import pandas

import numpy as np

import csv

file\_path="C:\\Users\\nishit\\Desktop\\Desktop Doc\\P1.csv"

df=pandas.read\_csv(file\_path)

p = df[df.columns[1:16384]]

b = [[100]] \* 16384

x = np.linalg.lstsq(p, b)[0]

from matplotlib import pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage

import numpy as np

X=df

Z = linkage(X, 'ward')

plt.figure(figsize=(25, 10))

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('sample index')

plt.ylabel('distance')

dendrogram(

    Z,

    leaf\_rotation=90.,  # rotates the x axis labels

    leaf\_font\_size=8.,  # font size for the x axis labels

)

plt.show()

import numpy as np

import pylab as pl

from sklearn import svm, datasets

## import some data to play with

#iris = datasets.load\_iris()

#X = iris.data[:, :2]  # we only take the first two features. We could

                     # avoid this ugly slicing by using a two-dim dataset

#Y = iris.target

X=X.transpose()

b=np.array([0,1,2])

Y=np.repeat(b,10)

h = 10  # step size in the mesh

# we create an instance of SVM and fit out data. We do not scale our

# data since we want to plot the support vectors

C = 1.0  # SVM regularization parameter

svc = svm.SVC(kernel='linear', C=C).fit(X, Y)

rbf\_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, Y)

#poly\_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, Y)

lin\_svc = svm.LinearSVC(C=C).fit(X, Y)

# create a mesh to plot in

x\_min, x\_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1

y\_min, y\_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),np.arange(y\_min, y\_max, h))

# title for the plots

titles = ['SVC with linear kernel','SVC with RBF kernel','SVC with polynomial (degree 3) kernel','LinearSVC (linear kernel)']

for i, clf in enumerate((svc, rbf\_svc,lin\_svc)):

    # Plot the decision boundary. For that, we will assign a color to each

    # point in the mesh [x\_min, m\_max]x[y\_min, y\_max].

    pl.subplot(2, 2, i + 1)

    Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

    M=svc.predict(X)

    # Put the result into a color plot

    Z = Z.reshape(xx.shape)

    pl.contourf(xx, yy, Z, cmap=pl.cm.Paired)

    pl.axis('off')

    # Plot also the training points

    pl.scatter(X.iloc[:, 0], X.iloc[:, 1], c=Y, cmap=pl.cm.Paired)