SKILL ACTIVITY NO: 5

Name:Shubham Ravasaheb Patil Date: 25/8/2021
PRN:
School: School of Data Science
Program: Machine Learning
Batch: ML 12
Module Name: Python Programming
Module Code: ML101

Title: Perform Principal component analysis on the Wine Dataset

Skills/Competencies to be acquired:

1. To gain an understanding of data and find clues from the data.
2. Assess assumptions on which statistical inference will be based.
3. To check the quality of data for further processing and cleaning if necessary.
4. To check for anomalies or outliers that may impact model.
5. Data Visualization.

Duration of activity: 1 Hour

1.What is the purpose of this activity?

Preview data.
Check total number of entries and column types.
Check any null values.
Perform Principal component analysis and perform clustering using first
3 principal component scores (both hierarchical 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 beginning who shows it has 3 clusters)

2.Steps performed in this activity.

1)EDA of the data
2)PCA on the data
3)Finding optimum number of clusters using KMeans and Agglomeretive hirerchical clustering

3.What resources / materials / equipment / tools did you use for this activity?

1)Google colab
2)jupyter notebook
3)machine learning libraries

4.What skills did you acquire?

1)PCA
2)Plotly

3)Clustring by feature extraction

5.Time taken to complete the activity? 1 hr

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('/content/wine.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39

In [ ]:
```
df.shape
```

Out[ ]:
```
(178, 14)
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Type', 'Alcohol', 'Malic', 'Ash', 'Alcalinity', 'Magnesium
', 'Phenols',
       'Flavanoids', 'Nonflavanoids', 'Proanthocyanins', 'Color',
'Hue',
       'Dilution', 'Proline'],
      dtype='object')
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Type              0
Alcohol           0
Malic             0
Ash               0
Alcalinity        0
Magnesium         0
Phenols           0
Flavanoids        0
Nonflavanoids     0
Proanthocyanins   0
Color             0
Hue               0
Dilution          0
Proline           0
dtype: int64
```

In [ ]:
```

```

There are no null values in our data

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Type             178 non-null    int64  
 1   Alcohol          178 non-null    float64
 2   Malic            178 non-null    float64
 3   Ash              178 non-null    float64
 4   Alcalinity       178 non-null    float64
 5   Magnesium        178 non-null    int64  
 6   Phenols          178 non-null    float64
 7   Flavanoids       178 non-null    float64
 8   Nonflavanoids    178 non-null    float64
 9   Proanthocyanins  178 non-null    float64
 10  Color            178 non-null    float64
 11  Hue              178 non-null    float64
 12  Dilution         178 non-null    float64
 13  Proline          178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

All the features of numerical type and our target variable is Type

In [4]:
```
x = df.drop(columns = ['Type'])
```

PCA

In [25]:
```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(x)
```

Out[25]:
```
PCA(copy=True, iterated_power='auto', n_components=None, random_st
ate=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [26]:
```
pca.explained_variance_
```

Out[26]:
```
array([9.92017895e+04, 1.72535266e+02, 9.43811370e+00, 4.99117861
e+00,
       1.22884523e+00, 8.41063869e-01, 2.78973523e-01, 1.51381266e
-01,
       1.12096765e-01, 7.17026032e-02, 3.75759789e-02, 2.10723661e
-02,
       8.20370314e-03])
```

In [41]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=3)
pca.fit(x)
```

Out[41]:
```
PCA(copy=True, iterated_power='auto', n_components=3, random_state
=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [44]:
```
pca.n_components_
```

Out[44]:
```
3
```

In [62]:
```
principle_df = pd.DataFrame(data = pca.fit_transform(x),columns = ['
pca 1','pca 2','pca 3'])
```

In [46]:
```
principle_df.head()
```

Out[46]:

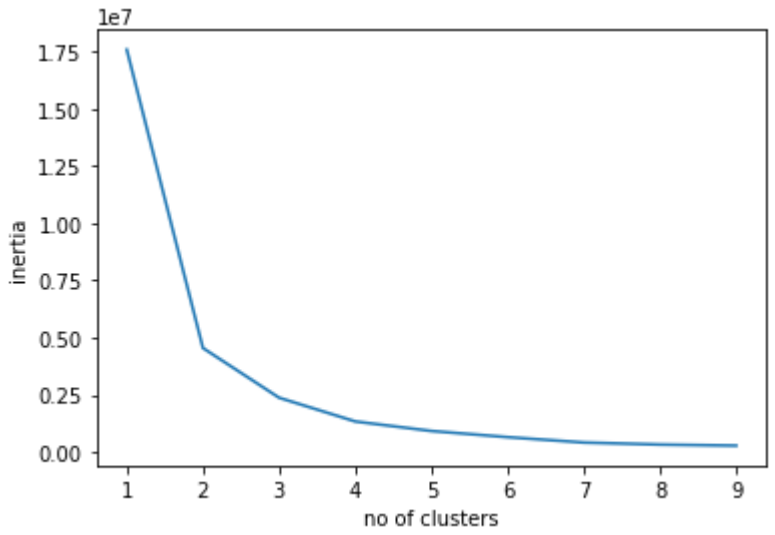
	pca 1	pca 2	pca 3
0	318.562979	21.492131	3.130735
1	303.097420	-5.364718	6.822835
2	438.061133	-6.537309	-1.113223
3	733.240139	0.192729	-0.917257
4	-11.571428	18.489995	-0.554422

Implementing K-Means

In [49]:
```
from sklearn.cluster import KMeans
n = []
inert = []
for i in range(1,10):
  model = KMeans(n_clusters = i)
  model.fit(principle_df)
  n.append(i)
  inert.append(model.inertia_)
```

In [74]:
```
plt.plot(n, inert)
plt.xlabel('no of clusters')
plt.ylabel('inertia')
plt.show
```

Out[74]:
```
<function matplotlib.pyplot.show>
```



Here we can see that after number of clusters n = 3 the plot declines linearly so we choose n=3 as number of clusters

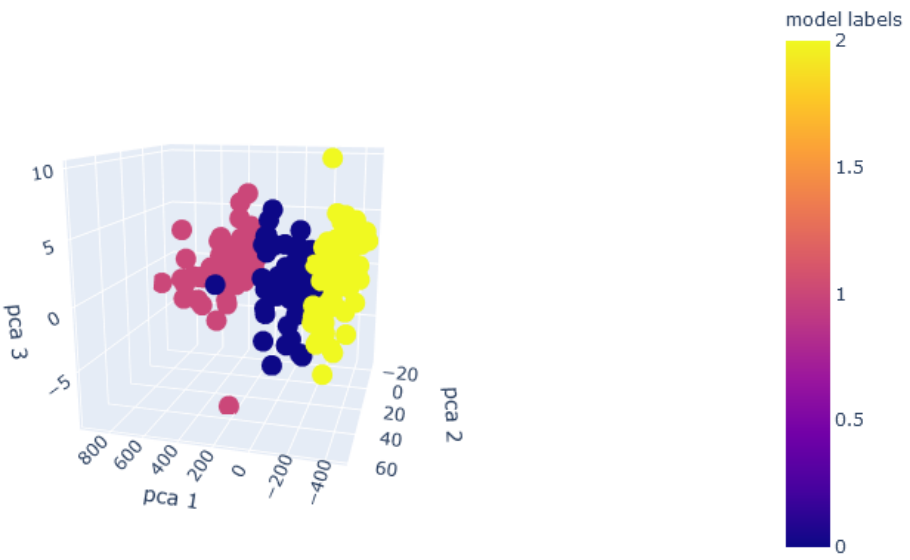
In [63]:
```
model = KMeans(n_clusters = 3)
model.fit(principle_df)
```

Out[63]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=3
00,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances=
'auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [64]:
```
principle_df['model labels'] = model.labels_
```

In [65]:
```
import plotly.express as px
```

In [ ]:
```
px.scatter_3d(principle_df,x="pca 1",y="pca 2",z="pca 3",color="mode
l labels")
```



Here we can see that the model clusters the data into 3 distinct clusters

Checking the value of number of clusters by the model and the original number of classes

```
In [ ]: df['Type'].unique()
```

```
Out[ ]: array([1, 2, 3])
```

```
In [ ]: model.n_clusters
```

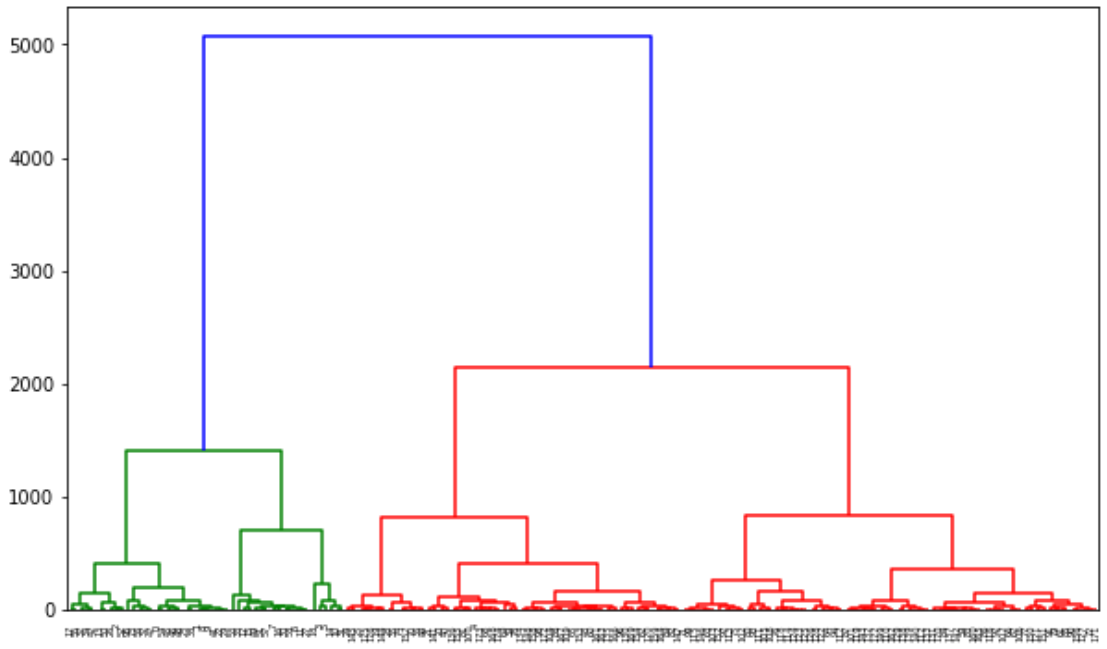
```
Out[ ]: 3
```

number of clusters from the model and the number classes in the original dataset are same

Implementing Hierarchial Agglomeretive clustering

```
In [67]: principle_df = pd.DataFrame(data = pca.fit_transform(x),columns = ['
         pca 1','pca 2','pca 3'])
```

In [68]:
```
from scipy.cluster import hierarchy
fig=plt.figure(figsize=(10,6))
den=hierarchy.dendrogram(hierarchy.linkage(principle_df,method='ward
'))
```



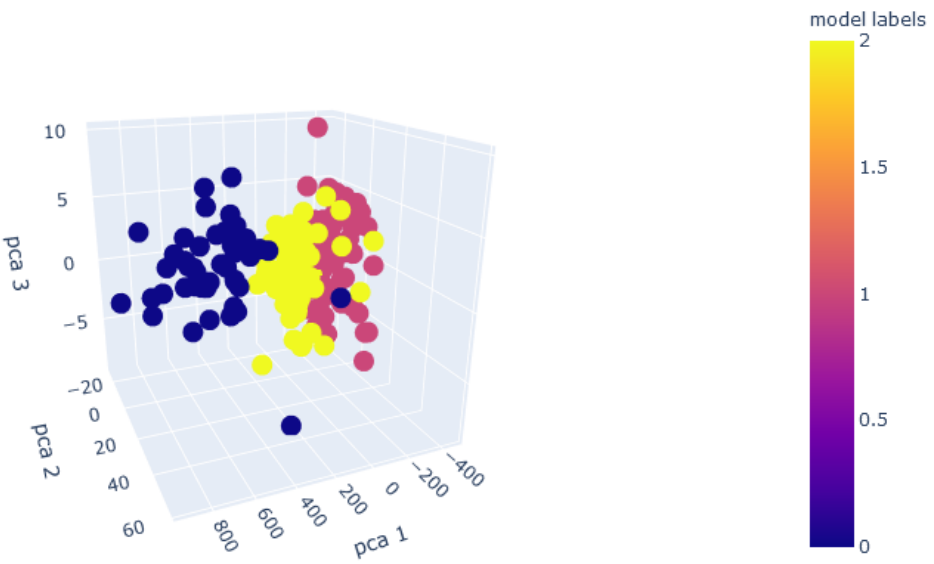
In the above dendogram below the value 2000 the branches get shorter and messy so we choose 2000 as the optimal value to divide the dendogram thus at that value we get number of cluster 3

In [69]:
```
from sklearn.cluster import AgglomerativeClustering
model=AgglomerativeClustering(n_clusters=3)
model.fit(principle_df)
```

Out[69]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='a
uto',
                        connectivity=None, distance_threshold=Non
e,
                        linkage='ward', memory=None, n_clusters=3)
```

In [70]:
```
principle_df['model labels'] = model.labels_
```

In [ ]:
```
px.scatter_3d(principle_df,x="pca 1",y="pca 2",z="pca 3",color="mode
l labels")
```



In the above plot you can see that the model is clustering the data into three different clusters neatly

Checking the value of number of clusters by the model and the original number of classes

```
In [75]: df['Type'].unique()
Out[75]: array([1, 2, 3])

In [77]: model.n_clusters
Out[77]: 3
```

number of clusters from the model and the number classes in the original dataset are same