**Practical No. 01**

**Aim:Design a simple machine learning model to train the training instances and test the same.**

# python library to generate random numbers

from random import randint

# the limit within which random numbers are generated

TRAIN\_SET\_LIMIT = 1000

# to create exactly 100 data items

TRAIN\_SET\_COUNT = 100

# list that contains input and corresponding output

TRAIN\_INPUT = list()

TRAIN\_OUTPUT = list()

# loop to create 100 data items with three columns each

for i in range(TRAIN\_SET\_COUNT):

a = randint(0, TRAIN\_SET\_LIMIT)

b = randint(0, TRAIN\_SET\_LIMIT)

c = randint(0, TRAIN\_SET\_LIMIT)

# creating the output for each data item

op = a + (2 \* b) + (3 \* c)

TRAIN\_INPUT.append([a, b, c])

# adding each output to output list

TRAIN\_OUTPUT.append(op)

# Sk-Learn contains the linear regression model

from sklearn.linear\_model import LinearRegression

# Initialize the linear regression model

predictor = LinearRegression(n\_jobs =-1)

# Fill the Model with the Data

predictor.fit(X = TRAIN\_INPUT, y = TRAIN\_OUTPUT)

# Random Test data

X\_TEST = [[ 10, 20, 30 ]]

# Predict the result of X\_TEST which holds testing data

outcome = predictor.predict(X = X\_TEST)

# Predict the coefficients

coefficients = predictor.coef\_

# Print the result obtained for the test data

print('Outcome : {}\nCoefficients : {}'.format(outcome, coefficients))

**Practical No. 02**

**Aim: Perform Data Loading, Feature selection (Principal Component analysis).**

import numpy

from pandas import read\_csv

from sklearn.decomposition import PCA

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

# load data

url = "pima-indians-diabetes.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(url, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

# feature extraction

pca = PCA(n\_components=3)

fit = pca.fit(X)

# summarize components

print("Explained Variance: %s" % fit.explained\_variance\_ratio\_)

print(fit.components\_)

**Practical No. 03**

**Aim: Perform Data Loading, Feature selection Feature Scoring and Ranking.**

from pandas import read\_csv

from sklearn.ensemble import ExtraTreesClassifier

# load data

url = "pima-indians-diabetes.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(url, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

# feature extraction

model = ExtraTreesClassifier(n\_estimators=10)

model.fit(X, Y)

print(names)

print(model.feature\_importances\_)

**Practical No. 04**

**Aim:Write a program to implement Decision Tree.**

from matplotlib import pyplot as plt

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

# Prepare the data data

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# Fit the classifier with default hyper-parameters

clf = DecisionTreeClassifier(random\_state=1234)

model = clf.fit(X, y)

fig = plt.figure()

\_ = tree.plot\_tree(clf,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

filled=True)

plt.show()

**Practical No. 05**

**Aim: For a given set of training data examples stored in a .CSV file implement Least Square Regression algorithm.**

# regression line

"""

To find regression line, we need to find a and b.

Calculate a, which is given by a = (\sum yi)/n - b \* (\sum xi)/n

Calculate b, which is given by

b = (n\*\sum(xi\*yi) - \sum (xi)\* \sum (yi))/(n\*\sum (xi)^{2}-(\sum xi)^{2})

Put value of a and b in the equation of regression line.

"""

# Function to calculate b

def calculateB(x, y, n):

# sum of array x

sx = sum(x)

# sum of array y

sy = sum(y)

# for sum of product of x and y

sxsy = 0

# sum of square of x

sx2 = 0

for i in range(n):

sxsy += x[i] \* y[i]

sx2 += x[i] \* x[i]

b = (n \* sxsy - sx \* sy)/(n \* sx2 - sx \* sx)

return b

# Function to find the

# least regression line

def leastRegLine(X,Y,n):

# Finding b

b = calculateB(X, Y, n)

meanX = int(sum(X)/n)

meanY = int(sum(Y)/n)

# Calculating a

a = meanY - b \* meanX

# Printing regression line

print("Regression line:")

print("Y = ", '%.3f'%a, " + ", '%.3f'%b, "\*X", sep="")

# Driver code

# Statistical data

import pandas as pd

# Step 1 :Import libraries and dataset

datas = pd.read\_csv('data.csv')

print(datas )

X = datas.TEMPERATURE

Y = datas.PRESSURE

n = len(X)

leastRegLine(X, Y, n)

**Practical No. 06**

**Aim: For a given set of training data examples stored in a .CSV file implement Linear Regression algorithm.**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Step 1 :Import libraries and dataset

datas = pd.read\_csv('data.csv')

print(datas )

#Step 2: Dividing the dataset into 2 components

X = datas.iloc[:, 1:2].values

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

#Step 3: Fitting Linear Regression to the dataset

from sklearn.linear\_model import LinearRegression

lin = LinearRegression()

lin.fit(X, y)

plt.scatter(X, y, color = 'blue')

plt.plot(X, lin.predict(X), color = 'red')

plt.title('Linear Regression')

plt.xlabel('Temperature')

plt.ylabel('Pressure')

plt.show()

**Practical No. 07**

**Aim: Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set.**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

iris = load\_iris()

df = pd.DataFrame(iris.data,columns=iris.feature\_names)

print(df)

df['target'] = iris.target

df['flower\_name'] =df.target.apply(lambda x: iris.target\_names[x])

print(df)

df0 = df[:50] # setosa

df1 = df[50:100] # versicolor

df2 = df[100:] # virginica

X = df.drop(['target','flower\_name'], axis='columns')

y = df.target

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

knn = KNeighborsClassifier(n\_neighbors=10)

knn.fit(X\_test, y\_test)

print(knn.score(X\_test, y\_test))

**Practical No. 08**

**Aim: Implement the classification model using clustering for the following techniques with K means clustering**

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

######

from sklearn.cluster import KMeans

data = list(zip(x, y))

inertias = []

for i in range(1,11):

kmeans = KMeans(n\_clusters=i)

kmeans.fit(data)

inertias.append(kmeans.inertia\_)

plt.plot(range(1,11), inertias, marker='o')

plt.title('Elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()

#####

data = list(zip(x, y))

kmeans = KMeans(n\_clusters=2)

kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels\_)

plt.show()

**Practical No. 09**

**Aim: Implement the classification model using clustering for the following techniques with hierarchical clustering with Prediction**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

from scipy.cluster.hierarchy import dendrogram, linkage

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))

hierarchical\_cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

labels = hierarchical\_cluster.fit\_predict(data)

print(labels)

plt.scatter(x, y, c=labels)

plt.show()