**Sinhgad College of Engineering**

**Department Of Computer Engineering**

Name Of Student:

Roll No.: Exam No.:

Class: Div: Batch:

Name Of Laboratory:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **Start Date** | **Completion Date** |
| 1 | To implement Linear Regression to find the equation of the best fit line for given data. |  |  |
| 2 | To implement Decision Tree Classifier. |  |  |
| 3 | To implement k-NN algorithm for classifying the points on given graph. |  |  |
| 4 | To implement K-means Algorithm for clustering. |  |  |
| 5 | To implement a Simplified Data Encryption Standard (S-DES) algorithm. |  |  |
| 6 | To implement a Simplified Advanced Encryption Standard (S-AES)  algorithm. |  |  |
| 7 | To implement a Diffie-Hellman Key Exchange algorithm. |  |  |
| 8 | To implement a RSA algorithm. |  |  |
| 9 | Mini Project on Machine Learning |  |  |
| 10 | Mini Project on Information and Cyber Security |  |  |

Graphical user interface, text, application, email

Description automatically generatedAssignment 5 : (Input-Output)

Assignment 6 : (Input-Output)

Text

Description automatically generated

Text

Description automatically generatedAssignment 7 : (Input-Output)

Text

Description automatically generated

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedAssignment 8 : (Input-Output)

Text, letter

Description automatically generated

Assignment 1 : (Input-Output)

#import the packges

import matplotlib.pyplot as plt

import pandas as pd

#Read Dataset

dataset=pd.read\_csv("hours.csv")

#index read

x=dataset.iloc[:,:-1].values #slice all column

y=dataset.iloc[:,1].values #last Column

#import packages of LR

from sklearn.linear\_model import LinearRegression

regressor=LinearRegression() #create object of LR

# Fit Function

regressor.fit(x,y)

#score Function

Accuracy=regressor.score(x,y)\*100

print('Accuracy')

print(Accuracy)

#Predict Function

y\_pred=regressor.predict([[10]])

print(y\_pred)

#input from user

hours=int(input("Enter the no of hours"))

# Coefficient

# intercept

eq=regressor.coef\_\*hours+regressor.intercept\_

print("Risk Score",eq[0])

plt.plot(x,y,'o')

plt.plot(x,regressor.predict(x));

plt.show()

Chart, scatter chart

Description automatically generated

Assignment 2 : (Input-Output)

#import packages

import pandas as pd

import numpy as np

from graphviz import Digraph

from IPython.display import Image

# Read dataset

dataset=pd.read\_csv("tree1.csv")

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

y=dataset.iloc[:,5]

#Label encoder

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

x=x.apply(le.fit\_transform)

print(x)

# 1 1 0 0

#import Decesion Tree Classifier

from sklearn.tree import DecisionTreeClassifier

# Create decision tree classifer object

regressor=DecisionTreeClassifier()

# Train model

regressor.fit(x.iloc[:,1:5],y)

x\_in=np.array([1,1,0,0])

y\_pred=regressor.predict([x\_in])

print("Predicted class for input [Age < 21, Income =Low,Gender = Female, Marital Status = Married]\n", x\_in," is",y\_pred[0])

from six import StringIO

#from IPython.display import Image

from sklearn.tree import export\_graphviz

import pydotplus

dot\_data=StringIO()

export\_graphviz(regressor,out\_file=dot\_data,filled=True,rounded=True,special\_characters=True)

#Draw Graph

graph=pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

# Show graph & Create png File

graph.write\_png("tree.png")

#Image(graph.create\_png())

Table

Description automatically generated with low confidence

Assignment 3 : (Input-Output)

#import the packages

import pandas as pd

import numpy as np

#Read dataset

dataset=pd.read\_csv("kdata.csv")

X=dataset.iloc[:,:-1].values

y=dataset.iloc[:,2].values

#import KNeighborshood Classifier and create object of it

from sklearn.neighbors import KNeighborsClassifier

#Creating model

classifier=KNeighborsClassifier(n\_neighbors=3)

# Training model

classifier.fit(X,y)

#predict the class for the point(6,6)

X\_test=np.array([6,6])

# Predictions for test data

y\_pred=classifier.predict([X\_test])

print("class of the point (6,6) : ",y\_pred)

# KNeighborsClassifier looks for the 5 nearest neighbors

#If set to uniform, all points in each neighbourhood have

#equal influence in predicting class i.e. predicted class is the class with highest number of points in the neighbourhood.

classifier=KNeighborsClassifier(n\_neighbors=3,weights='distance')

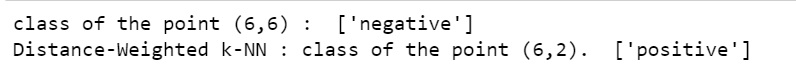
classifier.fit(X,y)

#predict the class for the point(6,2)

X\_test=np.array([6,2])

y\_pred=classifier.predict([X\_test])

print("Distance-Weighted k-NN : class of the point (6,2). ",y\_pred)



Assignment 4 : (Input-Output)

#import packages

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#create dataset using DataFrame

df=pd.DataFrame({'X':[0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3],

'y':[0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2]})

f1 = df['X'].values

f2 = df['y'].values

X = np.array(list(zip(f1, f2)))

print(X)

#centroid points

C\_x=np.array([0.1,0.3])

C\_y=np.array([0.6,0.2])

centroids=C\_x,C\_y

#plot the given points

colmap = {1: 'r', 2: 'b'}

plt.scatter(f1, f2, color='k')

plt.show()

#for i in centroids():

plt.scatter(C\_x[0],C\_y[0], color=colmap[1])

plt.scatter(C\_x[1],C\_y[1], color=colmap[2])

plt.show()

C = np.array(list((C\_x, C\_y)), dtype=np.float32)

print (C)

#plot given elements with centroid elements

plt.scatter(f1, f2, c='#050505')

plt.scatter(C\_x[0], C\_y[0], marker='\*', s=200, c='r')

plt.scatter(C\_x[1], C\_y[1], marker='\*', s=200, c='b')

plt.show()

#import KMeans class and create object of it

from sklearn.cluster import KMeans

model=KMeans(n\_clusters=2,random\_state=0)

model.fit(X)

labels=model.labels\_

print(labels)

#using labels find population around centroid

count=0

for i in range(len(labels)):

if (labels[i]==1):

count=count+1

print('No of population around cluster 2:',count-1)

#Find new centroids

new\_centroids = model.cluster\_centers\_

print('Previous value of m1 and m2 is:')

print('M1==',centroids[0])

print('M1==',centroids[1])

print('updated value of m1 and m2 is:')

print('M1==',new\_centroids[0])

print('M1==',new\_centroids[1])

Shape

Description automatically generated with medium confidenceChart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated