**Hierarchical Clustering**

Instructions:

Please share your answers filled inline in the word document. Submit Python code and R code files wherever applicable.

Please ensure you update all the details:

**Name: \_\_\_SHARANNBASAPPA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Batch Id: \_\_050121 10AM\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Topic: Hierarchical Clustering**

**1. Business Problem**

* 1. **Objective**
  2. **Constraints (if any)**

**2. Work on each feature of the dataset to create a data dictionary as displayed in the below image:**



**Using R and Python codes perform:**

**3. Data Pre-processing**

**2.1 Data Cleaning, Feature Engineering, etc.**

**4. Exploratory Data Analysis (EDA):**

**4.1. Summary**

**4.2. Univariate analysis**

**4.3. Bivariate analysis**

**5. Model Building**

**5.1 Build the model on the scaled data (try multiple options)**

**5.2 Perform the hierarchical clustering, visualize the clusters using dendrogram**

**5.3 Validate the clusters (try with different no. of clusters) – label the clusters and derive insights (compare the results from multiple approaches)**

**6. Share the benefits/impact of the solution - how or in what way the business (client) gets benefit from the solution provided.**

**Note:**

The assignment should be submitted in the following format:

* R code
* Python code
* Code Modularization should be maintained
* Documentation of the modules (elaborating on steps mentioned above)

**Problem Statement:**

1. Perform clustering for the airlines data to obtain optimum number of clusters. Draw the inferences from the clusters obtained. Refer to EastWestAirlines.xlsx dataset.

Solution: -

1. Business problem: - segmentation of data

Business Objectives: - customer satisfaction

Maximize: - clarity on flights feature clustering

Minimize: - inconvenience

Business constrains: - gain more customer

2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SI.No | Name of the feature | Descriptions | Type | Relevance |
| 1 | Id# | ID of flight | Nominal | Irrilivant ID, doesn’t Provide any usefull information |
| 2 | Balance | Flight balance | Continuous | Provides usefull information for analysis |
| 3 | Qual\_miles | Flight Qual\_miles | Continuous | Provides usefull information for analysis |
| 4 | cc1\_miles | Flight cc1\_miles | Continuous | Provides usefull information for analysis |
| 5 | cc2\_miles | Flight cc2\_miles | Continuous | Provides usefull information for analysis |
| 6 | cc3\_miles | Flight cc3\_miles | Continuous | Provides usefull information for analysis |
| 7 | Bonus\_miles | Flight bonus miles | Continuous | Provides usefull information for analysis |
| 8 | Bonus\_trans | Flight bonus trans | Continuous | Provides usefull information for analysis |
| 9 | Flight\_miles\_12mo | Flight\_miles\_12mo | Continuous | Provides usefull information for analysis |
| 10 | |  |  | | --- | --- | | Days\_since\_enroll |  | | Flight days since enroll | Continuous | Provides usefull information for analysis |
| 11 | Award? | Flight awards | Continuous | Provides usefull information for analysis |

3)Data Pre-Processing: -

data is continuous: no need of dummy variable and type casting

data has no missing values: df.isnull().sum()

- no need of any imputation method

Univ = pd.read\_excel("C:\\Users\\shilpa\\Desktop\\EastWestAirlines\_1.xlsx")

Data is to be normalized:-

def norm\_func(i):

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

return (x)

# Normalized data frame (considering the numerical part of data)

df\_norm = norm\_func(Univ.iloc[:, 0:])

df\_norm.describe()

4)

(4.1)

df.describe()

(4.2) to plot all columns box plot

for column in df:

plt.figure()

df.boxplot([column])

univariate analysis:- (found there are outliers)

Removed outliers by using replacement method:-

IQR = df["cc2\_miles"].quantile(0.75)-df["cc2\_miles"].quantile(0.25)

lower\_limit = df["cc2\_miles"].quantile(0.25)-(IQR\*1.5)

upper\_limit = df["cc2\_miles"].quantile(0.75)+(IQR\*1.5)

df\_replace3 = pd.DataFrame(np.where(df["cc2\_miles"]> upper\_limit, upper\_limit, np.where(df['cc2\_miles']< lower\_limit, lower\_limit, df["cc2\_miles"])))

sns.boxplot(df\_replace3)

df\_replace3.shape,df.shape

it is applied for each and every columns accept the nominal colum

4.3)bivariate analysis:-

Sns.pairplot(df.iloc[:,:])

5)

def norm\_func(i):

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

return (x)

# Normalized data frame (considering the numerical part of data)

df\_norm = norm\_func(df.iloc[:, 0:]) ###starts normalising from first column(indexing starts from 0)

df\_norm.describe()

# for creating dendrogram

from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch

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

# Dendrogram(all code are executed in spyder)

plt.figure(figsize=(15, 8));

plt.title('Hierarchical Clustering Dendrogram');

plt.xlabel('Index');

plt.ylabel('Distance')

sch.dendrogram(z,

leaf\_rotation = 0, # rotates the x axis labels

leaf\_font\_size = 10 # font size for the x axis labels

)

plt.show()

# Now applying AgglomerativeClustering choosing 5 as clusters from the above dendrogram

from sklearn.cluster import AgglomerativeClustering

h\_complete = AgglomerativeClustering(n\_clusters = 3, linkage = 'complete', affinity = "euclidean").fit(df\_norm)

h\_complete.labels\_

cluster\_labels = pd.Series(h\_complete.labels\_)

Univ['clust'] = cluster\_labels # creating a new column and assigning it to new column

Univ

Univ1 = Univ.iloc[:, [11,0,1,2,3,4,5,6,7,8,9,10]] ##bringing clusters column in first

Univ1

Univ1.head()

# Aggregate mean of each cluster

Univ1.iloc[:, 1:].groupby(Univ1.clust).mean()

# creating a csv file

Univ1.to\_csv("University.csv", encoding = "utf-8") ###convert from df to utf-8 formate file

import os

os.getcwd() ###location of saved file

Univ1.describe()

6)

Balance Qual\_miles ... Days\_since\_enroll Award?

clust ...

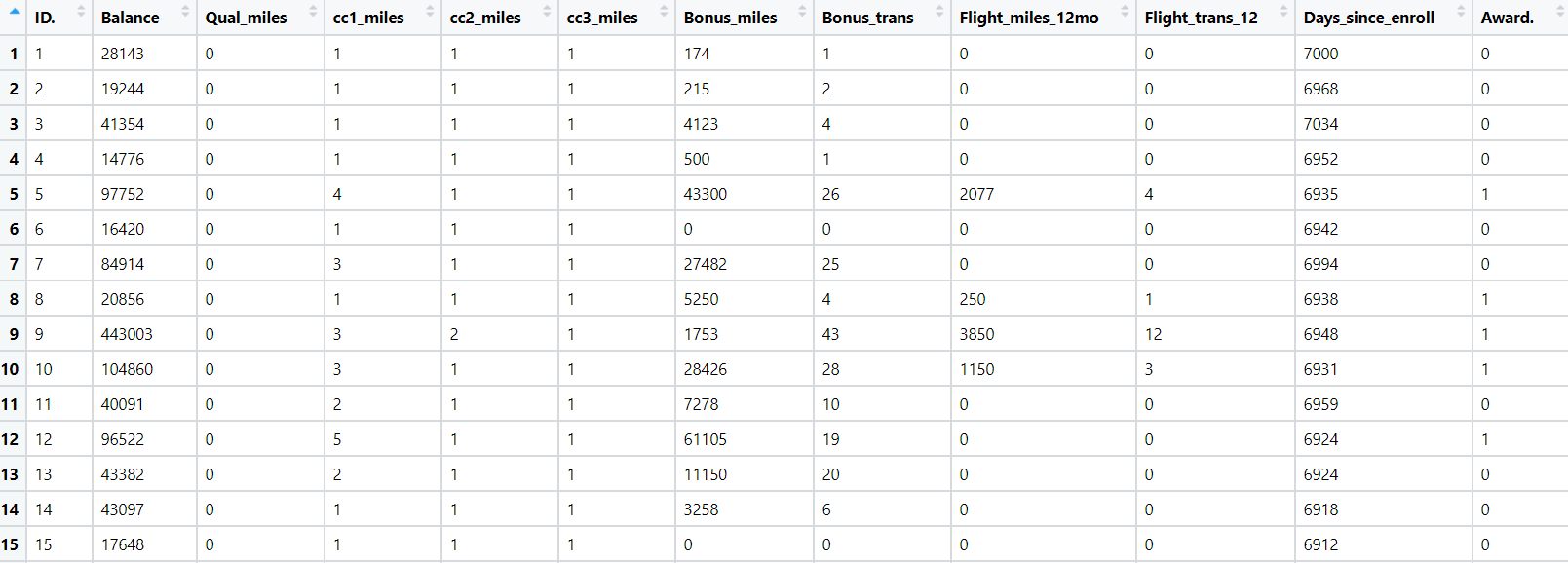
0 59791.056611 88.188836 ... 3824.887965 0.003167

1 97189.586113 239.728387 ... 4628.761743 1.000000

2 131999.500000 347.000000 ... 2200.250000 1.000000

Comment on result:- cluster 0 has less number of balance left with less award, and cluster 2 has more distance and it awarded once, even less days enrollment

Cluster 2 is best option to go



1. Perform clustering for the crime data and identify the number of clusters formed and draw inferences. Refer to crime\_data.csv dataset.

Solution:-

1)

Business problem:to catigorise crime\_data

Objectives: minimize crimes

Constrains: without affecting common peoples

2)

First column is nominal and all are continuous data type

3)data cleaning:- df.isnull().sum() >> there is no null

All are continuous, accept first column

4)exploratory data analysis[EDA]

4.1)df.discribe()

4.2)univariate analysis:

Box plot:-

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_excel("C:\\Users\\shilpa\\Desktop\\crime\_data.xlsx")

sns.boxplot(df.Rape);

plt.title("Rape");

plt.show()

IQR = df["Rape"].quantile(0.75)-df["Rape"].quantile(0.25)

lower\_limit = df["Rape"].quantile(0.25)-(IQR\*1.5)

upper\_limit = df["Rape"].quantile(0.75)+(IQR\*1.5)

df\_replace = pd.DataFrame(np.where(df["Rape"]> upper\_limit, upper\_limit, np.where(df['Rape']< lower\_limit, lower\_limit, df["Rape"])))

sns.boxplot(df\_replace)

df\_replace.shape,df.shape

from rape column outliers are removed

normalization:- def norm\_func(i):

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

return (x)

# Normalized data frame (considering the numerical part of data)

df\_norm = norm\_func(df.iloc[:, 1:]) ###starts normalising from second column(indexing starts from 0)

df\_norm.describe()

df\_norm

# for creating dendrogram

from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch

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

5)

# Dendrogram

plt.figure(figsize=(15, 8));

plt.title('Hierarchical Clustering Dendrogram');

plt.xlabel('Index');

plt.ylabel('Distance')

sch.dendrogram(z,

leaf\_rotation = 0, # rotates the x axis labels

leaf\_font\_size = 10 # font size for the x axis labels

)

plt.show()

# Now applying AgglomerativeClustering choosing 5 as clusters from the above dendrogram

from sklearn.cluster import AgglomerativeClustering

h\_complete = AgglomerativeClustering(n\_clusters = 3, linkage = 'complete', affinity = "euclidean").fit(df\_norm)

h\_complete.labels\_

cluster\_labels = pd.Series(h\_complete.labels\_)

Univ['clust'] = cluster\_labels # creating a new column and assigning it to new column

Univ

Univ1 = Univ.iloc[:, [5,0,1,2,3,4,]] ##bringing clusters column in first

Univ1

Univ1.head()

# Aggregate mean of each cluster

Univ1.iloc[:, 3:].groupby(Univ1.clust).mean()

# creating a csv file

Univ1.to\_csv("University.csv", encoding = "utf-8") ###convert from df to utf-8 formate file

import os

os.getcwd() ###location of saved file

Univ1.describe()

6)

Murder Assault UrbanPop Rape

clust

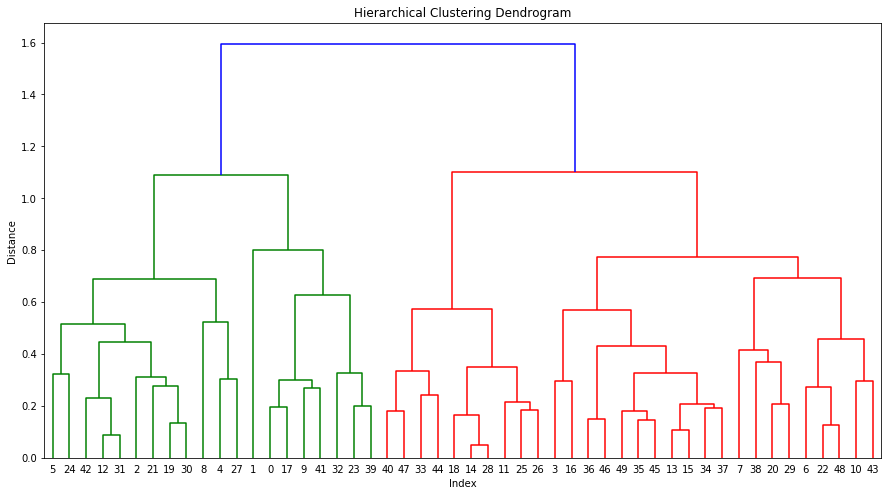
0 12.165 255.25 68.4 29.165

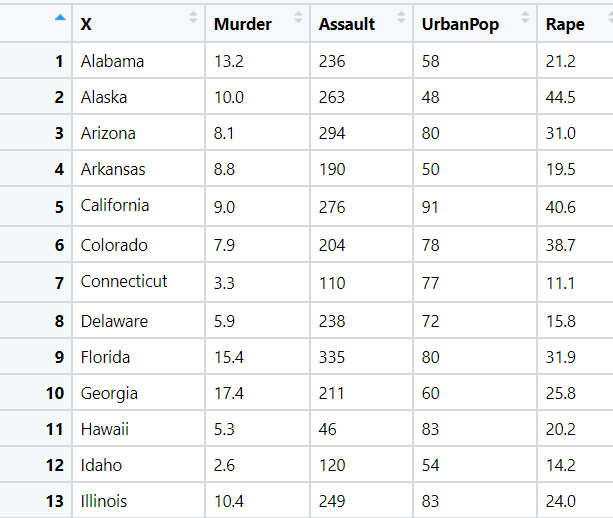
1 5.715 132.30 70.8 18.100

2 3.180 78.70 49.3 11.630

Places at cluster 0 has more number of crimes reported, and more number of polices officers to be deployed

Number 14 and 28 are having less distance so those are more similer





1. Perform clustering analysis on the telecom data set. The data is a mixture of both categorical and numerical data. It consists the number of customers who churn. Derive insights and get possible information on factors that may affect the churn decision. Refer to Telco\_customer\_churn.xlsx dataset.

Hint:

* Perform EDA and remove unwanted columns.

Solution:-

Business problem:-

Objectives:- cluster the customers to make offer

Constrain:- without disturbing customers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SI.no | Name of the feature | Descriptions | Type | Relevance |
| 1 | | Col.no:-1,4,5,8,12,24,25,26,27,28,29, #from 0 indexing | Informative information about business | Continuous | Usefull data to process to obtain constrains of the business |
| 2 | Colum no. 3,6,8,9,10,13,14,15,16,17,18,19,20,0,12,23,25  Indexing from 0 | Information is given in the form of sting | Discrete | Need to process the data to do analysis |
| 3 | 0,12,23,25 indexing from 0 | Indicates names of the item | Nominal | Not usefull for analysis purpose |

- datain = pd.read\_excel("C:\\Users\\shilpa\\Desktop\\telico\_customer\_churn.xlsx")

df = datain.drop(["Customer ID","Count","Quarter","Internet Type","Contract","Payment Method"], axis=1)

df

df = pd.get\_dummies(df1) ##dummies creation with n-1 variable, n is number if inputs

find outliers:-

IQR = df["Number of Referrals"].quantile(0.75)-df["Number of Referrals"].quantile(0.25)

lower\_limit = df["Number of Referrals"].quantile(0.25)-(IQR\*1.5)

upper\_limit = df["Number of Referrals"].quantile(0.75)+(IQR\*1.5)

df\_replace = pd.DataFrame(np.where(df["Number of Referrals"]> upper\_limit, upper\_limit, np.where(df['Number of Referrals']< lower\_limit, lower\_limit, df["Number of Referrals"])))

sns.boxplot(df\_replace)

df\_replace.shape,df.shape >>>>>> outliers are replaced by upper and lower values,

same has been done for all continuous data type columns

normalization:-

def norm\_func(i):

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

return (x)

# Normalized data frame (considering the numerical part of data)

df\_norm = norm\_func(df.iloc[:, 0:]) ###starts normalising from first column(indexing starts from 0)

df\_norm.describe()

df\_norm

* Use Gower dissimilarity matrix, In R use daisy () function.

import gower

from scipy.cluster.hierarchy import fcluster , dendrogram

gowers\_matrix = gower.gower\_matrix(telco\_data)

gowers\_linkage = linkage(gowers\_matrix)

gcluster = fcluster(gowers\_linkage , 3 , criterion = 'maxclust')

dendrogram(gowers\_linkage)

telco\_data["cluster"] = gcluster

telco\_data.iloc[: , 0:29].groupby(telco\_data.cluster).mean()



1. Perform clustering on mixed data convert the categorical variables to numeric by using dummies or Label Encoding and perform normalization techniques. The data set consists details of customers related to auto insurance. Refer to Autoinsurance.csv dataset.

Solution:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SI.no | Name of the feature | Descriptions | Type | Relevance |
| 1 | 3,9,12,13,14,15,16, | Numerical information about business | Continuous | Use full for business analysis |
| 2 | 2,4,5,6,8,21,22,23 | String format information about business | Discrete | To make use full data, pre processing is required |
| 3 | 0,1,7,10,11,17,18,19,20 | Just name of the items | Nominal | Not use full for dada analysis |

audata=pd.read\_excel("C:\\Users\\shilpa\\Desktop\\Dataset\_Assignment Clustering\\AutoInsurance.csv")

audata.drop(['Customer'] , axis= 1 , inplace = True)

new\_audata = audata.iloc[ : ,1:]

new\_audata.isna().sum()

new\_audata.columns

duplis = new\_audata.duplicated()

sum(duplis)

new\_audata = new\_audata.drop\_duplicates()

seaborn.boxplot(new\_audata["Customer Lifetime Value"]);plt.title("Boxplot");plt.show()

seaborn.boxplot(new\_audata["Income"]);plt.title("Boxplot");plt.show()

seaborn.boxplot(new\_audata["Monthly Premium Auto"]);plt.title("Boxplot");plt.show()

seaborn.boxplot(new\_audata["Months Since Last Claim"]);plt.title("Boxplot");plt.show()

seaborn.boxplot(new\_audata["Months Since Policy Inception"]);plt.title("Boxplot");plt.show()

seaborn.boxplot(new\_audata["Total Claim Amount"]);plt.title("Boxplot");plt.show()

plt.scatter(new\_audata["Customer Lifetime Value"] , new\_audata["Income"])

plt.scatter(new\_audata["Monthly Premium Autos"] , new\_audata["Months Since Last Claime"])

plt.scatter(new\_audata["Months Since Policy Inception"] , new\_audata["Total Claim Amount"])

IQR = new\_audata["Customer Lifetime Value"].quantile(0.75) - new\_audata["Customer Lifetime Value"].quantile(0.25)

L\_limit\_Customer\_Lifetime\_Value = new\_audata["Customer Lifetime Value"].quantile(0.25) - (IQR \* 1.5)

H\_limit\_Customer\_Lifetime\_Value = new\_audata["Customer Lifetime Value"].quantile(0.75) + (IQR \* 1.5)

new\_audata["Customer Lifetime Value"] = pd.DataFrame(np.where(new\_audata["Customer Lifetime Value"] > H\_limit\_Customer\_Lifetime\_Value , H\_limit\_Customer\_Lifetime\_Value ,

np.where(new\_audata["Customer Lifetime Value"] < L\_limit\_Customer\_Lifetime\_Value , L\_limit\_Customer\_Lifetime\_Value , new\_audata["Customer Lifetime Value"])))

seaborn.boxplot(new\_audata["Customer Lifetime Value"]);plt.title('Boxplot');plt.show()

IQR = new\_audata["Monthly Premium Auto"].quantile(0.75) - new\_audata["Monthly Premium Auto"].quantile(0.25)

L\_limit\_Monthly\_Premium\_Auto = new\_audata["Monthly Premium Auto"].quantile(0.25) - (IQR \* 1.5)

H\_limit\_Monthly\_Premium\_Auto = new\_audata["Monthly Premium Auto"].quantile(0.75) + (IQR \* 1.5)

new\_audata["Monthly Premium Auto"] = pd.DataFrame(np.where(new\_audata["Monthly Premium Auto"] > H\_limit\_Monthly\_Premium\_Auto , H\_limit\_Monthly\_Premium\_Auto ,

np.where(new\_audata["Monthly Premium Auto"] < L\_limit\_Monthly\_Premium\_Auto , L\_limit\_Monthly\_Premium\_Auto , new\_audata["Monthly Premium Auto"])))

seaborn.boxplot(new\_audata["Monthly Premium Auto"]);plt.title('Boxplot');plt.show()

IQR = new\_audata["Total Claim Amount"].quantile(0.75) - new\_audata["Total Claim Amount"].quantile(0.25)

L\_limit\_Total\_Claim\_Amount = new\_audata["Total Claim Amount"].quantile(0.25) - (IQR \* 1.5)

H\_limit\_Total\_Claim\_Amount = new\_audata["Total Claim Amount"].quantile(0.75) + (IQR \* 1.5)

new\_audata["Total Claim Amount"] = pd.DataFrame(np.where(new\_audata["Total Claim Amount"] > H\_limit\_Total\_Claim\_Amount , H\_limit\_Total\_Claim\_Amount ,

np.where(new\_audata["Total Claim Amount"] < L\_limit\_Total\_Claim\_Amount , L\_limit\_Total\_Claim\_Amount , new\_audata["Total Claim Amount"])))

seaborn.boxplot(new\_audata["Total Claim Amount"]);plt.title('Boxplot');plt.show()

dummy\_audata = pd.get\_dummies(new\_audata)

def norm\_func(i):

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

return (x)

audata\_norm = norm\_func(dummy\_audata)

from sklearn.cluster import AgglomerativeClustering

auto\_single = AgglomerativeClustering(n\_clusters=3 , linkage="single" , affinity="euclidean").fit(audata\_norm)

cluster\_auto\_single = pd.Series(auto\_single.labels\_)

new\_audata["cluster"] = cluster\_auto\_single

auto\_complete = AgglomerativeClustering(n\_clusters=3 , linkage="complete" , affinity="euclidean").fit(audata\_norm)

cluster\_auto\_complete = pd.Series(auto\_complete.labels\_)

new\_audata["cluster"] = cluster\_auto\_complete

auto\_average = AgglomerativeClustering(n\_clusters=3 , linkage="average" , affinity="euclidean").fit(audata\_norm)

cluster\_auto\_average = pd.Series(auto\_average.labels\_)

new\_audata["cluster"] = cluster\_auto\_average

auto\_centroid = AgglomerativeClustering(n\_clusters=3 , linkage="centroid" , affinity="euclidean").fit(audata\_norm)

cluster\_auto\_centroid = pd.Series(auto\_centroid.labels\_)

new\_audata["cluster"] = cluster\_auto\_centroid

new\_audata.iloc[: ,:23].groupby(new\_audata.cluster).mean()

import os

new\_audata.to\_csv("final\_audata.csv" , encoding="utf-8")

os.getcwd()

