PROJECT TITLE
HEART FAILURE PREDICTION

GROUP NO. 10

GROUP MEMBERS

MEMBER-01: MUHAMMAD BAZIL CHOUHAN	(EP#20101027)
MEMBER-02: AZIZ ALI	(EP#19101009)
MEMBER-03: SYED AMMAR ALI	(EP#20101052)
MEMBER-04: YUMAN SAEED	(EP#20101060)

DEGREE NAME

MASTERS OF COMPUTER SCIENCE(MCS)

YEAR

MCS FINAL YEAR (2020-22)

UNIVERSITY NAME

UNIVERSITY OF KARACHI(UOK)

DEPARTMENT NAME

UMAER BASHA INSTITUTE OF INFORMATION TECHNOLOGY(UBIT)

Table of Contents

CHAPTER 1

INTRODUCTION OF DATA MINING AND WAREHOUSING

We can define data mining and warehousing as an automatic or semi-automatic technical process which gathered the scattered or unreliable data and make that data into knowledgeable.

APPLICATION OF DATA MINING AND WAREHOUSING

HEALTH CARE SYSTEM

We can use data mining in health care to reduce cost. It is use in improvement of entire system. Researchers use data mining in database. It can use to predict the volume of patients in hospitals.

MARKET BASE ANALYSIS

It can use in market for costs reduction. Through this retailer will know what group of items they should keep by model.

EDUCATION

It is called EDM which purpose is to analysis student`s performance and predict student future learning. Its goal is to predict student`s future behavior and to predict student effectiveness on study.

MANUFACTORING ENGINEERING

In this its works to solve pattern the complex manufacturing process. It can be very use for to predict the product development span time.

CRM

Customer relationship management. It is use to maintain the relationship between business and customer for better customer service.

FRAUD DETECTION

Its use to save billion dollars to from fraud with help of data mining we can convert data into information.

LIE DETECTION

With the help of this we can detect lies

We collect data of past and present and make unstructured data into information and predict that either criminal speaking a truth or not.

DIFFERNCE BETWEEN DATA AND INFORMATION

DATA

It is unorganized form which is useless because it is not in structured form.

Data is not very useful until it is converted into information.

INFORMATION

It’s a form of structured data.

It’s very useful because it is arranged and easy so understand.

ADVANTAGES OF DATA MINING AND WAREHOUSING

It is use in health care, education, marketing, CRM. It is also use in businesses and banking. Information mining have a great deal of benefits when utilizing in a particular industry. Other than those benefits, information mining additionally has its own disservices e.g., protection, security, and abuse of data.

CHAPTER 2
LITERATURE REVIEW

Enormous information is arising as a very famous topic among experts and researchers. For models, many organizations utilize advanced advances to follow web-based media consistently, along these lines making longitudinal designs of millions of posts, tweets, or surveys George et al. (2014). Contemporaneously, an enormous number of examination papers are devoted on the progressive of large information in various industry and utilize 3Vs to portray huge information, the outrageous volume of information, the wide assortment of information types and the speed at which the information should be processed. Definitely, individuals from very different backgrounds these days are living in the period of enormous information.

In 2014, Bank of America designated another high level position, the Chief Analytics Officer(CAO), to Douglas Hague who is the previous senior VP of merchant analytics(Ferguson, 2014). He unequivocally communicated an acknowledgment of the significance of investigation in the utilization of enormous information, rather than simply writing about data(Ferguson, 2014). Exactly, organizations need to pick up taking investigation past the information and data which could come into the genuine bits of knowledge that sway direction. Sanders (2016 p.28) precisely gave the contrast between enormous information and examination, "Large information without investigation is only a huge measure of information. Examination without enormous information are basically numerical and factual apparatuses and applications". Henceforth, arranged by enormous information, we should place more endeavors on huge information analytics(BDA), which could be characterized as " another age of advances and structures, intended to monetarily remove esteem from exceptionally huge volumes of a wide assortment of information, by empowering high speed catch, disclosure and/or examination." (Carter, 2011).

Alongside the innovation of enormous information investigation being developed bit by bit, the idea of business knowledge emerged through the entire financial globe. Endeavors will more often than not investigate a wide range of profoundly concealed highlights from mass data organizations, with pervasive and wise examination abilities, for example, information mining, process mining, web mining or text mining.

CHAPTER 3

RANDOM FOREST

INTRODUCTION

Arbitrary timberlands or irregular choice woodlands are a group learning strategy for characterization, relapse and different undertakings that works by building a huge number of choice trees at preparing time. For relapse errands, the mean or normal expectation of the singular trees is returned. Random forest is bagged decision trees that built tree ang make large collection of decorrelated trees. its use to improve prediction of data mining. It’s a very enjoyable algorithm to use.

It is a supervised machine learning Algorithm used in classification and regression problems. It takes data randomly from the original dataset and create decision trees and based on average of the output generated by decision trees makes final decision.

Random forest works on the bagging principle.

BAGGING

It is also known as bootstrap aggregation. We create bootstraps by taking information randomly from the original data. The process of combining the output of the Decision tree which we created from the bootstrap and average the majority votes from the decision tree and make decision known as aggregation.

<u>For Example</u>:

A student wants to take admission in a specific course and he takes the advice about the course from the professional of that field and students of that particular course and gets the different voles, now he averages the votes and take decision accordingly.

STEPS INVOLVE IN RANDOM FOREST ALGORITHM:

1. In random forest n number of random records taken from the k number of records from the dataset.
2. Individual decision trees are constructed from each sample.
3. Each decision tree will generate an output.

4. The final out will be considered by majority votes or averaging for classification and regression problems

DECISION TREE

INTRODUCTION

If we can say that decision tree is process of learning a function that maps data item into one of several pre-defined classes. Every classification based on inductive-learning algorithm.in this we create a model known as classifier in which it predicts the available values of their input attributes. Decision tree is a powerful method to solve the real-life mythologies.

EXAMPLES OF CLASSIFICATION

PROBLEMS:

Classifying trends in financial market

Identified objects in large images

ADVANTAGES OF DECISION TREE

One of its advantages is their output is to read

It does not require normalization of data.

It does not require scaling of data

It’s very easy to explain someone like technical teams or stakeholders

DISADVANTAGES OF DECISION TREE

Sometimes its algorithm goes in complex solution

It’s a high time consume model

Its take higher time to train the mode.

PRUNING

Pruning is an information pressure procedure in AI and search calculations that lessens the size of choice trees by eliminating areas of the tree that are non-basic and excess to order cases.

CLASSIFICATION MODEL

In a Classification model, the association among classes and different properties of the examples can be characterized by a flowchart or as complicated and unstructured manual. Data-mining systems limit conversation to formalized, "executable" models of grouping. There are two altogether different manners by which they can be built. - Interviewing the pertinent master or specialists, and most information-based frameworks have been fabricated thusly (regardless of the notable troubles orderly on adopting this strategy.) - Alternatively, various recorded orders may be analyzed and a model developed inductively by summing up from explicit models that are of essential interest for information mining applications.

EXTENDED FOREST

Random forest uses same principle as decision tree. Random forest introduces us bagging. Random forest teaches us how to make leave trees bigger. Random forest use aggregates to improve prediction.

OUT OF THE BOX

It's very popular because it provides out of the box performance.

Random forest has become least variability in their prediction.

HYPERPARAMETER

We should consider hyperparameter when we train models.

There are main hyperparameters.

1. Trees in forest
2. Complexity of tree
3. How to do sampling scheme

TREE IN FOREST

First, we should consider number of trees in random forest.

No of trees should be large to control error.

COMPLEXITY IN TREE

In random forest there are one or more hyperparameter to control the complexity of tree.

SAMPLING SCHEME

In random forest sampling scheme is bootstrapping where total 100 percent data is observed.

TUNNING STRATEGIES

As we introduce more complex algorithm, we should me strategies.

We should know how to proceed our grid search.

SPLITE RULE

We should recall the default split rule which we use in decision tree to build tree.

CONCLUSION

1. Random forest provides out of shell prediction performance.
2. Random forest has more accuracy
3. Random forest also faster than bagging
4. It creates stability between correlation trees

CHAPTER 4

DBSCAN ALGORITHM

We can define DBSCAN as dense region covered or surrounded by the low-density region

It selects the no of clusters automatically to select core there should be 3 points in circle

The center point of circle is core. The neighbor points of core in circle are boundary points. The noise lies anywhere in circle.

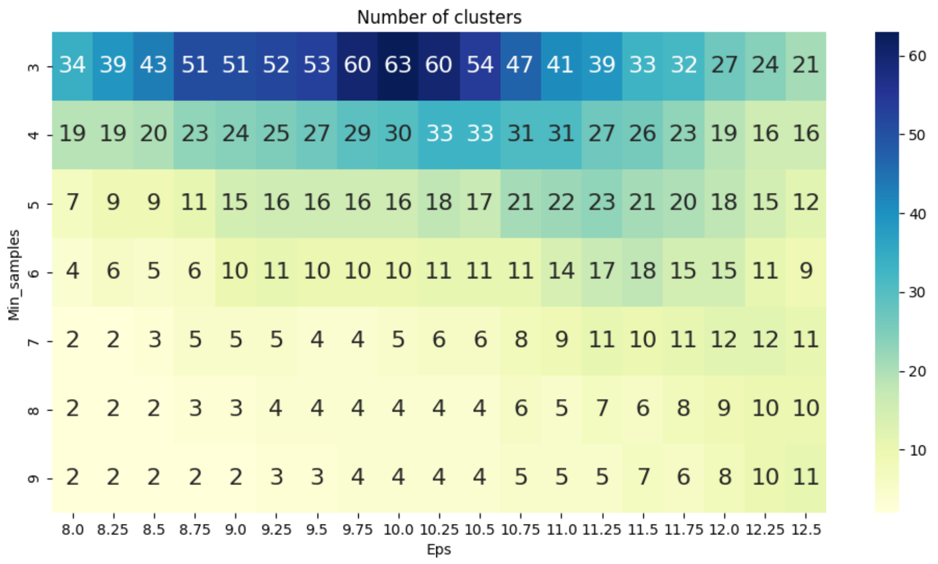
How it works?

You should select a point p randomly from arbitrarily. Find epsilon neighborhood of point p.

If they are core points then the cluster is formed.

Then search for other points in the data set if they turn out to be a border points then move on and search for other points in the database.

Repeat these actions until all the points in the database are processed.



Min_samples \ Eps	8.0	8.25	8.5	8.75	9.0	9.25	9.5	9.75	10.0	10.25	10.5	10.75	11.0	11.25	11.5	11.75	12.0	12.25	12.5
3	34	39	43	51	51	52	53	60	63	60	54	47	41	39	33	32	27	24	21
4	19	19	20	23	24	25	27	29	30	33	33	31	31	27	26	23	19	16	16
5	7	9	9	11	15	16	16	16	16	18	17	21	22	23	21	20	18	15	12
6	4	6	5	6	10	11	10	10	10	11	11	11	14	17	18	15	15	11	9
7	2	2	3	5	5	5	4	4	5	6	6	8	9	11	10	11	12	12	11
8	2	2	2	3	3	4	4	4	4	4	4	6	5	7	6	8	9	10	10
9	2	2	2	2	2	3	3	4	4	4	4	5	5	5	7	6	8	10	11

Figure 1

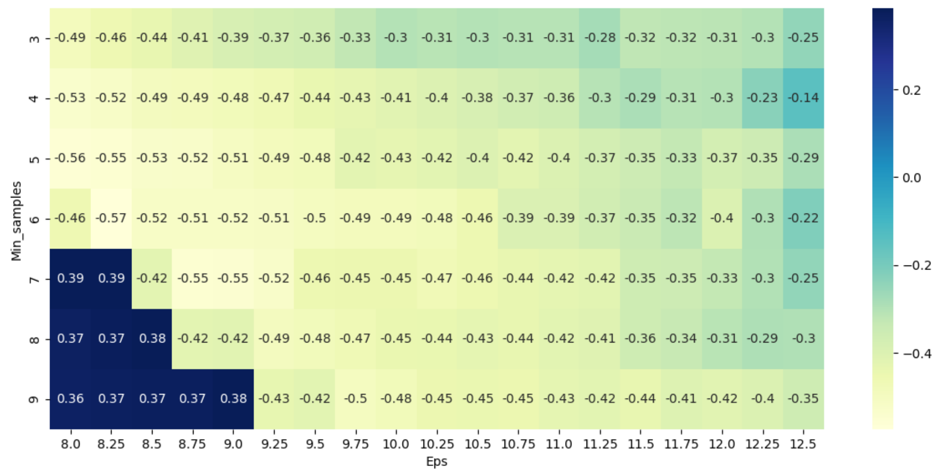


Figure 2

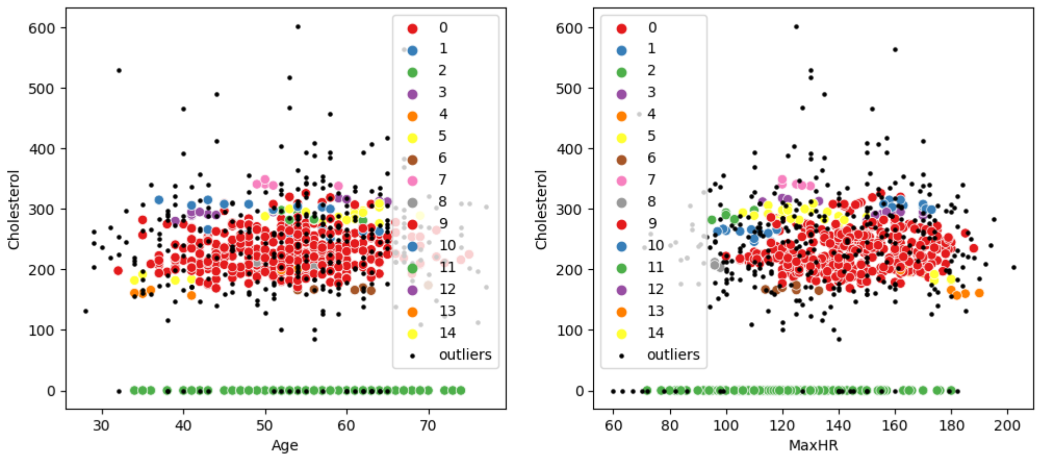


Figure 3

TIME COMPLEXITY

The time complexity is O(m) and in worst case the time complexity will be O(m)^2.

FUZZY K MEANS

Fuzzy k means we can say that it is an observation of soft clustering that can be assigned to multiple clusters.

ADVANTAGES:

- It’s very easy for implement
- Its scale very large datasets
- Guarantee of convergence

DISADVANTAGES:

- Choose k manually it’s a big disadvantage
- Always depends on initial values
- Clustering outliers

CENTER BASED CLUSTERING

We define that goal of center-based clustering is to minimize the objective function. It’s very efficient and work good on large or high dimensional data sets. We assume that cluster should be in convex shaped. We can represent cluster center as cluster. The example of this cluster is gaussian mixtures which is widely use in this clustering. We use Euclidean distance in this clustering.

Error generated in clustering which we can define as the sum of distance from the clusters Centre. K –means clustering use in in center clustering. If we talk about optimization in k-means its procedure is very greedy. In every iteration the of objective function decreases. It’s working depends only on numeric values. If we see the clusters or identified so, we found ball-shaped clusters. The initial values make or depend on result.

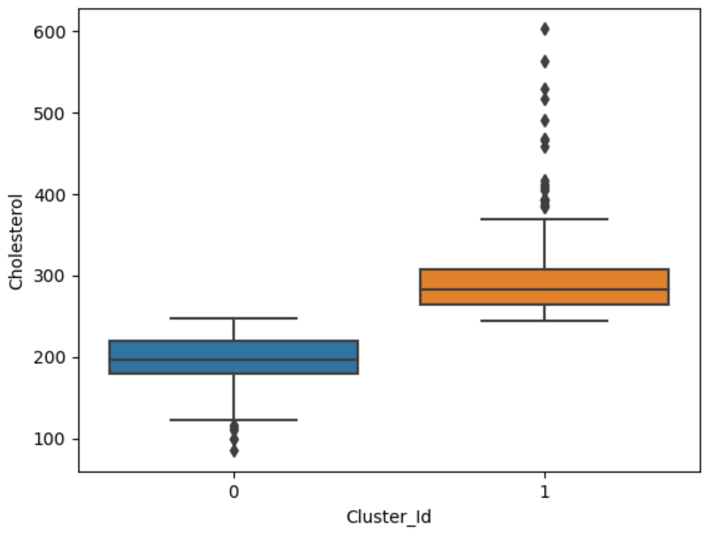


Figure 4

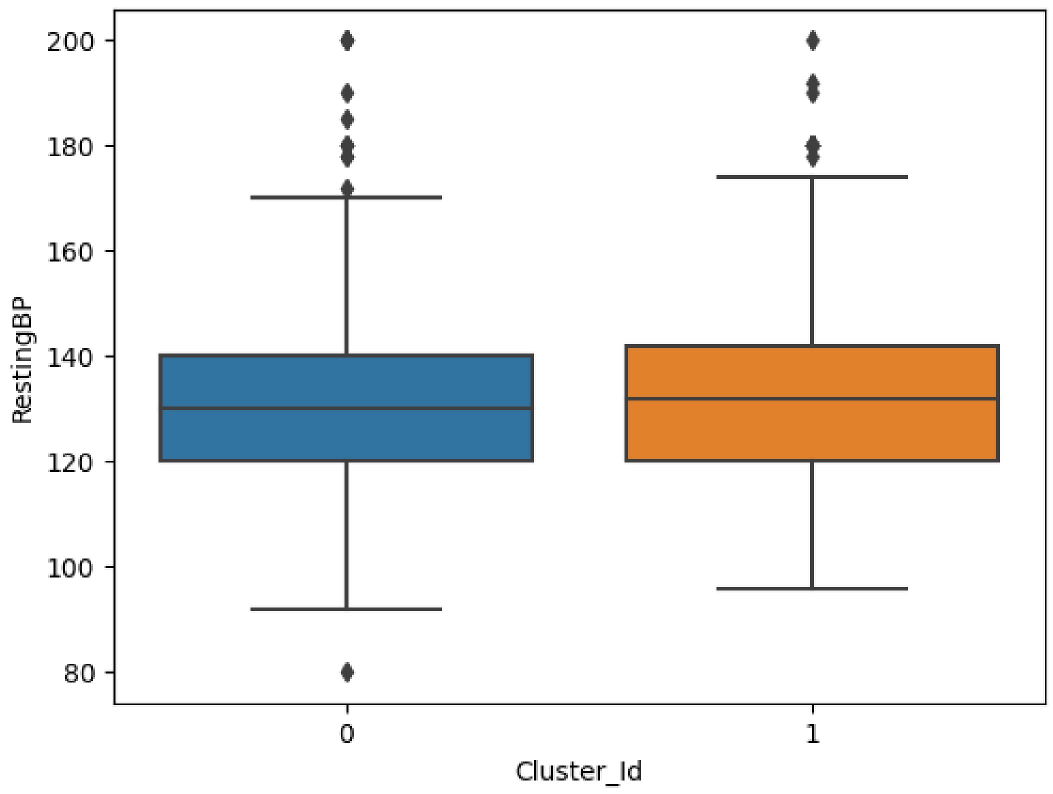


Figure 5

CONCLUSION

In this chapter we learn about DBSCAN, center base clustering and fuzzy k-means technique

CHAPTER 5

DATA PREPROCESSING

It is done to improve the quality of data in data warehouse. Data preprocessing in used for increasing efficiency and mining process is easy. And it is also removed noisy data, incomplete data and inconsistent data. Data in the real world is dirty.

Inconsistent: containing discrepancies in order or names. Data cleaning: It clean the data by filling in the missing values. Sometime noisy data resolving the inconsistency and removing the outliers. Data transformation: It is Data processing techniques that transfer or consolidate the data into alternate appropriate for mining. Data integration: Its is preprocessing method that involves merging of data form different sources in order to form a data solve like data warehouse.

OUTLIER DETECTION

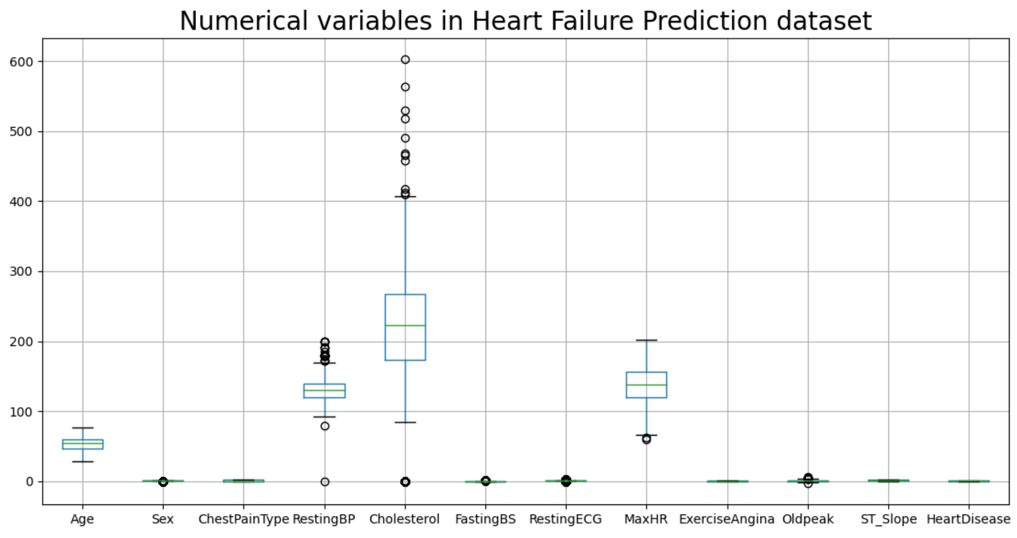


Figure 6

OUTLIER CORRECTION

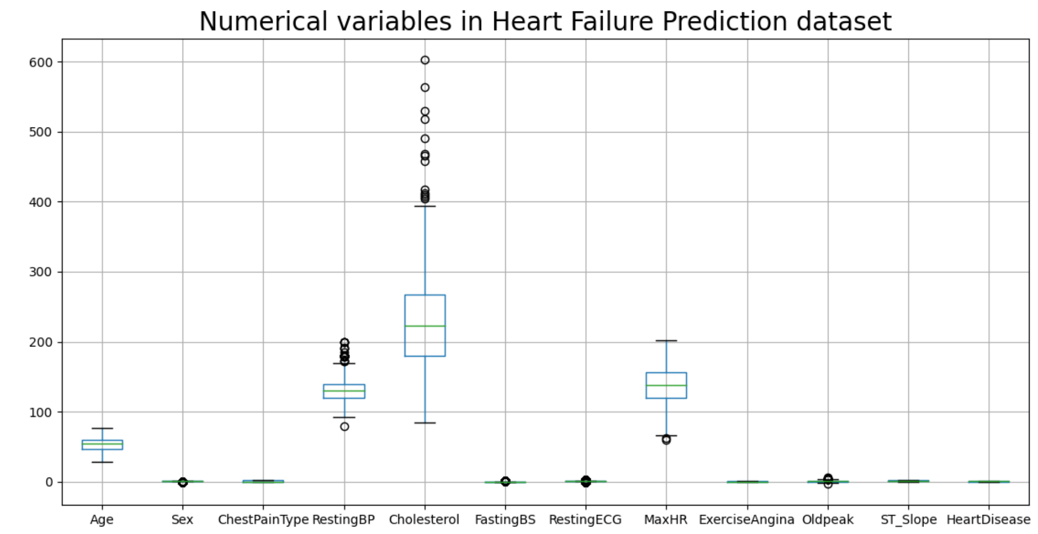


Figure 7

Issues in data integration

INTEGRATION AND OBJECT MATCHING

For example, we have two companies A and B. Company A store employee’s data in name with “employee ID” while company B store employee data in the name with “employee number”. In this scenario both are same attributes but how the system understand that both are same. So, scheme integration and object matching are major problem in such a situation.

In redundancy some unwanted attributes are included in our data that are not needed so we have to remove such data which Are not necessary.

DETECTION AND RESOLUTION OF DATA VALUE CONFLICT

Convert and modify the values. For example if we have if company show the amount in “RUPEES” while other company show in “DOLLAR” such in such a situation we have to correctly modify such a values.

Data reduction: reprocessing techniques that helps in obtaining related representation of data set from the available dataset.

- Integrity of original data should own after non reduction in data value
- It should produce same analytic result as an original data.
- Dimensionality Red: remove redundant attribute.
- Data compression: encoding of data.

DATA REDUCTION STRATEGIES

DATA CUBE AGGREGATION

The lowest level of data cube base cuboid. The aggregation data for an individual entity of interest. Multiple levels of aggregation in data cube.

DIMENSIONALITY REDUCTION

It eliminates redundant attribute which are weakly important across the data.

- Step forward selection.
- Stepwise backward elimination
- Decision tree induction.

NUMEROSITY REDUCTION

Replace the original data with small form of data representation. There are two methods.

PARAMETRIC

Parametric reduction: used to eliminate the data so that only parameter of data is required store instead of actual data.

NON-PARAMETRIC REDUCTION

Non-Parametric reduction: used to store thousand representations of data. It includes histogram, clustering, sampling and data aggregation.

DATA DISCRETIZATION

It divides the range of attribute into intervals so as to reduce number of values for a given continuous attribute.

- Splitting
- Merging
- Supervised
- Un supervised

SPLITTING

It’s a top-down approach where attribute is split in to range of values.

MERGING

It’s a top-down approach, initially we consider all values later remove some during merging.

SUPERVISED

If we know the class which they are lying the is called supervised

UNSUPERVISED

If we have no idea of class which they are lying it is called unsupervised.

CONCLUSION

In this chapter we learn about data preprocessing.

CHAPTER 6

LOGISTIC REGRESSION:

Logistic regression is a method used to recognized a data value based on observations of a data set. Logistic regression now a days become an important tool to use. The approach allows an algorithm being used in a machine learning application. A logistic regression model now a days predicts a dependent data variable by analyzing and make the relationship between one or more existing independent variable. For example, a logistic regression could be used to predict whether a team will win or lose a match or whether a college student will be admitted to a particular university.

PURPOSE AND EXAMPLE OF LOGISTIC REGRESSION

Logistic regression is one of the most commonly used machine learning for binary classification problems, which are problem with two class values, including prediction such as this or that yes or no or A and B.

NON-LINEAR REGRESSION

Nonlinear regression is a form of regression analysis in which data is fit to a model and then expressed as a mathematical function simple linear regression relates two variables (X and Y) with a straight line (y= mx+b), while nonlinear regression relates the two variable is a nonlinear (curved) relationship. The goal of the model is to make the sum of the squares as Small as possible. The sum of squares is a measure that attacks how far the Y observations vary from the nonlinear (curved) function that is used to predict Y. Nonlinear regression modelling is similar to linear regression modelling is that both seek to track a particular response from a set of variables graphically. Nonlinear model is more complicated than linear model to develop because the function is created through a series of approximations that may stem from trial and error.

Logistic regression with one predictor

Example Rizatriptan

Rizatriptan is a headache medicine that narrow blood vessel around the brain it is also used reduce the substance in the body make the headache, nausea and also many symptoms. Rizatriptan is used to treat migraine headache. It is not used to treat the tension or any other headache. Use Rizatriptan at that movement if doctor confirm of migraine headache.

Multiple logistic regression

Multinomial logistic regression is used to predict in probability of categorically membership and categorical placement on dependent variable on a multiple independent variable. The type of dependent variable may be binary or continuous or ratio scale. Multinomial logistic regression is simple logistic regression that allow more than two categories of the dependent of outcome variable. Like binary logistic regression and multinomial logistic regression.

Examples of multiple logistic regression

Example 1 : entering high school student make choices to among in general program like vocational and academic program. Their choice might be modeled using writing score and their social economic status.

Nonlinear regression

It's a form of regression data is fix and fit to a model and then make or expressed as mathematical function if for example of we have a linear regression have two variables (X and Y) with a straight line (y=mx+b), while in nonlinear regression has two variables in a nonlinear relationship.

Examples p24 antigen

These are sometimes called fourth generation tests

One distinctive HIV antigen is a viral protein called p24, making of p24 antigen assays useful in diagnosing primary HIV infection.

AZT therapy

AZT therapy were tested by means of. Newly developed plaque assay in cd4+ heal cells. Fifty percent inhibitory Dose values of meter 0.05 nano meter.

Mk-639 in HIV patients

Mk-639 is a potent and specific HIV protease to have good oral bioavailability. target HIV protease mk-639 is a protease inhibitor used a component of high active antiretroviral therapy to treat HIV infection and AIDS MK-639 appear to have significant dose related antiviral activity and is well tolerated.

CONCLUSION:

In this chapter we talk about logistic regression and find how to determine logistic regression.

CHAPTER 7

FUZZY C-MEAN

INTRODUCTION:

We call it extension of k-means. In fuzzy c-mean it allows you to assigned data set into more than one cluster. In fuzzy c-mean data point has probability to belonging to each cluster. Fuzzy c-mean is better than k-mean.

Fuzzy-c-means clustering can be viewed as a superior calculation contrasted with the k-Means calculation. Not at all like the k-Means calculation where the information focuses only on a place with one group, on account of the Fuzzy-c-means calculation, the data point can have a place with more than one cluster with a probability

OBJECTIVES OF FUZZY-C-MEANS

Mohsen Ghanea. Iranian National Institute for Oceanography. Minimizing objective function means increasing similarity among all the components within an object and reducing similarity between components of one object with others.

MEMBERSHIP VALUES IN THE FUZZY C-MEANS CLUSTERING

Membership grades are allotted to every one of the main items (labels). These participation grades demonstrate how much information focuses have a place with each group. Accordingly, focuses on the edge of a cluster, with lower participation grades, might be in the group less significantly than focuses in the focal point of cluster.

CALCULATION of FUZZY C- MEAN

- First you have to find initial cluster centers by using formula
- After you find centers now you find distance to dataset from each data set
- After that you find distance, you update the new membership matrix using formula
- Do it after and after until you get same centroids.

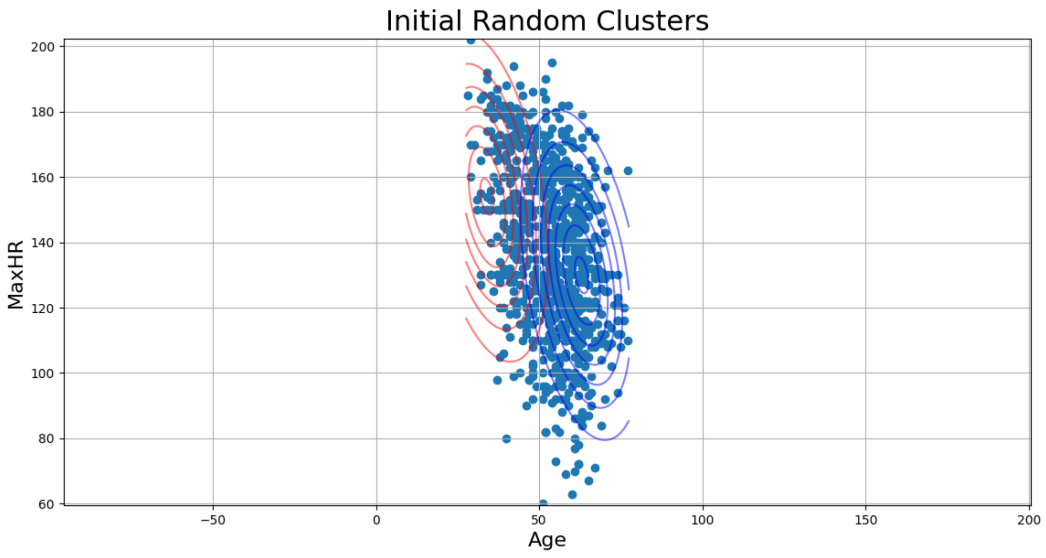


Figure 8

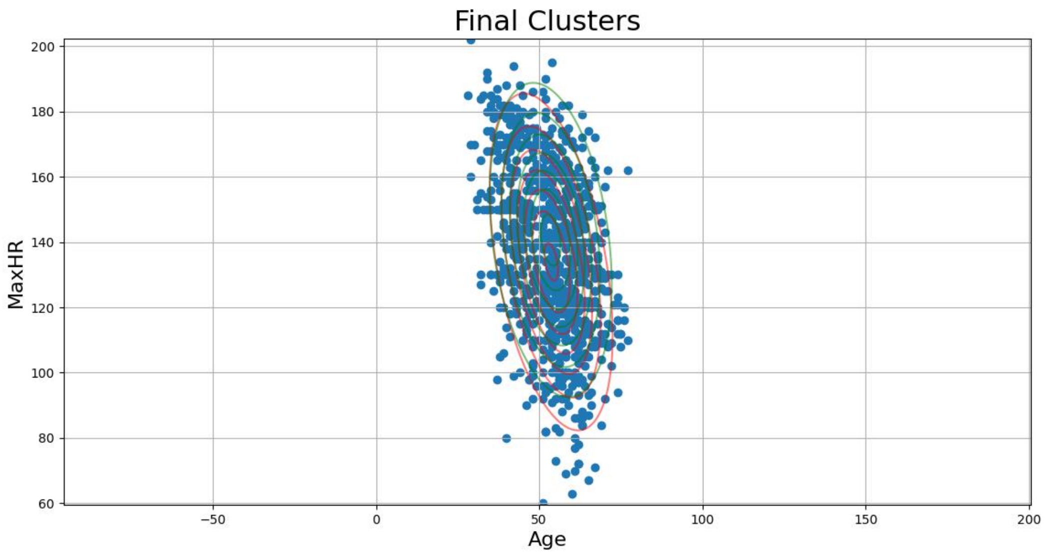


Figure 9

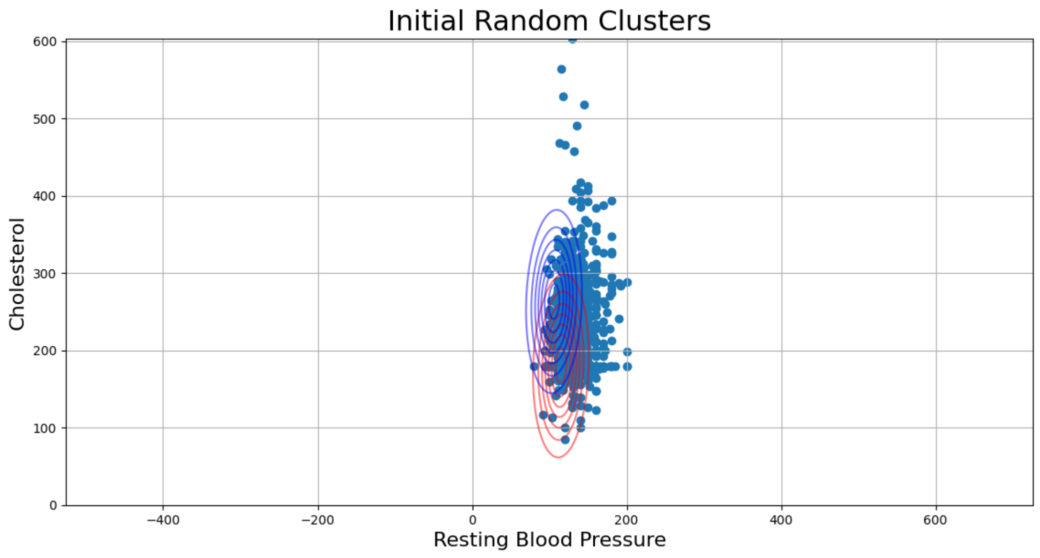


Figure 10

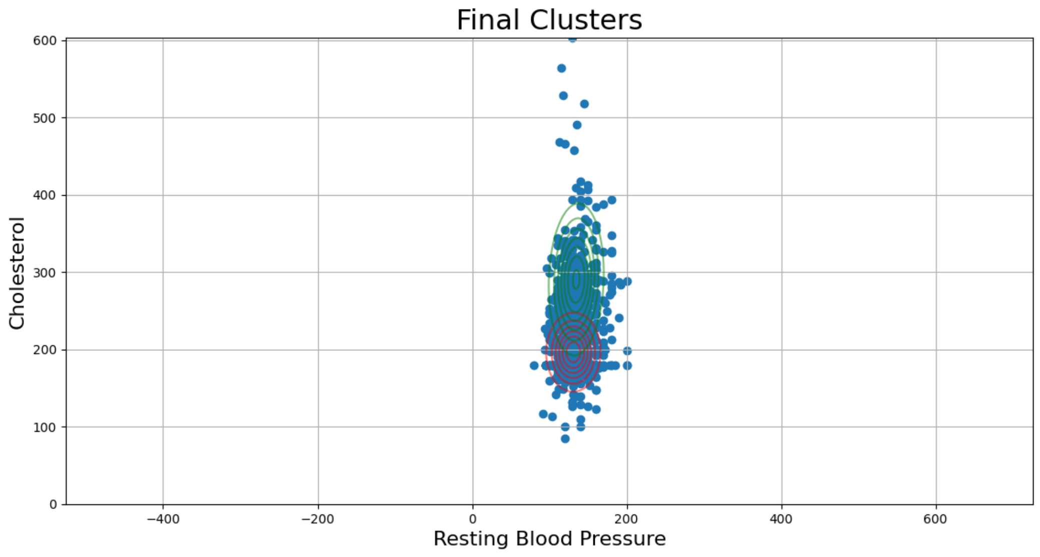


Figure 11

CONCLUSION

Fuzzy c-mean much better than k-means

In fuzzy c-mean cluster belong to each other and there can be more one cluster assigned.

CHAPTER 8

ENTROPY AND CLUSTER

ENTROPY FUNCTION ALGORITHM

Entropy measures the impurity of a collection of examples. It relies from the distribution of the random variable as we know

- S is a collection of training examples.
- P is Proportion of positive example in S
- P is Proportion of negative example in S
- Entropy and information gain

When Information gain measures the expected or outcoming reduction in entropy
Given entropy as a measure of impurity is a collection of training example, the information gains the expected reduction in entropy caused by partitioning the example according to a variable.
More precisely the information gain (S, A) of attribute A relative to a collection of example S in defined as
Gain (S, A) = entropy(S) - Sv\S (entropy S)

PRINCIPAL COMPONENT ANALYSIS

A component analysis is concerned that explain the variance, covariance structure of a set through a few linear combinations of these variables.
Principal Analysis is an unsupervised learning algorithm that is used for the dimensionality reduce or make machine learning. It is a process that converts the observations of correlated into a set of linearly uncorrelated with the help of orthogonal transformation. These new converted features are called the Components. It is one of the famous tools that is used for explore and analysis data and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

FACTOR ANALYSIS

Factor analysis is a technique to combine question or variable to create new factor.
Purpose
Data reduction: to reduce the number of variables to a manageable set of factors

Substantive interpretation: to identify underlying constructs in the data. This make easier to understand the data.

CLUSTER ANALYSIS

Cluster analysis is all about finding observations which are similar so that when you group them together, they should be homogeneous within group. And each group should be quite distinct or in other word any observations of one group should be quite dissimilar to any object of other group.

TYPES OF CLUSTERING METHOD

As we know that for a successful grouping, we need to attain two major goals: one, a similarity between one data point with another and two, a distinction of those similar data points with others which most certainly, heuristically differ from those points. The basis of such divisions begins with our ability to scale large datasets and that's a major beginning point for us. Once we are in it, we are represented with a challenge that our data sustain different kinds of attributes – categorical, continuous data, etc., and we should be able to deal with them. Now, as we know that now a our data these days is not limited in terms of dimensions.

The clusters that we require, should not only be able to distinguish data points but also, they should be inclusive. We are Sure, a distance metric will help a lot but the cluster shape is often limited due to a geometric shape and that's a reason many important data points get excluded. This problem too needs to be taken care of.

In our progress, we seeing that our data is highly "noisy" in nature. Many unwanted features have been residing in the data which makes it rather Herculean task to bring about any similarity between the data I points – leading to the creation of improper groups. As we move towards the end of the line, we are faced with a challenge of business interpretation. The outputs from the clustering algorithm should be understandable and should fit the business criteria and address the business problem correctly.

The various types of clustering are:

Connectivity-based Clustering (Hierarchical clustering)

Centroids-based Clustering (Partitioning methods)

Distribution-based Clustering

Density-based Clustering (Model-based methods)

Fuzzy Clustering

Constraint-based (Supervised Clustering)

CONNECTIVITY-BASED CLUSTERING (HIERARCHICAL CLUSTERING)

Hierarchical Clustering is a method of unsupervised machine learning clustering where it begins with a pre-defined top to bottom hierarchy of clusters. It then proceeds to perform a decomposition of the data I objects based on this hierarchy, hence obtaining the clusters. This method follows two approaches based on the I direction of progress, i.e., whether it is the top-down or bottom-up I flow of creating clusters. These are Divisive Approach and the Agglomerative Approach respectively.

CLUSTERING DENDOGRAM

SINGAL LINKAGE

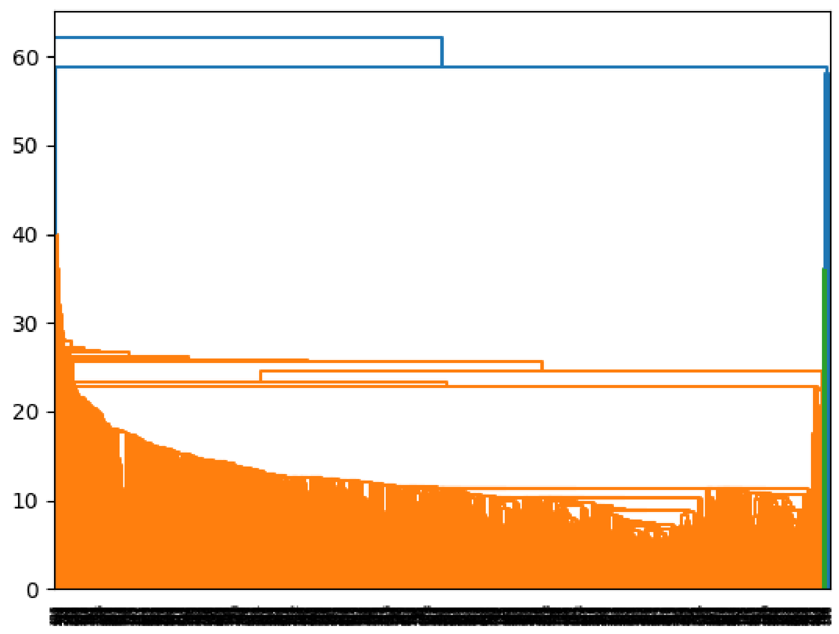


Figure 12

COMPLETE LINKAGE

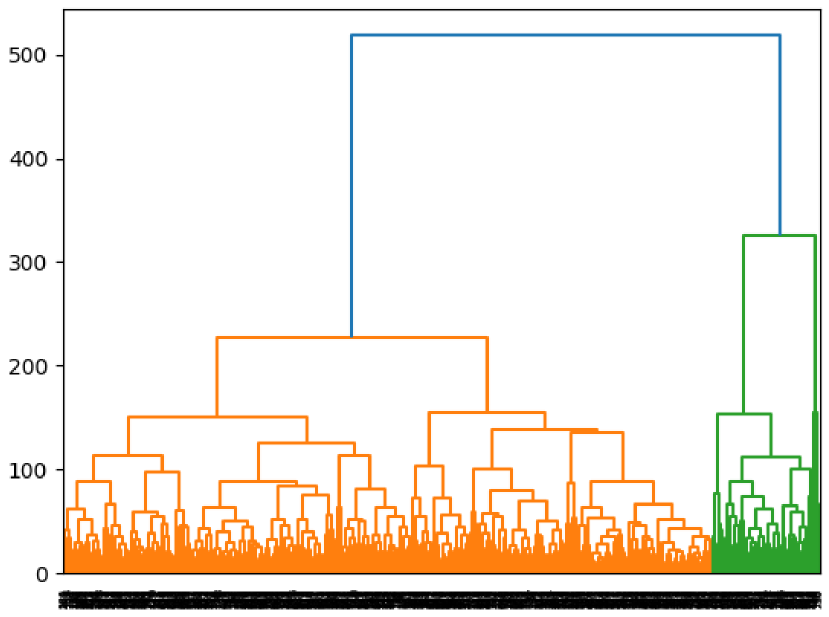


Figure 13

AVERAGE LINKAGE

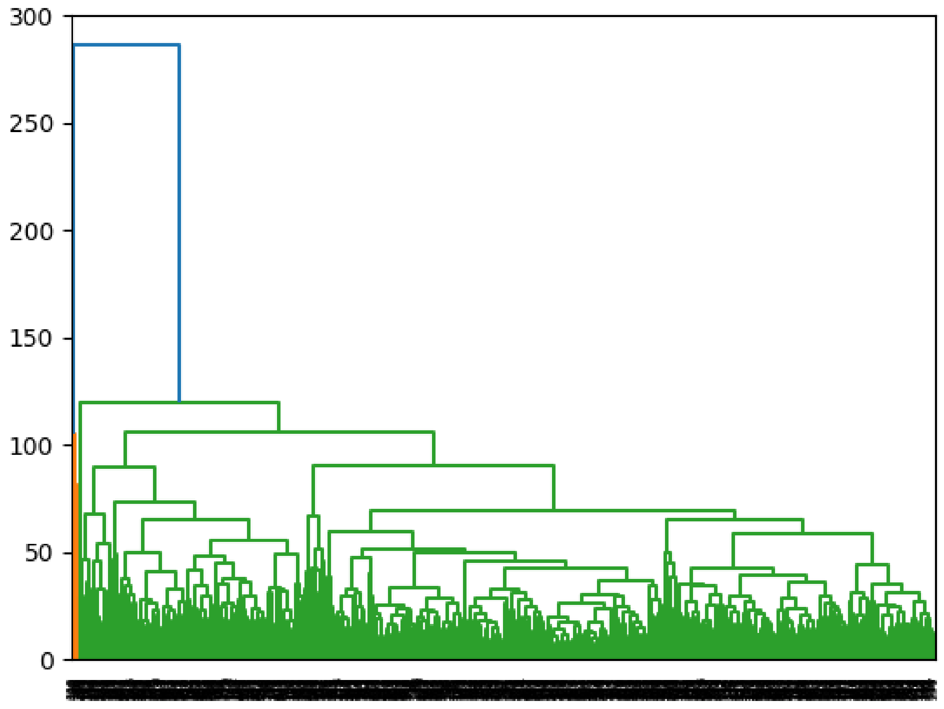


Figure 14

DIVISIVE APPROACH

This approach of hierarchical I clustering follows a top-down approach where we consider that all the data points belong to one large cluster and try I to divide the data into smaller groups based on a termination logic or, a point beyond which there will be no further division of data points. This termination logic can be based on the minimum sum of squares of error inside a cluster.

CONCLUSION

In this chapter we learn about Entropy,Clusters and their types.

CHAPTER 9
RESULTS AND CONCLUSIONS

DATASET

We took this dataset from www.kaggle.com.The dataset is about heart disease. This dataset will allow us to predict that someone has heart disease based on these variables ('Age', 'Sex',' ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease'). The data are organized in rows and columns and each row contains a mixture of rows and columns.

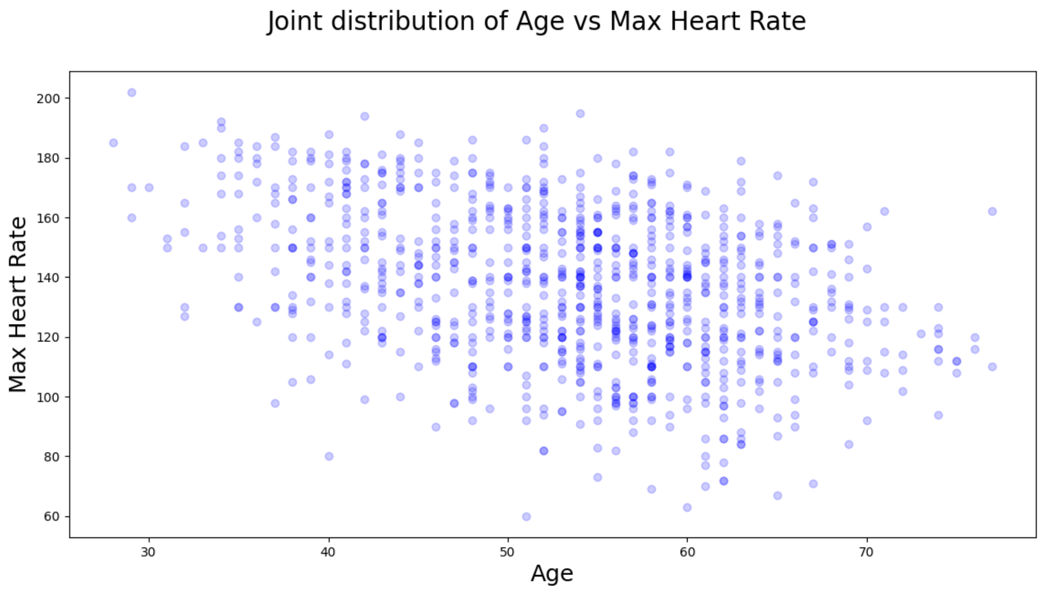


Figure 15

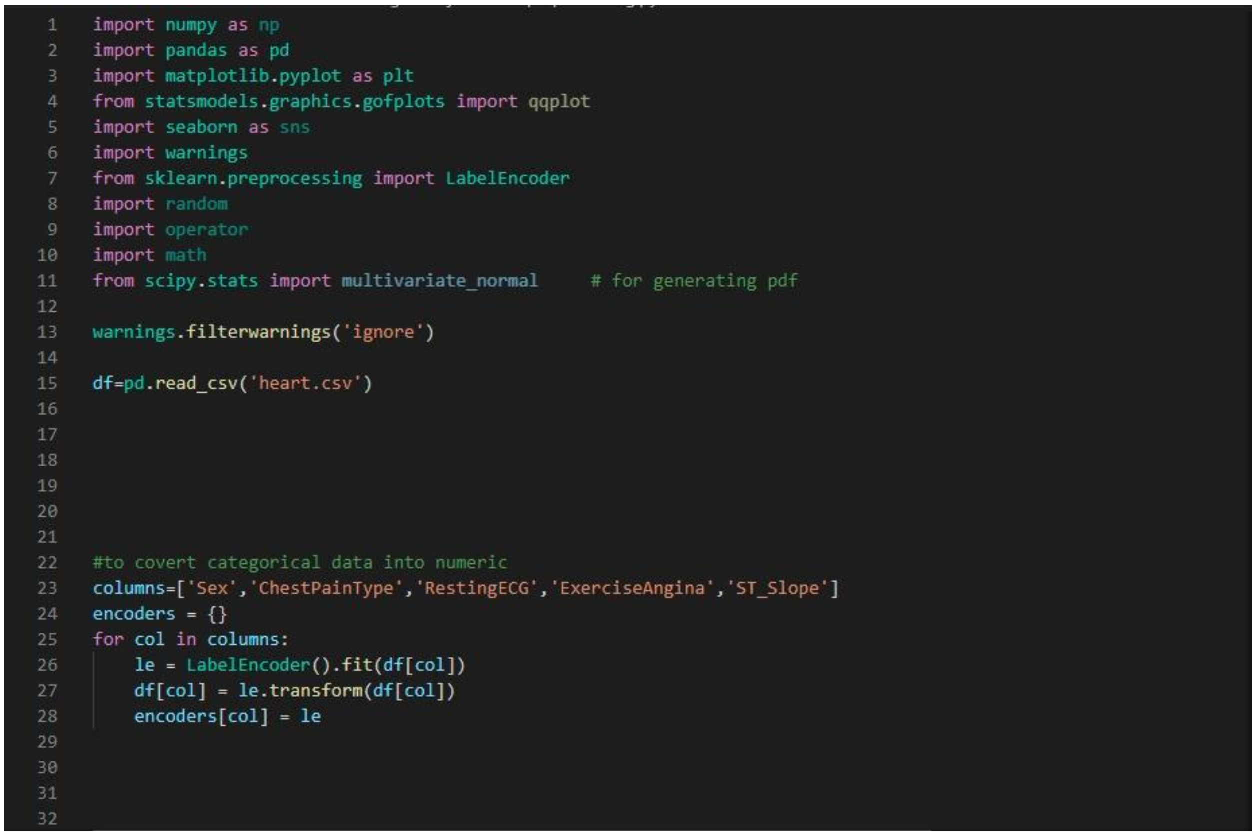
According to our dataset the figure 1 shows that heart rate is increased by age, As we can see the age group of between 50-60 have highest maximum heart rate, which lies under 80-180.

PRE PROCESSING

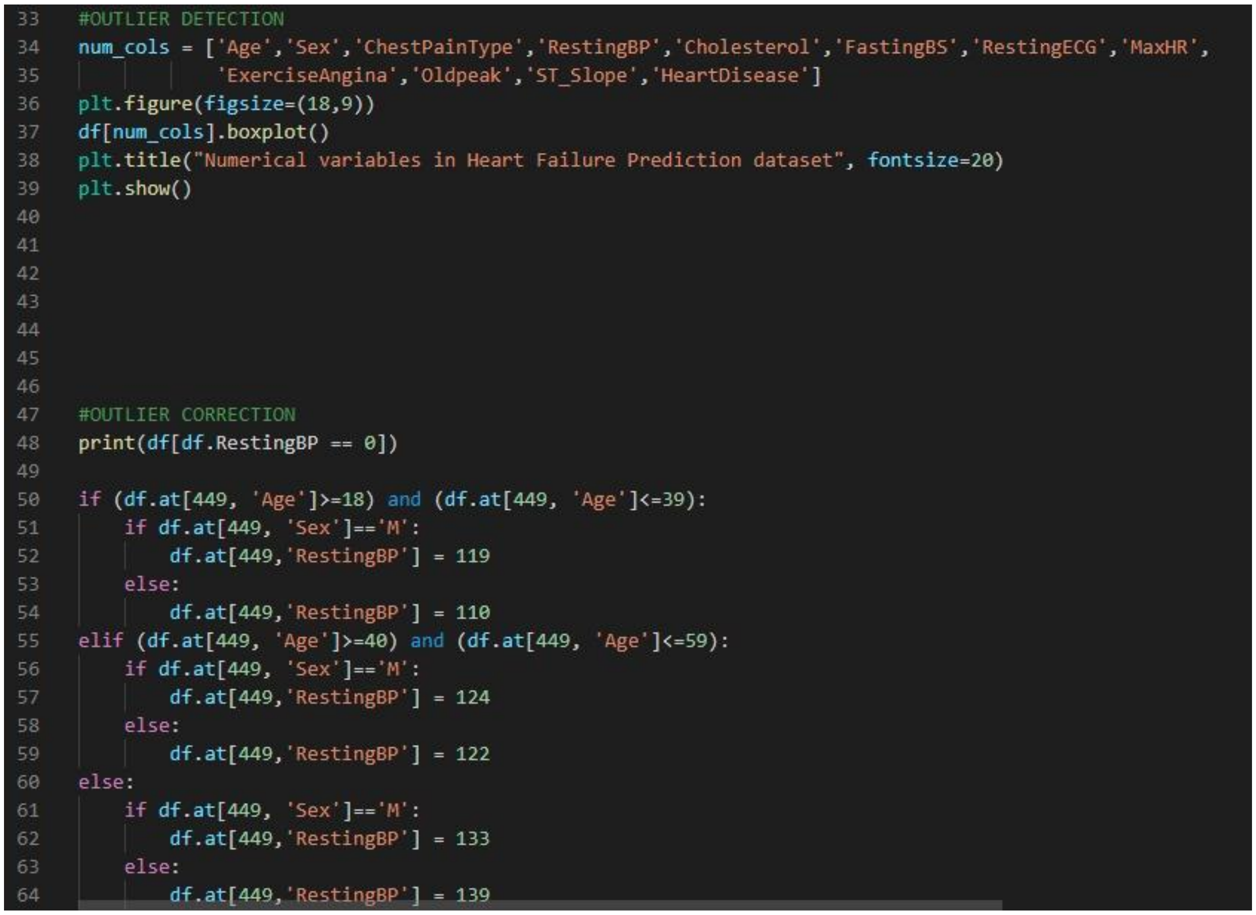
Our dataset contains columns: Age, Sex, ChestPainType, RestingBP, Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina, Oldpeak, ST_Slope, HeartDisease.

We used boxplot to detect the outliers, the outliers were present in RestingBP, Cholesterol and MaxHR attribute. We corrected the outliers in RestingBP by assigning the average BP of a person according to their age and gender. We assigned Average cholesterol 180 to deal with outliers in Cholesterol. The heart rate outliers were not actually outliers as we know a person heart rate can go down to 80, 81. It lies under the normal range of a person heart rate.

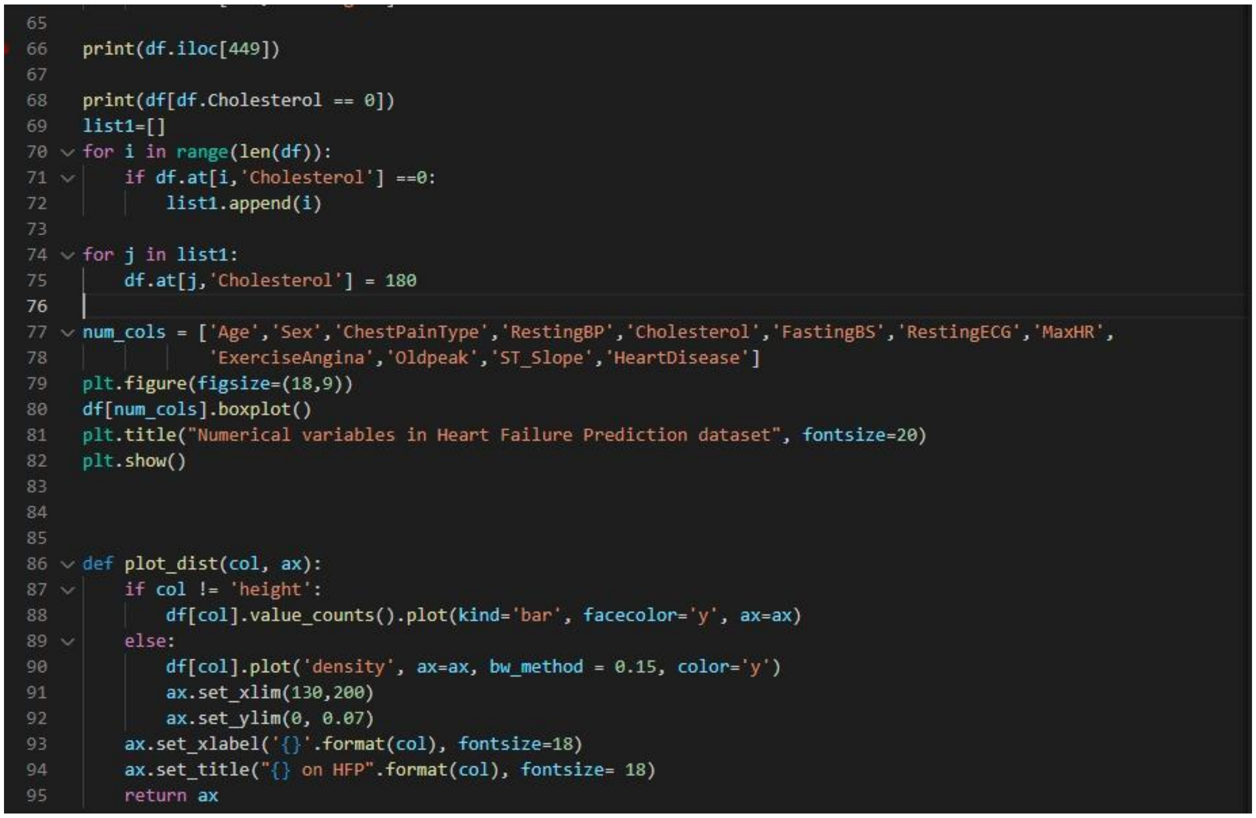
We used boxplot again to show all the removed outliers. We also used different types of plots to understand our data more specifically.



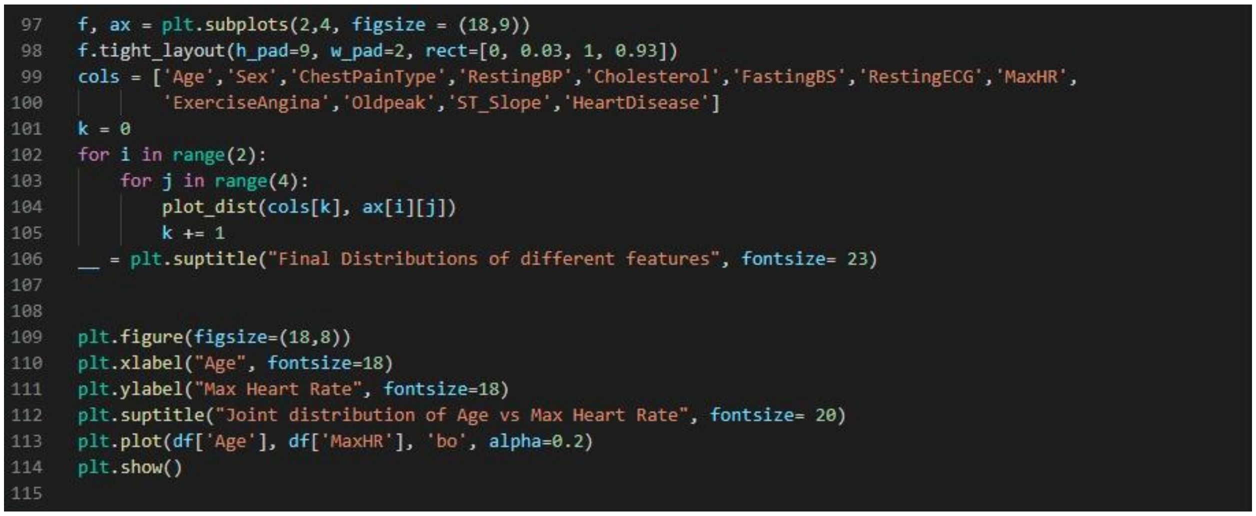
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
import warnings
from sklearn.preprocessing import LabelEncoder
import random
import operator
import math
from scipy.stats import multivariate_normal     # for generating pdf

warnings.filterwarnings('ignore')

df=pd.read_csv('heart.csv')






#to covert categorical data into numeric
columns=['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']
encoders = {}
for col in columns:
    le = LabelEncoder().fit(df[col])
    df[col] = le.transform(df[col])
    encoders[col] = le




```



```python
#OUTLIER DETECTION
num_cols = ['Age','Sex','ChestPainType','RestingBP','Cholesterol','FastingBS','RestingECG','MaxHR',
            'ExerciseAngina','Oldpeak','ST_Slope','HeartDisease']
plt.figure(figsize=(18,9))
df[num_cols].boxplot()
plt.title("Numerical variables in Heart Failure Prediction dataset", fontsize=20)
plt.show()







#OUTLIER CORRECTION
print(df[df.RestingBP == 0])

if (df.at[449, 'Age']>=18) and (df.at[449, 'Age']<=39):
    if df.at[449, 'Sex']=='M':
        df.at[449,'RestingBP'] = 119
    else:
        df.at[449,'RestingBP'] = 110
elif (df.at[449, 'Age']>=40) and (df.at[449, 'Age']<=59):
    if df.at[449, 'Sex']=='M':
        df.at[449,'RestingBP'] = 124
    else:
        df.at[449,'RestingBP'] = 122
else:
    if df.at[449, 'Sex']=='M':
        df.at[449,'RestingBP'] = 133
    else:
        df.at[449,'RestingBP'] = 139
```



```python

print(df.iloc[449])

print(df[df.Cholesterol == 0])
list1=[]
for i in range(len(df)):
    if df.at[i,'Cholesterol'] ==0:
        list1.append(i)

for j in list1:
    df.at[j,'Cholesterol'] = 180

num_cols = ['Age','Sex','ChestPainType','RestingBP','Cholesterol','FastingBS','RestingECG','MaxHR',
            'ExerciseAngina','Oldpeak','ST_Slope','HeartDisease']
plt.figure(figsize=(18,9))
df[num_cols].boxplot()
plt.title("Numerical variables in Heart Failure Prediction dataset", fontsize=20)
plt.show()



def plot_dist(col, ax):
    if col != 'height':
        df[col].value_counts().plot(kind='bar', facecolor='y', ax=ax)
    else:
        df[col].plot('density', ax=ax, bw_method = 0.15, color='y')
        ax.set_xlim(130,200)
        ax.set_ylim(0, 0.07)
    ax.set_xlabel('{}'.format(col), fontsize=18)
    ax.set_title("{} on HFP".format(col), fontsize= 18)
    return ax
```



```python
f, ax = plt.subplots(2,4, figsize = (18,9))
f.tight_layout(h_pad=9, w_pad=2, rect=[0, 0.03, 1, 0.93])
cols = ['Age','Sex','ChestPainType','RestingBP','Cholesterol','FastingBS','RestingECG','MaxHR',
        'ExerciseAngina','Oldpeak','ST_Slope','HeartDisease']
k = 0
for i in range(2):
    for j in range(4):
        plot_dist(cols[k], ax[i][j])
        k += 1
__ = plt.suptitle("Final Distributions of different features", fontsize= 23)


plt.figure(figsize=(18,8))
plt.xlabel("Age", fontsize=18)
plt.ylabel("Max Heart Rate", fontsize=18)
plt.suptitle("Joint distribution of Age vs Max Heart Rate", fontsize= 20)
plt.plot(df['Age'], df['MaxHR'], 'bo', alpha=0.2)
plt.show()

```

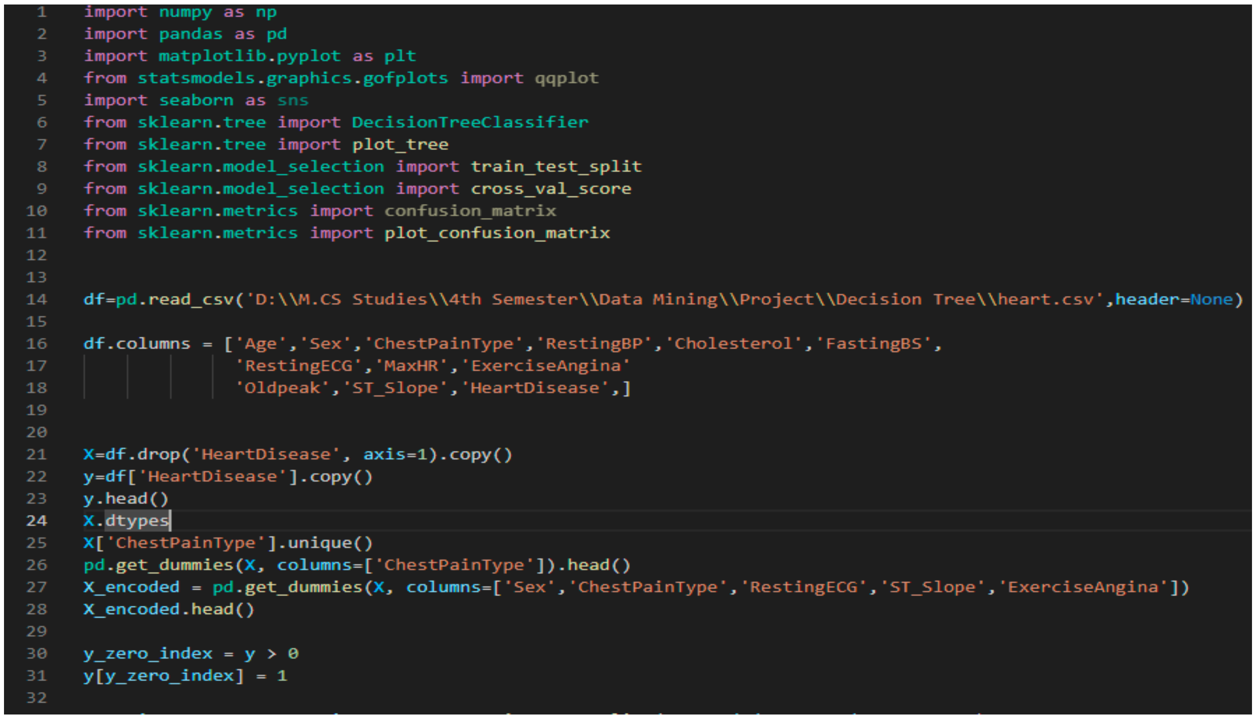
DECISION TREE

The above dataset is used in Decision tree.

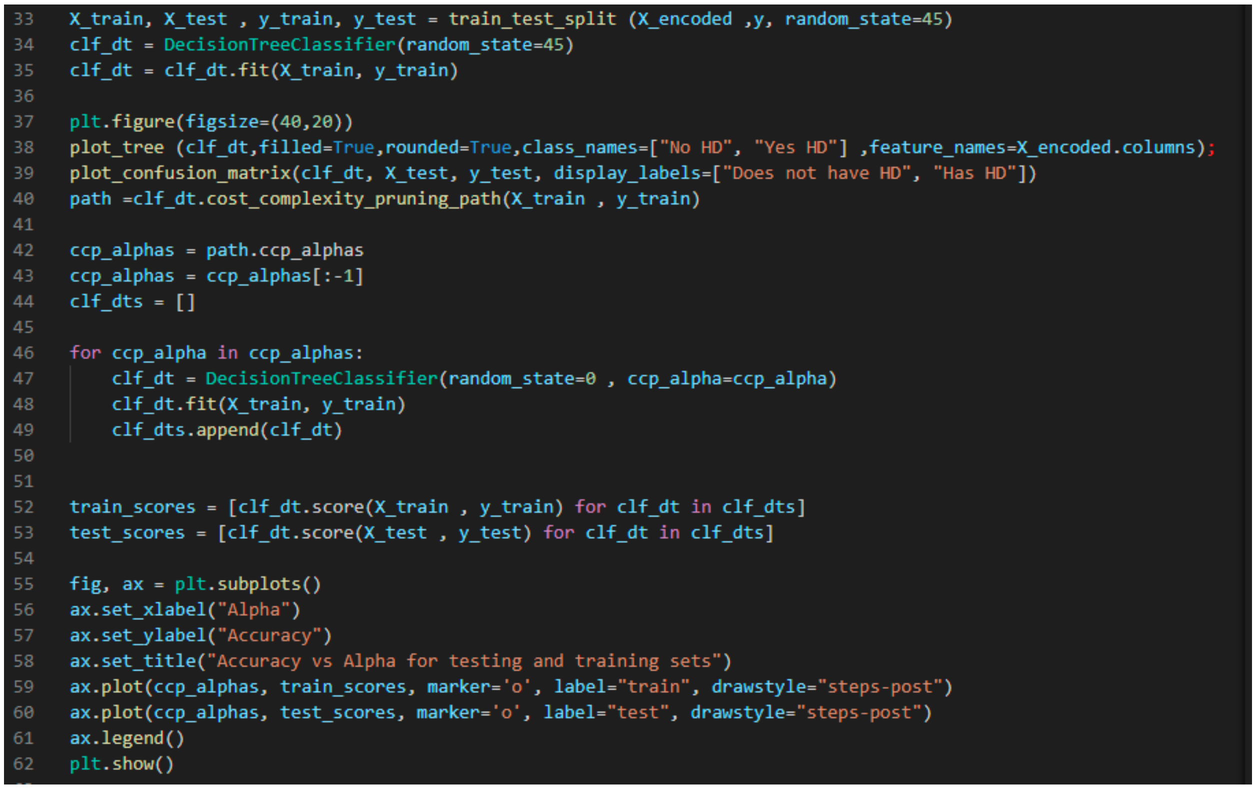
We have split the data into dependent and independent variables. We have used the data in columns to predict one variable (Heart Disease).

We have used One Hot Encoding (A *one hot encoding* is a representation of categorical variables as binary vectors).

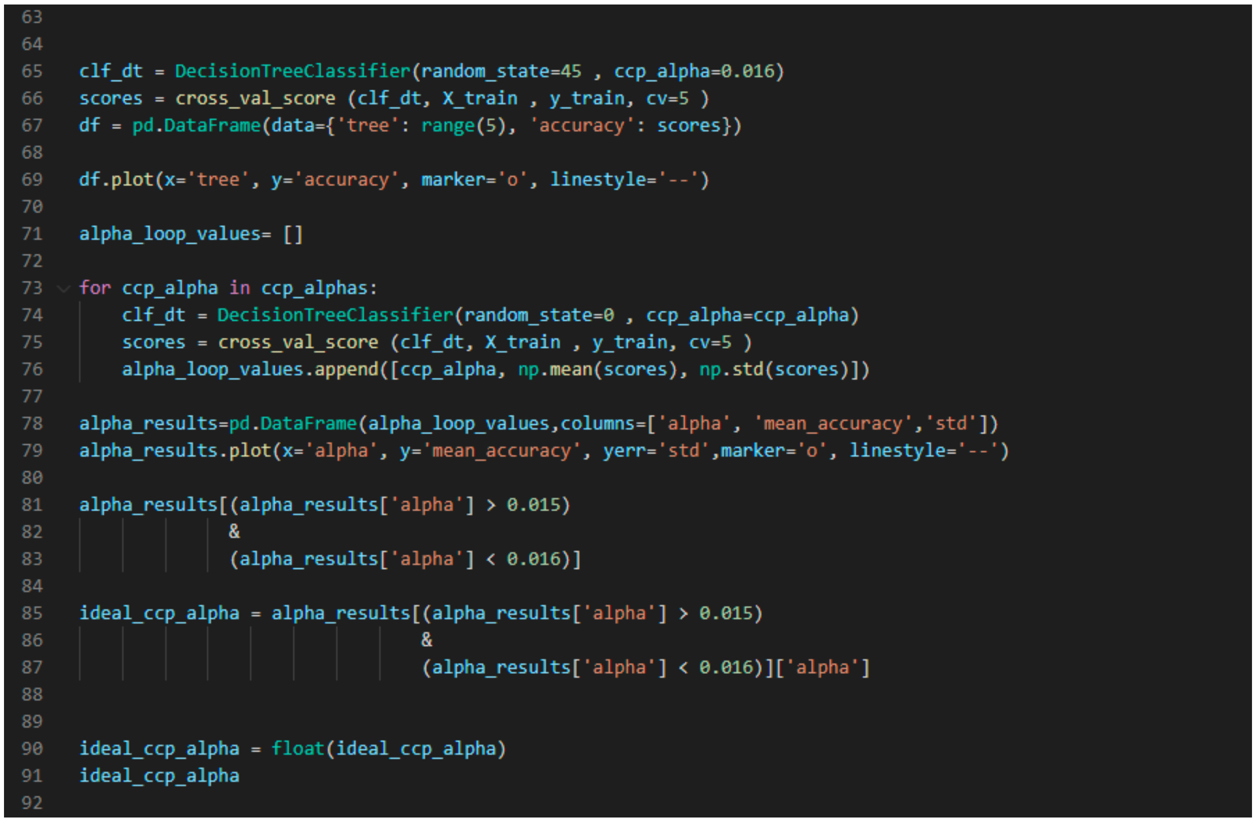
We split the data into training and testing sets and we plot a tree and it was huge so, then we use cost complexity pruning to simplify the tree which has more accuracy and less complicated to understand.



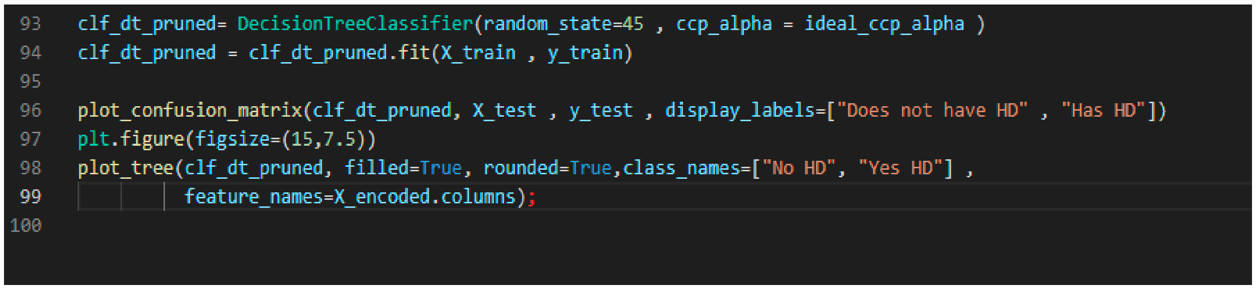
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix


df=pd.read_csv('D:\\M.CS Studies\\4th Semester\\Data Mining\\Project\\Decision Tree\\heart.csv',header=None)

df.columns = ['Age','Sex','ChestPainType','RestingBP','Cholesterol','FastingBS',
              'RestingECG','MaxHR','ExerciseAngina'
              'Oldpeak','ST_Slope','HeartDisease',]


X=df.drop('HeartDisease', axis=1).copy()
y=df['HeartDisease'].copy()
y.head()
X.dtypes
X['ChestPainType'].unique()
pd.get_dummies(X, columns=['ChestPainType']).head()
X_encoded = pd.get_dummies(X, columns=['Sex','ChestPainType','RestingECG','ST_Slope','ExerciseAngina'])
X_encoded.head()

y_zero_index = y > 0
y[y_zero_index] = 1

```



```python
X_train, X_test , y_train, y_test = train_test_split (X_encoded ,y, random_state=45)
clf_dt = DecisionTreeClassifier(random_state=45)
clf_dt = clf_dt.fit(X_train, y_train)

plt.figure(figsize=(40,20))
plot_tree (clf_dt,filled=True,rounded=True,class_names=["No HD", "Yes HD"] ,feature_names=X_encoded.columns);
plot_confusion_matrix(clf_dt, X_test, y_test, display_labels=["Does not have HD", "Has HD"])
path =clf_dt.cost_complexity_pruning_path(X_train , y_train)

ccp_alphas = path.ccp_alphas
ccp_alphas = ccp_alphas[:-1]
clf_dts = []

for ccp_alpha in ccp_alphas:
    clf_dt = DecisionTreeClassifier(random_state=0 , ccp_alpha=ccp_alpha)
    clf_dt.fit(X_train, y_train)
    clf_dts.append(clf_dt)


train_scores = [clf_dt.score(X_train , y_train) for clf_dt in clf_dts]
test_scores = [clf_dt.score(X_test , y_test) for clf_dt in clf_dts]

fig, ax = plt.subplots()
ax.set_xlabel("Alpha")
ax.set_ylabel("Accuracy")
ax.set_title("Accuracy vs Alpha for testing and training sets")
ax.plot(ccp_alphas, train_scores, marker='o', label="train", drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker='o', label="test", drawstyle="steps-post")
ax.legend()
plt.show()
```



```python


clf_dt = DecisionTreeClassifier(random_state=45 , ccp_alpha=0.016)
scores = cross_val_score (clf_dt, X_train , y_train, cv=5 )
df = pd.DataFrame(data={'tree': range(5), 'accuracy': scores})

df.plot(x='tree', y='accuracy', marker='o', linestyle='--')

alpha_loop_values= []

for ccp_alpha in ccp_alphas:
    clf_dt = DecisionTreeClassifier(random_state=0 , ccp_alpha=ccp_alpha)
    scores = cross_val_score (clf_dt, X_train , y_train, cv=5 )
    alpha_loop_values.append([ccp_alpha, np.mean(scores), np.std(scores)])

alpha_results=pd.DataFrame(alpha_loop_values,columns=['alpha', 'mean_accuracy','std'])
alpha_results.plot(x='alpha', y='mean_accuracy', yerr='std',marker='o', linestyle='--')

alpha_results[(alpha_results['alpha'] > 0.015)
              &
              (alpha_results['alpha'] < 0.016)]

ideal_ccp_alpha = alpha_results[(alpha_results['alpha'] > 0.015)
                                &
                                (alpha_results['alpha'] < 0.016)]['alpha']


ideal_ccp_alpha = float(ideal_ccp_alpha)
ideal_ccp_alpha

```

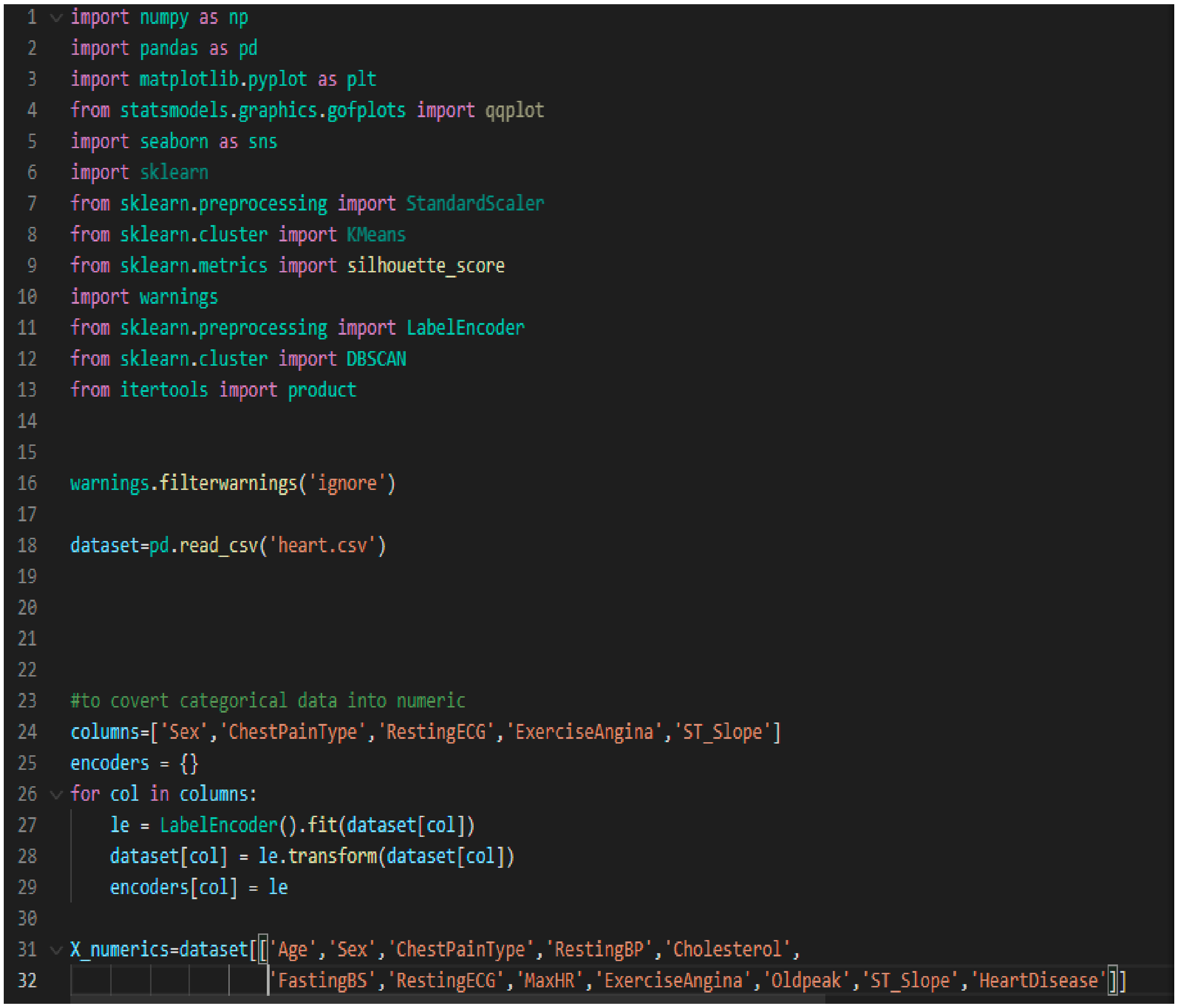


```
clf_dt_pruned= DecisionTreeClassifier(random_state=45 , ccp_alpha = ideal_ccp_alpha )
clf_dt_pruned = clf_dt_pruned.fit(X_train , y_train)

plot_confusion_matrix(clf_dt_pruned, X_test , y_test , display_labels=["Does not have HD" , "Has HD"])
plt.figure(figsize=(15,7.5))
plot_tree(clf_dt_pruned, filled=True, rounded=True,class_names=["No HD", "Yes HD"] ,
        feature_names=X_encoded.columns);
```

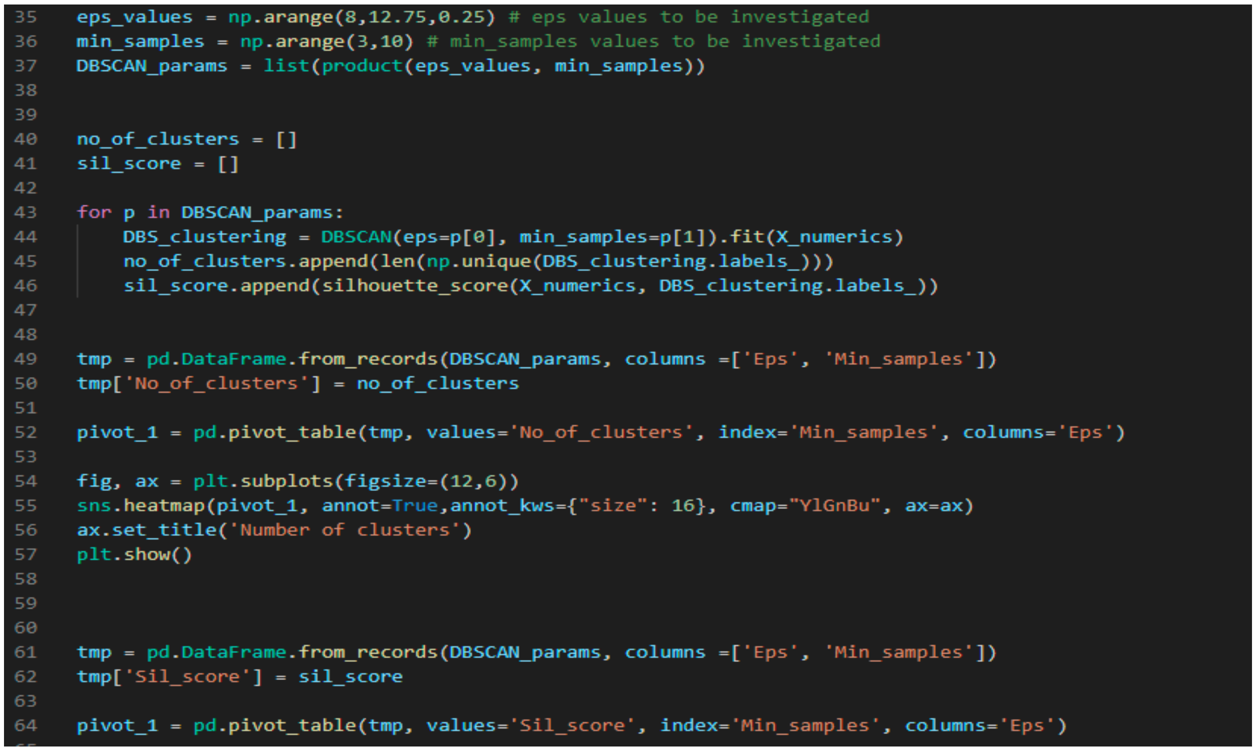
DBSCAN

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. In this type of clustering method, clusters are formed by locating all the points in a dense region surrounded low density points. This algorithm uses epsilon (ε) behaves as a radius of a circle, and minimum points which tells that minimum how many points are required to form a cluster. When a cluster is formed by selecting the core point, then all the points having shorter distance from the epsilon are added to the cluster as boundary points. Continue the process until all the points are processed. Euclidean distance formula is used to find the distance between points and epsilon. It contains noise points also which behave as outliers.

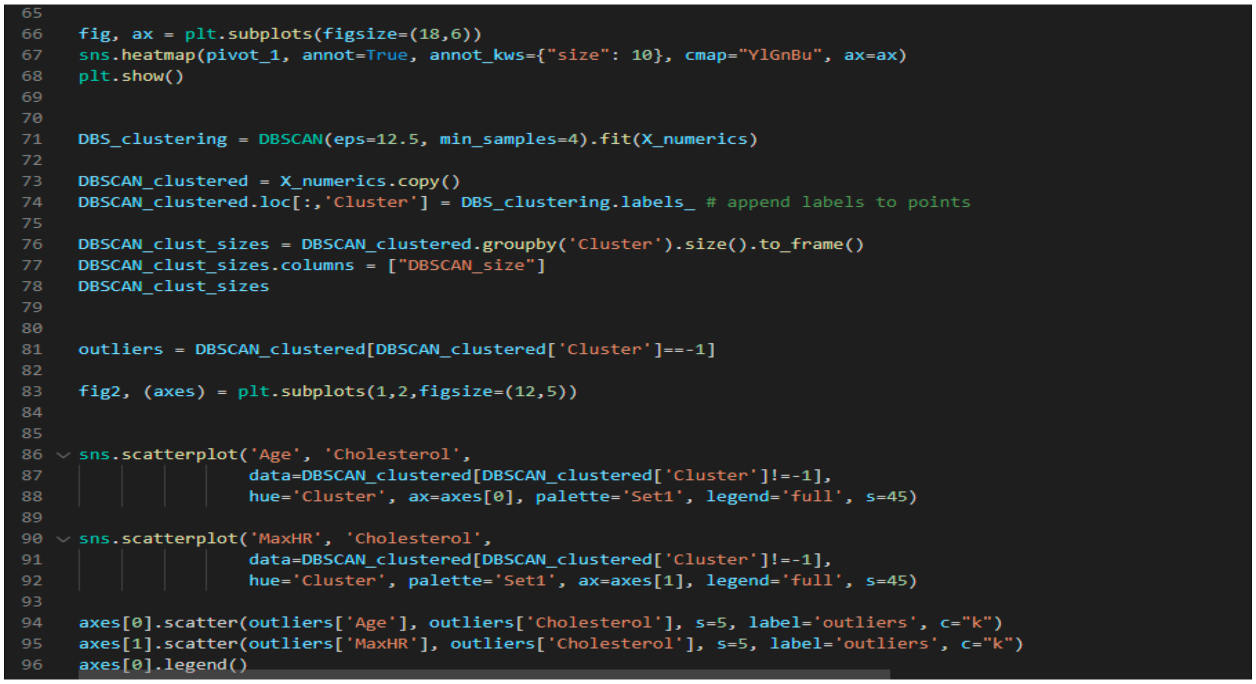
We implemented DBSCAN on our dataset. We created matrix of investigated combinations to select the best then generated the clusters. We used heatplot to show how many clusters are formed. Then we found the size of each cluster to have a better idea. In last 15 clusters were formed. We used Scatter Plot to show the cluster.



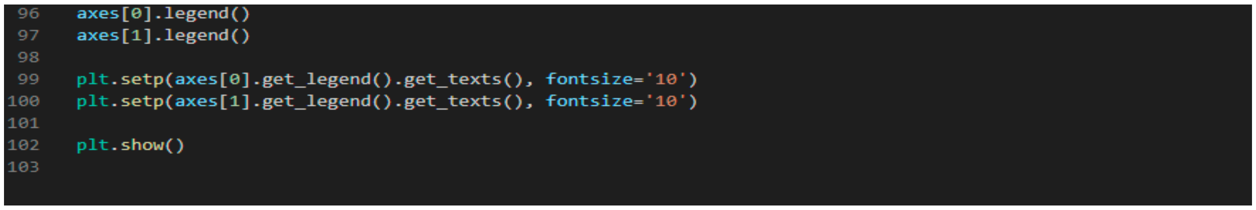
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import warnings
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import DBSCAN
from itertools import product


warnings.filterwarnings('ignore')

dataset=pd.read_csv('heart.csv')




#to covert categorical data into numeric
columns=['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']
encoders = {}
for col in columns:
    le = LabelEncoder().fit(dataset[col])
    dataset[col] = le.transform(dataset[col])
    encoders[col] = le

X_numerics=dataset[['Age','Sex','ChestPainType','RestingBP','Cholesterol',
                    'FastingBS','RestingECG','MaxHR','ExerciseAngina','Oldpeak','ST_Slope','HeartDisease']]
```



```python
eps_values = np.arange(8,12.75,0.25) # eps values to be investigated
min_samples = np.arange(3,10) # min_samples values to be investigated
DBSCAN_params = list(product(eps_values, min_samples))


no_of_clusters = []
sil_score = []

for p in DBSCAN_params:
    DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_numerics)
    no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
    sil_score.append(silhouette_score(X_numerics, DBS_clustering.labels_))


tmp = pd.DataFrame.from_records(DBSCAN_params, columns =['Eps', 'Min_samples'])
tmp['No_of_clusters'] = no_of_clusters

pivot_1 = pd.pivot_table(tmp, values='No_of_clusters', index='Min_samples', columns='Eps')

fig, ax = plt.subplots(figsize=(12,6))
sns.heatmap(pivot_1, annot=True,annot_kws={"size": 16}, cmap="YlGnBu", ax=ax)
ax.set_title('Number of clusters')
plt.show()



tmp = pd.DataFrame.from_records(DBSCAN_params, columns =['Eps', 'Min_samples'])
tmp['Sil_score'] = sil_score

pivot_1 = pd.pivot_table(tmp, values='Sil_score', index='Min_samples', columns='Eps')
```



```python

fig, ax = plt.subplots(figsize=(18,6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 10}, cmap="YlGnBu", ax=ax)
plt.show()


DBS_clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_numerics)

DBSCAN_clustered = X_numerics.copy()
DBSCAN_clustered.loc[:,'Cluster'] = DBS_clustering.labels_ # append labels to points

DBSCAN_clust_sizes = DBSCAN_clustered.groupby('Cluster').size().to_frame()
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes


outliers = DBSCAN_clustered[DBSCAN_clustered['Cluster']==-1]

fig2, (axes) = plt.subplots(1,2,figsize=(12,5))


sns.scatterplot('Age', 'Cholesterol',
                data=DBSCAN_clustered[DBSCAN_clustered['Cluster']!=-1],
                hue='Cluster', ax=axes[0], palette='Set1', legend='full', s=45)

sns.scatterplot('MaxHR', 'Cholesterol',
                data=DBSCAN_clustered[DBSCAN_clustered['Cluster']!=-1],
                hue='Cluster', palette='Set1', ax=axes[1], legend='full', s=45)

axes[0].scatter(outliers['Age'], outliers['Cholesterol'], s=5, label='outliers', c="k")
axes[1].scatter(outliers['MaxHR'], outliers['Cholesterol'], s=5, label='outliers', c="k")
axes[0].legend()
```



```python
axes[0].legend()
axes[1].legend()

plt.setp(axes[0].get_legend().get_texts(), fontsize='10')
plt.setp(axes[1].get_legend().get_texts(), fontsize='10')

plt.show()

```

K-MEANS CLUSTERING

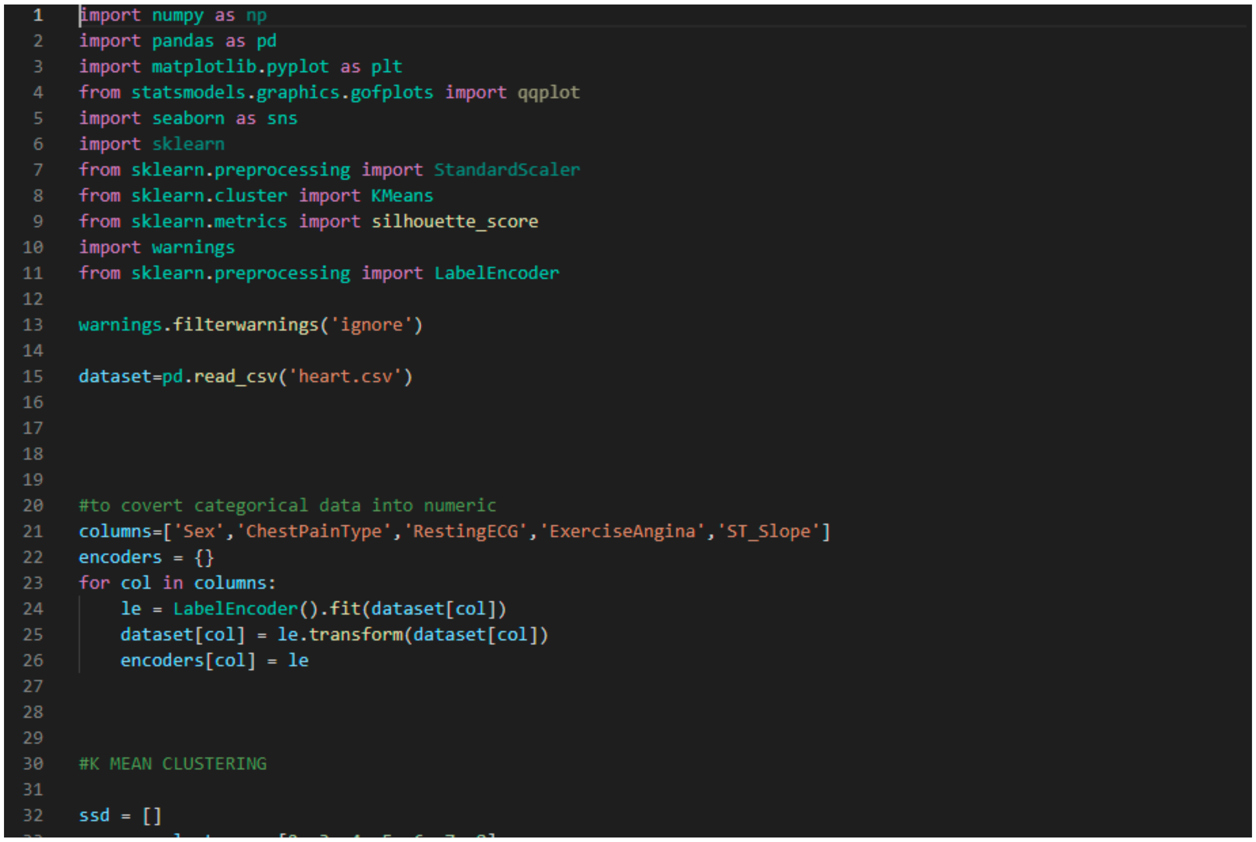
K-means clustering is one of the simplest unsupervised machine learning algorithms.

First we initialize clusters by selecting the random points from the data.

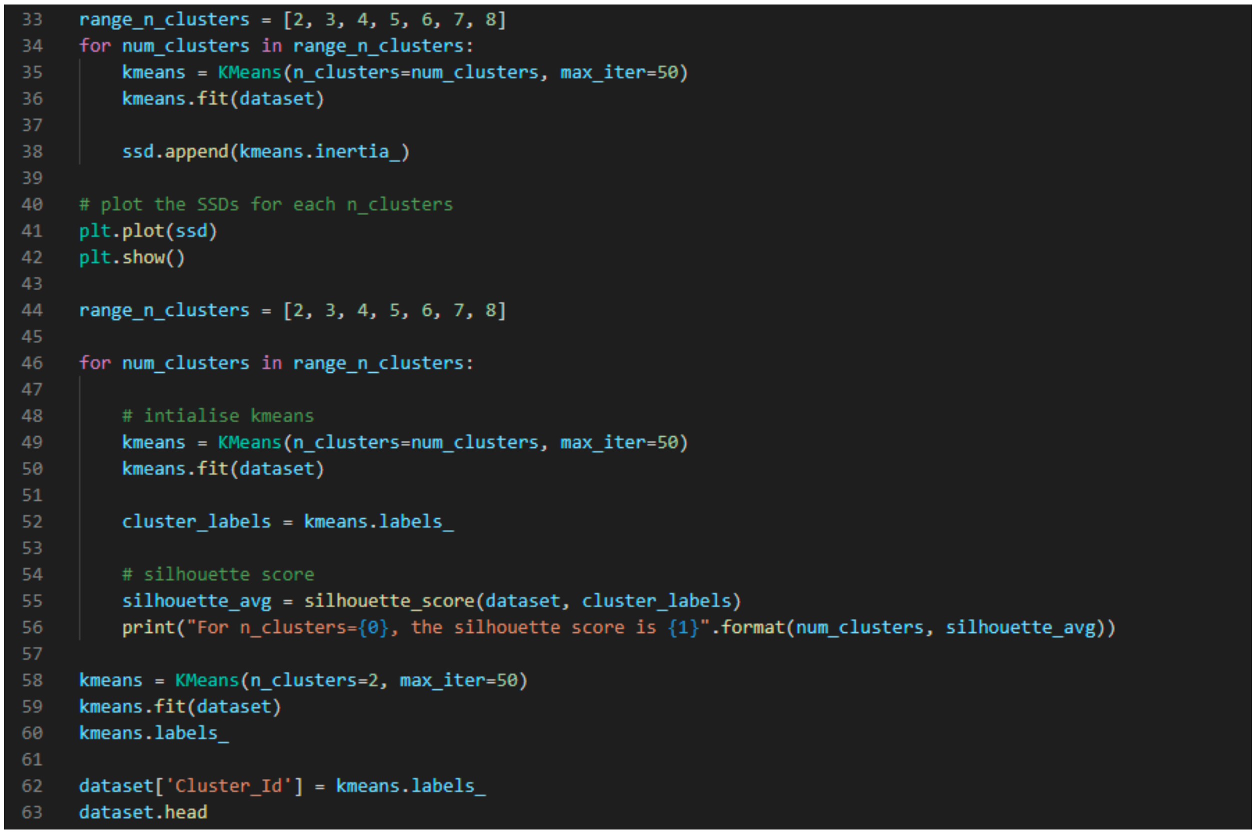
Then calculate distance to categorize each point to the cluster having shorter distance from it, then we recalculate the clusters by finding average between the cluster point and the point that have recently been added to the cluster.

Continue this process until all the points have been added to the clusters.

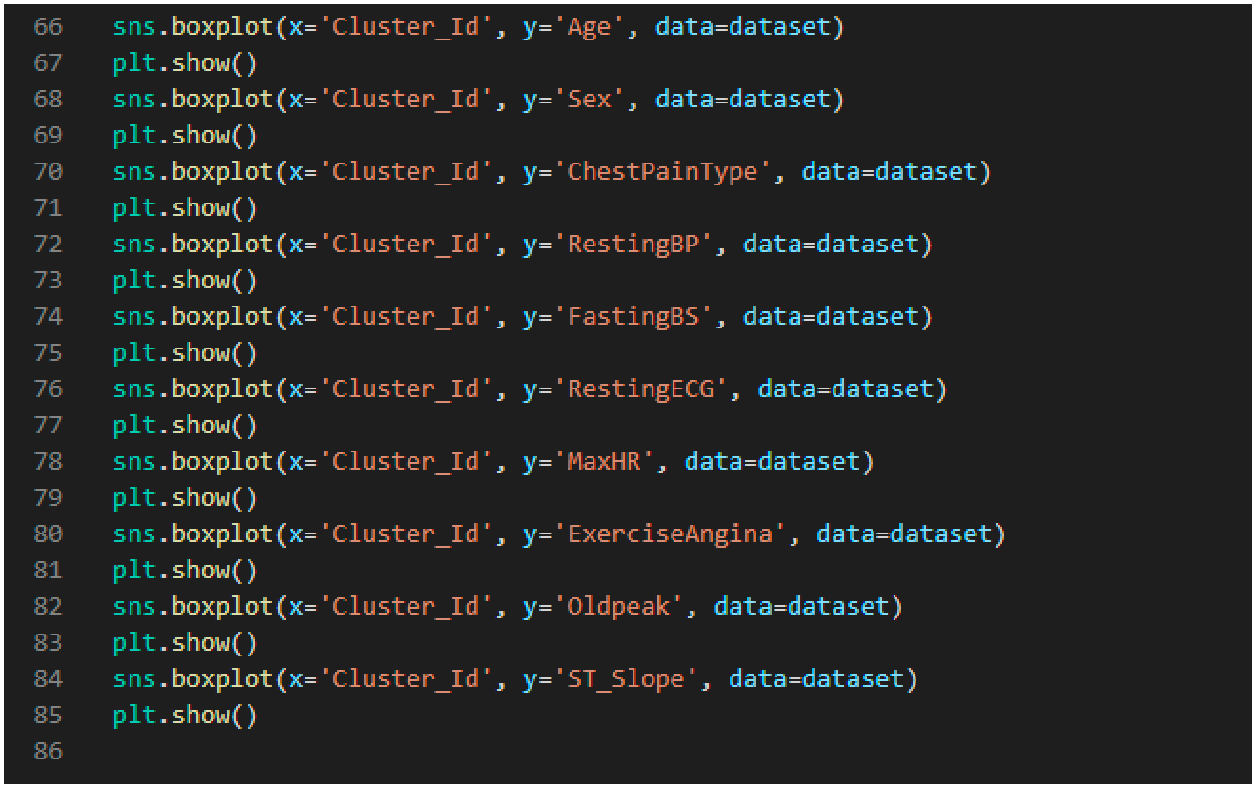
K-mean clustering had implemented to heart failure prediction dataset. No of clusters were 2, Heart Disease (1) and No Heart Disease (0). The result also showed that considering 2 clusters have higher silhouette score. We used boxplot to show the clusters.



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import warnings
from sklearn.preprocessing import LabelEncoder

warnings.filterwarnings('ignore')

dataset=pd.read_csv('heart.csv')




#to covert categorical data into numeric
columns=['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']
encoders = {}
for col in columns:
    le = LabelEncoder().fit(dataset[col])
    dataset[col] = le.transform(dataset[col])
    encoders[col] = le



#K MEAN CLUSTERING

ssd = []
```



```python
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(dataset)

    ssd.append(kmeans.inertia_)

# plot the SSDs for each n_clusters
plt.plot(ssd)
plt.show()

range_n_clusters = [2, 3, 4, 5, 6, 7, 8]

for num_clusters in range_n_clusters:

    # intialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(dataset)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = silhouette_score(dataset, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))

kmeans = KMeans(n_clusters=2, max_iter=50)
kmeans.fit(dataset)
kmeans.labels_

dataset['Cluster_Id'] = kmeans.labels_
dataset.head
```



```python
sns.boxplot(x='Cluster_Id', y='Age', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='Sex', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='ChestPainType', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='RestingBP', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='FastingBS', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='RestingECG', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='MaxHR', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='ExerciseAngina', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='Oldpeak', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Id', y='ST_Slope', data=dataset)
plt.show()
```

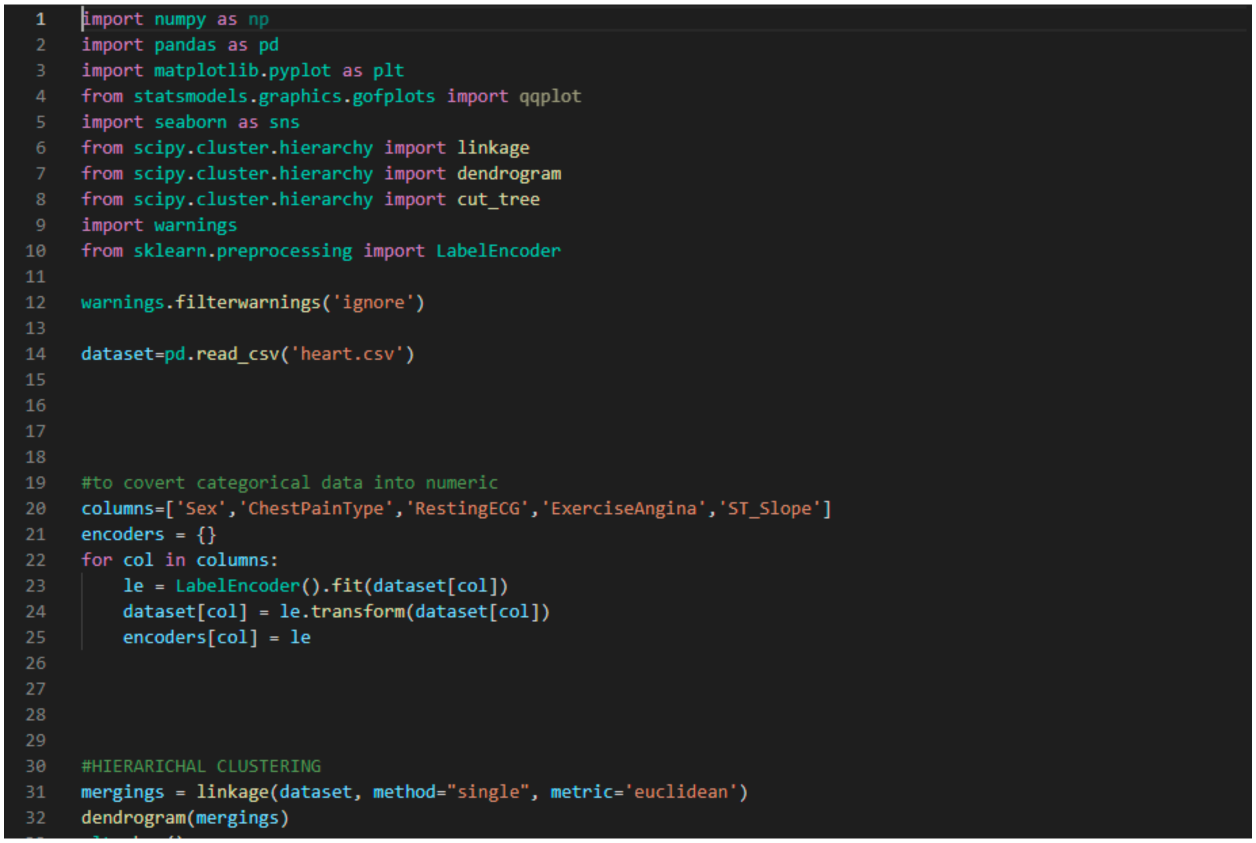
HIERARCHICAL CLUSTERING

Hierarchical clustering is a type of clustering method in which the clusters are formed by ordering from top to bottom, in a hierarchical way. Two types of hierarchical clustering are used. One is divisive and the other is agglomerative. We are using agglomerative approach in this report.

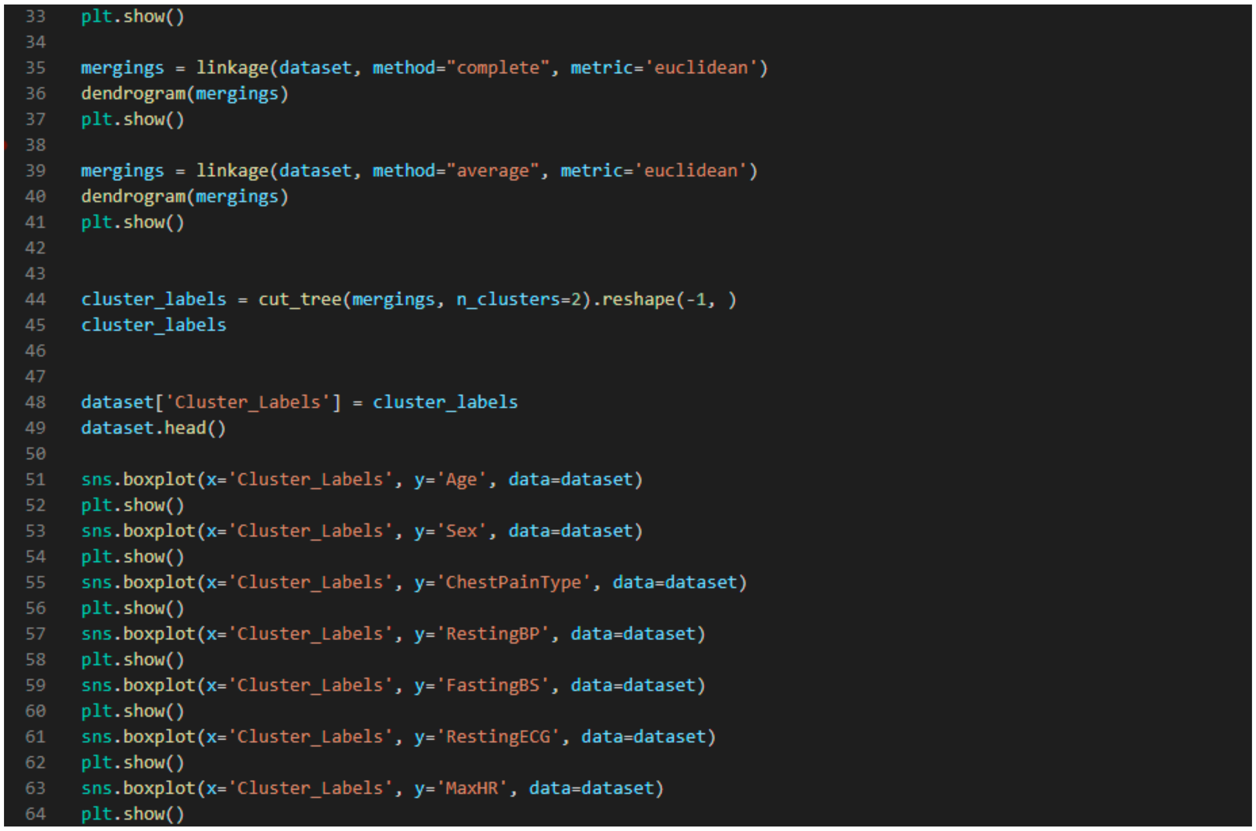
Three types of methods are used to perform agglomerative clustering

1. Single linkage 2. Complete linkage 3. Average linkage

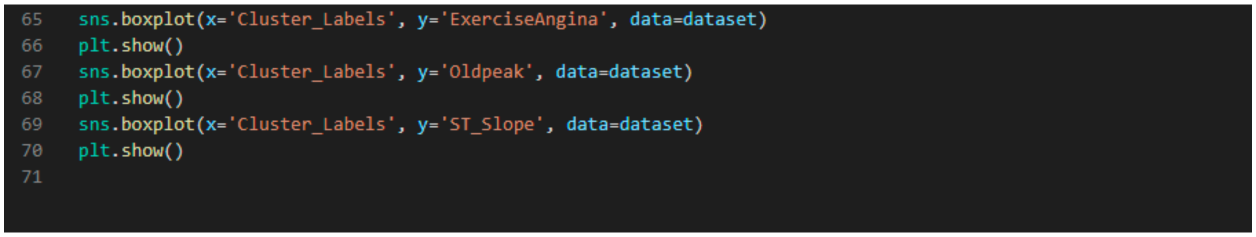
We used all these 3 methods to have a better idea and result. In single linkage we focus on considering the shortest distance between two points. Unlike the complete linkage hierarchical clustering that focuses on considering the longest distance between two points. Whereas in Average linkage we consider the average distance. In these hierarchical clustering methods we make dendrogram based on k clusters. We implemented single, complete and average linkage hierarchical clustering on our dataset to plot the dendogram based on 2 clusters. And showed all the results by using box plot



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
import warnings
from sklearn.preprocessing import LabelEncoder

warnings.filterwarnings('ignore')

dataset=pd.read_csv('heart.csv')




#to covert categorical data into numeric
columns=['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']
encoders = {}
for col in columns:
    le = LabelEncoder().fit(dataset[col])
    dataset[col] = le.transform(dataset[col])
    encoders[col] = le




#HIERARICHAL CLUSTERING
mergings = linkage(dataset, method="single", metric='euclidean')
dendrogram(mergings)
```



```python
plt.show()

mergings = linkage(dataset, method="complete", metric='euclidean')
dendrogram(mergings)
plt.show()

mergings = linkage(dataset, method="average", metric='euclidean')
dendrogram(mergings)
plt.show()


cluster_labels = cut_tree(mergings, n_clusters=2).reshape(-1, )
cluster_labels


dataset['Cluster_Labels'] = cluster_labels
dataset.head()

sns.boxplot(x='Cluster_Labels', y='Age', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='Sex', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='ChestPainType', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='RestingBP', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='FastingBS', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='RestingECG', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='MaxHR', data=dataset)
plt.show()
```

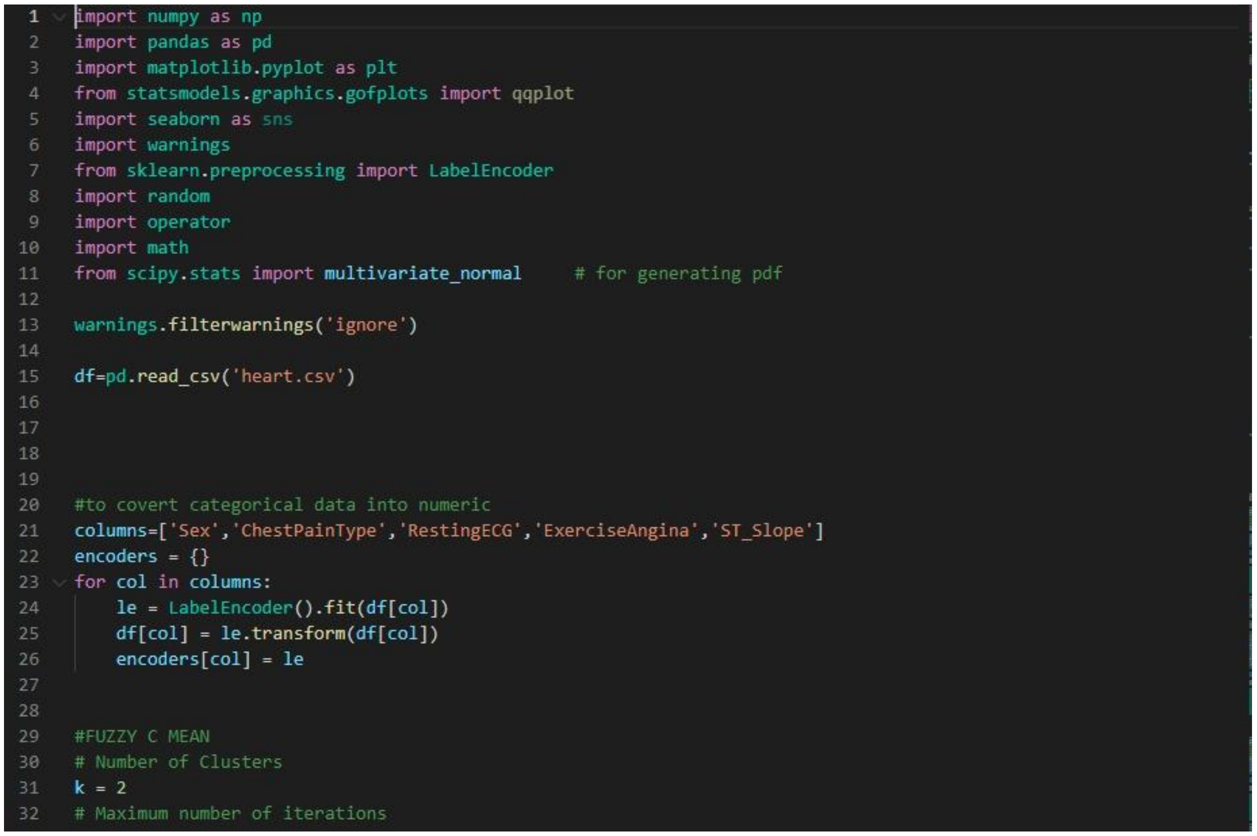


```
sns.boxplot(x='Cluster_Labels', y='ExerciseAngina', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='Oldpeak', data=dataset)
plt.show()
sns.boxplot(x='Cluster_Labels', y='ST_Slope', data=dataset)
plt.show()
```

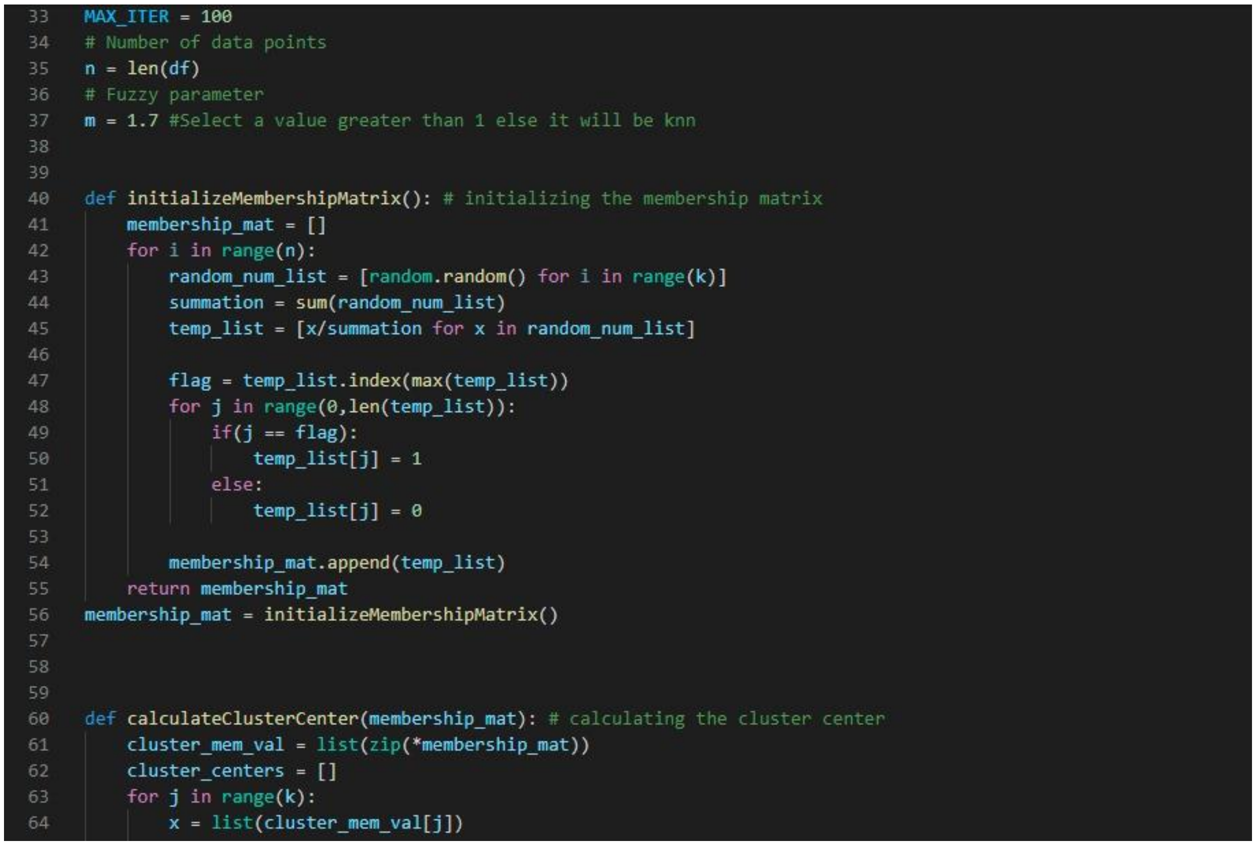
FUZZY C MEAN CLUSTERING

Fuzzy c mean is a type of clustering method in which a data point can belong to two or more clusters by initializing membership matrix then calculating cluster centers based on initial membership values. Update the membership value to attain the accuracy of clusters.

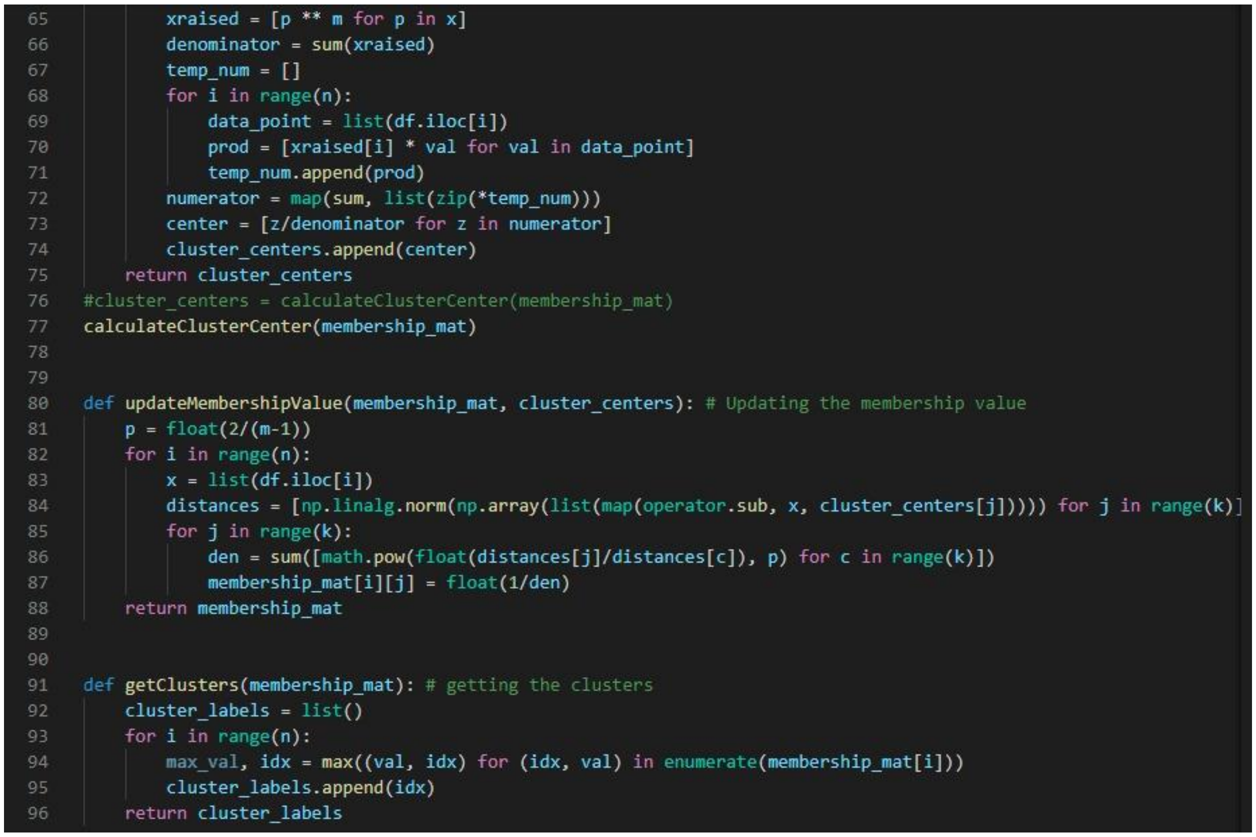
We implemented Fuzyy c mean on our dataset related heart failure prediction to cluster attributes. We used only few attributes to show the results. We performed clustering on Age and Max Heart Rate, and on Resting Blood Pressure and Cholesterol. We initialized the clusters at the origin, at random locations within a multi-variate Gaussian distribution with zero-mean and unit-variance and at random vectors chosen from the data. Scatter plot was used to plot the clusters.



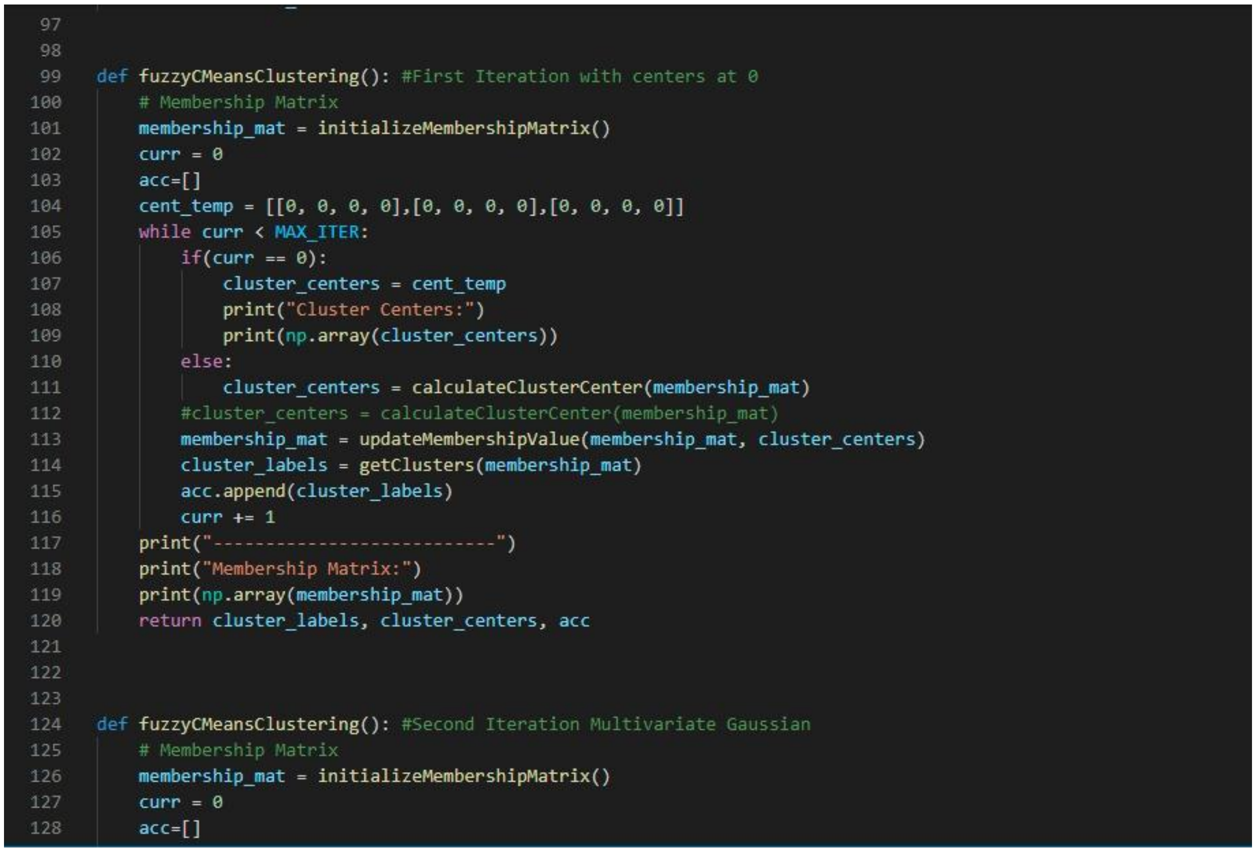
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns
import warnings
from sklearn.preprocessing import LabelEncoder
import random
import operator
import math
from scipy.stats import multivariate_normal     # for generating pdf

warnings.filterwarnings('ignore')

df=pd.read_csv('heart.csv')




#to covert categorical data into numeric
columns=['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']
encoders = {}
for col in columns:
    le = LabelEncoder().fit(df[col])
    df[col] = le.transform(df[col])
    encoders[col] = le


#FUZZY C MEAN
# Number of Clusters
k = 2
# Maximum number of iterations
```



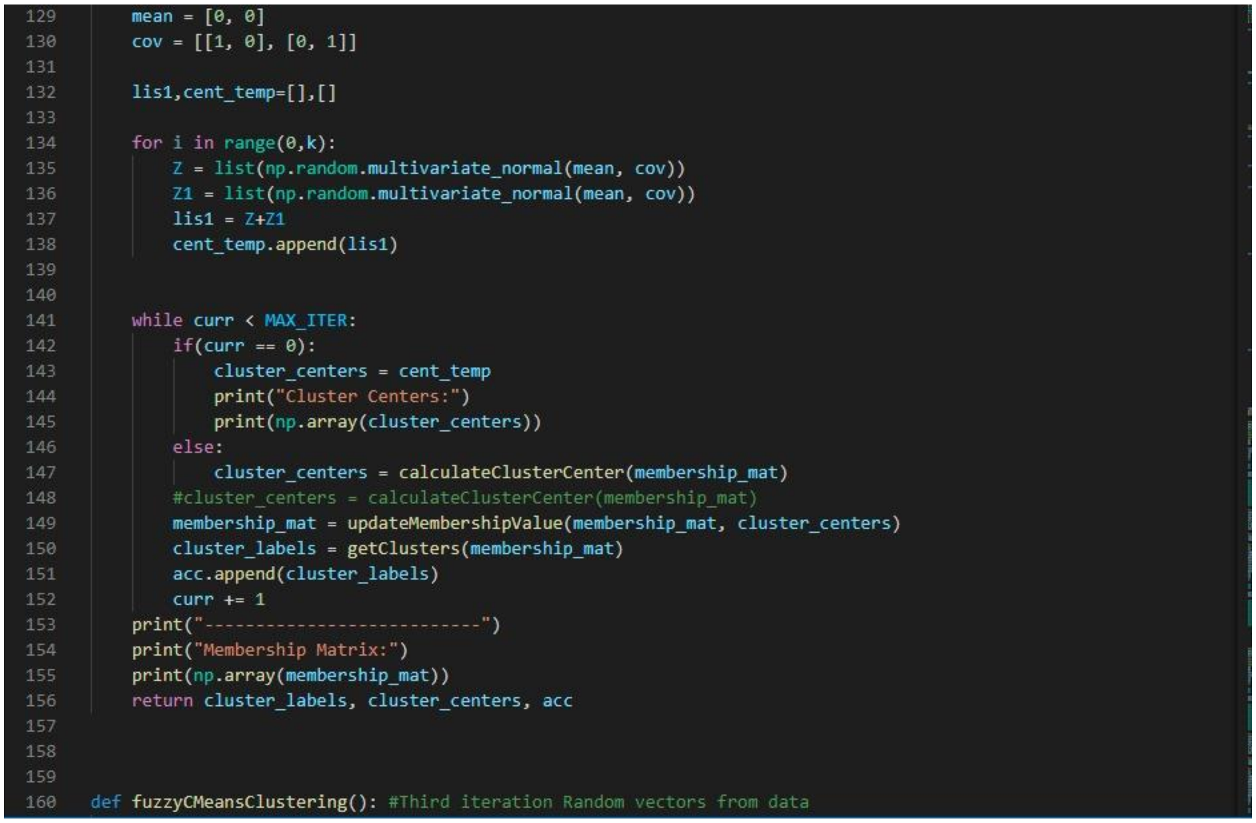
```
MAX_ITER = 100
# Number of data points
n = len(df)
# Fuzzy parameter
m = 1.7 #Select a value greater than 1 else it will be knn


def initializeMembershipMatrix(): # initializing the membership matrix
    membership_mat = []
    for i in range(n):
        random_num_list = [random.random() for i in range(k)]
        summation = sum(random_num_list)
        temp_list = [x/summation for x in random_num_list]

        flag = temp_list.index(max(temp_list))
        for j in range(0,len(temp_list)):
            if(j == flag):
                temp_list[j] = 1
            else:
                temp_list[j] = 0

        membership_mat.append(temp_list)
    return membership_mat
membership_mat = initializeMembershipMatrix()



def calculateClusterCenter(membership_mat): # calculating the cluster center
    cluster_mem_val = list(zip(*membership_mat))
    cluster_centers = []
    for j in range(k):
        x = list(cluster_mem_val[j])
```



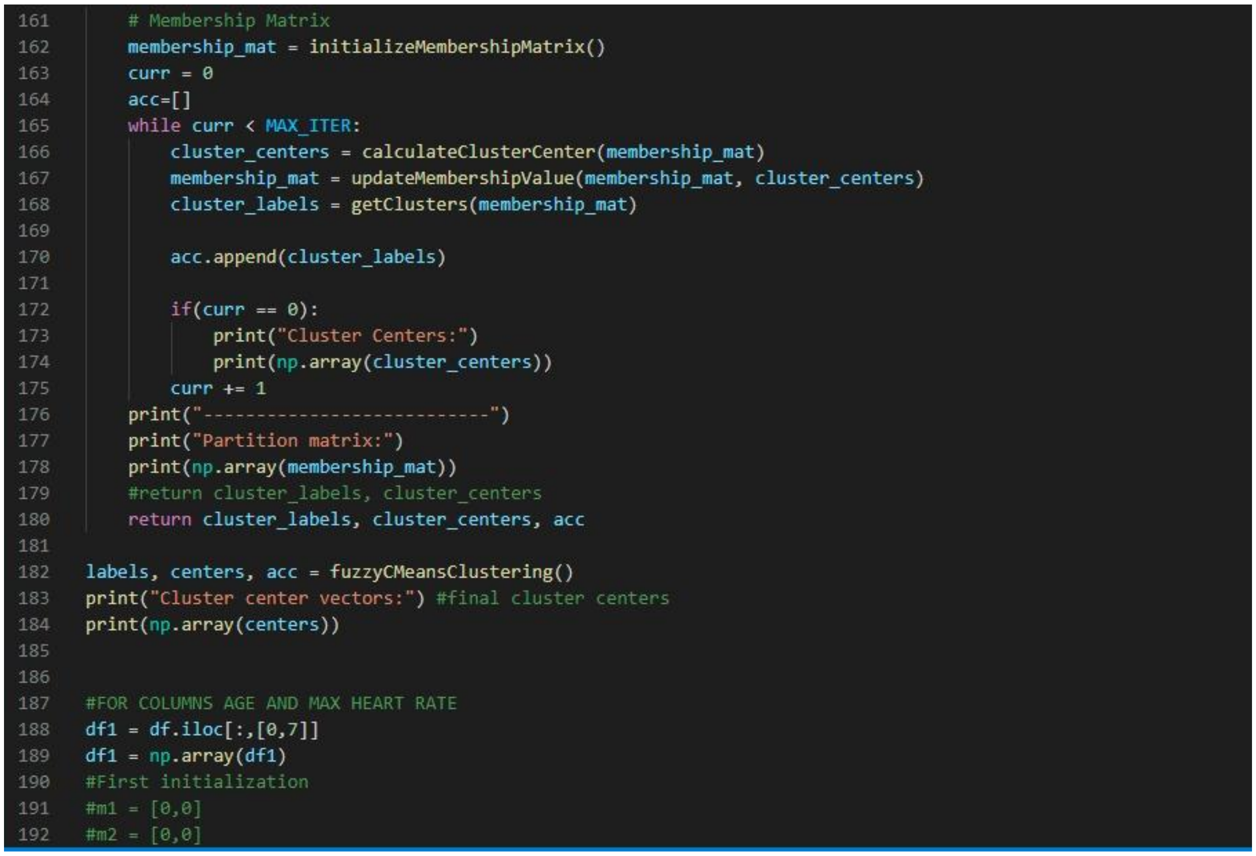
```
        xraised = [p ** m for p in x]
        denominator = sum(xraised)
        temp_num = []
        for i in range(n):
            data_point = list(df.iloc[i])
            prod = [xraised[i] * val for val in data_point]
            temp_num.append(prod)
        numerator = map(sum, list(zip(*temp_num)))
        center = [z/denominator for z in numerator]
        cluster_centers.append(center)
    return cluster_centers
#cluster_centers = calculateClusterCenter(membership_mat)
calculateClusterCenter(membership_mat)


def updateMembershipValue(membership_mat, cluster_centers): # Updating the membership value
    p = float(2/(m-1))
    for i in range(n):
        x = list(df.iloc[i])
        distances = [np.linalg.norm(np.array(list(map(operator.sub, x, cluster_centers[j])))) for j in range(k)]
        for j in range(k):
            den = sum([math.pow(float(distances[j]/distances[c]), p) for c in range(k)])
            membership_mat[i][j] = float(1/den)
    return membership_mat


def getClusters(membership_mat): # getting the clusters
    cluster_labels = list()
    for i in range(n):
        max_val, idx = max((val, idx) for (idx, val) in enumerate(membership_mat[i]))
        cluster_labels.append(idx)
    return cluster_labels
```



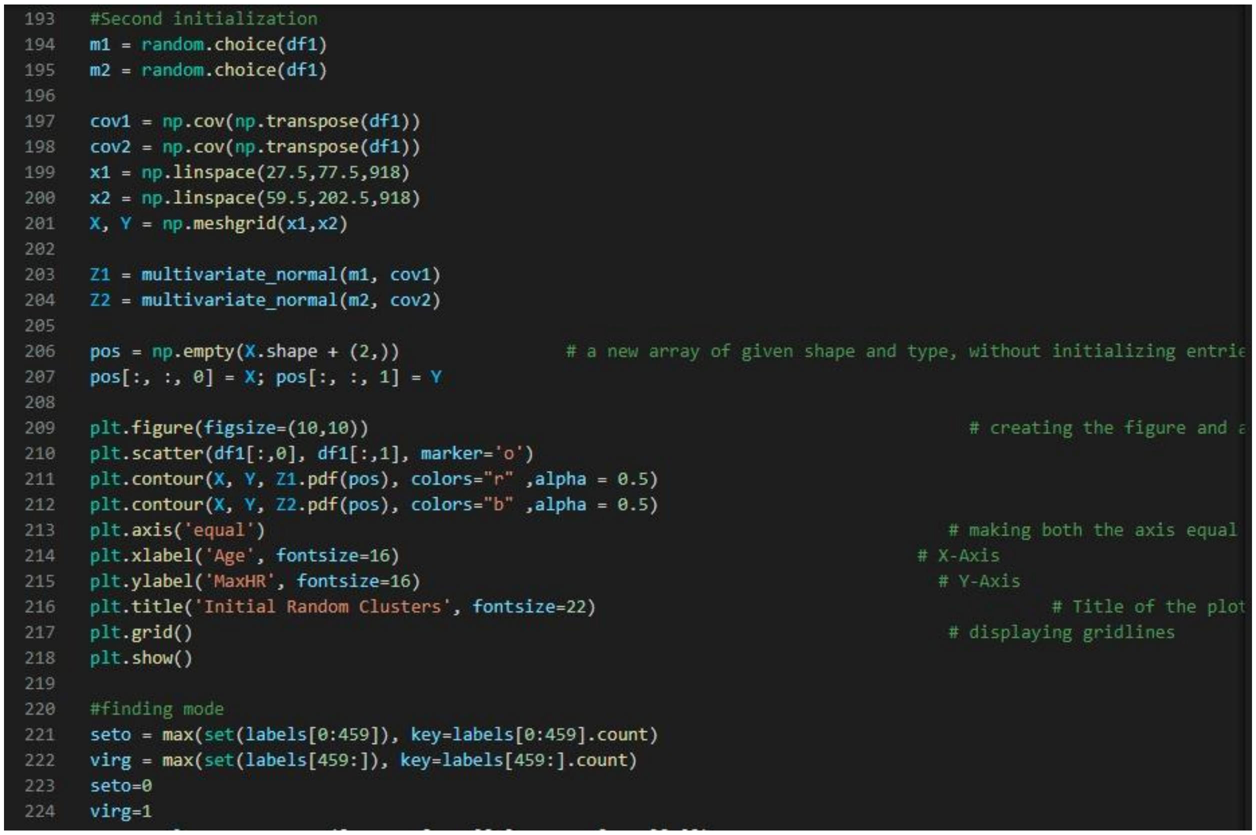
```


def fuzzyCMeansClustering(): #First Iteration with centers at 0
    # Membership Matrix
    membership_mat = initializeMembershipMatrix()
    curr = 0
    acc=[]
    cent_temp = [[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]]
    while curr < MAX_ITER:
        if(curr == 0):
            cluster_centers = cent_temp
            print("Cluster Centers:")
            print(np.array(cluster_centers))
        else:
            cluster_centers = calculateClusterCenter(membership_mat)
        #cluster_centers = calculateClusterCenter(membership_mat)
        membership_mat = updateMembershipValue(membership_mat, cluster_centers)
        cluster_labels = getClusters(membership_mat)
        acc.append(cluster_labels)
        curr += 1
    print("--------------------------")
    print("Membership Matrix:")
    print(np.array(membership_mat))
    return cluster_labels, cluster_centers, acc



def fuzzyCMeansClustering(): #Second Iteration Multivariate Gaussian
    # Membership Matrix
    membership_mat = initializeMembershipMatrix()
    curr = 0
    acc=[]
```



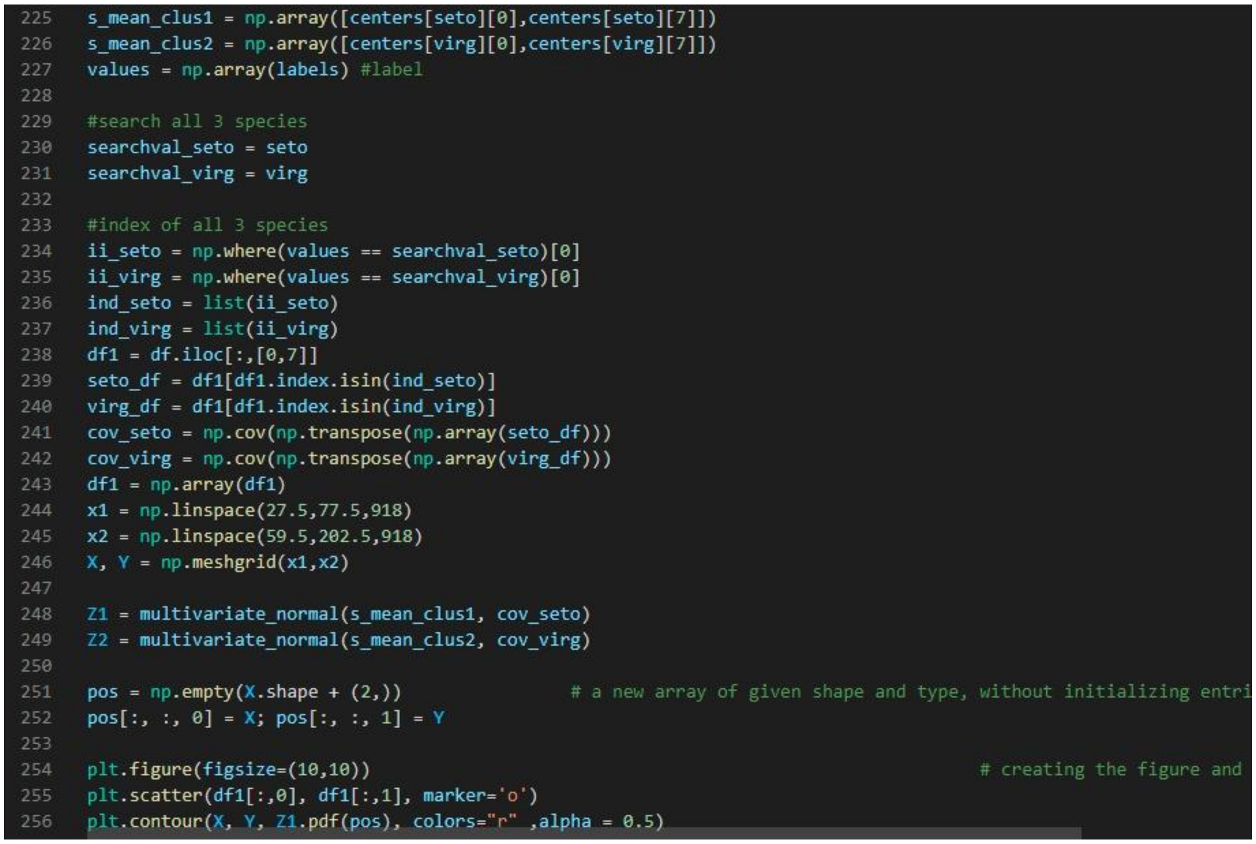
```
    mean = [0, 0]
    cov = [[1, 0], [0, 1]]

    lis1,cent_temp=[],[]

    for i in range(0,k):
        Z = list(np.random.multivariate_normal(mean, cov))
        Z1 = list(np.random.multivariate_normal(mean, cov))
        lis1 = Z+Z1
        cent_temp.append(lis1)


    while curr < MAX_ITER:
        if(curr == 0):
            cluster_centers = cent_temp
            print("Cluster Centers:")
            print(np.array(cluster_centers))
        else:
            cluster_centers = calculateClusterCenter(membership_mat)
        #cluster_centers = calculateClusterCenter(membership_mat)
        membership_mat = updateMembershipValue(membership_mat, cluster_centers)
        cluster_labels = getClusters(membership_mat)
        acc.append(cluster_labels)
        curr += 1
    print("---------------------------")
    print("Membership Matrix:")
    print(np.array(membership_mat))
    return cluster_labels, cluster_centers, acc



def fuzzyCMeansClustering(): #Third iteration Random vectors from data
```



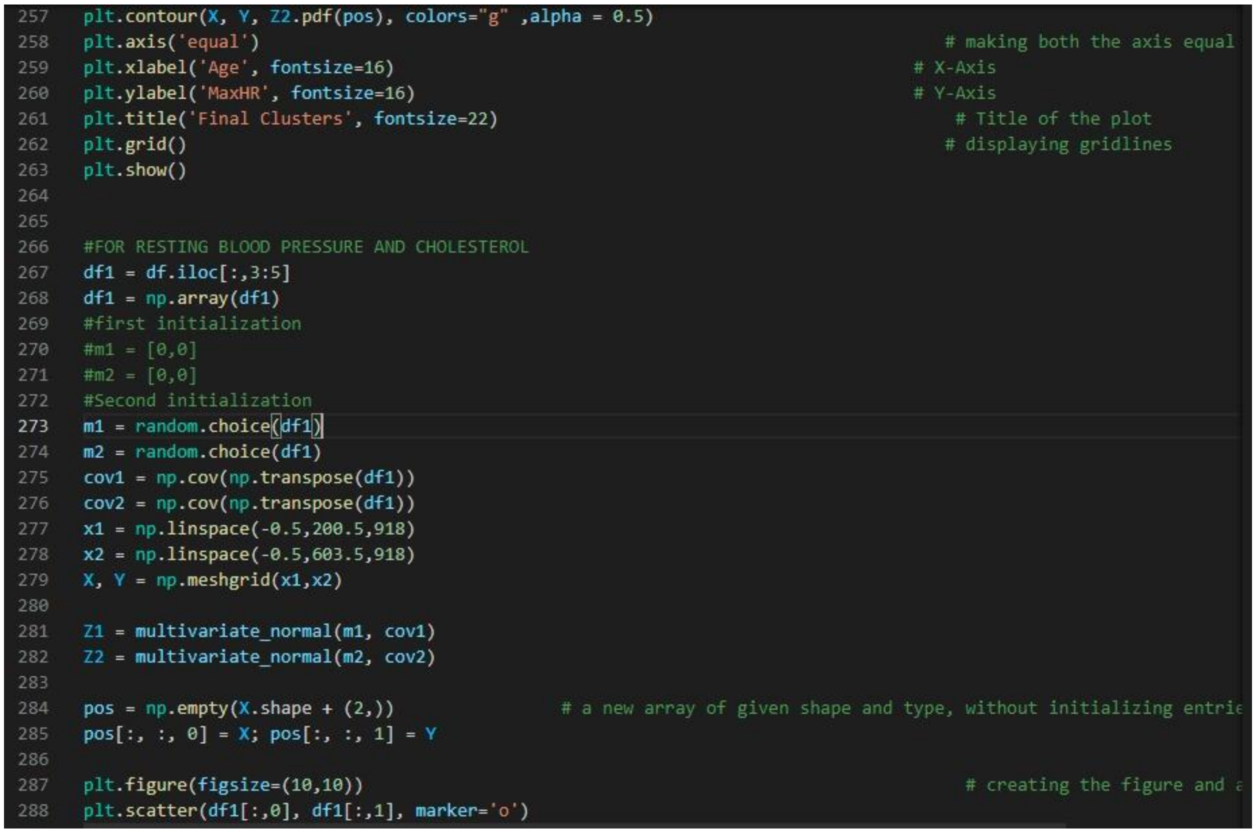
```
    # Membership Matrix
    membership_mat = initializeMembershipMatrix()
    curr = 0
    acc=[]
    while curr < MAX_ITER:
        cluster_centers = calculateClusterCenter(membership_mat)
        membership_mat = updateMembershipValue(membership_mat, cluster_centers)
        cluster_labels = getClusters(membership_mat)

        acc.append(cluster_labels)

        if(curr == 0):
            print("Cluster Centers:")
            print(np.array(cluster_centers))
        curr += 1
    print("---------------------------")
    print("Partition matrix:")
    print(np.array(membership_mat))
    #return cluster_labels, cluster_centers
    return cluster_labels, cluster_centers, acc

labels, centers, acc = fuzzyCMeansClustering()
print("Cluster center vectors:") #final cluster centers
print(np.array(centers))


#FOR COLUMNS AGE AND MAX HEART RATE
df1 = df.iloc[:,[0,7]]
df1 = np.array(df1)
#First initialization
#m1 = [0,0]
#m2 = [0,0]
```



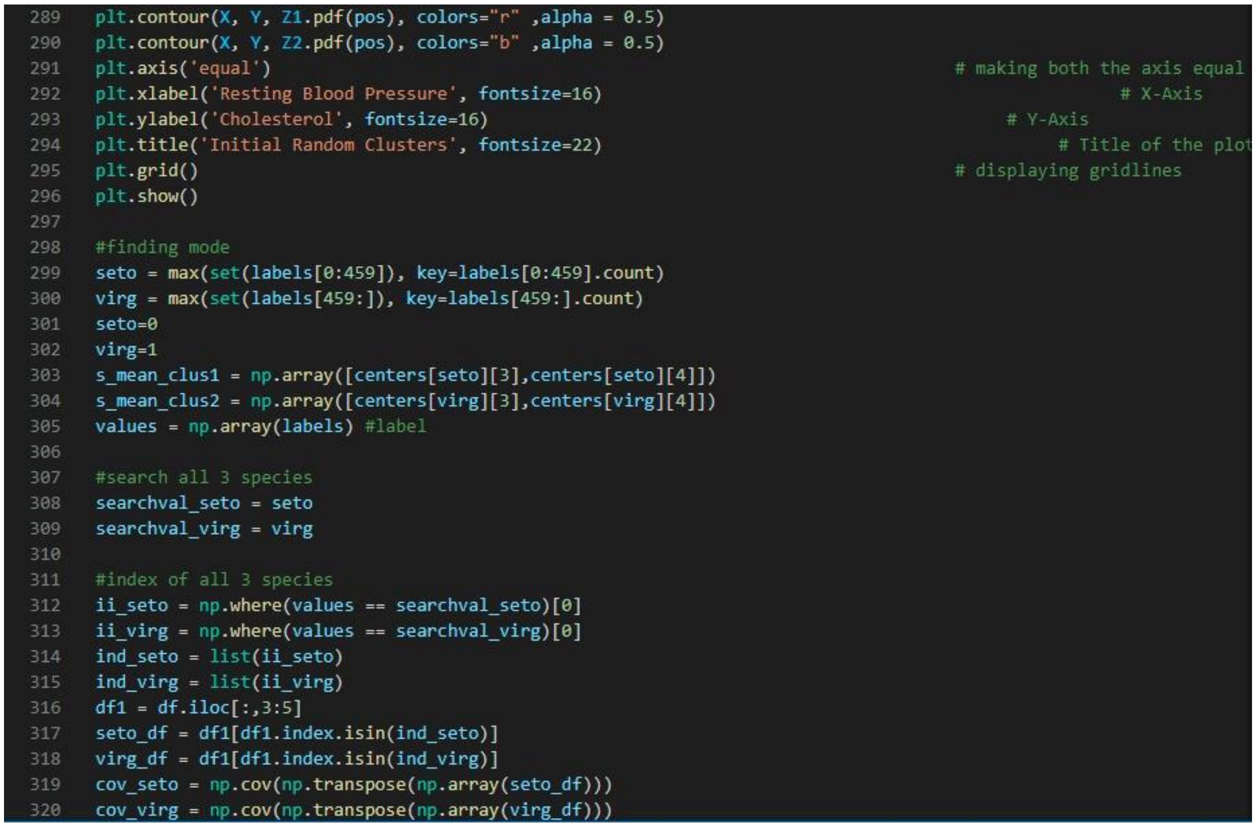
```python
#Second initialization
m1 = random.choice(df1)
m2 = random.choice(df1)

cov1 = np.cov(np.transpose(df1))
cov2 = np.cov(np.transpose(df1))
x1 = np.linspace(27.5,77.5,918)
x2 = np.linspace(59.5,202.5,918)
X, Y = np.meshgrid(x1,x2)

Z1 = multivariate_normal(m1, cov1)
Z2 = multivariate_normal(m2, cov2)

pos = np.empty(X.shape + (2,))                # a new array of given shape and type, without initializing entrie
pos[:, :, 0] = X; pos[:, :, 1] = Y

plt.figure(figsize=(10,10))                                                # creating the figure and a
plt.scatter(df1[:,0], df1[:,1], marker='o')
plt.contour(X, Y, Z1.pdf(pos), colors="r" ,alpha = 0.5)
plt.contour(X, Y, Z2.pdf(pos), colors="b" ,alpha = 0.5)
plt.axis('equal')                                                         # making both the axis equal
plt.xlabel('Age', fontsize=16)                                          # X-Axis
plt.ylabel('MaxHR', fontsize=16)                                          # Y-Axis
plt.title('Initial Random Clusters', fontsize=22)                                    # Title of the plot
plt.grid()                                                                # displaying gridlines
plt.show()

#finding mode
seto = max(set(labels[0:459]), key=labels[0:459].count)
virg = max(set(labels[459:]), key=labels[459:].count)
seto=0
virg=1
```



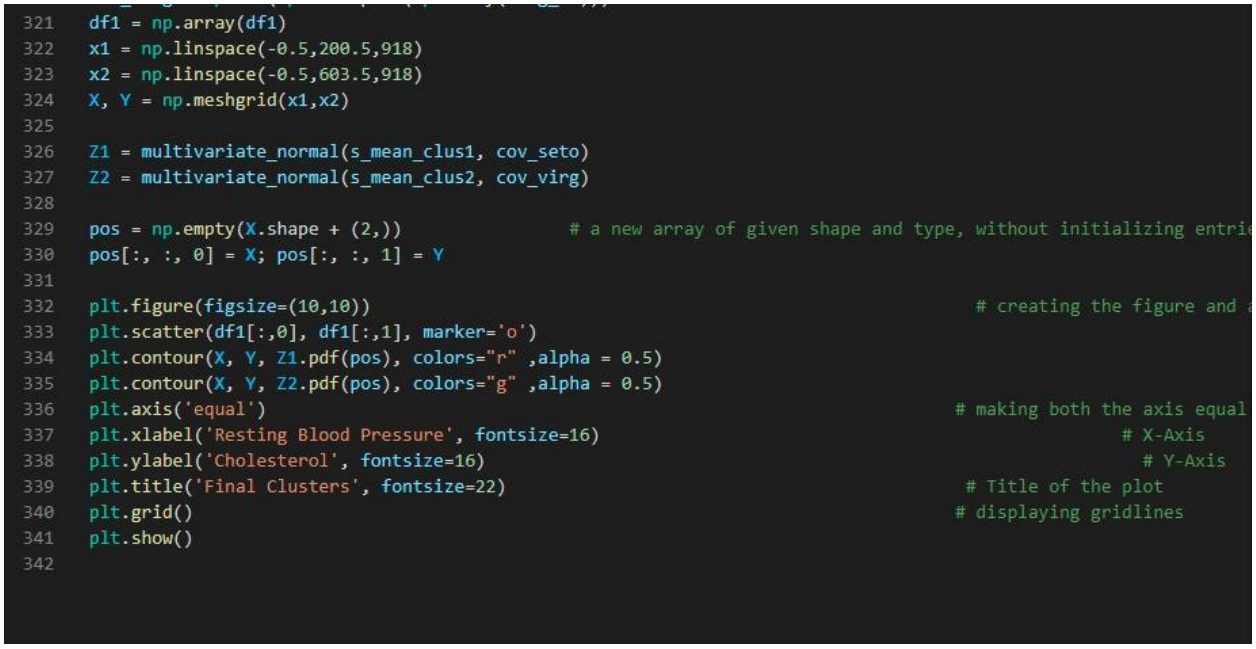
```python
s_mean_clus1 = np.array([centers[seto][0],centers[seto][7]])
s_mean_clus2 = np.array([centers[virg][0],centers[virg][7]])
values = np.array(labels) #label

#search all 3 species
searchval_seto = seto
searchval_virg = virg

#index of all 3 species
ii_seto = np.where(values == searchval_seto)[0]
ii_virg = np.where(values == searchval_virg)[0]
ind_seto = list(ii_seto)
ind_virg = list(ii_virg)
df1 = df.iloc[:,[0,7]]
seto_df = df1[df1.index.isin(ind_seto)]
virg_df = df1[df1.index.isin(ind_virg)]
cov_seto = np.cov(np.transpose(np.array(seto_df)))
cov_virg = np.cov(np.transpose(np.array(virg_df)))
df1 = np.array(df1)
x1 = np.linspace(27.5,77.5,918)
x2 = np.linspace(59.5,202.5,918)
X, Y = np.meshgrid(x1,x2)

Z1 = multivariate_normal(s_mean_clus1, cov_seto)
Z2 = multivariate_normal(s_mean_clus2, cov_virg)

pos = np.empty(X.shape + (2,))                # a new array of given shape and type, without initializing entri
pos[:, :, 0] = X; pos[:, :, 1] = Y

plt.figure(figsize=(10,10))                                                # creating the figure and
plt.scatter(df1[:,0], df1[:,1], marker='o')
plt.contour(X, Y, Z1.pdf(pos), colors="r" ,alpha = 0.5)
```



```
plt.contour(X, Y, Z2.pdf(pos), colors="g" ,alpha = 0.5)
plt.axis('equal')                                                        # making both the axis equal
plt.xlabel('Age', fontsize=16)                                         # X-Axis
plt.ylabel('MaxHR', fontsize=16)                                       # Y-Axis
plt.title('Final Clusters', fontsize=22)                                   # Title of the plot
plt.grid()                                                               # displaying gridlines
plt.show()


#FOR RESTING BLOOD PRESSURE AND CHOLESTEROL
df1 = df.iloc[:,3:5]
df1 = np.array(df1)
#first initialization
#m1 = [0,0]
#m2 = [0,0]
#Second initialization
m1 = random.choice(df1)
m2 = random.choice(df1)
cov1 = np.cov(np.transpose(df1))
cov2 = np.cov(np.transpose(df1))
x1 = np.linspace(-0.5,200.5,918)
x2 = np.linspace(-0.5,603.5,918)
X, Y = np.meshgrid(x1,x2)

Z1 = multivariate_normal(m1, cov1)
Z2 = multivariate_normal(m2, cov2)

pos = np.empty(X.shape + (2,))             # a new array of given shape and type, without initializing entrie
pos[:, :, 0] = X; pos[:, :, 1] = Y

plt.figure(figsize=(10,10))                                                # creating the figure and a
plt.scatter(df1[:,0], df1[:,1], marker='o')
```



```
plt.contour(X, Y, Z1.pdf(pos), colors="r" ,alpha = 0.5)
plt.contour(X, Y, Z2.pdf(pos), colors="b" ,alpha = 0.5)
plt.axis('equal')                                                        # making both the axis equal
plt.xlabel('Resting Blood Pressure', fontsize=16)                                      # X-Axis
plt.ylabel('Cholesterol', fontsize=16)                                       # Y-Axis
plt.title('Initial Random Clusters', fontsize=22)                                  # Title of the plot
plt.grid()                                                               # displaying gridlines
plt.show()

#finding mode
seto = max(set(labels[0:459]), key=labels[0:459].count)
virg = max(set(labels[459:]), key=labels[459:].count)
seto=0
virg=1
s_mean_clus1 = np.array([centers[seto][3],centers[seto][4]])
s_mean_clus2 = np.array([centers[virg][3],centers[virg][4]])
values = np.array(labels) #label

#search all 3 species
searchval_seto = seto
searchval_virg = virg

#index of all 3 species
ii_seto = np.where(values == searchval_seto)[0]
ii_virg = np.where(values == searchval_virg)[0]
ind_seto = list(ii_seto)
ind_virg = list(ii_virg)
df1 = df.iloc[:,3:5]
seto_df = df1[df1.index.isin(ind_seto)]
virg_df = df1[df1.index.isin(ind_virg)]
cov_seto = np.cov(np.transpose(np.array(seto_df)))
cov_virg = np.cov(np.transpose(np.array(virg_df)))
```



```
df1 = np.array(df1)
x1 = np.linspace(-0.5,200.5,918)
x2 = np.linspace(-0.5,603.5,918)
X, Y = np.meshgrid(x1,x2)

Z1 = multivariate_normal(s_mean_clus1, cov_seto)
Z2 = multivariate_normal(s_mean_clus2, cov_virg)

pos = np.empty(X.shape + (2,))                # a new array of given shape and type, without initializing entrie
pos[:, :, 0] = X; pos[:, :, 1] = Y

plt.figure(figsize=(10,10))                                              # creating the figure and a
plt.scatter(df1[:,0], df1[:,1], marker='o')
plt.contour(X, Y, Z1.pdf(pos), colors="r" ,alpha = 0.5)
plt.contour(X, Y, Z2.pdf(pos), colors="g" ,alpha = 0.5)
plt.axis('equal')                                                       # making both the axis equal
plt.xlabel('Resting Blood Pressure', fontsize=16)                                      # X-Axis
plt.ylabel('Cholesterol', fontsize=16)                                                  # Y-Axis
plt.title('Final Clusters', fontsize=22)                                 # Title of the plot
plt.grid()                                                              # displaying gridlines
plt.show()
```