\_test,y\_pred)

print(ac)

**OUTPUT:**

[2 2 0 1 1 1 0 2 2 1 2 0 0 1]

Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]

Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]

[(2, 1),

(2, 1),

(0, 1),

(1, 2),

(1, 0),

(1, 0),

(0, 0),

(2, 2),

(2, 0),

(1, 2),

(2, 2),

(0, 2),

(0, 1),

(1, 2)]

Predicted Value: [1]

**PROGRAM NO : 4 DATE : 13 – 01 – 2022**

**AIM : Predict the class label of an unseen observation using Naive-bayes.**

**SOURCE CODE :**

weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',

'Rainy','Sunny','Overcast','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

from sklearn import preprocessing

#creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers.

weather\_encoded=le.fit\_transform(weather)

print(weather\_encoded)

temp\_encoded=le.fit\_transform(temp)

label=le.fit\_transform(play)

print("Temp:",temp\_encoded)

print("Play:",label)

features=zip(weather\_encoded,temp\_encoded)

features = list(features)

features

from sklearn.naive\_bayes import GaussianNB

#Create a Gaussian Classifier

model = GaussianNB()

# Train the model using the training sets

model.fit(features,label)

#Predict Output

predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild

print("Predicted Value:", predicted)

**OUTPUT:**

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

Labels: ['setosa' 'versicolor' 'virginica']

[2]

[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

1.0

**PROGRAM NO : 5 DATE : 13 – 01 – 2022**

**AIM : Predict the accuracy of standard dataset using naive bayes.**

**SOURCE CODE :**

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

from sklearn.datasets import load\_iris

# Loading data

irisData = load\_iris()

print("Features: ", irisData.feature\_names)

# print the label type of wine(class\_0, class\_1, class\_2)

print("Labels: ", irisData.target\_names)

# Create feature and target arrays

X = irisData.data

y = irisData.target

# Split into training and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state=42)

from sklearn.naive\_bayes import GaussianNB

#Create a Gaussian Classifier

gnb = GaussianNB()

#Train the model using the training sets

gnb.fit(X\_train, y\_train)

#Predict the response for test dataset

print(gnb.predict([[7.7,2.6,6.9,2.3]]))

y\_pred1 = gnb.predict(X\_test)

print(y\_pred1)

from sklearn.metrics import accuracy\_score

ac1 = accuracy\_score(y\_test,y\_pred1)

print(ac1)

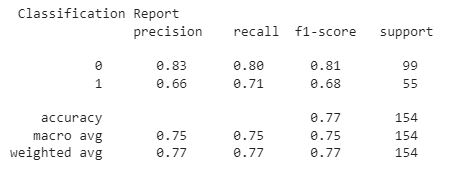
**OUTPUT:**

0.7662337662337663

confusion matrix:

[[79 20]

[16 39]]



**PROGRAM NO : 6 DATE : 14 – 01 – 2022**

**AIM : Generate classification report and confusion matrix of diabetes dataset using naive bayes algorithm.**

**SOURCE CODE :**

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

from sklearn.naive\_bayes import GaussianNB

import pandas as pd

df=pd.read\_csv("./diabetes.csv")

df.head()

X=df.drop("Outcome",axis=1)

y=df["Outcome"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state=42)

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

y\_pred1 = gnb.predict(X\_test)

print(y\_pred1)

from sklearn.metrics import accuracy\_score

ac1 = accuracy\_score(y\_test,y\_pred1)

print(ac1)

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

matrix = confusion\_matrix(y\_test,y\_pred1)

print("confusion matrix: \n",matrix)

cr = classification\_report(y\_test,y\_pred1)

print( C"lassification Report \n ",cr)

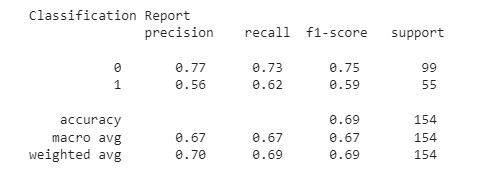
**OUTPUT:**

0.6883116883116883

confusion matrix :

[[72 27]

[21 34]]



**PROGRAM NO : 7 DATE : 14 – 01 – 2022**

**AIM : Generate classification report and confusion matrix of diabetes dataset using KNN algorithm**

**SOURCE CODE :**

from sklearn.neighbors import KNeighborsClassifier

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

import pandas as pd

df = pd.read\_csv("./diabetes.csv")

x = df.drop("Outcome",axis =1)

y = df["Outcome"]

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

knn = KNeighborsClassifier (n\_neighbors = 7)

knn.fit(x\_train,y\_train)

y\_pred = knn.predict(x\_test)

from sklearn.metrics import accuracy\_score

ac = accuracy\_score(y\_test,y\_pred)

print(ac)

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

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

print("confusion matrix : \n",matrix)

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

print("Classification Report \n",cr)

**OUTPUT:**

(768, 8)

(768,)

DecisionTreeClassifier()

Accuracy on train data using Gini: 1.0

Accuracy on test data using Gini: 0.703125

Accuracy on train data using entropy 1.0

Accuracy on test data using entropy 0.734375

**PROGRAM NO : 8 DATE : 20 – 01 – 2022**

**AIM : Implement Decision tree algorithm, find accuracy in pima Indian diabetes dataset.**

**SOURCE CODE :**

import pandas as pd

df=pd.read\_csv("./diabetes.csv")

X=df.drop("Outcome",axis=1)

y=df["Outcome"]

display (X.shape, y.shape)

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

from sklearn.tree import DecisionTreeClassifier

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,random\_state = 50, test\_size = 0.25)

classifier = DecisionTreeClassifier()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import accuracy\_score

print('Accuracy on train data using Gini: ',accuracy\_score(y\_train, y\_pred = classifier.predict (X\_train)))

print('Accuracy on test data using Gini: ',accuracy\_score(y\_test,y\_pred))

**OUTPUT:**

array([120197.8256403 , 88644.21802942, 74146.61453254, 118492.2252289 ,

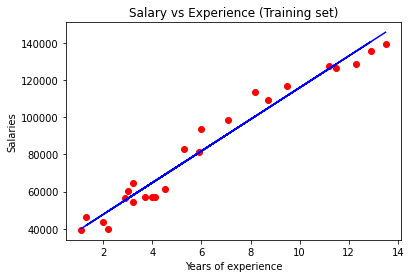
98025.02029212, 72441.01412114, 63913.01206415, 43445.80712736,

64765.81226984, 112522.623789 , 107405.82255481])\

Coefficients: [8528.00205699]

intercept: 30653.80404187284

array([121872, 91738, 66029, 122391, 101302, 67938, 63218, 37731,

 55794, 112635, 105582])

**PROGRAM NO : 9 DATE : 27 – 01 – 2022**

**AIM : Implement simple linear regression using salary dataset.**

**SOURCE CODE :**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read\_csv('./Salary.csv')

dataset.head()

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=0.3,random\_state=0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train,y\_train) #actually produces the linear eqn for the data

 # predicting the test set results

y\_pred = regressor.predict(X\_test)

y\_pred

print('Coefficients: ', regressor.coef\_)

print('intercept:', regressor.intercept\_)

print((y\_test)

plt.scatter(X\_train, y\_train, color='red') # plotting the observation line

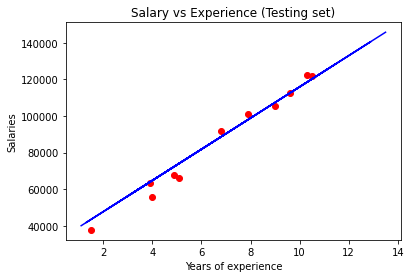
plt.plot(X\_train, regressor.predict(X\_train), color='blue') # plotting the regression line

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



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()

**OUTPUT:**

LinearRegression()

intercept: 2.652789668879496

coefficients:

[('TV', 0.04542559602399794),

('Radio', 0.18975772766893614),

('Newspaper', 0.004603078953112072)]

prediction:[10.62160072 20.00625302 16.91850882 19.17040746 20.94974131 13.12284284

11.80740696 12.32019766 20.57806782 20.95662688 10.79096475 19.54868702

6.42403866 15.23133391 8.97226257 7.89897862 16.23599497 12.02636477

17.09702178 11.26080277 16.97826292 9.75655721 20.82389762 17.20916742

15.13816239 21.97290698 19.20181841 10.07501899 19.39017185 14.8673761

14.36798893 7.55604543 9.96742165 14.76342565 7.20995576 13.60003295

7.49088656 11.70865932 13.46091883 15.2229793 17.18088277 13.56738329

14.30942267 13.72909849 11.88559349 8.77039705 12.1244102 19.20252289

9.08376601 5.15367352 16.22852749 18.14111213 12.94835466 16.86274503

17.86462435 12.33930625 4.3575739 11.25904494 16.11560622 13.56602169]

**PROGRAM NO : 10 DATE : 28 – 01 – 2022**

**AIM : Implement Multiple Linear Regression using Advertising dataset.**

**SOURCE CODE :**

import pandas as pd

dst = pd.read\_csv('./Advertising.csv')

dst.head()

x = dst.iloc[:,:-1]

y = dst.iloc[:,-1]

x.head()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=100)

from sklearn.linear\_model import LinearRegression

mr = LinearRegression()

mr.fit(x\_train,y\_train)

print('intercept:',mr.intercept\_)

print("coefficients:")

list(zip(x,mr.coef\_))

y\_pred = mr.predict(x\_test)

print("prediction:{}".format(y\_pred))

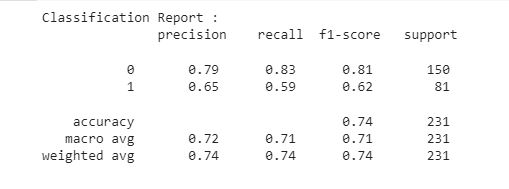
**OUTPUT:**

from sklearn.metrics import accuracy\_score

Confusion Matrix :

[[124 26]

[ 33 48]]



**PROGRAM NO : 11 DATE : 03 – 02 – 2022**

**AIM : Implement SVM Classification using diabetes dataset.**

**SOURCE CODE :**

import pandas as pd

ds = pd.read\_csv('./diabetes.csv')

ds.head()

x = ds.iloc[:,:-1]

x.head()

y = ds.iloc[:,-1]

y.head()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = 0.3, random\_state = 100)

from sklearn.svm import SVC

classifier = SVC ( kernel = 'linear')

classifier.fit(x\_train, y\_train)

y\_pred = classifier.predict(x\_test)

from sklearn.metrics import accuracy\_score

ac = accuracy\_score(y\_test,y\_pred)

print("Accuracy Score : ",ac)

from sklearn.metrics import confusion\_matrix

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

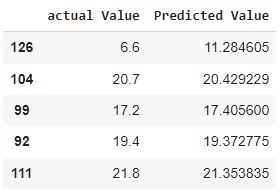
print("Confusion Matrix : \n",cm)

from sklearn.metrics import classification\_report

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

print("Classification Report : \n",cr)

**OUTPUT:**



**PROGRAM NO : 12 DATE : 04 – 02 – 2022**

**AIM : Implement SVM Regression using Advertising dataset.**

**SOURCE CODE :**

import pandas as pd

dataset = pd.read\_csv("/content/Advertising.csv")

x=dataset.iloc[:,:-1]

y=dataset.iloc[:,-1]

dataset.head()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = 0.3, random\_state = 100)

from sklearn.svm import SVR

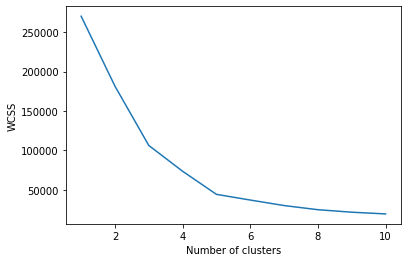
Re = SVR(kernel='linear')

Re.fit(x\_train, y\_train)

y\_pred= Re.predict(x\_test)

diff=pd.DataFrame({'actual Value': y\_test, 'Predicted Value': y\_pred})

diff.head()

**OUTPUT:**

[2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2

3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4

1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1

4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]

**PROGRAM NO : 13 DATE : 08 – 02 – 2022**

**AIM : Implement KMeans using mall customer dataset.**

**SOURCE CODE :**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

    kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

    kmeans.fit(X)

    wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

plt.plot(range(1, 11), wcss)

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters = 5, init = "k-means++", random\_state = 42)

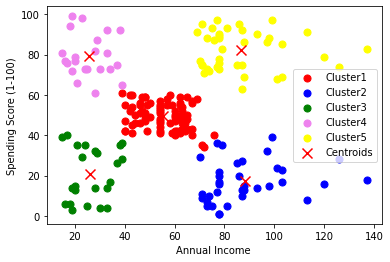
y\_kmeans = kmeans.fit\_predict(X)

print(y\_kmeans)

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 50, c = 'red', label = 'Cluster1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 50, c = 'blue', label = 'Cluster2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 50, c = 'green', label = 'Cluster3')



plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 50, c = 'violet', label = 'Cluster4')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 50, c = 'yellow', label = 'Cluster5')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 100,marker='x',  c = 'red', label = 'Centroids')

plt.xlabel('Annual Income ')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**OUTPUT:**

['Purpose', 'Ease-of-use', 'Speed', 'Learning Curve', 'Documentation', 'JavaScript Support', 'CPU and Memory Usage', 'Size of Web Scraping Project Supported']

**PROGRAM NO : 14 DATE : 23 – 02 – 2022**

**AIM : Implement Web Scrapping using python**

**SOURCE CODE :**

pip install autoscraper

from autoscraper import AutoScraper

url="http://oxylabs.io/blog/python-web-scraping"

wanted\_list=['Purpose']

Scraper=AutoScraper()

result=Scraper.build(url,wanted\_list)

print(result)

**OUTPUT:**

[('Hello', 'NNP'), ('.', '.')]

[('MCA', 'NNP'), ('S3', 'NNP'), ('fantastic', 'JJ'), ('.', '.')]

[('We', 'PRP'), ('learn', 'VBP'), ('many', 'JJ'), ('new', 'JJ'), ('concepts', 'NNS'), ('implement', 'JJ'), ('practical', 'JJ'), ('exams', 'NN'), ('.', '.')]

[('1st', 'CD'), ('data', 'NNS'), ('science', 'NN'), ('new', 'JJ'), ('paper', 'NN'), ('.', '.')]

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']

['The sun', 'sun rises', 'rises east']

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']

['The sun rises', 'sun rises east']

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']

['The sun rises east']

**PROGRAM NO : 15 DATE : 24 – 02 – 2022**

**AIM : Implement problem on Natural Language Processing – part of speech, tagging Ngram using NLTK**

**SOURCE CODE :**

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize

nltk.download('stopwords')

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

stop\_words = set(stopwords.words('english'))

txt = "Hello. MCA S3 is fantastic. We learn many new concepts and implement them in  our practical exams. "\

"1st of all the data science is a new paper."

tokenized = sent\_tokenize(txt)

for i in tokenized:

   wordsList = nltk.word\_tokenize(i)

   wordsList = [w for w in wordsList if not w in stop\_words]

   tagged = nltk.pos\_tag(wordsList)

   print(tagged)

def generate\_N\_grams(text,ngram=1):

  words=[word for word in text.split(" ") if word not in set(stopwords.words('english'))]

  print("Sentence after removing stopwords:",words)

  temp=zip(\*[words[i:] for i in range(0,ngram)])

  ans=[' '.join(ngram) for ngram in temp]

  return ans

generate\_N\_grams("The sun rises in the east",2)

generate\_N\_grams("The sun rises in the east",3)

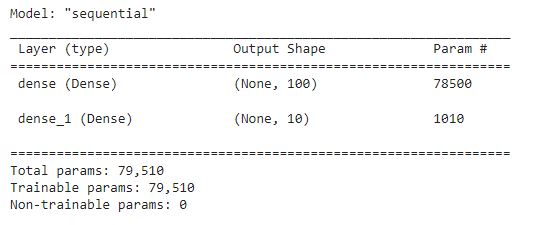
generate\_N\_grams("The sun rises in the east",4)

**OUTPUT:**

**­**

X\_train shape (60000, 28, 28)

2



**PROGRAM NO : 16 DATE : 24 – 02 – 2022**

**AIM : Program on CNN to classify images from any standard dataset in the public domain using keras framework.**

**SOURCE CODE :**

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.utils import np\_utils

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

print("X\_train shape", X\_train.shape)

print("y\_train shape", y\_train.shape)

print("X\_test shape", X\_test.shape)

print("y\_test shape", y\_test.shape)

import matplotlib.pyplot as plt

plt.imshow(X\_train[5], cmap=plt.cm.binary)

print(y\_train[5])

X\_train = X\_train.reshape(60000, 784)

X\_test = X\_test.reshape(10000, 784)

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

X\_train.shape

n\_classes = 10

Y\_train = np\_utils.to\_categorical(y\_train, n\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, n\_classes)

model = Sequential()

model.add(Dense(100,input\_shape=(784,), activation='relu'))

model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical\_crossentropy', metrics=['accuracy'], optimizer='adam')

model.fit(X\_train, Y\_train, batch\_size=100, epochs=10)

test\_loss, test\_acc = model.evaluate(X\_test, Y\_test)

print("TEST ACCURACY",round(test\_acc,3))

print("TEST LOSS",round(test\_loss,3))