Problem statement:

Perform Principal component analysis and perform clustering using first

3 principal component scores (both heirarchial and k mean clustering(scree plot or elbow curve) and obtain

optimum number of clusters and check whether we have obtained same number of clusters with the original data

(class column we have ignored at the begining who shows it has 3 clusters)df

Codes:

# -\*- coding: utf-8 -\*-

"""

Created on Sat Oct 17 02:56:31 2020

@author: sunil

"""

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from scipy.spatial.distance import cdist

wine=pd.read\_csv("wine.csv")

wine.describe()

wine.head(5)

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale

wine\_data=wine.iloc[:,1:]

#building an PCA model

wine\_norm=scale(wine\_data)

pca=PCA().fit\_transform(wine\_norm)

pca=PCA()

pca\_val=pca.fit\_transform(wine\_norm)

pca\_val.shape

var=pca.explained\_variance\_ratio\_

var

pca.components\_[0]

var1=np.cumsum(np.round(var,decimals=4)\*100)

var1

plt.plot(var1,color="red")

x=np.array(pca\_val[:,0:])

y=np.array(pca\_val[:,1:])

plt.plot(x,y,"ro")

plt.plot(np.arange(25),x,"ro")

######################### k-means-clustering####################################

def norm\_func(i):

x = (i-i.min()) / (i.max() - i.min())

return (x)

df\_norm = norm\_func(wine.iloc[:,1:])

k = list(range(2,13))

k

TWSS = [] # variable for storing total within sum of squares for each kmeans

for i in k:

kmeans = KMeans(n\_clusters = i)

kmeans.fit(df\_norm)

WSS = [] # variable for storing within sum of squares for each cluster

for j in range(i):

WSS.append(sum(cdist(df\_norm.iloc[kmeans.labels\_==j,:],kmeans.cluster\_centers\_[j].reshape(1,df\_norm.shape[1]),"euclidean")))

TWSS.append(sum(WSS))

# Scree plot

plt.plot(k,TWSS, 'ro-');plt.xlabel("No\_of\_Clusters");plt.ylabel("total\_within\_SS");plt.xticks(k)

#########################hierarchical clustering################################

from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch

type(df\_norm)

z = linkage(df\_norm, method="complete",metric="euclidean")

plt.figure(figsize=(15, 5));plt.title('Hierarchical Clustering Dendrogram');plt.xlabel('Index');plt.ylabel('Distance')

sch.dendrogram(

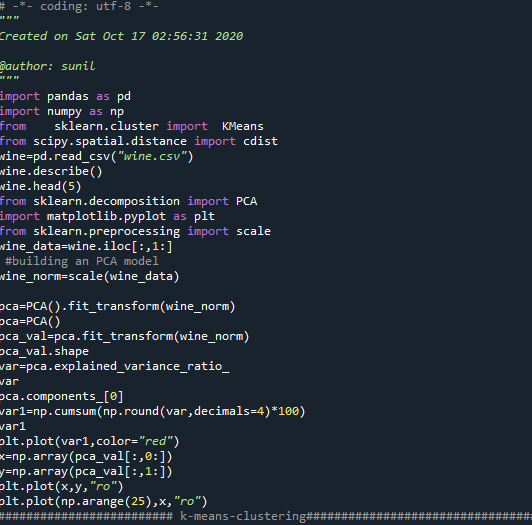
z,

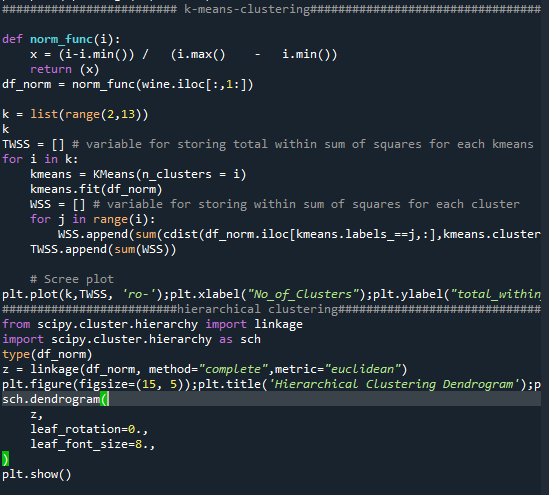
leaf\_rotation=0.,

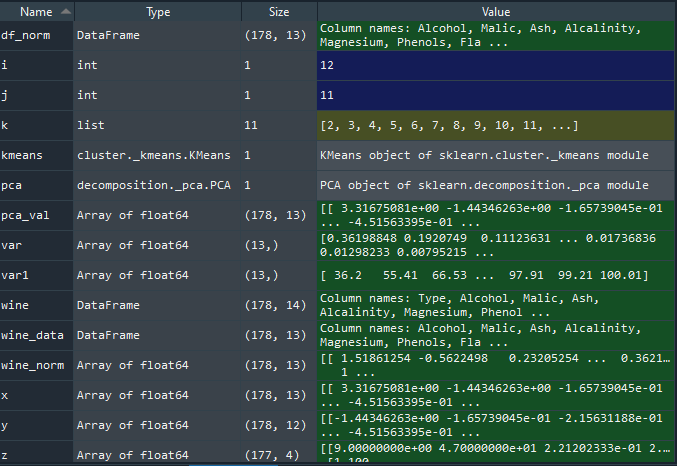
leaf\_font\_size=8.,

)

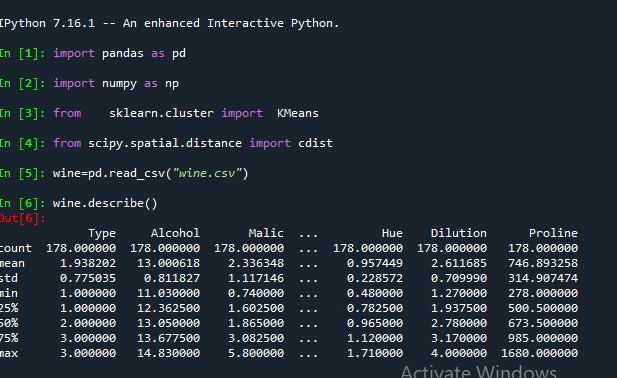
plt.show()

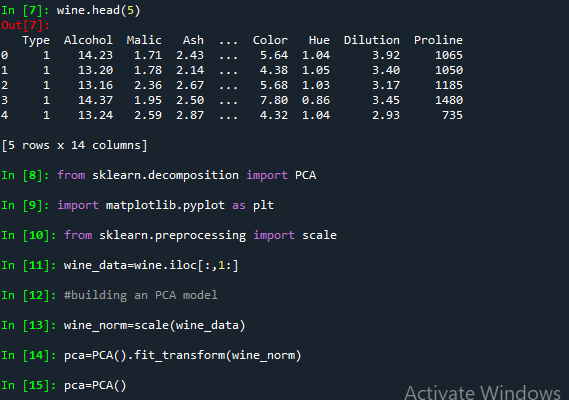


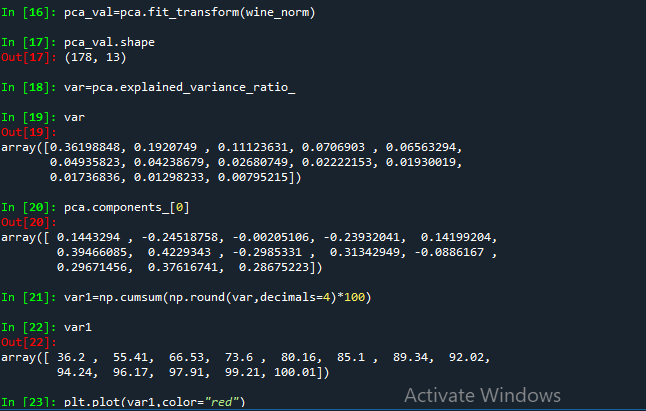


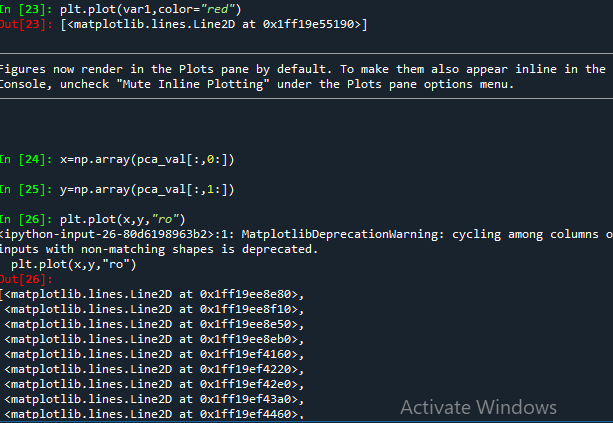


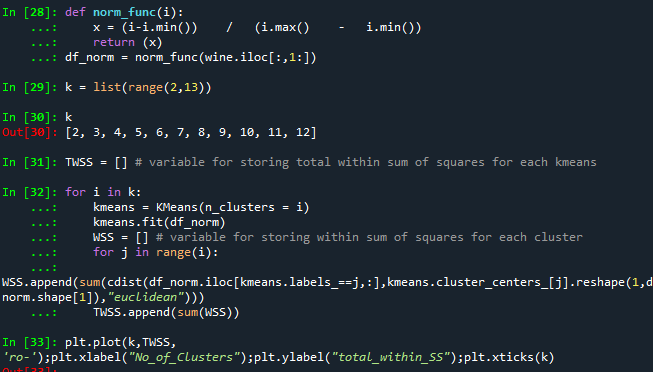
OUTPUT:

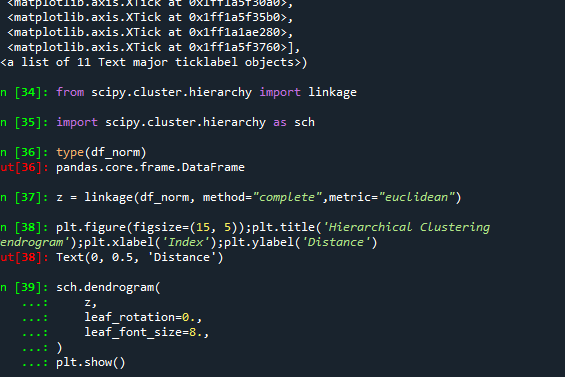


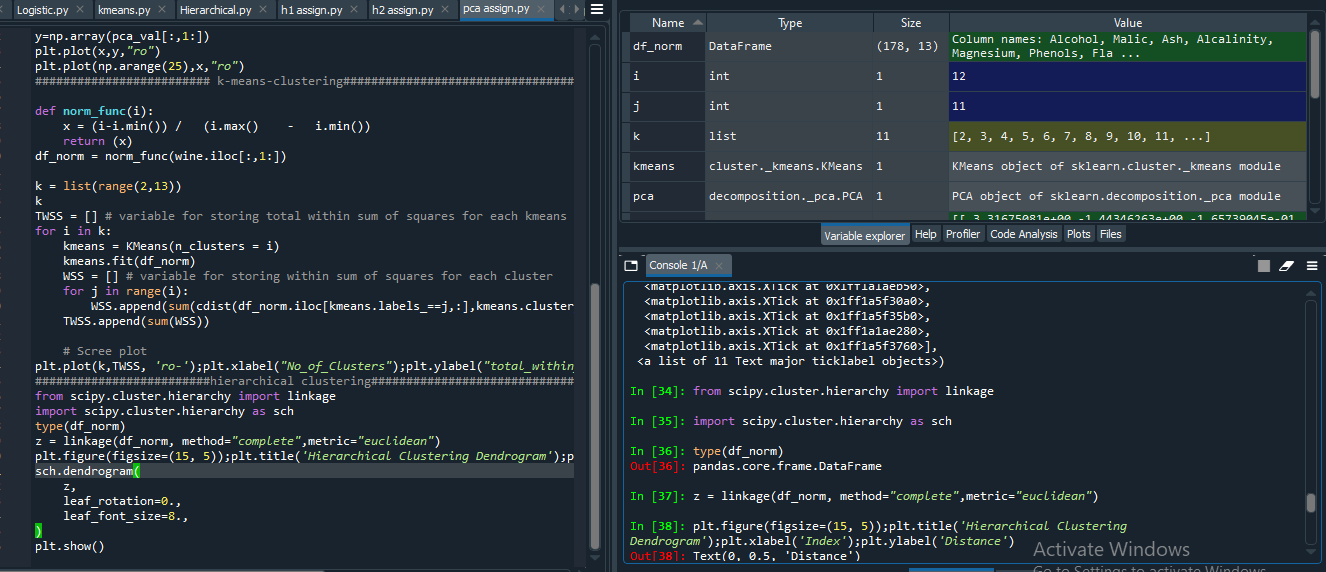












PLOTS:

