Dissertation On

Prediction of Patients' Mortality Rate Discharging in Odd and Even Hours Using Gaussian Mixture Model (GMM) and K-Means Algorithm

Thesis submitted towards partial fulfillment of the requirement for the degree of

M.Tech in Information Technology (Courseware Engineering)

Submitted by

KUNDAN MUKHERJEE

EXAMINATION ROLL NO.: M4CWE22006

UNIVERSITY REGISTRATION NO.: 154488 of 2020-21

Under the guidance of

Dr. SASWATI MUKHERJEE

School of Education Technology

Jadavpur University

Course affiliated to

Faculty of Engineering and Technology

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Prediction of Patients' Mortality Rate Discharging in Odd and Even Hours Using Gaussian Mixture Model (GMM) and K-Means Algorithm"** is a bonafide work carried out by **Kundan Mukherjee** under supervision and guidance of **Dr. Saswati Mukherjee** for partial fulfillment of the requirements for the degree of **M.Tech in Information Technology (Courseware Engineering)** in **School of Education Technology**, during the academic session 2020-2022.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr. Saswati Mukherjee

Supervisor

Jadavpur University

Kolkata:700032

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Director

School of Education Technology

Jadavpur University

Kolkata:700032

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dean-FISLM

Jadavpur University

Kolkata:700032

CERTIFICATE OF APPROVAL

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Committee of final examination _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

for evaluation of thesis _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

DECLARATION OF ORIGINALITY AND COMPLLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of him **M.Tech in Information Technology(Courseware Engineering)** studies during academic session 2020-2022.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Kundan Mukherjee

Examination Roll Number: M4CWE22006

Thesis Title: Prediction of Patients' Mortality Rate Discharging in Odd and Even Hours Using Gaussian Mixture Model (GMM) and Means Algorithm.

_ _

Signature **Date**

ACKNOWLEDGEMENT

I feel fortunate while presenting this dissertation at **School of Education Technology, Jadavpur University, Kolkata,** in the partial fulfillment of the requirement for the degree of **M.Tech in Information Technology (Courseware Engineering).**

I hereby take this opportunity to show my gratitude towards my mentor, **Dr. Saswati Mukherjee,** who has guided and helped me with all possible suggestions, support, aspiring advice and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Dr. Ranjan Parekh, Director of School of Education Technology** for his timely encouragement, support and advice. I would also like to thank **Prof. Matangini Chattopadhyay** and **Mr. Joydeep Mukherjee** for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from M.Tech in Information Technology(Courseware Engineering) and Master in Multimedia Development. I do wish to thank all the departmental support staffs and everyone else who has different contributions to this dissertation.

Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieve this height.

Date: 27.06.2022

Place: Jadavpur, Kolkata

Kundan Mukherjee

M.Tech in Information Technology (Courseware Engineering)

School of Education Technology

Jadavpur University

Kolkata:700032

Contents

Executive Summary

The dissertation work is implemented with K-means clustering and Gaussian Mixture Model clustering algorithm. Both algorithms are unsupervised learning algorithm.

The users do not need to supervise the model. Unsupervised learning allows the model to work on its own to discover undetected pattern and information. Unlabeled data is dealt with it.

The proposed approach takes into account data from different patients. The main goal is to come up with a method for clustering these real time data accurately and efficiently. Objective of the dissertation is to predict the mortality rate of patients who have been discharged in odd and even hours. It has used two different unsupervised machine learning algorithms i.e. **K-Means** Clustering algorithm and **Gaussian Mixture Model (GMM)** algorithm to cluster the data.

The proposed approach is evaluated using various evaluation score methods namely Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, Fowlskes Mallows Index and Rand Score. It is observed that Rand score has better accuracy than other.

1. Introduction

1.1. Overview

Clustering is used to find structure in a dataset where class labels are unavailable. It is fundamentally a data exploration technique. There are several definitions of cluster and several ways to generate it. A cluster is defined as a group of similar entities and it is different from other groups.

An alternative definition is based on points. A group of points is defined as a cluster if the distances among the points of that group are minimum but the distances with other points of another set are maximum.

There is another definition of cluster where it is defined with density. Two clusters are separated due to density variance. High density regions are separated with low density regions.

From above discussion it is observed that cluster analysis is used for model classification of various kinds based on their internal and external distances. So, the dots or points of various kinds are classified or clustered by using distance function, so that similar type of entities are fallen in same group and different group be different. The pattern similarity is measured by using distance function of dots of entities.

K-means algorithm and Gaussian Mixture Model algorithm are used in this thesis.

1.2. Problem Statement

Prediction of patients' mortality rate discharging in odd and even hours using Gaussian Mixture Model (GMM) and K-Means algorithm.

1.3. Objectives

The objectives are:

i. Prediction of patients' mortality rate discharged in odd and even hours data using K-means algorithm and Gaussian Mixture Model (GMM) algorithm.

ii. To validate the model externally with synthetic dataset generated by using Gretel's artificial intelligence algorithm.

2. Background Concept

Some key concepts will be discussed to understand the details of the algorithm that have been implemented for the prediction of patients' mortality rate in ITU as well as normal beds. Preprocessing technique and clustering technique will be discussed.

2.1. K-Means Clustering

There are two types of K-Means clustering i.e.

(a) Static K-Means clustering

K-means algorithm is a popular clustering technique where Euclidean distance is used to separate a group of points in a manner so that the summation of distances among each point and the centroid of the individually assigned cluster is minimum.

It is a iterative two steps process. These two steps are repeated until the solutions of two consecutive iteration produce same class assignments.

It is observed in the initialization phase of K-means algorithm that k random points of the dataset are selected arbitrary as the centroids of arbitrary classes.

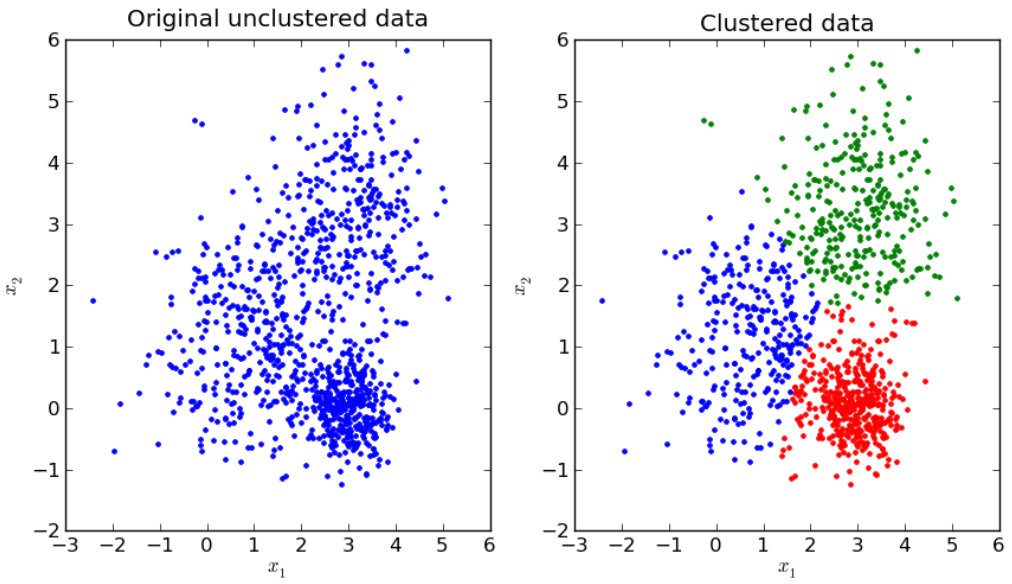


Fig:1, Random selection of 3 cluster centroids for initial seeding

After initialization two step iterative procedure starts. The first step is called "Assignment" step. In this step, every points in the dataset is assigned to one of the separated classes. Each point is assigned to the class with its nearest centroid to that point. The distance is based on Euclidean distance.

This step is expressed mathematically where J is the mathematical term that is wished to minimize;

$$J = arg_j min\{\sum_{j=1}^{k}\sum_{i=1}^{n}|x_i - c_j|^2\}\ldots\ldots(1)$$

In the above equation [1], the class j is assigned an individual point x to is the class for which the Euclidean distance between point x and the k class centroids is minimized.

The second step is "Update " step. In this step it is observed that all the assigned points of k separate classes start calculation for a new centroid position for that class. The centroid of a class is the mean position of that class. It can be expressed mathematically as,

$$C_j = \frac{\sum x_i}{n_j}|_{j=1:k}\ldots\ldots(2)$$

Where C_j is the centroid of cluster j, x_j is the set of points belonging to class j, and n_j is the number of points assigned to cluster j. This operation is performed for all of the k classes, i.e. j=1:k.

After update step the previous assignment step based on Euclidean distance is no longer valid as the update step moves the centroids.

These two steps i.e. assignment step and update step are repeated iteratively and eventually outcome two consecutive iterations become same.

When this case happens, the standard K-means algorithm is halted and original dataset is separated successfully into k separate classes or clusters.

(b) Dynamic K-Means clustering

Standard K-means algorithm has a weakness that the users set the number of clusters arbitrarily which results in a suboptimal solution.

To overcome the problem, Ahamad & Hareesha [4] proposed an approach to determine the optimal number of classes, k, to assign data points to arbitrary sets.

The optimal value of k is determined by "inter" cluster distance and "intra" cluster distance.

Inter cluster distance is measured between separate cluster centroids and intra cluster distance is measured between all the points of a cluster and the centroid of that cluster.

For dynamic k-means algorithm, first standard K-means algorithm is performed first for two clusters (k=2), which is the smallest number of possible classes, and then increase the number of clusters by 1 i.e. k=k+1 clusters consecutively until certain conditions for inter cluster distances and intra cluster distances are met. The inter cluster distance is measured with Euclidean distance i.e. distance between each possible pair of clusters in the set. The number of combinations of a pair of clusters is determined by the well-known "handshake" problem, i.e.

$$N_{pairs} = \frac{k(k-1)}{2} \ldots\ldots (3)$$

Inter Cluster distance is defined as:

$$D_{inter}(i, ii) = min\{|c_i - c_{ii}|^2\}|_{i=1:k-1, ii=k+1:k} \ldots\ldots (4)$$

Where i is the index of a single cluster's centroid, and ii is the index of the cluster which is being compared. Note that the assignments of i and ii in above equation [4] ensure that i will not be equal to ii, so that a single cluster is never compared to itself. Because the clustering process aims to separate the original dataset into a discrete number of well-defined classes or clusters, a good

separation is achieved when distances between centroids are maximum. Because multiple distances are measured, the worst case is selected, i.e., the minimum observed inter-cluster distance to compare the separation achieved with different k's. The completion criteria for inter cluster distance is:

$$D_{inter,k} > D_{inter,k-1} \ldots\ldots (5)$$

Intra cluster distance is measured about the compactness of the points within a specific cluster. Standard deviation is a commonly used to measure the separation of data points within a cluster, and is defined intra cluster distance.

$$D_{intra(j)} = max \sqrt{\frac{1}{n-1} \sum_{i=1}^{n_j} (x_i - c_j)}|_{j=1:k} \ldots\ldots (6)$$

In a successfully separated dataset, the variation within individual classes should be minimum. In the same fashion as before, the worst of all cases is evaluated, i.e., the maximum of the k intra cluster distances observed in the clustered dataset. Separation of the data is improved by increasing k when

$$D_{intra,k} > D_{intra,k-1} \ldots\ldots (7)$$

The dynamic K-means algorithm is said to be completed when criteria of both the inter distances and intra distances are observed to be true at the same time, i.e. when the separation between separate clusters is increased and the compactness of individual clusters is improved by increasing k. Used in combination, the constraints on inter cluster distances and intra-cluster distance attempt to find a number of clusters which improves the separation between separate clusters while packing individual clusters as closely as possible.

2.2. Gaussian Mixture Model (GMM) Clustering

Gaussian Mixture Model is based on Gaussian distribution which is a very popular.

Gaussian distribution is a very important distribution for continuous variables in the probability theory. This is known as the law of errors and the most popular probability distribution in statistics. Let, x is a single real valued variable. In Gaussian distribution, μ is the mean of the variable and the standard deviation is σ. It is defined as:

$$\Phi(x|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})} \ldots\ldots(8)$$

Where θ = {μ, σ}.

The graph of the Gaussian distribution is shown below:

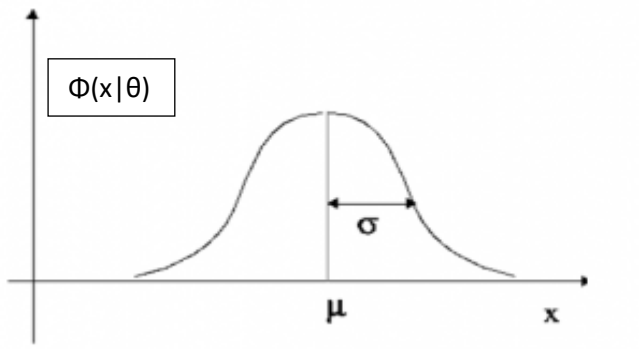


Fig:2, Graph of the Gaussian distribution for the mean μ and standard deviation σ

From the above equation [8] it is observed that the Gaussian distribution needs to satisfy the two requirements for a valid probability density.

$$\Phi(x|\theta) > 0 \ldots\ldots(9)$$

And,

$$\int_{-\alpha}^{\alpha} \Phi(x|\theta)dx = 1 \ldots\ldots(10)$$

Within the Gaussian distribution defined, the average value of x is

$$\mathbb{E}[x] = \int_{-\alpha}^{\alpha} \Phi(x|\theta)xdx = \mu \ldots\ldots(11)$$

And the variance of x is

$$\text{var}[x^2] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \int_{-\alpha}^{\alpha} \Phi(x|\theta)x^2dx - \mu^2 = \sigma^2 \ldots\ldots(12)$$

Gaussian Mixture Models (GMM) is generally used for clustering and classification. It has a group of K components. Each component is modeled by a Gaussian distribution parameterized by its mean vector and covariance matrix. A prior probability associated with each of the components. It can be written the prior probability of the k^{th} component as $\pi(k)$ $\forall k \in \{1, 2, ..., K\}$. The probability density function(pdf) of the kth component is written as $p(x|k)$ and it is just the probability density function of a Gaussian distribution $N(\mu_k, C_k)$ parameterized by mean μ_k and covariance matrix C_k. The sum of prior probabilities of all the components must be 1. Thus,

$$\sum_{k=1}^{K} \pi(k) = 1 \ldots\ldots (13)$$

For such a mixture, the joint-density is:

$$p(x) = \sum_{k=1}^{K} \pi(k) p(x|k) \ldots\ldots (14)$$

Expectation Maximization

The EM algorithm proceeds by iterating following two steps:

(a) Expectation Step (E-Step)

Expectation Maximization (EM) algorithm is used to obtain GMM parameters. The parameters of a K components GMM are (π_k, μ_k, C_k) $\forall k \in \{1, 2, ..., K\}$. Let the dataset be $\{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$. Using Bayes theorem, the probability that a point x_i belongs to component k is given by:

$$p(k|x_i) = \frac{p(x_i|k)\pi(k)}{p(x_i)} \ldots\ldots (15)$$

(b)Maximization Step(M-Step)

Here, $p(x_i)$ is given by joint density. If the log likelihood of the GMM for the training data is written and is differentiated it w.r.t. the parameters, a set of coupled equations for the parameters are produced. The equations of the parameters for the k^{th} component are given by:

$$\pi(k) = \frac{1}{n} \sum_{i=1}^{n} p(k|x_i) \ldots\ldots (16)$$

$$\mu_k = \frac{\sum_{i=1}^{n} p(k|x_i) x_i}{\sum_{i=1}^{n} p(k|x_i)} \ldots\ldots (17)$$

$$C_k = \frac{\sum_{i=1}^{n} p(k|x_i)(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{n} p(k|x_i)} \ldots\ldots(18)$$

K data points are selected arbitrarily to use as the initial means (μ_k). Thiese K data points must be distinct. The initial covariance matrices for all the components are diagonal where the (j, j)th element of each matrix is the variance of the j^{th} feature of the data. Next, $p(x_i|k)$ and $\pi(k)$ are computed. Now from the Expectation step, it is observed that the probability of each point belonging to each cluster ($p(k|x_i)$ $\forall i \in \{1, 2, ..., n\}$ and $\forall k \in \{1, 2, ..., K\}$) is calculated using E-Step equation [15]. In the Maximization step, the parameters of the GMM is calculated using equations of M-Step[17]. In this way, by iteration of the Expectation step and the Maximization step, the value of the log likelihood of the data converges.

2.3. Evaluation Score Methods

Evaluation of a machine learning model is very important. It indicates the authenticity of the model. As the dissertation is on unsupervised learning algorithm. So, confusion matrix based evaluation is not appropriate. Thus evaluation score methods are used.

This dissertation uses 5 evaluation score methods namely Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, Fowlskes Mallows Index and Rand Score.

Silhouette Coefficient

The Silhouette Coefficient is used to measure the similarity of an object to its own cluster. This coefficient ranges from −1 to +1. The high value indicates that the object is grouped to its own cluster very well.

Calinski-Harabasz Index

The **Calinski**-Harabasz index also known as the Variance Ratio Criterion. This is defined as the ratio of the sum of intra clusters dispersion and of inter cluster dispersion for all clusters. If the score is high then the performance in good.

Davies-Bouldin Index

The Davies-Bouldin index measures clustering algorithms. It is used to evaluate the goodness of split by an unsupervised learning algorithm for a given number of clusters. This index ranges from 0 to 1. If the score is high then it indicates that the cluster is bad.

Fowlskes Mallows Index

Fowlskes Mallows Index is also an evaluation method. It is also like Davies-Bouldin index. If the score is high then it indicates that the cluster is bad. Its range is 0 to 1.

Rand Score

Rand score is used to measure similarity among clusters. It is using permutation model of clustering to check the similarity using pair wise comparison among clusters and its range is from 0 to 1. 0 means clusters do not agree on clustering of any pair of entities and 1 means clusters agree on clustering of every pair of entities.

3. Literature Survey

Wang et al. [1] proposed an unsupervised deep learning algorithm. It introduced a new principle on Gaussian Mixture Model (GMM) based deep modeling. In maximizing step, the first target enabled GMM to achieve the best possible modelling of the data representations and each Gaussian component corresponded to a compact cluster. The separability of Gaussian components enhanced by maximizing the second term and so the inter cluster distance increased.

Jiang et al.[2] proposed a new network structure for both representation learning and Gaussian Mixture Model (GMM) based representation modelling. The data representation towards associating Gaussian centers of Gaussian component had been adjusted.

Li et al. [3] proposed an improved K means clustering algorithm by combining largest minimum distance algorithm and traditional K means algorithm. To determine the initial focal point the shortcoming of the traditional K means algorithm solved by this improved algorithm. Traditional K means algorithm has two major disadvantages i.e. greater dependence to choose the initial focal point and the other is easy to be trapped in local minimum. The proposed improved algorithm solved these two problems.

Shafeeq et al. [4] proposed a Dynamic K means clustering technique to cluster the data points efficiently. The proposed method worked for both the cases i.e. for advanced known number of clusters as well as unknown number of clusters. The user had the flexibility either to fix the number of clusters or input the minimum number of required clusters. In the former case it worked same as K means algorithm. In the latter case the algorithm computed the new cluster centers by incrementing the cluster counter by one in each iteration until it satisfies the validity of cluster quality.

Fahim et al. [5] proposed an idea that makes K means more efficient, especially for dataset containing large number of clusters. Since, in each iteration, the K means algorithm computed the distances between data point and all centers, this was computationally very expensive especially for huge datasets. Why not benefit from previous iteration of K means algorithm? For each data point, the distances to the nearest cluster were kept. At the next iteration, the distance to the previous nearest cluster computed. If the new distance was less than or equal to the previous distance, the point stays in its cluster, and there was no need to compute its distances to the other cluster centers. This saved the time required to compute distances to k−1 cluster centers. The proposed idea came from the

fact that the K means algorithm discovered spherical shaped cluster, whose center was the gravity center of points in that cluster, this center moved as new points were added tour removed from it. This motion made the center closer to some points and far apart from the other points, the points that became closer to the center will stay in that cluster, so there was no need to find its distances to other cluster centers. The points far apart from the center may change the cluster, so only for these points their distances to other cluster centers calculated, and assigned to the nearest center.

Desautels et al. [6] proposed a transfer learning-based machine learning algorithm. They achieved good discrimination. Unplanned readmission prediction can be used to target resources more efficiently.

Flores et al. [9] proposed a method that evaluate heterogeneous and missing clinical data using unsupervised machine learning algorithm to detect important artery disease.

4. Proposed Methodology

The proposed work is predicted mortality rate of patients who are discharged in odd hours and even hours. The odd and even hour time slots are designed as: Even hour denotes 09:00 am to 07:00 pm time slot and odd hours denotes 07:00 pm to 09:00 am time slot of next day.

It is observed that mortality rate of the patients who are discharged in odd hours is higher than the patients who are discharged in even hours.

There are existing literature on supervised learning that deals with this problem. However, very few works have been done with unsupervised learning.

However, the accuracy score for K-means algorithm is not appropriate. So, combined Gaussian mixture model with K-means provides better Rand score.

Google colab platform is used to implement the proposed work.

Firstly, the raw dataset is collected from a hospital in Kolkata. Then various imputation algorithms are applied on the raw dataset to make it in a structured form. After the dataset is prepared, unsupervised machine learning algorithm is applied to predict required outcome. So, before training the dataset, various preprocessing steps are executed so that the dataset can be made ready to fit in into the processing steps. Once the dataset is processed the final outcome is produced. The entire process of the proposed work is depicted in figure 3.

The pipe line is depicted as below diagram:

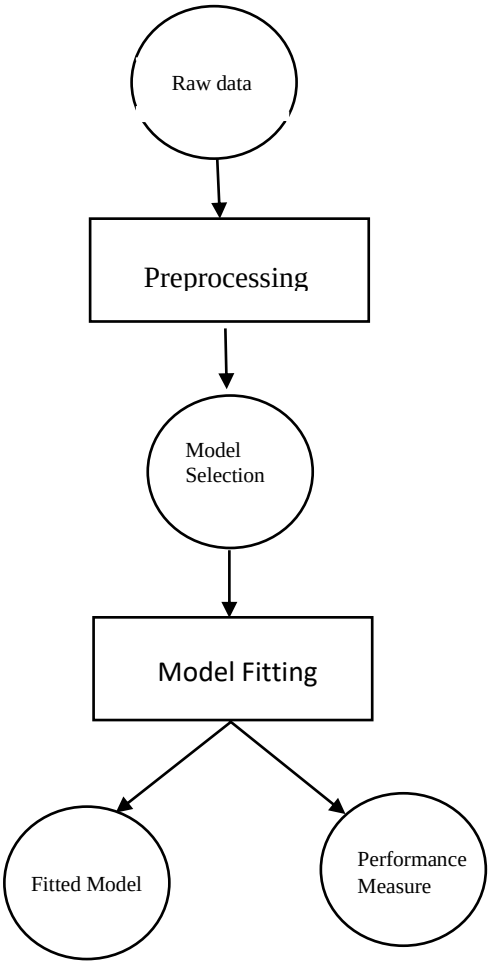


Fig:3. Model pipe line

Hence from above figure [3] it is seen that the raw data is followed by preprocessing steps.

Before applying preprocessing steps, one should study the dataset very well like size of the dataset, types of the columns etc.

It is observed that there are 4783 rows and 18 columns in the dataset. All columns are not important for the model so some columns are kept and some are discarded. The columns which are considered namely 'Years', 'A4', 'PMR', 'Exptd ITU LOS', and 'ITU_STAY'. It is necessary to check if there is any missing data and type of data whether categorical or numerical. If the data is categorical it is converted to numerical.

For missing data, imputation is also applied.

It is observed that gender of the patients is written as ‘Male’ and ‘Woman’. So, it has changed and marked ‘Male’ as 0 and ‘Woman’ as 1.

All columns namely 'Years', 'A4', 'PMR', 'Exptd ITU LOS', and 'ITU_STAY' are combined as feature dataset.

The target feature is named as 'ITU-OUTCOME'. It is observed that there are 6 outcomes namely ‘Discharge (DIS)’, ‘Discharge on reserve bond (DORB)’, ‘Expire (EXP)’, ‘Transfer to home (TRH)’, ‘Transfer to ward (TRW)’, and ‘Transfer to Other ITU’. These outcomes are labeled as ‘Discharge (DIS)’-1, ‘Discharge on reserve bond (DORB)’-2, ‘Expire (EXP)’-3, ‘Transfer to home (TRH)’-4, ‘Transfer to ward (TRW)’-5, and ‘Transfer to Other ITU’-6. All feature columns are normalized with standard scaler function due to large variance in the data.

In model selection, K-means clustering is chosen to obtain 6 clusters. Evaluation score methods are used to evaluate the model. Evaluation score methods used to evaluate the model are Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, Fowlskes Mallows Index and Rand Score.

However, it is observed that the evaluation scores for K-means algorithm is not appropriate.

So, another unsupervised learning algorithm is used i.e., Gaussian Mixture Model (GMM). GMM is also used to cluster the data in 6 clusters. Rand score of GMM algorithm is given better accuracy than Rand score of K-means algorithm.

After the clusters are formed and the target clusters are achieved then K-fold cross validation method is applied on external synthetic dataset generated with artificial intelligence algorithm.

5. Experimental Result and Evaluation

The dissertation has been done with K-means algorithm and Gaussian Mixture Model algorithm. The confusion matrix can't justify the cluster accuracy of unsupervised algorithm. So, some evaluation score methods namely Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, Fowlskes Mallows Index and Rand Score are applied to achieve the cluster accuracy.

After exploring several scores, it is observed that the Rand score for GMM algorithm is providing the best clustering accuracy. It is already mentioned that Rand score is used to measure similarity among clusters. It is using permutation model of clustering to check the similarity using pair wise comparison among clusters and its range is from 0 to 1. 0 means clusters do not agree on clustering of any pair of entities and 1 means clusters agree on clustering of every pair of entities.

SL. No.	Name of Evaluation Score Methods	K-Means	GMM
1	Silhouette Coefficient	0.250	0.110
2	Calinski-Harabasz Index	1742.870	1098.528
3	Davies-Bouldin Index	1.112	1.586
4	Fowlskes Mallows Index	0.309	0.340
5	Rand Score	0.619	0.624

Table 1. Evaluation Scores

Observing the table [1], it could be seen all the other coefficients like Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index and Fowlskes Mallows Index of K-means or in GMM, all values are bit away from the good accuracy score. So, seeing that in the observation Rand Score has been considered as the clustering accuracy score.

6. Conclusion and Future work

From the implementation code it has been seen that original data is used for fitting the model and synthetic data is used for validation. As the Rand score of GMM is higher than K-means algorithm, it can be said that GMM is better model for this data than K-means algorithm.

This work can be further experimented with deep unsupervised learning algorithm for large dataset. Different unsupervised learning algorithm like Hierarchical algorithm, DBscan algorithm etc. can be used to predict the mortality rate.

7. References

1. J.Wang, J.Jiang, "An unsupervised deep learning framework via integrated optimization of representation learning and GMM-based modeling", Asian Conference on Computer Vision, ACCV, pp: 249–265, 2018.

2. J. Wang, J. Jiang, "Unsupervised deep clustering via adaptive GMM modeling and optimization", Neurocomputing, vol. 433, pp:199-211, 2021.

3. Y. Li, H. Wu, "A Clustering Method Based on K-Means Algorithm", International Conference on Solid State Devices and Materials Science, pp:1104-1109, 2012.

4. B.M.A. Shafeeq, K.S. Hareesha, "Dynamic Clustering of Data with Modified K-Means Algorithm", International Conference on Information and Computer Networks, 2012.

5. A.M.Fahim, A.M.Salem, F.A.Torkey, "An efficient enhanced k-means clustering algorithm", Journal of Zhejiang University Science , vol.10, pp:1626-1633, 2006.

6. T. Desautels, R.Das, J.Calvert, M.Trivedi, C.Summers, D.J. Wales, A.Erocole, "Prediction of early unplanned intensive care unit readmission in a UK tertiary care hospital: a cross-sectional machine learning approach", BMJ Journal,vol.7, 2017.

7. Y.A.Khan, S.Z.Abbas, B.C.Truong, "Machine learning-based mortality rate prediction using optimized hyper-parameter", Elsevier Public Health Emergency Collection, vol. 197, 2020.

8. J.Chen, L.Milot, H.M.C. Cheung, A.L.Martel, "Unsupervised Clustering of Quantitative Imaging Phenotypes Using Autoencoder and Gaussian Mixture Model", Medical Image Computing and Computer Assisted Intervention, pp 575–582,2019.

9. A.M. Flores, A. Schular, A.V. Eberhard, J.W. Olin, J.P. Cooke, N.J. Leeper, N. H. Shah, E.G. Ross, "Unsupervised Learning for Automated Detection of Coronary Artery Disease Subgroups", Journal of the American Heart Association, vol.10, 2021.

10. S. Hyun, P. Kaewprag, C. Cooper, B. Hixon, S. M. Bruce, "Exploration of critical care data by using unsupervised machine learning", Computer Methods and Programs in Biomedicine, vol.194,2020

Appendix

Data of patients from different hospital is collected. Apart from that synthetic data has generated for validation and have taken ['Years', 'A4', 'PMR', 'Exptd ITU LOS','ITU_STAY'] as features.

Mortality rate prediction using unsupervised algorithm

Importing Required Libraries

```
# Importing Panda
import pandas as pd

# Importing Numpy
import numpy as np

# Importing Matplotlib
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

#Importing SKLearn
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

# Importing Seaborn
import seaborn as sns

#K-Means
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans


#GMM
from matplotlib.patches import Ellipse
from sklearn.mixture import GaussianMixture

#Unsupervised score
from sklearn.metrics import silhouette_score,calinski_harabasz_score,da
vies_bouldin_score,fowlkes_mallows_score,rand_score

import warnings
```

```
warnings.filterwarnings('ignore')
```

Importing Dataset & Visualization

```
# Importing files using pandas
data = pd.read_csv("/content/Draft transfer time database.csv")

# Shows top 5 dataset items
data.head(5)
```

Information of the Data set

```
# Dataset dimensions - (rows, columns)
data.shape
(4783, 18)

# Information about the columns
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4783 entries, 0 to 4782
Data columns (total 18 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   SL_ NO                                 4783 non-null   int64
 1   NAME                                   4782 non-null   object
 2   Years                                  4767 non-null   float64
 3   SEX                                    4770 non-null   object
 4   DT_OF_ADM                              4763 non-null   object
 5   TIME OF ADM                            4749 non-null   object
 6   ADM_DT_ITU                             4758 non-null   object
 7   RE-ADM                                 4749 non-null   float64
 8   READM_WITHIN_2_DAYS                    4746 non-null   float64
 9   A4                                     4691 non-null   float64
 10  PMR                                    4686 non-null   float64
 11  Exptd ITU LOS                          4680 non-null   float64
 12  ITU-OUTCOME                            4754 non-null   object
 13  PHYSICALLY TRANSFER OUT /DISCHARGETIME  4711 non-null   object
 14  DEATH WITHIN 24 HRS                    4726 non-null   object
 15  ITU_STAY                               4768 non-null   object
 16  TOTAL_HOSP_STAY                        4253 non-null   object
 17  HOSP_OUTCOME                           4662 non-null   object
dtypes: float64(6), int64(1), object(11)
memory usage: 672.7+ KB

# To check the dataset wheather it has any null value or not
data.isnull().sum()
SL_ NO                                   0
NAME                                     1
Years                                   16
SEX                                     13
DT_OF_ADM                               20
TIME OF ADM                             34
ADM_DT_ITU                              25
RE-ADM                                  34
READM_WITHIN_2_DAYS                     37
```

```
A4                                          92
PMR                                         97
Exptd ITU LOS                              103
ITU-OUTCOME                                 29
PHYSICALLY TRANSFER OUT /DISCHARGETIME      72
DEATH WITHIN 24 HRS                         57
ITU_STAY                                    15
TOTAL_HOSP_STAY                            530
HOSP_OUTCOME                               121
```

```
# Statistical description of dataset
data.describe()
```

	SL_ NO	Years	RE-ADM	READM_WITHIN_ 2_DAYS	A4	PMR	Exptd ITU LOS
count	4783.000000	4767.000000	4749.000000	4746.000000	4691.000000	4686.000000	4680.000000
mean	2392.000000	66.287392	0.069278	0.016013	55.595685	17.250294	4.720308
std	1380.877499	15.735995	0.253953	0.125540	26.714380	120.401071	1.785322
min	1.000000	10.000000	0.000000	0.000000	0.000000	0.060000	0.110000
25 %	1196.500000	58.000000	0.000000	0.000000	37.000000	2.810000	3.420000
50 %	2392.000000	69.000000	0.000000	0.000000	52.000000	6.745000	4.520000
75 %	3587.500000	78.000000	0.000000	0.000000	69.000000	16.765000	5.800000
max	4783.000000	110.000000	1.000000	1.000000	181.000000	7076.000000	30.780000

Data Pre-processing

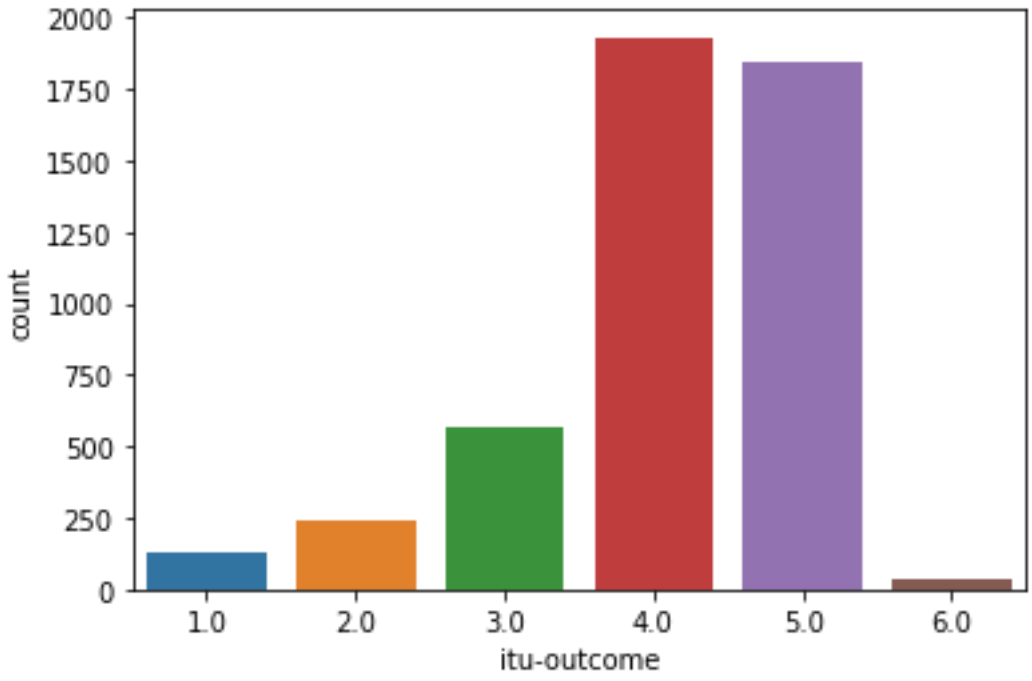
```
# Grouping the output
print(data.groupby('ITU-OUTCOME').size())
ITU-OUTCOME
1.40 pm                    1
6:00 PM                    1
6pm                        1
8:00 PM                    1
DIS                      129
DORB                     243
EXP                      571
TRH                     1930
TRW                     1841
Transfer to Other ITU     36
dtype: int64
```

```
# Convert the categorical output into numerical

# DIS(1),DORB(2),EXP(3),TRH(4),TRW(5),Transfer to Other ITU(6)


def new_itu_outcome(x):
    if x == '1.40 pm':
        return 1
    if x == '6:00 PM':
        return 1
    if x == '6pm':
        return 2
    if x == '8:00 PM':
        return 2
    if x == 'DIS':
        return 1
    if x == 'DORB':
        return 2
    if x == 'EXP':
        return 3
    if x == 'TRH':
        return 4
    if x == 'TRW':
        return 5
    if x == 'Transfer to Other ITU':
        return 6


# Adding new column for output
data['itu-outcome'] = data['ITU-OUTCOME'].apply(new_itu_outcome)


# Printing the information of output column
print(data.groupby('itu-outcome').size())
```

```
itu-outcome
1.0     131
2.0     245
3.0     571
4.0    1930
5.0    1841
6.0      36
dtype: int64
```

```
# Outcome countplot (v8 in bar graph)

sns.countplot(x = 'itu-outcome',data = data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8990f74650>
```




```
# visualize columns
data.columns
```

```
Index(['SL_ NO', 'NAME', 'Years', 'SEX', 'DT_OF_ADM', 'TIME OF ADM',
       'ADM_DT_ITU', 'RE-ADM', 'READM_WITHIN_2_DAYS', 'A4', 'PMR',
       'Exptd ITU LOS', 'ITU-OUTCOME',
       'PHYSICALLY TRANSFER OUT /DISCHARGETIME', 'DEATH WITHIN 24 HRS',
       'ITU_STAY', 'TOTAL_HOSP_STAY', 'HOSP_OUTCOME', 'itu-outcome'],
      dtype='object')
```

CodeText

```
# To change; Male = 0 & Female = 1
data["SEX"] = np.where(data["SEX"] == "M", 0, 1)

# Show the Data set
data


#To show the number of null values in the dataset
data.isnull().sum()
```

```
SL_ NO                                    0
NAME                                      1
Years                                    16
SEX                                       0
DT_OF_ADM                                20
TIME OF ADM                              34
ADM_DT_ITU                               25
RE-ADM                                   34
READM_WITHIN_2_DAYS                      37
A4                                       92
PMR                                      97
Exptd ITU LOS                           103
ITU-OUTCOME                              29
PHYSICALLY TRANSFER OUT /DISCHARGETIME    72
DEATH WITHIN 24 HRS                      57
```

```
ITU_STAY                                  15
TOTAL_HOSP_STAY                          530
HOSP_OUTCOME                             121
itu-outcome                               29
dtype: int64
```

K-Means Model

```
#define predictor and response variables
#X = data[['Years', 'A4', 'PMR', 'Exptd ITU LOS','ITU_STAY']]
#y = data['itu-outcome']
#X = data.iloc[:, [2,9,10,11,15]].values
#Y = data.iloc[:, 18].values

# To distinguish the features and target columns
X_KM = data.drop(['SL_ NO', 'NAME','SEX', 'DT_OF_ADM', 'TIME OF ADM',
       'ADM_DT_ITU', 'RE-ADM', 'READM_WITHIN_2_DAYS',
       'ITU-OUTCOME',
       'PHYSICALLY TRANSFER OUT /DISCHARGETIME', 'DEATH WITHIN 24 HRS',
       'TOTAL_HOSP_STAY', 'HOSP_OUTCOME', 'itu-outcome'], axis = 1)
Y_KM = data['itu-outcome']
# To show feature column
print(X_KM)
     Years    A4    PMR  Exptd ITU LOS ITU_STAY
0      63.0  25.0   2.31           4.07        2
1      80.0  33.0   1.63           4.66        9
2      50.0  77.0  52.08          13.09       15
3      43.0  54.0   2.75           3.85        4
4       NaN  33.0   2.74           9.56        4
...     ...   ...    ...            ...      ...
4778   76.0  91.0  30.55           5.52        3
4779   68.0  60.0  13.10           5.58        2
4780   19.0  51.0   6.10           6.95        4
4781   28.0  32.0   3.50           4.47        5
4782   81.0  60.0  20.19           5.05        2

[4783 rows x 5 columns]

# To show target column
print(Y_KM)
0       5.0
1       5.0
2       4.0
3       3.0
4       4.0
       ...
4778    4.0
4779    5.0
4780    5.0
4781    4.0
4782    4.0
Name: itu-outcome, Length: 4783, dtype: float64
```

For Feature Dataset

```
# To show shape of feature column
X_KM.shape
(4783, 5)

# To check the feature data wheather it has any null value or not
X_KM.isnull().sum()
Years             16
A4                92
PMR               97
Exptd ITU LOS    103
ITU_STAY          15
dtype: int64

# Information about the feature data
X_KM.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4783 entries, 0 to 4782
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Years          4767 non-null   float64
 1   A4             4691 non-null   float64
 2   PMR            4686 non-null   float64
 3   Exptd ITU LOS  4680 non-null   float64
 4   ITU_STAY       4768 non-null   object
dtypes: float64(4), object(1)
memory usage: 187.0+ KB

# Converting categorical data into numerical data
cat_cols_itust=[ 'ITU_STAY']
le=preprocessing.LabelEncoder()
X_KM[cat_cols_itust]=X_KM[cat_cols_itust].apply(le.fit_transform)
# Replacing NaN with mean values for feature data
X_KM["Years"].fillna(X_KM["Years"].mean(), inplace = True)
X_KM["A4"].fillna(X_KM["A4"].mean(), inplace = True)
X_KM["PMR"].fillna(X_KM["PMR"].mean(), inplace = True)
X_KM["Exptd ITU LOS"].fillna(X_KM["Exptd ITU LOS"].mean(), inplace = Tr
ue)
X_KM["ITU_STAY"].fillna(X_KM["ITU_STAY"].mean(), inplace = True)
```

Scaling the Feature Dataset

```
#from sklearn.preprocessing import StandardScaler
scaler_km = StandardScaler()
# Fit the scaling function on feature dataset
X_KM = scaler_km.fit_transform(X_KM)
```

For Target Dataset

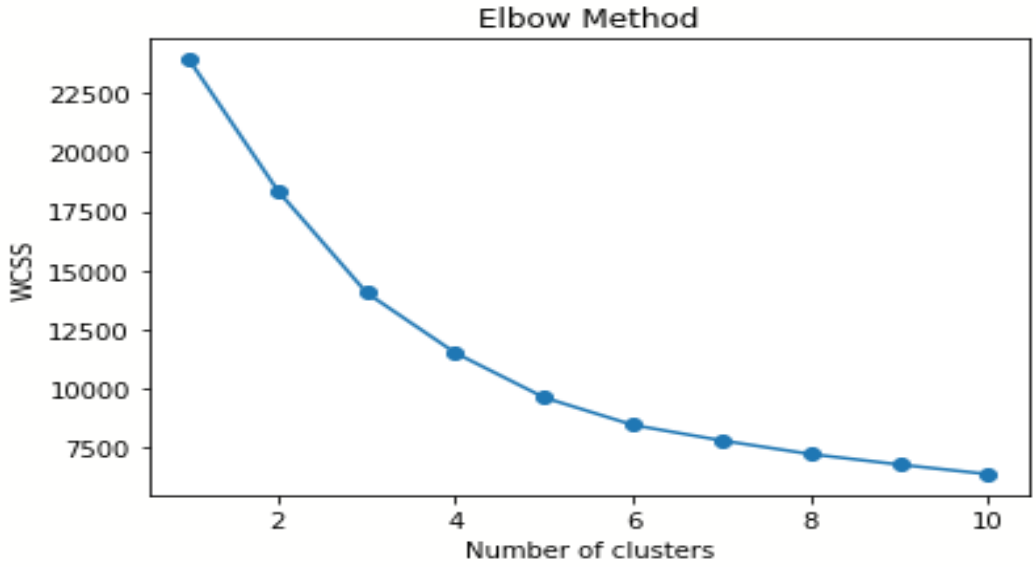
```
# To check the feature data wheather it has any null value or not
Y_KM.isnull().sum()
29

# Replacing NaN with mean values for target data
```

```
Y_KM.fillna(Y_KM.mean(), inplace = True)
# Converting target dataset into numpy dataset
Y_KM = Y_KM.to_numpy()
```

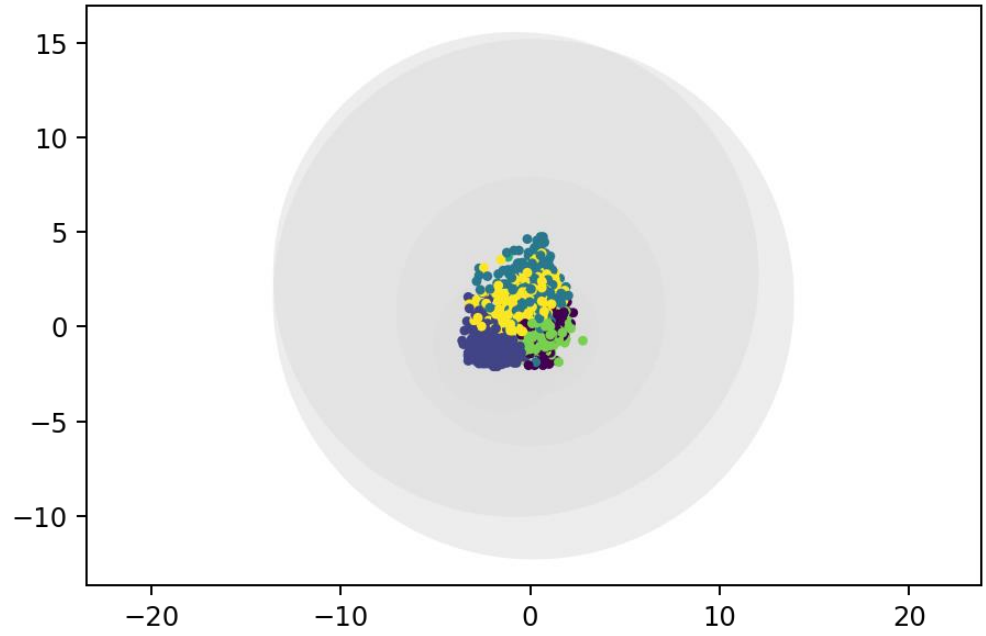
Build K-Means Model

```
#from scipy.spatial.distance import cdist
def plot_kmeans(kmeans, X_KM, ax=None):
    labels = kmeans.fit_predict(X_KM)
# plot the input data
    plt.figure(dpi=175)
    ax = ax or plt.gca()
    ax.axis('equal')
    ax.scatter(X_KM[:, 0], X_KM[:, 1], c=labels, s=7, cmap='viridis', z
order=2)
# plot the representation of the k-means model
    centers = kmeans.cluster_centers_
    radii = [
        cdist(X_KM[labels == i], [center]).max()
        for i, center in enumerate(centers)
    ]
    for c, r in zip(centers, radii):
        ax.add_patch(plt.Circle(
            c, r, fc='#DDDDDD', lw=3, alpha=0.5, zorder=1
        ))
# To append the inertia in a list
list1=[]
for i in range(1,11):
    KMeans_Clustering=KMeans(n_clusters=i,random_state=42)
    KMeans_Clustering.fit(X_KM)
    list1.append(KMeans_Clustering.inertia_)
```

```
#Elbow Method
plt.plot(range(1,11 ),list1 ,marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
#from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=6, random_state=42).fit(X_KM)
labels = kmeans.predict(X_KM)
plot_kmeans(kmeans, X_KM)
```




```
KMeans_Clustering=KMeans(n_clusters=6,random_state=10)
y_kmeans=KMeans_Clustering.fit_predict(X_KM)
print("The predicted clusters are:\n",y_kmeans)
```

```
print("The centers are:\n",KMeans_Clustering.cluster_centers_)
The predicted clusters are:
 [4 3 5 ... 0 0 4]
The centers are:
 [[-1.57960973 -0.99414562 -0.11819025 -0.54728368 -0.34134948]
 [ 0.04845876  0.80276913  0.09784571  1.09274114  1.09354021]
 [-0.7504067   2.75595363 46.51051329  1.13528439  0.42400999]
 [ 0.5027821  -0.19598107 -0.07766593 -0.40546731  0.85549838]
 [ 0.3924078  -0.36606964 -0.08781168 -0.4775008  -0.80509469]
 [ 0.17783785  1.44745266  0.22357487  1.05405175 -0.8839328 ]]

labels = KMeans_Clustering.labels_
print("Labels: ",labels)
Labels:  [4 3 5 ... 0 0 4]

se = []
rnd = []
fms = []
sil_scores = []
calinski_score = []
davies_score = []
index = range(2, 11)
for i in index:
    kmeans = KMeans(n_clusters=i, random_state=42)
    labels = kmeans.fit_predict(X_KM)
    se.append(kmeans.inertia_)
    sil_scores.append(silhouette_score(X_KM, labels))
    calinski_score.append(calinski_harabasz_score(X_KM, labels))
    davies_score.append(davies_bouldin_score(X_KM, labels))
    fms.append(fowlkes_mallows_score(Y_KM,labels))
    rnd.append(rand_score(Y_KM,labels))

    print('Intertia at K =', i, ':', kmeans.inertia_)
    print("Silhouette Coefficient: %0.3f" % silhouette_score(X_KM, labe
ls))
    print("Calinski-
Harabasz Index: %0.3f" % calinski_harabasz_score(X_KM, labels))
    print("Davies-
Bouldin Index: %0.3f" % davies_bouldin_score(X_KM, labels))
    print("Fowlskes Mallows Index: %0.3f" % fowlkes_mallows_score(Y_KM,
 labels))
    print("Rand Score: %0.3f" % rand_score(Y_KM, labels))
    print("-----------------------------------------------------------
---")
Intertia at K = 2 : 18370.941391047272
Silhouette Coefficient: 0.273
Calinski-Harabasz Index: 1442.834
Davies-Bouldin Index: 1.460
Fowlskes Mallows Index: 0.454
Rand Score: 0.533
-----------------------------------------------------------------
Intertia at K = 3 : 14056.135890755791
```

```
Silhouette Coefficient: 0.273
Calinski-Harabasz Index: 1676.345
Davies-Bouldin Index: 1.066
Fowlskes Mallows Index: 0.452
Rand Score: 0.532
-----------------------------------------------------------------
Intertia at K = 4 : 11514.731558092633
Silhouette Coefficient: 0.245
Calinski-Harabasz Index: 1715.514
Davies-Bouldin Index: 1.081
Fowlskes Mallows Index: 0.377
Rand Score: 0.570
-----------------------------------------------------------------
Intertia at K = 5 : 9629.515994345926
Silhouette Coefficient: 0.267
Calinski-Harabasz Index: 1772.069
Davies-Bouldin Index: 1.061
Fowlskes Mallows Index: 0.351
Rand Score: 0.601
-----------------------------------------------------------------
Intertia at K = 6 : 8467.793494473337
Silhouette Coefficient: 0.250
Calinski-Harabasz Index: 1742.870
Davies-Bouldin Index: 1.112
Fowlskes Mallows Index: 0.309
Rand Score: 0.619
-----------------------------------------------------------------
Intertia at K = 7 : 7813.136000647943
Silhouette Coefficient: 0.251
Calinski-Harabasz Index: 1640.458
Davies-Bouldin Index: 1.115
Fowlskes Mallows Index: 0.293
Rand Score: 0.626
-----------------------------------------------------------------
Intertia at K = 8 : 7238.8681946697725
Silhouette Coefficient: 0.235
Calinski-Harabasz Index: 1571.494
Davies-Bouldin Index: 1.123
Fowlskes Mallows Index: 0.273
Rand Score: 0.636
-----------------------------------------------------------------
Intertia at K = 9 : 6796.718015384615
Silhouette Coefficient: 0.219
Calinski-Harabasz Index: 1502.994
Davies-Bouldin Index: 1.123
Fowlskes Mallows Index: 0.254
Rand Score: 0.640
-----------------------------------------------------------------
Intertia at K = 10 : 6391.447021877876
Silhouette Coefficient: 0.214
Calinski-Harabasz Index: 1454.047
Davies-Bouldin Index: 1.171
Fowlskes Mallows Index: 0.237
Rand Score: 0.644
```

GMM Model

```
# To distinguish the features and target columns
X_GMM = data.drop(['SL_ NO', 'NAME','SEX', 'DT_OF_ADM', 'TIME OF ADM',
       'ADM_DT_ITU', 'RE-ADM', 'READM_WITHIN_2_DAYS',
       'ITU-OUTCOME',
       'PHYSICALLY TRANSFER OUT /DISCHARGETIME', 'DEATH WITHIN 24 HRS',
       'TOTAL_HOSP_STAY', 'HOSP_OUTCOME', 'itu-outcome'], axis = 1)
Y_GMM = data['itu-outcome']
# To show feature column
print(X_GMM)
      Years    A4    PMR  Exptd ITU LOS ITU_STAY
0      63.0  25.0   2.31           4.07        2
1      80.0  33.0   1.63           4.66        9
2      50.0  77.0  52.08          13.09       15
3      43.0  54.0   2.75           3.85        4
4       NaN  33.0   2.74           9.56        4
...     ...   ...    ...            ...      ...
4778   76.0  91.0  30.55           5.52        3
4779   68.0  60.0  13.10           5.58        2
4780   19.0  51.0   6.10           6.95        4
4781   28.0  32.0   3.50           4.47        5
4782   81.0  60.0  20.19           5.05        2

[4783 rows x 5 columns]

# To show target column
print(Y_GMM)
0       5.0
1       5.0
2       4.0
3       3.0
4       4.0
       ...
4778    4.0
4779    5.0
4780    5.0
4781    4.0
4782    4.0
Name: itu-outcome, Length: 4783, dtype: float64
```

For Feature Dataset

```
# To show shape of feature column
X_GMM.shape
(4783, 5)

# To check the feature data wheather it has any null value or not
X_GMM.isnull().sum()
Years             16
A4                92
PMR               97
Exptd ITU LOS    103
ITU_STAY          15
dtype: int64
```

```
# Information about the feature data
X_GMM.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4783 entries, 0 to 4782
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Years          4767 non-null   float64
 1   A4             4691 non-null   float64
 2   PMR            4686 non-null   float64
 3   Exptd ITU LOS  4680 non-null   float64
 4   ITU_STAY       4768 non-null   object
dtypes: float64(4), object(1)
memory usage: 187.0+ KB

# Converting categorical data into numerical data
cat_cols_itust=[ 'ITU_STAY']
le=preprocessing.LabelEncoder()
X_GMM[cat_cols_itust]=X_GMM[cat_cols_itust].apply(le.fit_transform)

# Replacing NaN with mean values for feature data
X_GMM["Years"].fillna(X_GMM["Years"].mean(), inplace = True)
X_GMM["A4"].fillna(X_GMM["A4"].mean(), inplace = True)
X_GMM["PMR"].fillna(X_GMM["PMR"].mean(), inplace = True)
X_GMM["Exptd ITU LOS"].fillna(X_GMM["Exptd ITU LOS"].mean(), inplace =
True)
X_GMM["ITU_STAY"].fillna(X_GMM["ITU_STAY"].mean(), inplace = True)
```

Scaling the Feature Dataset

```
#from sklearn.preprocessing import StandardScaler
scaler_gmm = StandardScaler()

# Fit the scaling function on feature dataset
X_GMM = scaler_gmm.fit_transform(X_GMM)
```

For Target Dataset

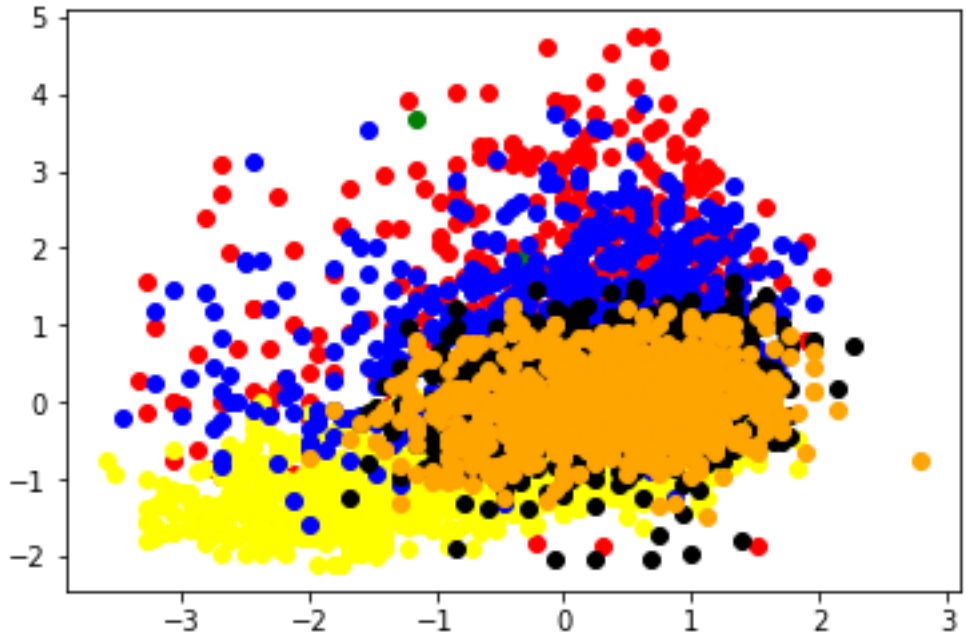
```
# To check the feature data wheather it has any null value or not
Y_GMM.isnull().sum()
0

# Replacing NaN with mean values for target data
Y_GMM.fillna(Y_GMM.mean(), inplace = True)

# Converting target dataset into numpy dataset
Y_GMM = Y_GMM.to_numpy()
```

Build GMM Model

```
#GMM
#from matplotlib.patches import Ellipse
```

```
def plot_gmm(gmm, X_GMM, label=True, ax=None):
    def draw_ellipse(position, covariance, ax=None, **kwargs):
        ax = ax or plt.gca()
        if covariance.shape == (2, 2):
                U, s, Vt = np.linalg.svd(covariance)
                angle = np.degrees(np.arctan2(U[1, 0], U[0, 0]))
                width, height = 2 * np.sqrt(s)
        else:
                angle = 0
                width, height = 2 * np.sqrt(covariance)
        for nsig in range(1, 4):
                ax.add_patch(Ellipse(
                position, nsig * width, nsig * height,
                angle, **kwargs))

                ax = ax or plt.gca()
                labels = gmm.fit(X_GMM).predict(X_GMM)
                if label:
                        ax.scatter(X_GMM[:, 0], X_GMM[:, 1], c=labels
, s=7, cmap='viridis', zorder=2)
                else:
                        ax.scatter(X_GMM[:, 0], X_GMM[:, 1], s=7, zor
der=2)

                        ax.axis('equal')
                        w_factor = 0.2 / gmm.weights_.max()
                        for pos, covar, w in zip(gmm.means_, gmm.cova
riances_, gmm.weights_):
                                        draw_ellipse(pos, covar, alpha=w *
 w_factor)
#from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=6, covariance_type='full', random_st
ate=1).fit(X_GMM)
labels_gmm = gmm.fit_predict(X_GMM)
plt.figure(dpi=175)
plot_gmm(gmm, X_GMM)
probabilities = gmm.predict_proba(X_GMM)
probabilities
array([[8.30221951e-04, 9.30937099e-01, 0.00000000e+00, 3.55284496e-05,
        1.98941815e-03, 6.62077326e-02],
       [2.10326377e-09, 8.50144629e-02, 0.00000000e+00, 2.08100235e-02,
        8.94175510e-01, 1.30416580e-09],
       [9.99991282e-01, 0.00000000e+00, 0.00000000e+00, 8.71807453e-06,
        3.14465702e-18, 3.03070401e-20],
       ...,
       [9.40274049e-05, 1.48168324e-02, 0.00000000e+00, 9.79558238e-01,
        5.53045477e-03, 4.47383996e-07],
       [8.77386596e-08, 9.96044685e-01, 0.00000000e+00, 3.74164880e-03,
        2.13578625e-04, 1.29470513e-10],
       [1.02443996e-01, 7.47943511e-87, 0.00000000e+00, 1.56848621e-02,
```

```
        5.00435900e-02, 8.31827552e-01]])
d = pd.DataFrame(X_GMM)
#gmm = GaussianMixture(n_components = 6)

# Fit the GMM model for the dataset
# which expresses the dataset as a
# mixture of 6 Gaussian Distribution
gmm.fit(d)

# Assign a label to each sample
labels = gmm.predict(d)
d['labels']= labels
d0 = d[d['labels']== 0]
d1 = d[d['labels']== 1]
d2 = d[d['labels']== 2]
d3 = d[d['labels']== 3]
d4 = d[d['labels']== 4]
d5 = d[d['labels']== 5]

# plot six clusters in same plot
plt.scatter(d0[0], d0[1], c ='red')
plt.scatter(d1[0], d1[1], c ='yellow')
plt.scatter(d2[0], d2[1], c ='green')
plt.scatter(d3[0], d3[1], c ='blue')
plt.scatter(d4[0], d4[1], c ='black')
plt.scatter(d5[0], d5[1], c ='orange')

#plt.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis')
<matplotlib.collections.PathCollection at 0x7f89897b2ed0>
```




```
y_gmm=gmm.fit_predict(X_GMM)
print("The predicted clusters are:\n",y_gmm)
```

```
The predicted clusters are:
 [1 4 0 ... 3 1 5]
# print the converged log-likelihood value
print(gmm.lower_bound_)

# print the number of iterations needed
# for the log-likelihood value to converge
print(gmm.n_iter_)

-2.9701484671960756
15
fms_sc = fowlkes_mallows_score(Y_GMM,labels_gmm)
sil_sc = silhouette_score(X_GMM, labels_gmm)
cal_sc = calinski_harabasz_score(X_GMM, labels_gmm)
dav_sc = davies_bouldin_score(X_GMM, labels_gmm)
rnd_sc = rand_score(Y_GMM,labels_gmm)

print(fms_sc)
print(sil_sc)
print(cal_sc)
print(dav_sc)
print(rnd_sc)
0.33982852496969634
0.11000980851137653
1098.5275221279567
1.5864629855642436
0.6240964946866311
```

Validation

Importing Required Library

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

Importing Dataset & Visualization

```
# Importing files using pandas
data_val = pd.read_csv("/content/Synthetic data.csv")

# Shows top 5 dataset items
data_val.head(5)
```

Information of the Data set

```
# Dataset dimensions - (rows, columns)
data_val.shape
(10000, 18)
# Information about the columns
```

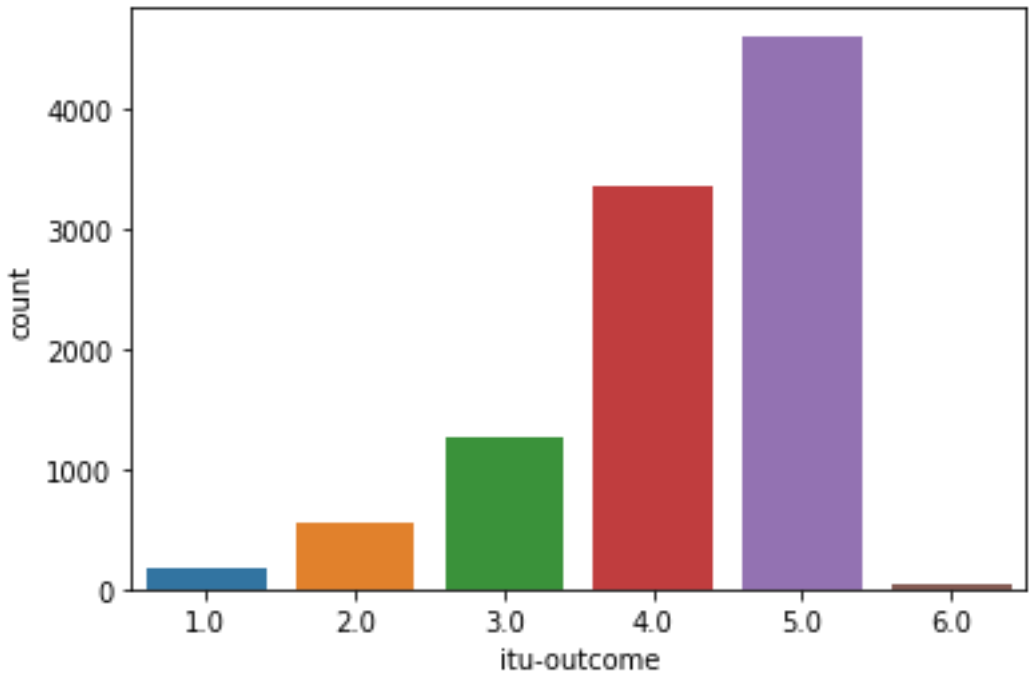
```
data_val.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   SL_ NO                                    10000 non-null  int64
 1   NAME                                      10000 non-null  object
 2   Years                                     9999 non-null   float64
 3   SEX                                       9999 non-null   object
 4   DT_OF_ADM                                 9999 non-null   object
 5   TIME OF ADM                               9995 non-null   object
 6   ADM_DT_ITU                                9998 non-null   object
 7   RE-ADM                                    9995 non-null   float64
 8   READM_WITHIN_2_DAYS                       9994 non-null   float64
 9   A4                                        9872 non-null   float64
 10  PMR                                       9871 non-null   float64
 11  Exptd ITU LOS                             9866 non-null   float64
 12  ITU-OUTCOME                               9999 non-null   object
 13  PHYSICALLY TRANSFER OUT /DISCHARGETIME    9919 non-null   object
 14  DEATH WITHIN 24 HRS                       9981 non-null   float64
 15  ITU_STAY                                  9996 non-null   float64
 16  TOTAL_HOSP_STAY                           9123 non-null   float64
 17  HOSP_OUTCOME                              9799 non-null   object
dtypes: float64(9), int64(1), object(8)
memory usage: 1.4+ MB

# To check the dataset wheather it has any null value or not
data_val.isnull().sum()
SL_ NO                                      0
NAME                                        0
Years                                       1
SEX                                         1
DT_OF_ADM                                   1
TIME OF ADM                                 5
ADM_DT_ITU                                  2
RE-ADM                                      5
READM_WITHIN_2_DAYS                         6
A4                                        128
PMR                                       129
Exptd ITU LOS                             134
ITU-OUTCOME                                 1
PHYSICALLY TRANSFER OUT /DISCHARGETIME     81
DEATH WITHIN 24 HRS                        19
ITU_STAY                                    4
TOTAL_HOSP_STAY                           877
HOSP_OUTCOME                              201
dtype: int64

# Statistical description of dataset
data_val.describe()
```

Data Pre-processing

```
# Grouping the output
```

```
print(data_val.groupby('ITU-OUTCOME').size())
ITU-OUTCOME
DIS                       176
DORB                      544
EXP                      1255
TRH                      3367
TRW                      4610
Transfer to Other ITU      47
dtype: int64

# Convert the categorical output into numerical

# DIS(1),DORB(2),EXP(3),TRH(4),TRW(5),Transfer to Other ITU(6)


def new_itu_outcome(x):
    if x == '1.40 pm':
        return 1
    if x == '6:00 PM':
        return 1
    if x == '6pm':
        return 2
    if x == '8:00 PM':
        return 2
    if x == 'DIS':
        return 1
    if x == 'DORB':
        return 2
    if x == 'EXP':
        return 3
    if x == 'TRH':
        return 4
    if x == 'TRW':
        return 5
    if x == 'Transfer to Other ITU':
        return 6
# Adding new coluumn for output
data_val['itu-outcome'] = data_val['ITU-
OUTCOME'].apply(new_itu_outcome)
# Printing the information of output column
print(data_val.groupby('itu-outcome').size())
itu-outcome
1.0     176
2.0     544
3.0    1255
4.0    3367
5.0    4610
6.0      47
dtype: int64

# Outcome countplot (v8 in bar graph)
```

```
sns.countplot(x = 'itu-outcome',data = data_val)
<matplotlib.axes._subplots.AxesSubplot at 0x7f8989650450>
```




```
# visualize columns
data_val.columns
Index(['SL_ NO', 'NAME', 'Years', 'SEX', 'DT_OF_ADM', 'TIME OF ADM',
       'ADM_DT_ITU', 'RE-ADM', 'READM_WITHIN_2_DAYS', 'A4', 'PMR',
       'Exptd ITU LOS', 'ITU-OUTCOME',
       'PHYSICALLY TRANSFER OUT /DISCHARGETIME', 'DEATH WITHIN 24 HRS',
       'ITU_STAY', 'TOTAL_HOSP_STAY', 'HOSP_OUTCOME', 'itu-outcome'],
      dtype='object')
# To change; Male = 0 & Female = 1
data_val["SEX"] = np.where(data_val["SEX"] == "M", 0, 1)

# Show the Data set
data_val
```

K means validation start

```
#define predictor and response variables
#X = data[['Years', 'A4', 'PMR', 'Exptd ITU LOS','ITU_STAY']]
#y = data['itu-outcome']
#X = data.iloc[:, [2,9,10,11,15]].values
#Y = data.iloc[:, 18].values

# To distinguish the features and target columns
X_VAL = data_val.drop(['SL_ NO', 'NAME','SEX', 'DT_OF_ADM', 'TIME OF AD
M',
       'ADM_DT_ITU', 'RE-ADM', 'READM_WITHIN_2_DAYS',
       'ITU-OUTCOME',
       'PHYSICALLY TRANSFER OUT /DISCHARGETIME', 'DEATH WITHIN 24 HRS',
```

```
       'TOTAL_HOSP_STAY', 'HOSP_OUTCOME', 'itu-outcome'], axis = 1)
Y_VAL = data_val['itu-outcome']
# To show feature column
print(X_VAL)
```

	Years	A4	PMR	Exptd ITU LOS	ITU_STAY
0	65.0	NaN	NaN	NaN	2.0
1	47.0	159.0	46.50	4.20	2.0
2	59.0	26.0	2.31	2.54	1.0
3	89.0	85.0	18.81	3.55	5.0
4	65.0	44.0	3.39	3.92	3.0
...	...	...	...	...	...
9995	64.0	60.0	8.60	4.44	3.0
9996	32.0	11.0	0.79	3.11	2.0
9997	48.0	31.0	2.17	4.44	2.0
9998	73.0	34.0	4.86	3.22	2.0
9999	84.0	78.0	19.05	7.98	2.0

[10000 rows x 5 columns]

```
# To show target column
print(Y_VAL)
```

```
0       5.0
1       3.0
2       5.0
3       4.0
4       5.0
       ...
9995    4.0
9996    5.0
9997    5.0
9998    5.0
9999    4.0
```

Name: itu-outcome, Length: 10000, dtype: float64

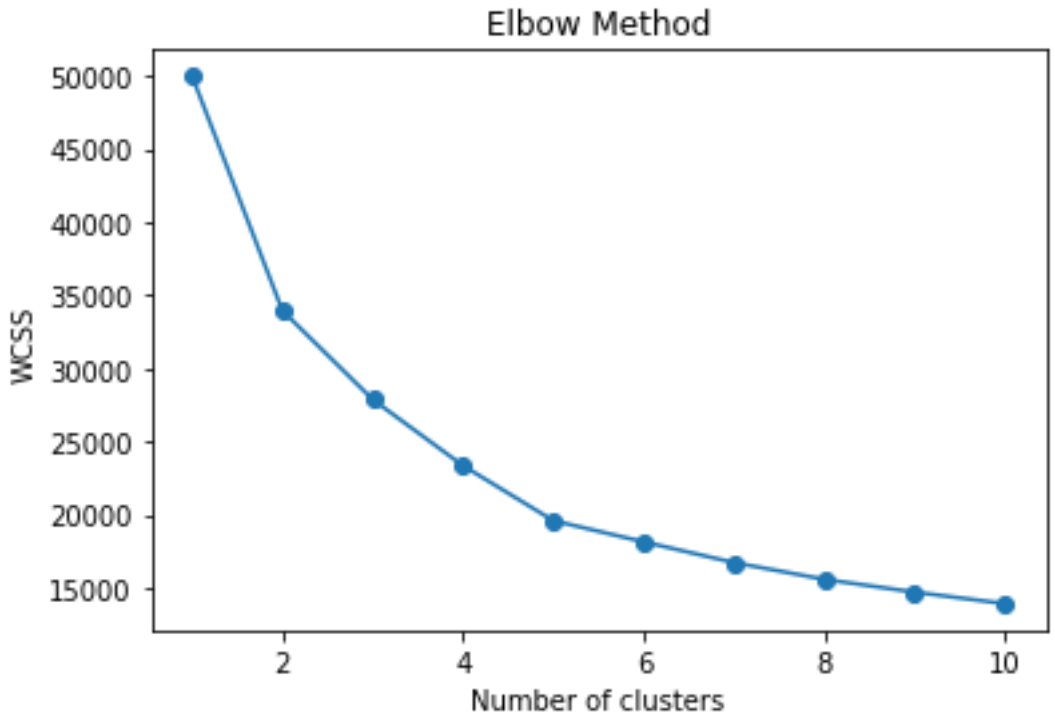
```
# To show shape of feature column
X_VAL.shape
```

(10000, 5)

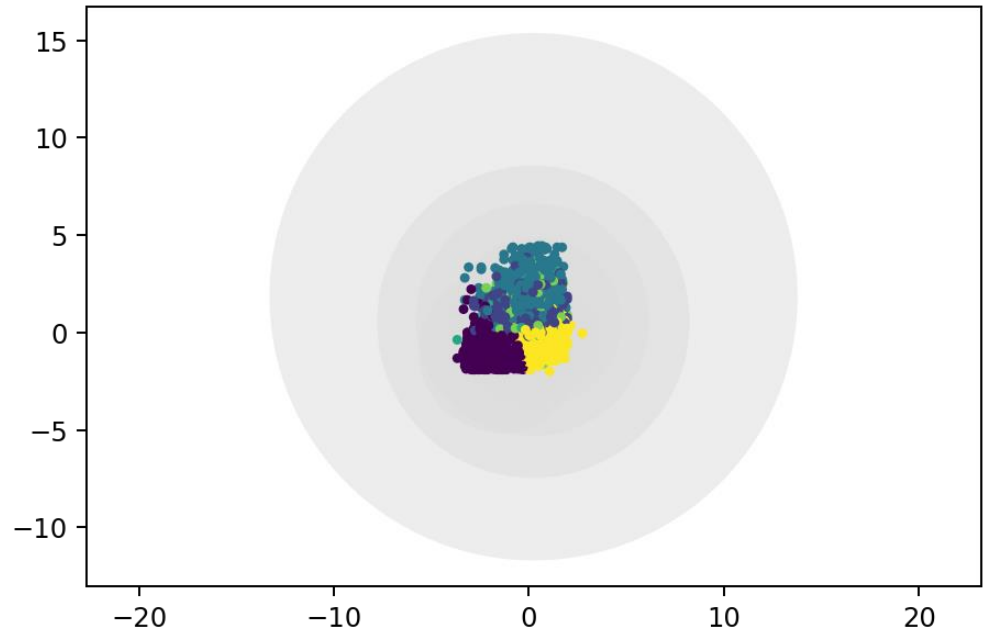
```
# To check the feature data wheather it has any null value or not
```

```
X_VAL.isnull().sum()
Years              1

A4               128

PMR              129

Exptd ITU LOS    134

ITU_STAY           4

dtype: int64

# Information about the feature data
X_VAL.info()
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 5 columns):

 #   Column         Non-Null Count  Dtype

---  ------         --------------  -----

 0   Years          9999 non-null   float64

 1   A4             9872 non-null   float64

 2   PMR            9871 non-null   float64

 3   Exptd ITU LOS  9866 non-null   float64

 4   ITU_STAY       9996 non-null   float64

dtypes: float64(5)

memory usage: 390.8 KB

# Replacing NaN with mean values for feature data
X_VAL["Years"].fillna(X_VAL["Years"].mean(), inplace = True)
X_VAL["A4"].fillna(X_VAL["A4"].mean(), inplace = True)
X_VAL["PMR"].fillna(X_VAL["PMR"].mean(), inplace = True)
X_VAL["Exptd ITU LOS"].fillna(X_VAL["Exptd ITU LOS"].mean(), inplace =
True)
X_VAL["ITU_STAY"].fillna(X_VAL["ITU_STAY"].mean(), inplace = True)
#from sklearn.preprocessing import StandardScaler
scaler_val = StandardScaler()
# Fit the scaling function on feature dataset
X_VAL = scaler_val.fit_transform(X_VAL)
# To check the feature data wheather it has any null value or not
Y_VAL.isnull().sum()
1

# Replacing NaN with mean values for target data
Y_VAL.fillna(Y_VAL.mean(), inplace = True)
# Converting target dataset into numpy dataset
Y_VAL = Y_VAL.to_numpy()
#from scipy.spatial.distance import cdist
```

```
def plot_kmeans(kmeans, X_VAL, ax=None):
    labels = kmeans.fit_predict(X_VAL)
# plot the input data
    plt.figure(dpi=175)
    ax = ax or plt.gca()
    ax.axis('equal')
    ax.scatter(X_VAL[:, 0], X_VAL[:, 1], c=labels, s=7, cmap='viridis',
 zorder=2)
# plot the representation of the k-means model
    centers = kmeans.cluster_centers_
    radii = [
        cdist(X_VAL[labels == i], [center]).max()
        for i, center in enumerate(centers)
    ]
    for c, r in zip(centers, radii):
        ax.add_patch(plt.Circle(
            c, r, fc='#DDDDDD', lw=3, alpha=0.5, zorder=1
        ))
# To append the inertia in a list
list1=[]
for i in range(1,11):
    KMeans_Clustering=KMeans(n_clusters=i,random_state=42)
    KMeans_Clustering.fit(X_VAL)
    list1.append(KMeans_Clustering.inertia_)
#Elbow Method
plt.plot(range(1,11 ),list1 ,marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
#from sklearn.cluster import KMeans
kmeans_val = KMeans(n_clusters=6, random_state=42).fit(X_VAL)
labels = kmeans_val.predict(X_VAL)
plot_kmeans(kmeans_val, X_VAL)
```




```
KMeans_Clustering=KMeans(n_clusters=6,random_state=10)
y_kmeans_val=KMeans_Clustering.fit_predict(X_VAL)
print("The predicted clusters are:\n",y_kmeans_val)
print("The centers are:\n",KMeans_Clustering.cluster_centers_)
```

```
The predicted clusters are:

 [2 4 0 ... 0 2 1]
```

```
The centers are:
 [[-1.45137871 -0.9481988  -0.56537713 -0.51040939 -0.2644665 ]
 [ 0.2314782   0.5518114   0.1300014   0.93703714 -0.09057667]
 [ 0.4700568  -0.35120182 -0.43628312 -0.57315913 -0.25026021]
 [ 0.1771355   0.64660817  0.67483556  0.55257571  7.50815661]
 [ 0.23560306  1.84060537  2.4756553   1.03526655  0.10351903]
 [ 0.35199465  0.5705076   0.38743664  0.54600104  2.16244207]]
```

```
labels = KMeans_Clustering.labels_
print("Labels: ",labels)
```

```
Labels:  [2 4 0 ... 0 2 1]
```

```
se = []
rnd = []
fms = []
sil_scores = []
calinski_score = []
davies_score = []
index = range(2, 11)
for i in index:
    kmeans_val = KMeans(n_clusters=i, random_state=42)
    labels = kmeans_val.fit_predict(X_VAL)
    se.append(kmeans_val.inertia_)
    sil_scores.append(silhouette_score(X_VAL, labels))
    calinski_score.append(calinski_harabasz_score(X_VAL, labels))
    davies_score.append(davies_bouldin_score(X_VAL, labels))
    fms.append(fowlkes_mallows_score(Y_VAL,labels))
    rnd.append(rand_score(Y_VAL,labels))

    print('Intertia at K =', i, ':', kmeans_val.inertia_)
    print("Silhouette Coefficient: %0.3f" % silhouette_score(X_VAL, lab
els))
    print("Calinski-
Harabasz Index: %0.3f" % calinski_harabasz_score(X_VAL, labels))
    print("Davies-
Bouldin Index: %0.3f" % davies_bouldin_score(X_VAL, labels))
    print("Fowlskes Mallows Index: %0.3f" % fowlkes_mallows_score(Y_VAL
, labels))
    print("Rand Score: %0.3f" % rand_score(Y_VAL, labels))
    print("----------------------------------------------------------
---")
```

```
Intertia at K = 2 : 33953.52918946557

Silhouette Coefficient: 0.383

Calinski-Harabasz Index: 4725.068

Davies-Bouldin Index: 1.264
```

```
Fowlskes Mallows Index: 0.539
Rand Score: 0.551
-----------------------------------------------------------------
Intertia at K = 3 : 27866.634619038785
Silhouette Coefficient: 0.276
Calinski-Harabasz Index: 3970.127
Davies-Bouldin Index: 1.210
Fowlskes Mallows Index: 0.428
Rand Score: 0.574
-----------------------------------------------------------------
Intertia at K = 4 : 23340.45043983354
Silhouette Coefficient: 0.292
Calinski-Harabasz Index: 3805.853
Davies-Bouldin Index: 1.166
Fowlskes Mallows Index: 0.422
Rand Score: 0.583
-----------------------------------------------------------------
Intertia at K = 5 : 19575.50621996712
Silhouette Coefficient: 0.285
Calinski-Harabasz Index: 3883.588
Davies-Bouldin Index: 1.141
Fowlskes Mallows Index: 0.382
Rand Score: 0.606
-----------------------------------------------------------------
Intertia at K = 6 : 18124.463200951253
Silhouette Coefficient: 0.288
Calinski-Harabasz Index: 3515.301
Davies-Bouldin Index: 1.143
Fowlskes Mallows Index: 0.377
Rand Score: 0.611
-----------------------------------------------------------------
Intertia at K = 7 : 16705.388169444897
Silhouette Coefficient: 0.226
```

```
Calinski-Harabasz Index: 3319.429
Davies-Bouldin Index: 1.223
Fowlskes Mallows Index: 0.320
Rand Score: 0.620
----------------------------------------------------------------
Intertia at K = 8 : 15561.02253589927
Silhouette Coefficient: 0.220
Calinski-Harabasz Index: 3159.133
Davies-Bouldin Index: 1.262
Fowlskes Mallows Index: 0.297
Rand Score: 0.628
----------------------------------------------------------------
Intertia at K = 9 : 14683.726789409382
Silhouette Coefficient: 0.216
Calinski-Harabasz Index: 3003.727
Davies-Bouldin Index: 1.260
Fowlskes Mallows Index: 0.283
Rand Score: 0.632
----------------------------------------------------------------
Intertia at K = 10 : 13897.19501407573
Silhouette Coefficient: 0.219
Calinski-Harabasz Index: 2883.623
Davies-Bouldin Index: 1.276
Fowlskes Mallows Index: 0.279
Rand Score: 0.634
```

```
kfold_validation=KFold(10)
from sklearn.model_selection import cross_val_score
results=cross_val_score(kmeans_val,X_VAL,Y_VAL,cv=kfold_validation)
print(results)
print(np.mean(results))
```

```
[-1325.03587007 -1357.82474656 -1354.78344387 -1392.09356238
 -1391.74449098 -1445.04973692 -1405.10256054 -1549.71578347
 -1354.71847555 -1409.75414579]
```

-1398.5822816114287