# Topic: K - Means Clustering

1.) Perform clustering (Both hierarchical and K means clustering) for the airlines data to obtain optimum number of clusters.

Draw the inferences from the clusters obtained.

**Ans:**

**Data Preprocessing:**

* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables or we can factorize the variables whereas, in python we need to write label encoder () code for creation of dummy variables.
* Removing unnecessary columns and checking for the NA value

**Normalizing the data:**

* Using the normalization function, the data is normalized and the data will come under same level.

**Elbow curve to decide the k value:**

$ cluster : int [1:3999] 2 2 2 2 3 2 3 2 4 3 ...

$ centers : num [1:4, 1:11] -0.3669 -0.1372 0.4638 1.1934 -0.0598 ...

..- attr(\*, "dimnames")=List of 2

.. ..$ : chr [1:4] "1" "2" "3" "4"

.. ..$ : chr [1:11] "Balance" "Qual\_miles" "cc1\_miles" "cc2\_miles" ...

$ totss : num 43978

$ withinss : num [1:4] 4603 7304 8034 9095

$ tot.withinss: num 29036

$ betweenss : num 14942

$ size : int [1:4] 1484 1239 1108 168

$ iter : int 3

$ ifault : int 0

- attr(\*, "class")= chr "kmeans"

* The str () function gives the structure of the kmeans which includes various parameters like withinss, betweenss, etc, analyzing which you can find out the performance of kmeans.
* betweenss: Between sum of squares i.e. Intracluster similarity
* withinss: Within sum of square i.e. Intercluster similarity
* totwithinss: Sum of all the withinss of all the clusters i.e. Total intra-cluster similarity
* A good clustering, will have a lower value of withinss and higher value of betweenss which depends on the number of clusters ‘k’ chosen initially. Let us see how we can find the optimal value of ‘k’.

**Finding the optimal value of ‘k’**

An optimal value of ‘k’ is the value which gives us a converged set of clusters with minimum distortion. Greater the distortion, worse will be the clusters formed

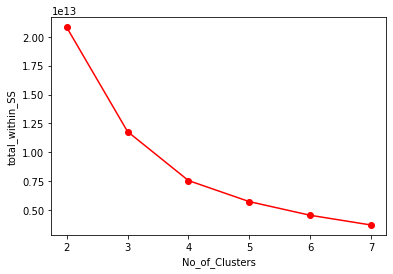
twss <- NULL

for (i in 2:8) {

twss <- c(twss, kmeans(normalized\_data, centers = i)$tot.withinss)

}

twss



This is the plot between ‘k’, the number of clusters and the ‘totwithinss’ (or distortion) for each value of k. You can see when the number of clusters is less, there is a gradual decrease in distortion but as we keep on increasing the value of k, the rate of reduction of distortion values becomes constant.

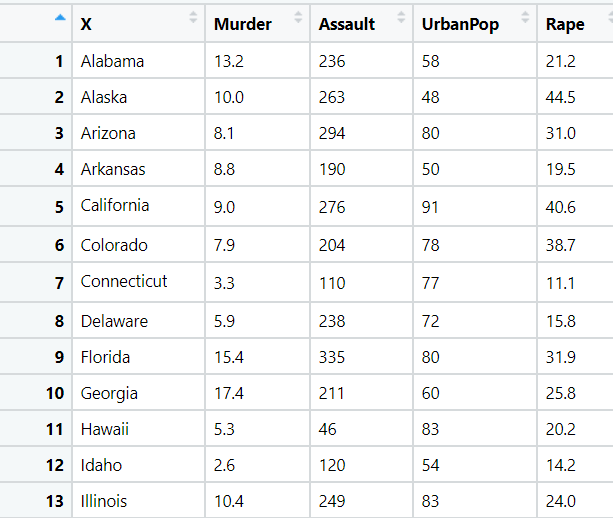
This value of k beyond which the distortion rate becomes constant is the optimal value. Here k=4.

**Inferences:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grp | **Bal** | **Qual** | **Cc1** | **Cc2** | **Cc3** | **B\_mls** | **B\_trns** | **f.mls** | **f.trnd** | **Days** | **awards** |
| **1** | 36627.94 | 97.86 | 1.26 | 1.00 | 1.00 | 4802.02 | 6.75 | 171.03 | 0.51 | 2227.85 | 0.14 |
| **2** | 59778.02 | 112.46 | 1.35 | 1.03 | 1.00 | 6254.46 | 8.35 | 265.72 | 0.81 | 5722.81 | 0.30 |
| **3** | 120343.73 | 149.27 | 3.87 | 1.00 | 1.00 | 42571.81 | 19.23 | 357.63 | 1.04 | 4770.17 | 0.688 |
| **4** | 193869.24 | 752.11 | 2.31 | 1.03 | 1.25 | 38792.83 | 27.95 | 5121.83 | 15.19 | 4690.83 | 0.80 |

* Cluster 4 has max no of balance, bonus miles, flight miles, flight transactions and also with highest percentage of awards will definitely shows they are the Premium customers.
* Cluster 1 and cluster 2 seems to be the customers comes under average category as per the above data in terms of balance, bonus miles, flight miles, flight trans and the awards comparing both Cluster 3 and cluster 4.
* Cluster 3 are the customers who are to be the next set of Premium customers compare to Cluster 1

2.) Perform K-Means Clustering for the crime data and identify the number of clusters formed and draw inferences.



**Ans:**

**Data Preprocessing:**

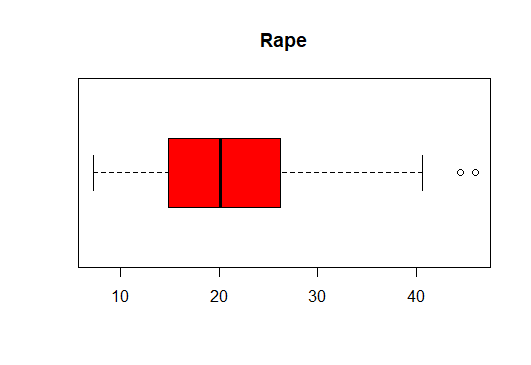
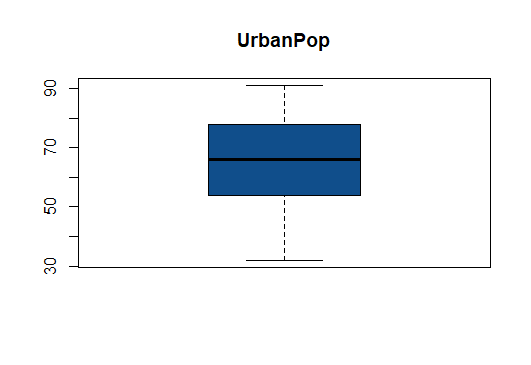
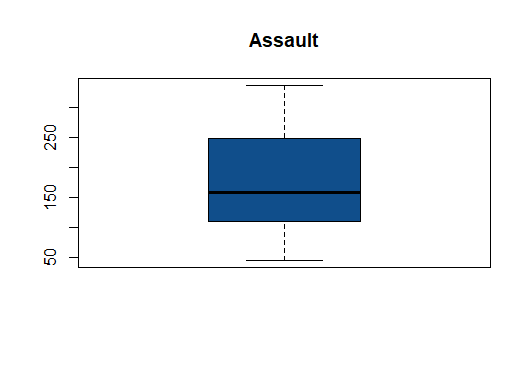
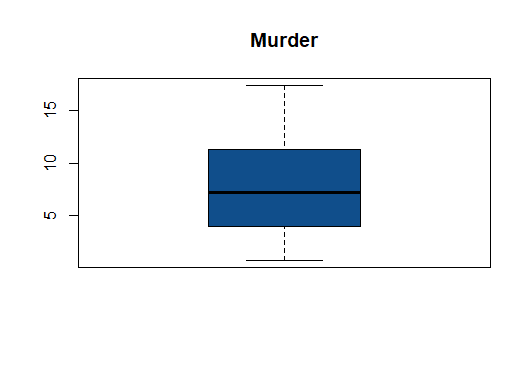
* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables or we can factorize the variables whereas, in python we need to write label encoder () code for creation of dummy variables.
* Removing unnecessary columns and checking for the NA value

**Normalizing the data:**

* Using the normalization function, the data is normalized and the data will come under same level.

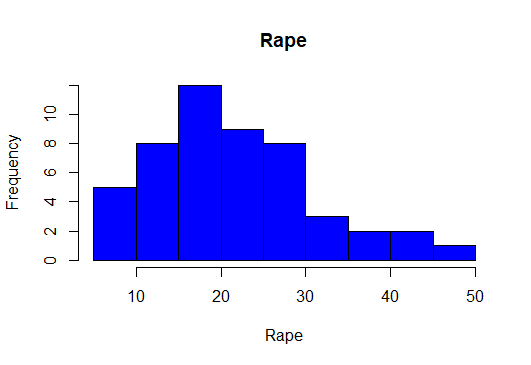
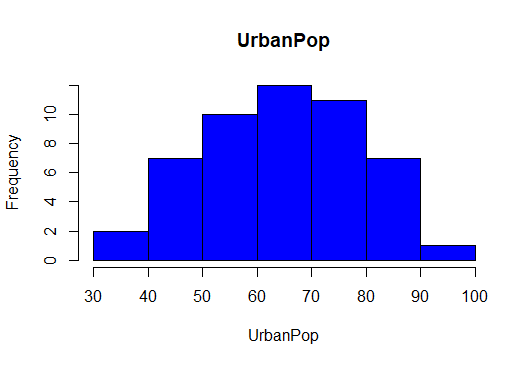
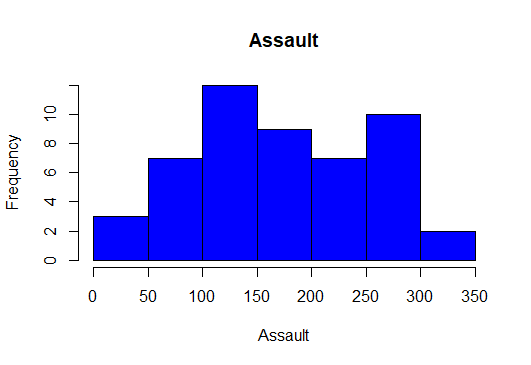
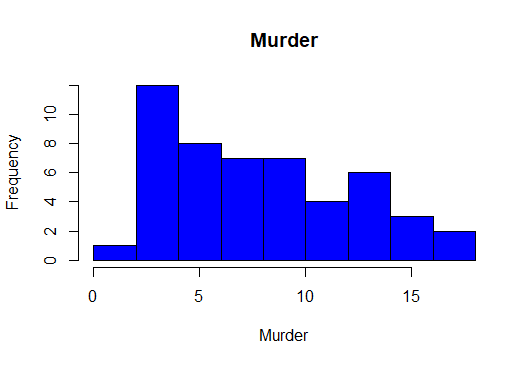
**Exploratory Data Analysis:**

**Box Plot Representation:**

****

* From the above graphical Representation, its clearly shows there exists no outliers

**Histogram Representation:**

****

* From the above histogram, its clearly shows the data is normally distributed.

**Elbow curve to decide the k value:**

$ cluster : int [1:50] 3 3 3 1 3 3 1 1 3 3 ...

$ centers : num [1:3, 1:4] -0.447 -0.962 1.005 -0.347 -1.107 ...

..- attr(\*, "dimnames")=List of 2

.. ..$ : chr [1:3] "1" "2" "3"

.. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"

$ totss : num 196

$ withinss : num [1:3] 19.6 12 46.7

$ tot.withinss: num 78.3

$ betweenss : num 118

$ size : int [1:3] 17 13 20

$ iter : int 3

$ ifault : int 0

- attr(\*, "class")= chr "kmeans"

* The str () function gives the structure of the kmeans which includes various parameters like withinss, betweenss, etc, analyzing which you can find out the performance of kmeans.
* betweenss: Between sum of squares i.e. Intracluster similarity
* withinss: Within sum of square i.e. Intercluster similarity
* totwithinss: Sum of all the withinss of all the clusters i.e. Total intra-cluster similarity
* A good clustering, will have a lower value of withinss and higher value of betweenss which depends on the number of clusters ‘k’ chosen initially. Let us see how we can find the optimal value of ‘k’.

**Finding the optimal value of ‘k’**

An optimal value of ‘k’ is the value which gives us a converged set of clusters with minimum distortion. Greater the distortion, worse will be the clusters formed

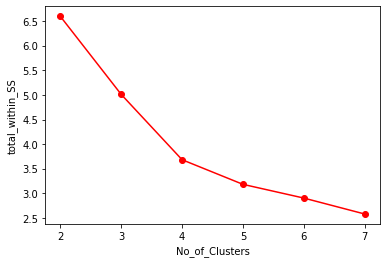
twss <- NULL

for (i in 2:8) {

twss <- c(twss, kmeans(normalized\_data, centers = i)$tot.withinss)

}

twss

****

This is the plot between ‘k’, the number of clusters and the ‘totwithinss’ (or distortion) for each value of k. You can see when the number of cluster is less, there is a gradual decrease in distortion but as we keep on increasing the value of k, the rate of reduction of distortion values becomes constant.

This value of k beyond which the distortion rate becomes constant is the optimal value. Here k=3.

**Inferences:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group** | **Murder** | **Assault** | **Urban Pop** | **Rape** |
| 1 | 5.841176 | 141.88235 | 72.47059 | 18.82353 |
| 2 | 3.600000 | 78.53846 | 52.07692 | 12.17692 |
| 3 | 12.165000 | 255.25000 | 68.40000 | 29.16500 |

* Cluster 3 has a huge crime rate with the moderate population.
* Cluster 2 has a low crime rate with the low population.
* Comparing to cluster 3, cluster 1 has high population with low crime rate

3.) Analyze the information given in the following ‘Insurance Policy dataset’ to

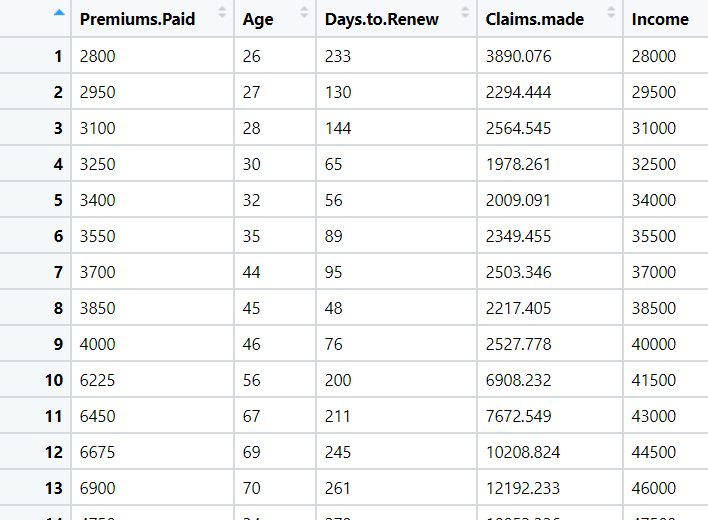
create clusters of persons falling in the same type.

The description of the attributes in the dataset are as follows: Premium Paid – Amount paid by the person

Age – Indicates the age of the person

Days to renew – Days remaining to renew the policy

Claims made – Indicates the claims already made by the person Income – Net income of the person



**Ans:**

**Data Preprocessing:**

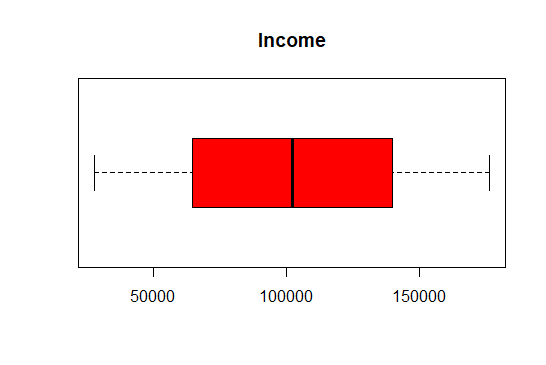
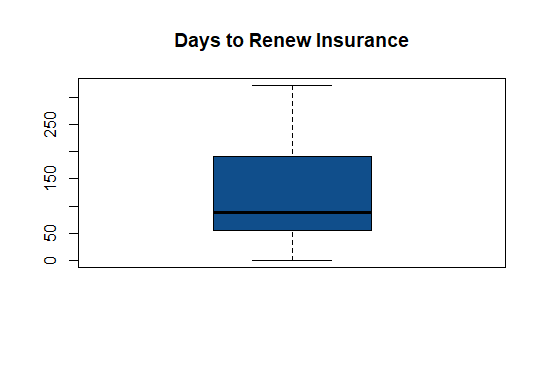
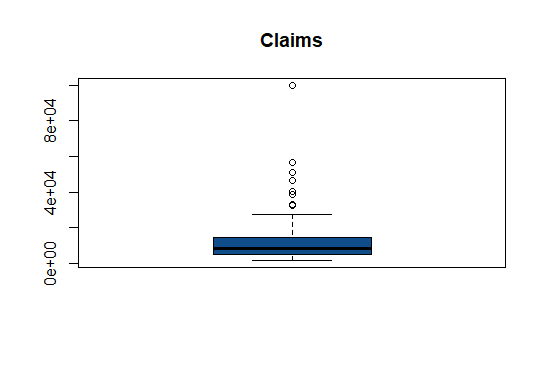
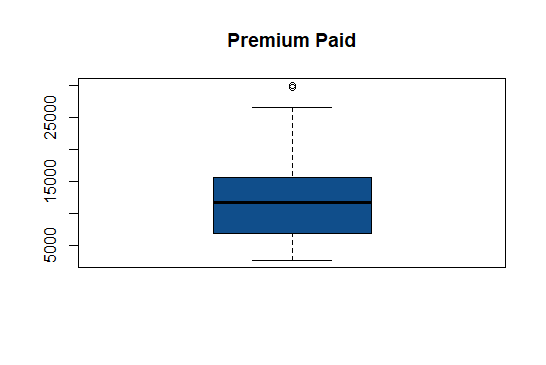
* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables or we can factorize the variables whereas, in python we need to write label encoder () code for creation of dummy variables.
* Removing unnecessary columns and checking for the NA value

**Normalizing the data:**

* Using the normalization function, the data is normalized and the data will come under same level.

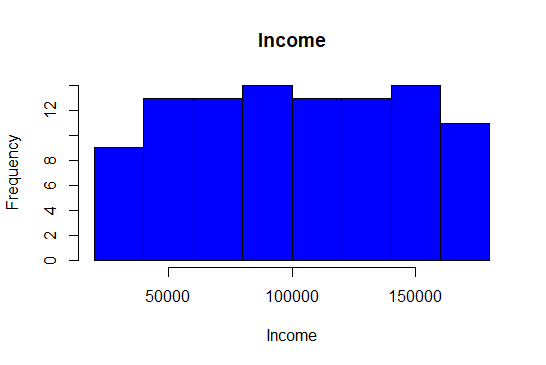
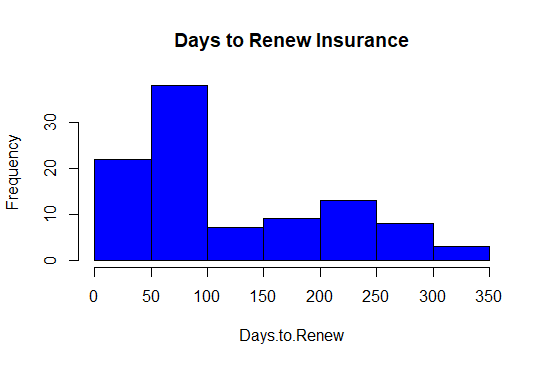
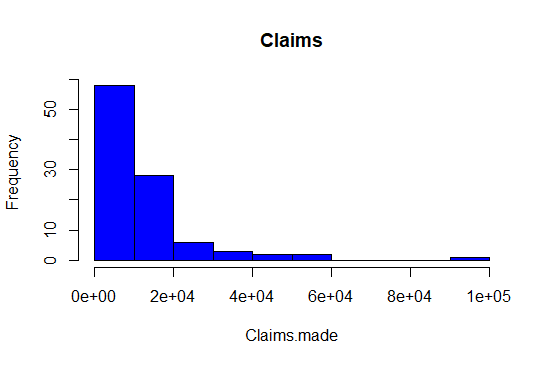
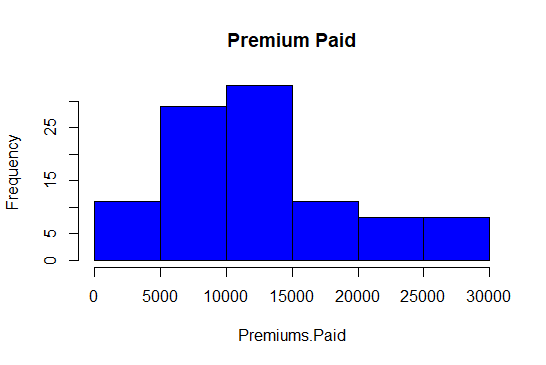
**Exploratory Data Analysis:**

**Box Plot Representation:**



* From the above graphical Representation, its clearly shows there exists outliers in premium paid and claims.

**Histogram Representation:**

****

* From the above histogram we can clearly see that the data is right skewed in case of claims made and days to renew and the data is normally distributed incase of premium paid and income.

**Elbow curve to decide the k value:**

$ cluster : int [1:100] 2 2 2 2 2 2 2 2 2 2 ...

$ centers : num [1:3, 1:5] 1.3695 -1.0027 0.0658 1.1213 -0.3723 ...

..- attr(\*, "dimnames")=List of 2

.. ..$ : chr [1:3] "1" "2" "3"

.. ..$ : chr [1:5] "Premiums.Paid" "Age" "Days.to.Renew" "Claims.made" ...

$ totss : num 495

$ withinss : num [1:3] 108.8 61.2 79.8

$ tot.withinss: num 250

$ betweenss : num 245

$ size : int [1:3] 22 33 45

$ iter : int 3

$ ifault : int 0

- attr(\*, "class")= chr "kmeans"

* The str () function gives the structure of the kmeans which includes various parameters like withinss, betweenss, etc, analyzing which you can find out the performance of kmeans.
* betweenss: Between sum of squares i.e. Intracluster similarity
* withinss: Within sum of square i.e. Intercluster similarity
* totwithinss: Sum of all the withinss of all the clusters i.e. Total intra-cluster similarity
* A good clustering, will have a lower value of withinss and higher value of betweenss which depends on the number of clusters ‘k’ chosen initially. Let us see how we can find the optimal value of ‘k’.

**Finding the optimal value of ‘k’**

An optimal value of ‘k’ is the value which gives us a converged set of clusters with minimum distortion. Greater the distortion, worse will be the clusters formed

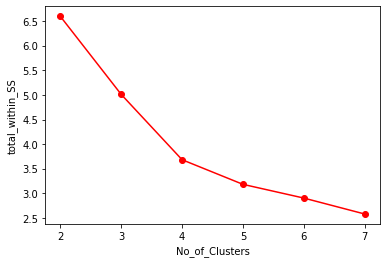
twss <- NULL

for (i in 2:8) {

twss <- c(twss, kmeans(normalized\_data, centers = i)$tot.withinss)

}

twss

****

This is the plot between ‘k’, the number of clusters and the ‘totwithinss’ (or distortion) for each value of k. You can see when the number of cluster is less, there is a gradual decrease in distortion but as we keep on increasing the value of k, the rate of reduction of distortion values becomes constant.

This value of k beyond which the distortion rate becomes constant is the optimal value. Here k=3.

**Inferences:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group** | **Age** | **Days to renew** | **Claims made** | **Income** |
| 1 | 61.68182 | 185.77273 | 29882.215 | 146227.27 |
| 2 | 40.93939 | 104.87879 | 4686.951 | 52181.82 |
| 3 | 42.28889 | 99.82222 | 9907.138 | 117466.67 |

* Cluster 1 with the age above 60+ years made highest number of claims even though their income levels are high.
* Cluster 3 with age group of 42+ years have moderate claims made with the moderate-income levels
* Cluster 2 with the age group 40 years have low income thought the claims raised are less.

**Hints:**

1. Business Problem
   1. Objective
   2. Constraints (if any)
2. Data Pre-processing

2.1 Data cleaning, Feature Engineering, EDA etc.

1. Model Building
   1. Partition the dataset
   2. Model(s) - Reasons to choose any algorithm
   3. Model(s) Improvement steps
   4. Model Evaluation
   5. Python and R codes
2. Deployment

4.1 Deploy solutions using R shiny and Python Flask.

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

**Note:**

1. For each assignment the solution should be submitted in the format
2. Research and Perform all possible steps for improving the model(s) accuracy Ex: Feature Engineering, Hyper Parameter tuning, etc.
3. All the codes (executable programs) are running without errors
4. Documentation of the module should be submitted along with R & Python codes, elaborating on every step mentioned here