import pandas as pd

from sklearn.preprocessing import LabelEncoder.

from sklearn.decomposition import pca

import seaborn as sns.

from sklearn.preprocessing import standardscalar.

import matplotlib.pyplot as plt.

import numpy as np

from sklearn.cluster.import K-Means.

from sklearn.decomposing import PCA

import warnings

warnings.filterwarning('ignore')

import scipy.cluster.hierarchy as shc.

from sklearn.cluster import Meanshift. Agglomerativeclustering

import statistics.

import numpy as np.

=) X = np.array([[1,2],[1,4],[1,4],[7,0],[10,2],[10,4],[10,4],[10,0]])

=) X = X[:,0]

Y = X[:,1]

sns.scatterplot(X,Y)

Step: 1

=) import pandas as pd.

data = pd.read_csv("Mall_customers.csv")

data.head()

=) km = KMeans(n_clusters=3, random_state=0)

km.fit(X)

=) km.cluster_centers

X = X[:,0]

Y = X[:,0]

sns.scatterplot(X,Y, hue = km.labels)

=) km.predict([[2,3]])

=) data.shape

=) data.columns

=) data.dtypes.

=) data.info()

=) data.value_counts()

Lab11. Shopping Mall Customer Segmentation using Clustering

Objectives

In this lab, you will detect clusters from customer data and perform analytics to understand customers who visit malls, using KMeans and Agglomerative Clustering methods.

Learning Objectives

After completing this lab, you will be able to

- Perform Skew analysis and interpretation
- Check if data normalization is required
- Build KMeans model with the required no. of clusters
- Visualize clusters based on selected features
- Select the best value for K using inertia error values and Elbow method
- Perform Cluster Analysis to understand cluster statistics
- Reduce dimensions of data using PCA and build KMeans model
- Build MeanShift clustering model on dimensions reduced data
- Create hierarchical clusters using Agglomerative Clustering
- Visualize hierarchical clusters using Dendrogram

Business Use Case

You are given the data of customers who have visited your mall. The details of customers such as age, annual income, spending score and gender are collected for each customer. They have asked you to analyse this data and give insights. This will help them to design promotion strategies, marketing models and others so as to reach out specific group of customers. Also, those group of customers will be targeted via surveys to collect further details. Based on that feedback we can decide whether the new strategy is good for that customer segment or not, even before the strategy is released.

Step1. [Understand Data]

- Using Pandas, import ***"Mall_Customers.csv"*** file and print properties such as head, shape, columns, dtype, info and value_counts.
- For example: **customers_data = pd.read_csv("Mall_Customers.csv")**

Step2. [Label encode gender]

- Genre (ie., gender) is a string, so label encode into binary

Step3. [Check for variance]

- Use describe() on your data frame and check for variance. If variance is high for float columns, you need to normalize. Otherwise, ignore

Step4. [Check skewness]

- Check if float columns are skewed. Use skew() on your data frame. If skew value is greater than 0.75, then you can perform log transformation on those skew columns.

Step5. [Pair plot]

- Draw pair plot and observe correlations.

Step6. [Build KMeans]

- Create and fit KMeans (n_clusters variable can be set with any value)
- Print label_ and cluster_centers_ values

Step7. [Scatter plot]

- Draw scatter plot between any two features with hue as "labels_". This figure shows 5 clusters.

Step 2:

=> data.drop(['Customers ID'], axis=1, Inplace = True)

=> label_encodn = LabelEncodn()

data['Genre'] = label_encoder.fit_transform(data['Genre'])

data

=> data.course.value_counts()

=> df = data.copy()

Step: 3

=> data.describe('include = 'all')

=> data.info()

=> data.var()

=> data.Cor()

Step: 4

=> data.show()

=> data.sort_values(by= ["Genre", "Age", "Annual Income (k$)",

encoding= False, Inplace = True)

Step: 5

=> sns_pairplot(data)

Step: 6

=> model. = kMeans(n_clusters = 5)

=> model.fit(data)

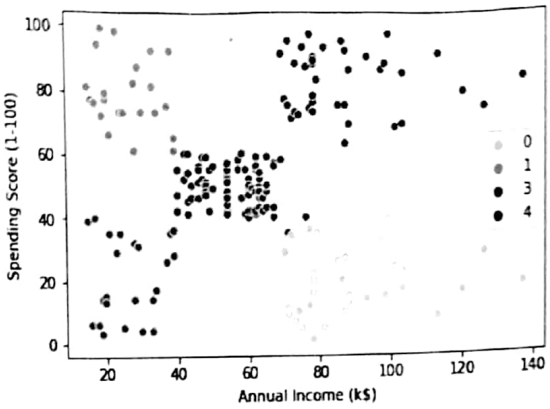
=> model.labels_

=> model.clusters_centers_

=> sns.scatterplot(data['Age'], data["Annual Income(k$)"],

plt.show()

hue = model.labels_)



Step8. [Cluster Analysis].

Now, predict cluster labels for the same data. For example,

```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(customers_data)
pred = kmeans2.predict(customers_data)
```

Now, add a new column for pred in a new dataframe, such as

```
frame = pd.DataFrame(customers_data)
frame['cluster'] = pred
```

This will create a new column to frame. That means, you have added a cluster prediction column, whose values say the cluster number to which the row belongs to. That is, that customer belongs to that cluster number.

Now, group customers based on cluster number. Remember, here we have 5 clusters from 0 to 4.

For each cluster group, print the following details.

```
Average age:  45.21739130434783
Average annual income:  26.304347826086957
Deviation of the mean for annual income:  7.893811054517766
No of customers ie shape:  (23, 5)
From those customers we have 9 male and 14 female
```

Step9. [Find the best number of clusters]

Compute inertia value as shown below

```
SSE = []
for clust in range(1,20):
    km = KMeans(n_clusters=clust, init="k-means++")
    km = km.fit(customers_data)
    SSE.append(km.inertia_)
```

Plot a line chart between cluster number and its inertia value. You will get graph as below. Can you identify the best number of clusters from this graph?

Step: 7

=> Sns.Scatterplot (data['Annual Income (k$)'], data['spending Score(1-100)'], hue = model.labels').

plt.show()

Step: 8

=> data.head()

=> x = data.iloc[:, [2,3]].values.

=> kmeans2 = kMeans(n_clusters=5, init='k-means++')

kMeans2.fit(x)

pred = kmeans2.predict(x)

=> frame = pd.DataFrame(x).

frame['cluster'] = pred.

data['Clusters'] = pred

=> frame.cluster.values_counts().

=> frame.

=> d_frame c0 = data[data['Cluster'] == 0]

d.frame C1 = data[data['cluster'] == 1]

d.frame C2 = data[data['Cluster'] == 2]

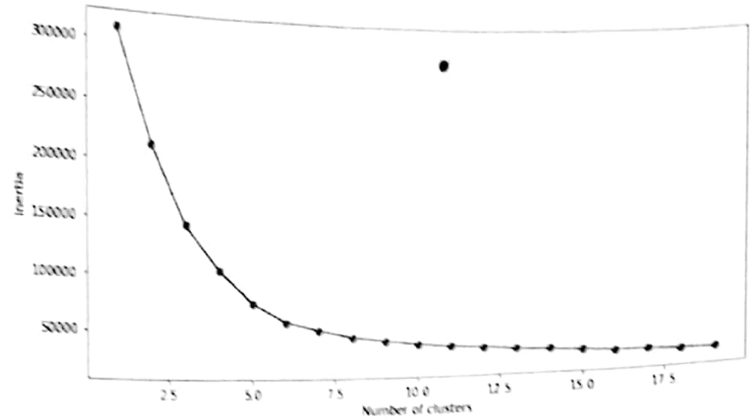
d.frame C3 = data[data['Cluster'] == 3]

d.frame C4 = data[data['cluster'] == 4]

=> Print("Average age for cluster 0:", d_frame['Age'].mean())

Print("Average Annual Income for cluster 0:", d_frame['Annual Income (k$)'].mean())

print("Deviation of the mean for annual Income:", statistics.stdev(d_frameC0['Annual Income (k$)']))

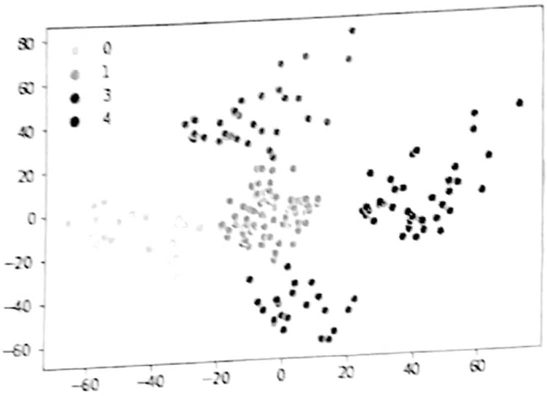


Step10. [Reduce Dimensions using PCA]

- Reduce 4 dimensions into 2 dimensions using PCA
- Create KMeans model, fit on the reduced dataset
- Print cluster_centers_ and labels_

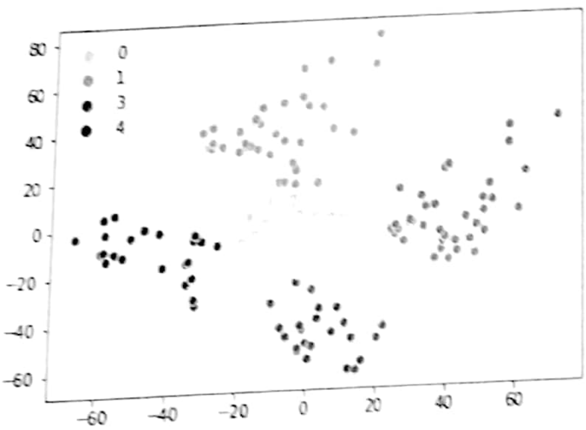
Step11. [Scatter plot]

- Draw a scatter plot between the 2 reduced dimensions, with hue as label_
- Your scatter plot may look like below



Step12. [MeanShift clustering]

- Create MeanShift clustering model and fit on the reduced data of PCA and visualize clusters on the reduced data, as shown below.



print(" No.of.customers i.e. shape of cluster 0: ", d.frame[0.shape)

print(" From those customers we have", d.frame(0). Genre.value_counts

([0], "female")

=> print(" Average age for cluster 1:", d.frame(1, Age.mean()).

print(" Deviation of the Mean for annual Income :", statistics. stdev.

(d.frame(1 (' Annual Income (k$)')])).

print ("No.of customers i.e. shape of cluster 1 :", d.frame 1. shape)

print(" From those Customers we have", d.frame(1. Genre, value_counts

()[1]

"male and", d.frame(1, Genre.value_counts()[.0], "female")

=> print(" Average age for cluster 2:", d.frame (2. Age.mean())

print(" Average Annual Income for cluster 2:", d.frame (2 ("Annual_Income

(k$) "]mean())

print (" Deviation of the Mean for annual income :", statistics. stdev.

(d.frame2 ["Annual Income (k$)"]))

=> Print(" Average age for cluster 3:", d.frame (3. Age . mean())

print (" Average Annual Income for Cluster 3:", d.frame()['Annual Income

(k$)']).mean()]

print(" Deviation of the mean. for annual Income:", Statistics. stdev.

(d.frame3 [' Annual Income (k$)']))

Print (" No. of Customers. i.e. shape of cluster 3:", d.frame() shape)

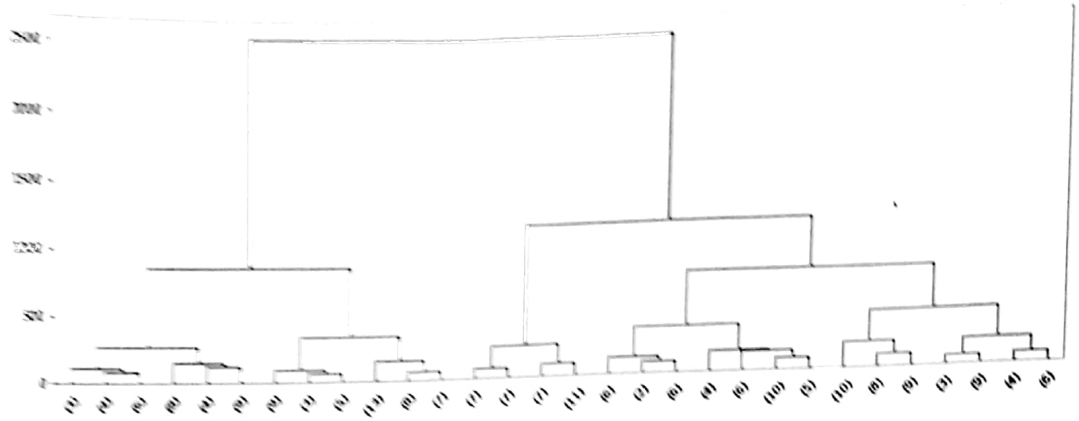
Print (" From those customers we have", d.frame (). Genre.

value_counts (3[2], " male and", d.frame Genre. value_

counts()[0] female")

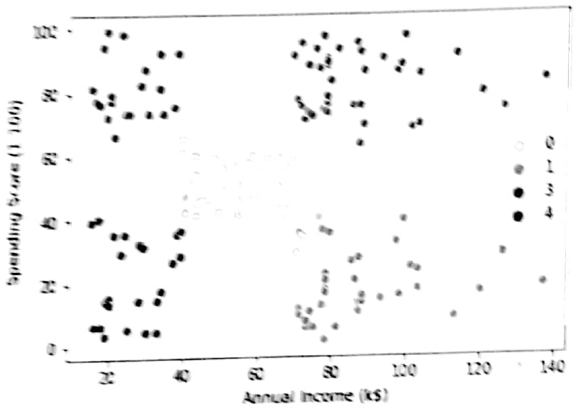
Step 13. [Predict hierarchical clusters using AgglomerativeClustering]

- Create 5 clusters using AgglomerativeClustering class. Your dendrogram will look like as below.



Step 14. [Visualize scatter plot with hue as agglomerativeclustering labels_]

- Visualize agglomerative clusters using the predicted label. Select any two features for X and Y with hue as labels_. Your scatter plot will look like below



5) print ("Average age for Cluster 4:", d.frame[4].Age.mean()).

print ("Average Annual Income for cluster 4:" d.frame[4]['Annual Income (k$)'].mean()).

Print (" Decision of the Mean for annual Income:" statistics.mode(d.frame[4]['Annual Income (k$)']))

print(" No. of customers in Shape of cluster 4:", d.frame[4].shape).

print("From these customers we have", d.frame[4].Curve.value_counts()[1] "male and", d.frame[4].Curve.value_counts().counts()[0], "female")

Step 9:

```
=> SSE = [ ]
   for cluster in range(1, 20):
      km = KMeans(n_clusters = cluster, init = "k-means++")
      km.fit(data)
      SSE.append(km.inertia)
=> plt.figure(figsize=(10,6))
=> plt.plot(np.arange(1,20), SSE, "o-")
=> plt.xlabel("Number of clusters")
=> plt.ylabel("Inertia")
   plt.show()
```

Step 10:

```
=> Y=data
=> pca = PCA(n_components=2)
   principalComponents = pca.fit_transform(Y)
   PCA_components = pd.DataFrame(principalComponents)
=> PCA_components
=> model1 = KMeans(n_clusters=5)
=> model1.fit(PCA_components)
=> model1.cluster_centers_
=> model1.labels_
```

Step 11:

```
=> sns.scatterplot(PCA_components[0], PCA_components[1],
                   hue = model1.labels)
```

Step 12:

```
=> model2 = MeanShift(bandwidth = 25)
   model2.fit(PCA_components)
```

NOTE

=) cluster_centers = model3.cluster_centers.

=) sns.scatterplot (pca_components[0], pca_components[1], hue=model3.labels).

Step: 13:

=) model3 = AgglomerativeClustering(n_clusters = 5, linkage = 'ward', compute_full_tree = True)

model3.fit(df)

=) model3.labels =

=) frame = df.copy()

frame['Cluster'] = model3.labels_

=) frame.value_counts()

=) plt.figure(figsize=(10,5))

plt.title('Dendrogram').

dend = shc.dendrogram(shc.linkage(frame[:30], method = 'ward'))

Step: 14

=) sns.Scatterplot(frame['Annual Income (k$)'], frame['Spending Score (1-100)'], hue = model3.labels_)