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

ANS : import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale

wine=pd.read\_csv('E:\\assignments\\wine.csv')

print(wine.describe())

wine.head()

wine['Type'].value\_counts()

Wine= wine.iloc[:,1:]

Wine

wine.shape

wine.info()

wine\_ary=Wine.values

wine\_ary

wine\_norm=scale(wine\_ary)

wine\_norm

pca = PCA()

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

pca\_values

pca.components\_

var = pca.explained\_variance\_ratio\_

var

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

Var

plt.plot(Var,color="blue");

final\_df=pd.concat([wine['Type'],pd.DataFrame(pca\_values[:,0:3],columns=['PC1','PC2','PC3'])],axis=1)

final\_df

import seaborn as sns

fig=plt.figure(figsize=(16,12))

sns.scatterplot(data=final\_df);

sns.scatterplot(data=final\_df, x='PC1', y='PC2', hue='Type');

pca\_values[: ,0:1]

x= pca\_values[:,0:1]

y= pca\_values[:,1:2]

plt.scatter(x,y);

# Hierarchical Clustering

import scipy.cluster.hierarchy as sch

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import normalize

plt.figure(figsize=(10,8))

dendrogram=sch.dendrogram(sch.linkage(wine\_norm,'complete'))

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

hclusters

y=pd.DataFrame(hclusters.fit\_predict(wine\_norm),columns=['clustersid'])

y['clustersid'].value\_counts()

wine2=wine.copy()

wine2['clustersid']=hclusters.labels\_

wine2

# K-Means Clustering

from sklearn.cluster import KMeans

wcss=[]

for i in range (1,6):

kmeans=KMeans(n\_clusters=i,random\_state=2)

kmeans.fit(wine\_norm)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,6),wcss)

plt.title('Elbow Graph')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS');

# K-3

clusters3=KMeans(3,random\_state=30).fit(wine\_norm)

clusters3

clusters3.labels\_

wine3=wine.copy()

wine3['clusters3id']=clusters3.labels\_

wine3

wine3['clusters3id'].value\_counts()