# INT – 417

**Soft Computing Techniques**



**Assignment – I**

## End-to-end Heart disease prediction

**Submitted By:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Name** | **Registration**  **Number** |
| 1 | Samhita Ghosh | 11915175 |

**Section: KM015**

Github link: https://github.com/samhitaghosh/Heart-Disease-Prediction

**Declaration**

**To whom so ever it may concern**

I, **SAMHITA GHOSH**, registration number - **11915175**, hereby declare that the work done by me on **“Heart Disease Prediction”** from **FEBRUARY, 2022** to **APRIL, 2020** is a record of original work for the partial fulfilment of the requirements for the award of the degree, **Bachelor of Technology (CSE).**

Name of the Student: Samhita Ghosh (11915175)

Dated: 29th March, 2022.

**ACKNOWLEDGEMENT**

The success and outcome of this project required a lot of guidance and assistance from many people. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I would not forget to remember **Dr. Sagar Dhanraj Pande** who is a PhD Scholar in Computer Science and Engineering at Lovely Professional University for his encouragement, timely support and guidance.

I am thankful to **Department of Computer science and Engineering** for providing me an opportunity to do the community development project.

I am thankful and fortunate enough to get constant encouragement, support, guidance from all teaching staff of **“Computer science and engineering”** Which helped me in successfully completing our project work.

I would also like to thank to my parents and friends who helped me a lot in finalizing the project within the limited period.

Name: Samhita Ghosh

Reg No: 11915175

**Index**

|  |  |  |
| --- | --- | --- |
| SL. NO. | CONTENT | Page No. |
| 1. | Cover Page | 01 |
| 2. | Declaration | 02 |
| 3. | Acknowledgement | 03 |
| 4. | List of Figures | 05 |
| 5. | Introduction | 06 |
| 6. | Libraries Used | 10 |
| 7. | Methodologies | 13 |
| 8. | Result | 42 |
| 9. | Conclusion | 43 |
| 10. | References | 44 |

**List of figures:**

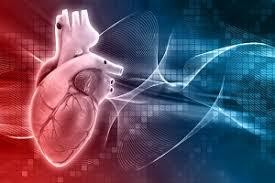
|  |  |  |
| --- | --- | --- |
| SL. NO. | FIGURE NO. | PAGE NO. |
| 1. | Figure 1 | 06 |
| 2. | Figure 2 | 08 |
| 3. | Figure 3 | 09 |
| 4. | Figure 4 & Figure 5 | 16 |
| 5. | Figure 6 & Figure 7 | 17 |
| 6. | Figure 8 & Figure 9 | 18 |
| 7. | Figure 10 | 19 |
| 8. | Figure 11 | 20 |
| 9. | Figure 12 | 21 |
| 10. | Figure 13 | 22 |
| 11. | Figure 14 & Figure 15 | 23 |
| 12. | Figure 16 | 24 |
| 13. | Figure 17 | 25 |
| 14. | Figure 18 & Figure 19 | 26 |
| 15. | Figure 20 | 27 |
| 16. | Figure 21 & Figure 22 | 28 |
| 17. | Figure 23 & Figure 24 & Figure 25 | 29 |
| 18. | Figure 26 | 30 |
| 19. | Figure 27 | 31 |
| 20. | Figure 28 | 32 |
| 21. | Figure 29 & Figure 30 | 33 |
| 22. | Figure 31 & Figure 32 | 34 |
| 23. | Figure 33 & Figure 34 | 35 |
| 24. | Figure 35 | 36 |
| 25. | Figure 36 | 37 |
| 26. | Figure 37 & Figure 38 | 38 |
| 27. | Figure 39 & Figure 40 | 39 |

# INTRODUCTION

The project is using Machine learning in health care sector for prediction.

The layman definition of the term Heart disease refers to several types of heart conditions, which are not normal. A biological definition of the same is – A type of disease that affects the heart or blood vessels. Smoking, high cholesterol, unhealthy diet and obesity are some of the factors which leads to the risk of heart disease.

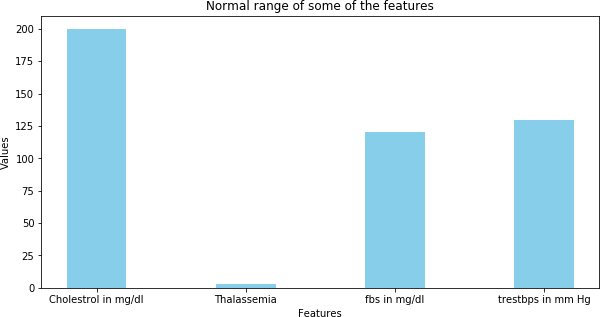
According to WHO (World Health Organization), Cardiovascular-diseases are the leading cause of death globally, taking an estimated 17.9 million lives each year. If the disease is detected at the early stage then the risk of death can be reduced, as by identifying the patients proper treatment can be given to the same.



**Figure 1**

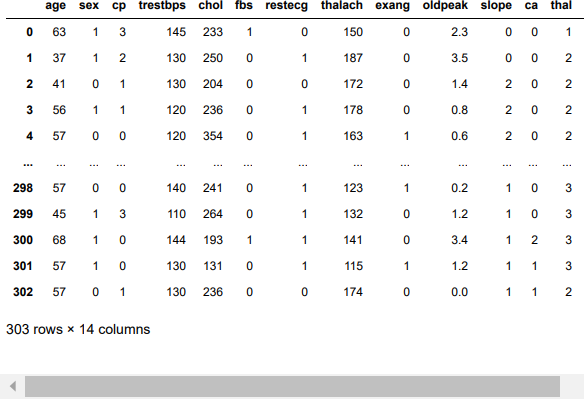
So, in order to make the prediction of this disease in human easier, we can use machine learning algorithms to which we can feed a large number of data, reports of the patients, and train them and predict the disease in future.

* **Features used in our project:**
  1. age: Age of the person is taken in years.
  2. sex: 1 => Male; 0 => Female
  3. cp: Chest Pain Type
* 0 => **Typical Angina**: chest pain caused due to decrease in blood supply to the heart.
* 1 => **Atypical Angina**: chest pain not related to heart.
* 2 => **Non-anginal pain**: typically esophageal spasms (not related to heart)
* 3 => **Asymptomatic**: chest pain not showing signs of disease.
  1. trestbps: Resting blood pressure (in mmHg on admission to the hospital) anything above 130-140 is typically cause for concern.
  2. chol : Serum cholesterol in mg/dl
* serum = LDL(Low Density Lipids) + HDL(High Density Lipids) +.2 \* triglycerides.
* above 200 is a cause for concern.
  1. fbs: Fasting blood sugar > 120mg/dl
* 1 => true, 2 -> false
* Above 126 mg/dl signals diabetes.
  1. restecg: Resting electrocardiographic results
* 0 => Nothing to note.
* 1 => ST –T wave abnormality; can range from mild symptoms to severe problems; signals abnormal heart beats.
* 2 => Possible or definite left ventricular hypertrophy; enlarged heart’s main pumping chamber.
  1. exang: Exercise induced angina.
* 0 => no , 1 => yes.
  1. oldpeak: ST depression induced by exercise relative to rest looks at stress of heart during exercise; unhealthy heart will stress more
  2. slope : the slope of the peak exercise ST segment
* 0 => Upslopping: better heart rate with exercise (uncommon).
* 1 => Flatslopping : minimal change (typical healthy heart).
* 2=>Downsloping: signs of unhealthy heart.
  1. ca: number of major vessels (0 – 3) colored by fluoroscopy colored vessel means the doctor can see the blood passing through; more blood movement means more better (no clots)
  2. thal: thalium stress result.
* 1 – 3: normal
* 6: fixed defect.
* 7: reversible defect.



**Figure 2**

# DATA SET:



**Figure 3**

**https:/**[**/www.k**](http://www.kaggle.com/ronitf/heart-disease-uci)**a**[**ggle.com/ronitf/heart-disease-uci**](http://www.kaggle.com/ronitf/heart-disease-uci)

**ROLE OF THE STUDENT:**

|  |  |  |
| --- | --- | --- |
| S.No | Name of the student | Role |
| 1 | Samhita Ghosh | Data Processing, Model Training & Testing and Report. |

**LIBRARIES USED:**

## Numpy:

**Num**erical **Py**thon is a library which is used for working with arrays. The array can be multi-dimensional as well. In python, generally we use list but it is a bit slower hence using numpy arrays is more convenient as it is faster than lists.

Using this library one can perform operations on array like:

* Operations related to linear algebra.
* Mathematical and logical operations.
* Shape manipulations

Some of the operations that can be performed using this library are:

*import numpy as np*

*arr = np.array([1,2,3,4,5]) print(arr[0]) print(np.sort(arr)) print(np.average(arr))*

## Pandas:

Pandas is used in order to store the data and preprocess it so that it can be used to train and test the model. The main object used is DataFrame for data manipulation. Using pandas, we can read and write data and also manage the data with missing values. Also, used for merging and joining the data sets.

Some of the operations that we can perform are:

* Reading data into a data frame from .csv files.
* Creating a data frame to store rows and columns.
* We can also use series.

*import pandas as pd df=pd.read\_csv(“File\_name.csv”) print(df.columns.tolist()) print(df.value\_counts())*

## Matplotlib:

Matplotlib is a plotting library of python. Using, matplotlib we can visualize the data through pie charts, histograms scatter plots. This, is a library which is very useful in data processing. Most of the utilities lies under the sub-module pyplot.

Some of the operations that can be performed using this library are:

* Visualizing the data using sub plots and bar graph and many more.

*import matplotlib.pyplot as plt import numpy as np*

*xpoints = np.array([0, 6]) ypoints = np.array([0, 250])* plt.plot(xpoints, ypoints) plt.show()

## Seaborn:

Seaborn is a visualization library which is used to plot the relationship between variables, which can be numerical or categorical.

Different types of plots are:

* Relational plots: to understand relation between two variables.
* Categorical plot: deals with categorical values.
* Matrix plot: It is an array of scatterplots.

*import seaborn as sns sns.set(style="dark")*

*fmri = sns.load\_dataset("fmri")*

*sns.lineplot(x="timepoint", y="signal", hue="region", style="event", data=fmri)*

## Scikit-learn:

Scikit-learn is a library which is used to build models including classification, regression and many more.

Some of the models are:

* Clustering: Such as KMeans clustering.
* Cross Validation: It is basically used to compare and select a model for a given predictive modeling problem.

*from sklearn import tree*

*t= tree.DecisionTreeClassifier(criterion=”entropy”) t=t.fit(train\_attributes,train\_labels) t.score(test\_attributes,test\_labels) t.predict(Example\_attributes) cross\_val\_score(t,all\_attributes,all\_labels)*

# METHODOLOGIES USED IN THE PROJECT

Before I discuss about the methodologies, let me explain about the libraries used in the program.

Firstly, I used:

1. **Numpy**: library used to implement the functionality of array. Also, it helps to use functions for linear algebra and matrices.
2. **Pandas**: library used to implement different tools for data analysis and manipulation tools.
3. **Matplotlib**: library used to visualize data that are built on Numpy.
4. **Seaborn**: library used to implement graphs in more statistical way.
5. **Sklearn**: library uses different and efficient tools for machine learning and statistical models.
   * **sklearn.linear\_model** :

A class of the sklearn module, it contains different functions for performing machine learning with linear models. Using this class to implement the **LogisticRegression** method.

* + **sklearn.neighbors** :

A class [that](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neighbors) provides functionality for unsupervised and supervised neighbors-based learning methods. Using this class to implement **KNeighborsClassifier** method.

* + **sklearn.ensemble** :

A class that helps to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. Using this class to implement **RandomForestClassifier** method.

## sklearn.model\_selection.train\_test\_split() :

It splits arrays or matrices into random train and test subsets.

syntax:

*sklearn.model\_selection.train\_test\_split(\*arrays, test\_size=None, train\_size=None, random\_state=None, shuffle=True, stratify=None)*

## sklearn.model\_selection.cross\_val\_score () :

Used to evaluate a score by cross-validation syntax:

*sklearn.model\_selection.cross\_val\_score(estimator, X, y=None, \*, group s=None, scoring=None, cv=None, n\_jobs=None, verbose=0, fit\_params=None, pre\_dispatch='2\*n\_jobs', error\_score=nan)*

## sklearn.model\_selection.RandomizedSearchCV() :

Used in randomized search on hyper parameters. It implements a

“fit” and a “score” method. It also implements “score\_samples”, “predict”, “predict\_proba”, “decision\_function”, “transform” and “inverse\_transform” if they are implemented in the estimator used.

syntax:

*sklearn.model\_selection.RandomizedSearchCV(estimator, param\_distrib utions, \*, n\_iter=10, scoring=None, n\_jobs=None, refit=True, cv=None, verbose=0, pre\_dispatch='2\*n\_jobs', random\_state=None, error\_score= nan, return\_train\_score=False)*

## sklearn.model\_selection.GridSearchCV() :

Exhaustive search over specified parameter values for an estimator, important members are fit, predict.

syntax:

*sklearn.model\_selection.GridSearchCV(estimator, param\_grid, \*, scoring=None, n\_jobs=None, refit=True, cv=None, verbose=0, pre\_dispatch=' 2\*n\_jobs', error\_score=nan,*

*return\_train\_score=False)*

## sklearn.metrics :

Module implements functions assessing prediction error for specific purposes.

## sklearn.metrics.confusion\_matrix :

Used to compute confusion matrix to evaluate the accuracy of a classification.

syntax:

*sklearn.metrics.confusion\_matrix(y\_true, y\_pred, \*, labels=None,*

*sample\_weight=None, normalize=None)*

## sklearn.metrics.classification\_report :

Used to build a text report showing the main classification metrics.

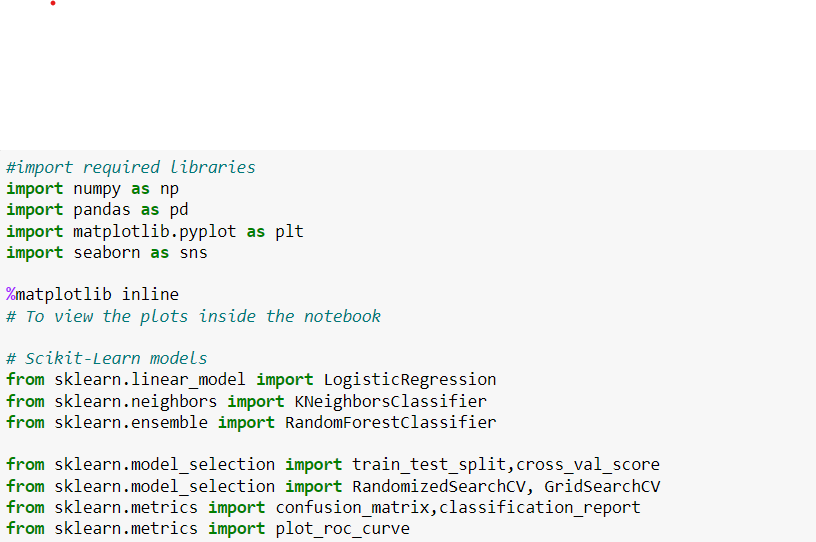
syntax:

*sklearn.metrics.classification\_report(y\_true, y\_pred, \*, labels=None, target\_names=None, sample\_weight=None, digits=2, output\_dict=False, ze ro\_division='warn')*

## sklearn.metrics.plot\_roc\_curve :

syntax:

*sklearn.metrics.plot\_roc\_curve(estimator, X, y, \*, sample\_weight=None, drop\_intermediate=True, response\_method='auto', name=None, ax=None, pos\_label=None, \*\*kwargs)*



**Figure 4**

* **pd.read\_csv("filename.csv”)**:

Data frame is a two-dimensional data structure. Using this, I can store tabular data. The data is stored in csv file and then in order to retrieve the data into the program then I use pandas, pd – alias name for pandas. In order to read the csv file, I use **read\_csv()** method which takes parameter filename.

Syntax:

**Figure 5**

* **value\_counts():**

This function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default. We used this to find how many persons do have the disease from the ‘target’ column.

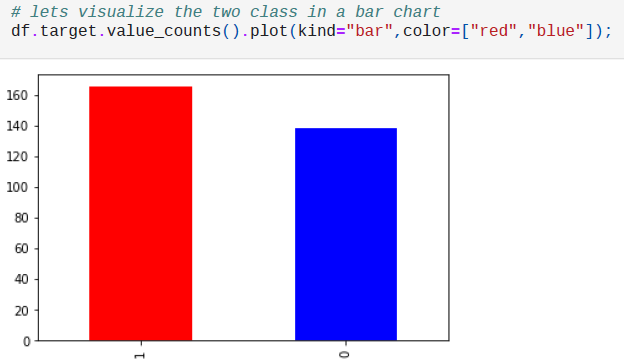
Syntax:

**Figure 6**

* **df.plot(kind=’#’):**

Pandas uses the plot() method to create diagrams # here I need to mention the type of chart in the kind parameter such as hist for histogram, scatter for scatter plot. Here, I use the pandas plot method to plot the data that we got in the previous statement where we generated the data to count the persons having the disease.

Syntax:

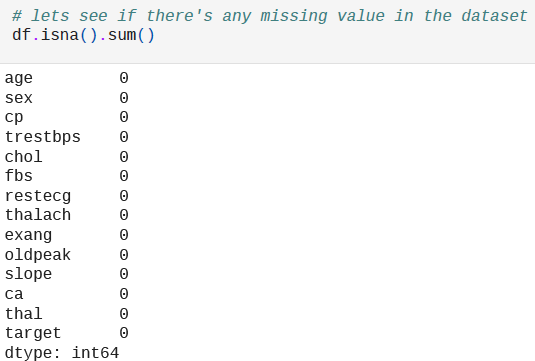


**Figure 7**

## df.isna().sum():

It returns the number of missing values in each column. We use it to check any record might have null values, because it may lead to improper analysis of each person towards the disease.

**Figure 8**

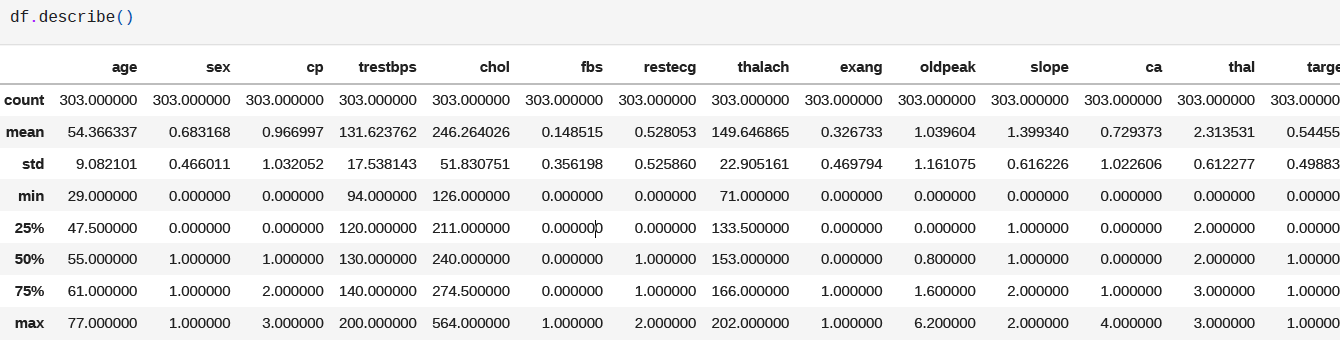


## df.describe():

This method is used to generate descriptive statistics which include those that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding NaN values.

syntax:

*DataFrame.describe(percentiles=None, include=None, exclude=None, d atetime\_is\_numeric=False)*

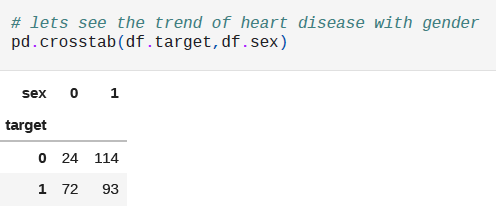


**Figure 9**

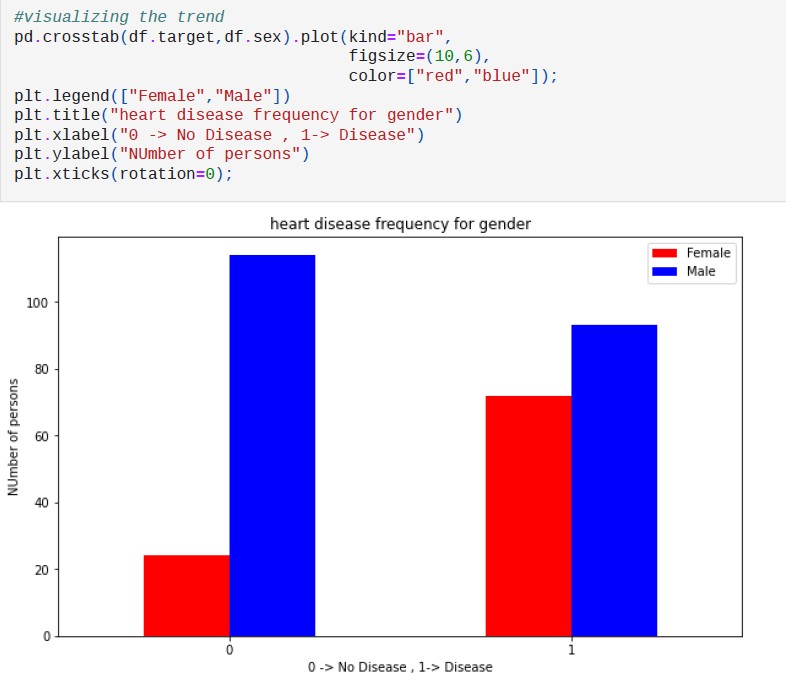
## pandas.crosstab

It computes a simple cross tabulation of two (or more) factors. By default, computes a frequency table of the factors unless an array of values and an aggregation function are passed. Here, we used this method to analyse the trend of heart disease with gender which will be used in plotting the graph, shown in the later picture.

syntax:

*pandas.crosstab(index, columns, values=None, rownames=None, colnam es=None, aggfunc=None, margins=False, margins\_name='All', dropna= True, normalize=False)*

**Figure 10**



**Figure 11**

We see that using the crosstab we got the tabular information as represented in the first picture above, and the data generated is used to plot the bar graph using the plot() method of the pandas. The plt is aliased name for matplotlib library.

## plt.legend():

helps to place the legends of the graphs.

## plt.title():

takes the title of the name of the chart as the input parameter.

## plt.xlabel() or plt.ylabel():

used to label the x-axis or y-axis of the generated chart respectively, the labels are taken into the parameter.

## plt.subplots():

It creates a figure and a set of subplots. This utility wrapper makes it convenient to create common layouts of subplots, including the enclosing figure object, in a single call. We tried to analyse the cholesterol with age and also heart rate with age using scatter plot but in two different plots, so we use subplots method to display the data.

Syntax:

*matplotlib.pyplot.subplots(nrows=1, ncols=1, \*, sharex=False, sharey= False, squeeze=True, subplot\_kw=None, gridspec\_kw=None, \*\*fig\_kw)*



**Figure 12**

## x1.scatter :

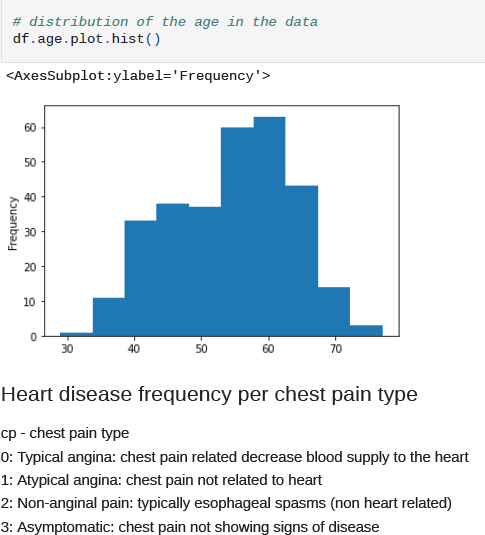
A scatter plot of *y* vs. *x* with varying marker size and/or colour, we defined it two times in order to plot two different data in the same scatter plot i.e., person with disease and without disease.

Syntax:

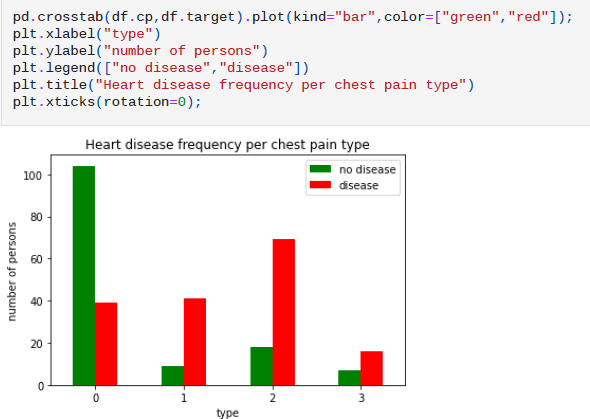
*Axes.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=N one, vmin=None,vmax=None, alpha=None, linewidths=None, \*, edgecol ors=None,plotnonfinite=False, data=None, \*\*kwargs)*

Then we plotted age data into histogram chart using the same pandas plot method.

Figure 13



Using the same methods i.e., crosstab and plot methods to plot the heart disease with the type of chest pains, this data will help to analyze and under which type of pain might lead to chronic disease.

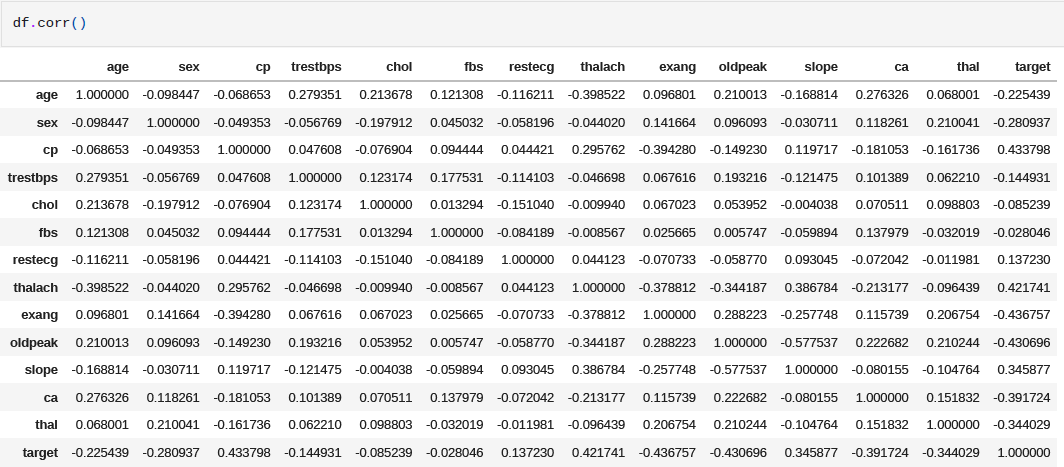


**Figure 14**

## df.corr():

Compute pairwise correlation of columns, excluding NA/null values. Syntax:

*DataFrame.corr(method='pearson', min\_periods=1)*



**Figure 15**

The data that I generated in the previous method i.e., df.corr(), used it to plot matrices type map, this kind of maps will help us to understand the complete tabular data into visualization form.

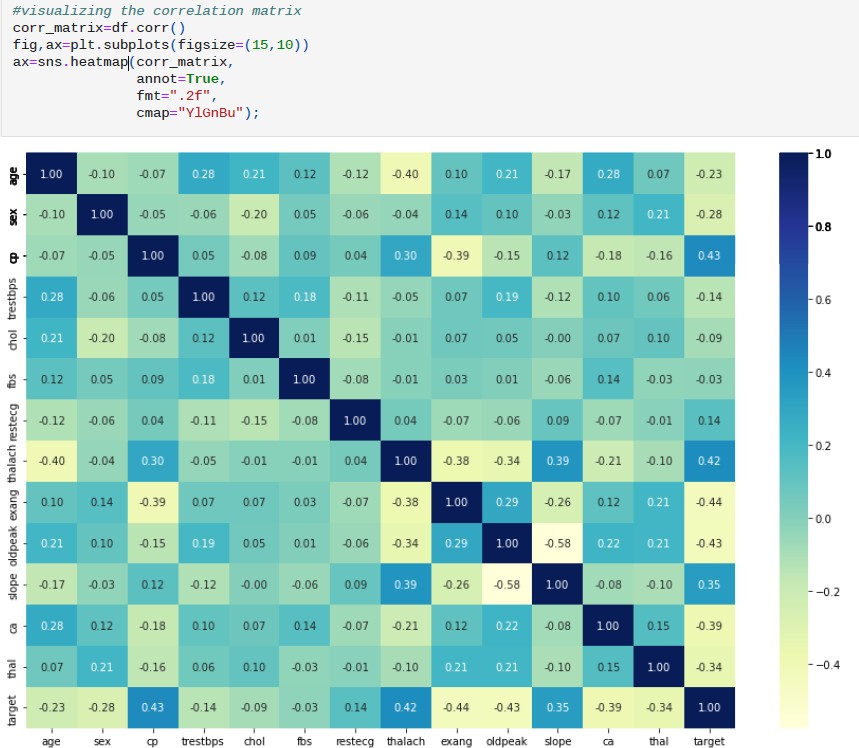
## Seaborn.heatmap()

Plot rectangular data as a color-encoded matrix.

Syntax:

*seaborn.heatmap(data, \*, vmin=None, vmax=None, cmap=None, center*

*=None, robust=False, annot=None, fmt='.2g', annot\_kws=None, linewi dths=0, linecolor='white', cbar=True, cbar\_kws=None, cbar\_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, ax=N one, \*\*kwargs)*



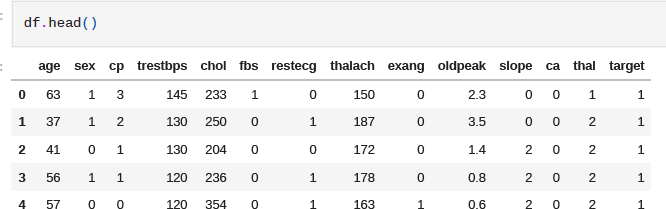
**Figure 16**

## Df.head() :

The head() function is used to get the first n rows.

This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it. I took two variables i.e., x and y to store target values that contains represents persons without disease and complete values of target column respectively.

Syntax:



**Figure 17**

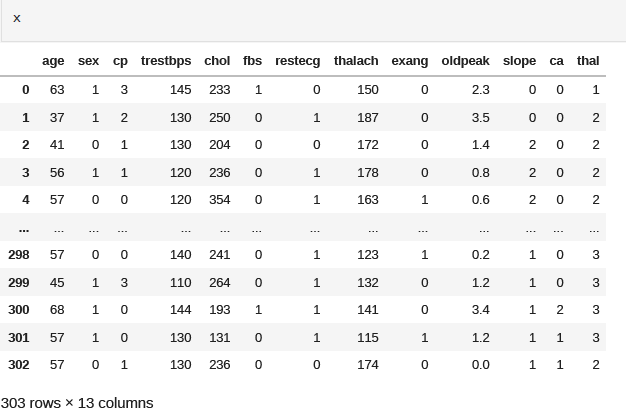
## DataFrame.drop():

Drop specified labels from rows or columns.

Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names

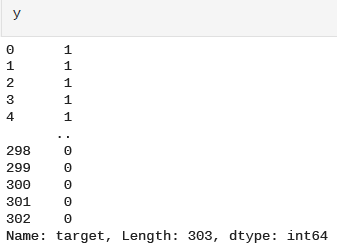
Syntax:

*DataFrame.drop(labels=None, axis=0, index=None, columns=None, lev el=None, inplace=False, errors='raise')*



**Figure 18**

The values of that stored in the y variable:



**Figure 19**

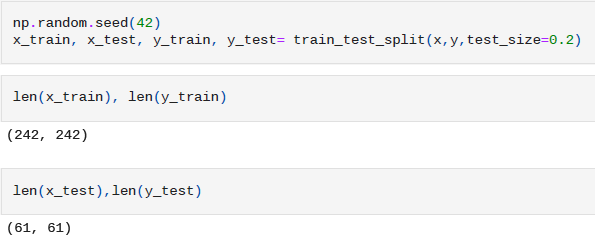
Now I need to split the input and output data for training and testing.

## np.random.seed :

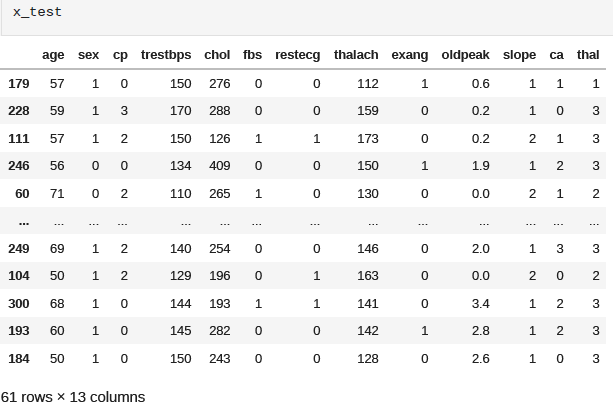
This is used in the generation of a pseudo-random encryption key. Encryption keys are an important part of computer security. These are the kind of secret keys which used to protect data from unauthorized access over the internet. It makes optimization of codes easy where random numbers are used for testing. The output of the code sometime depends on input. So, the use of random numbers for testing algorithms can be complex. Also seed function is used to generate same random numbers again and again and simplifies algorithm testing process.

Then I used the train\_test\_split method as discuss earlier to separate the input and output of the training and the testing data.

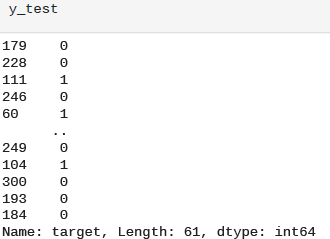
## Len() :

This method returns the count of the values of the variable or length of the data (if string variable is passed) that is passed as the parameter inside the method.

**Figure 20**

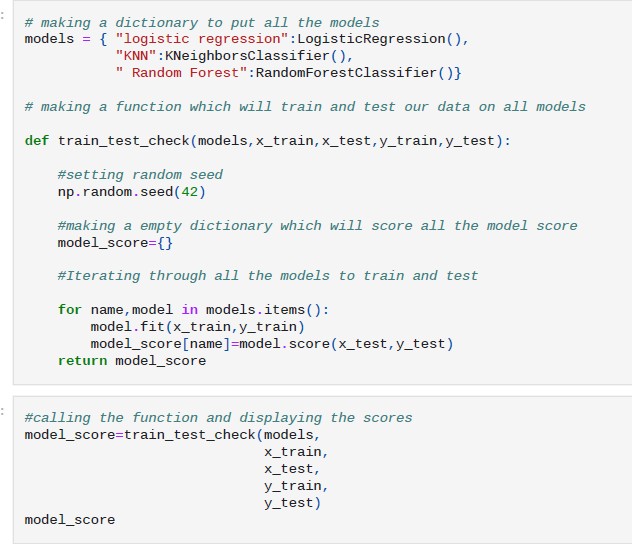
The values that are stored in the x\_test variable:

The values that are stored in the y\_test variable:

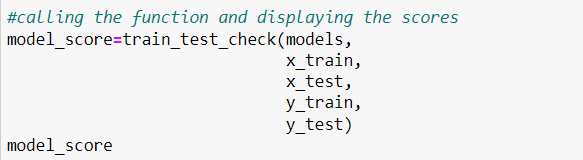


**Figure 21**

Now, I will be applying various classification models and see which have better accuracy



**Figure 22**



**Figure 23**

I created a dictionary named ‘models’ to store the value returned by the methods:

**LogisticRegression(), KNeighborsClassifier(), RandomForest Classifier().**

I called the user defined method i.e., train\_test\_check() that takes the dictionary models and the variables that are used to store the input and output of the training and the testing data.

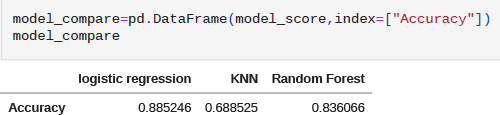
Inside the user defined method, I used **model.fit** method which is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. And storing the data into another dictionary named **model\_score**.



**Figure 24**

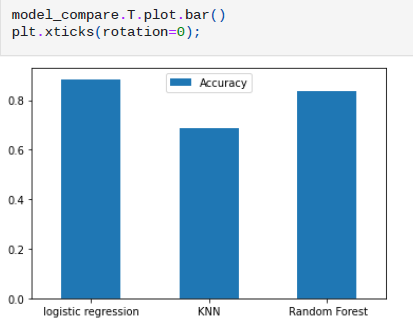
This helps to fill the values of the **LogisticRegression, KNeighborsClassifier, RandomForestClassifier** inside the dictionary.

Then I visualize the accuracy of the model using the dataframe method and storing the data into a variable **model\_compare**.



**Figure 25**

Now plotting the data into graph:



**Figure 26**

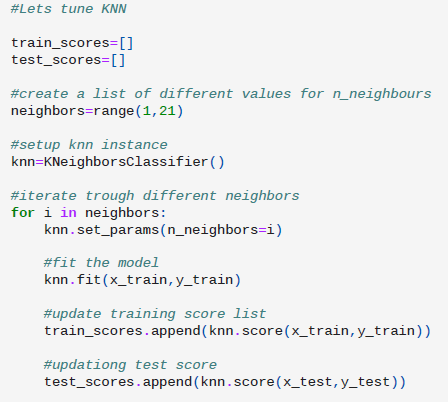
This is the accuracy that we got from the normal data, let’s see if I could get more accuracy if I change our models.

**First one:**

**Hyperparameter tuning** is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning.

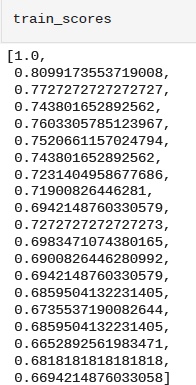
The hyperparameter method that I created, iterates through the nieghbours and then uses **knn.set\_params** to set the parameters of this estimator. The method works on simple estimators as well as on nested object.

Again, I check the accuracy and stores the score obtained into two different lists.

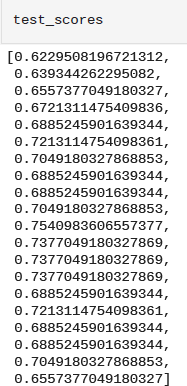


**Figure 27**

The values present in the list named **train\_scores** as used in the hyperparameter tuning:

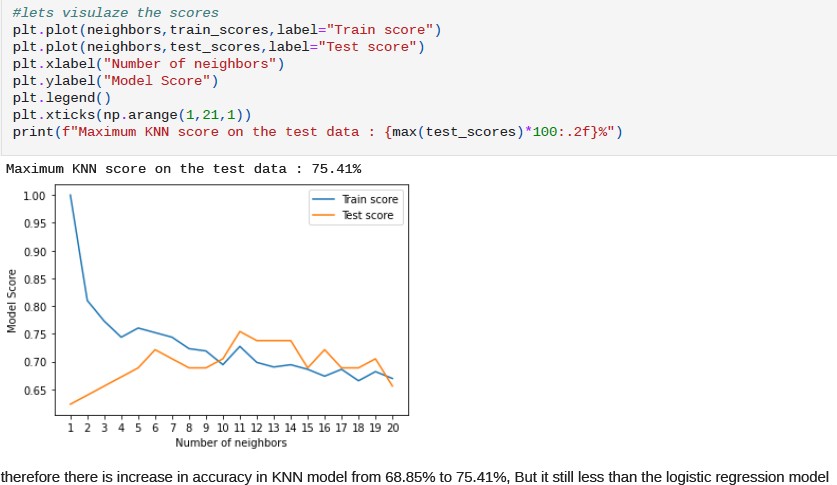


The values present in the list named **test\_scores** as used in the hyperparameter tuning:



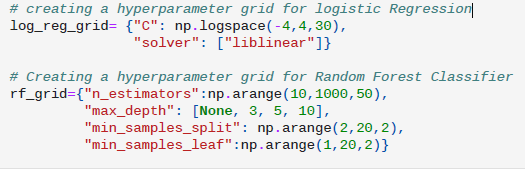
**Figure 28**

Now plotting the data generated by the hyperparameter tuning into a graph for better visualization of the data and analysis. I used the defined matplotlib methods to plot the data.



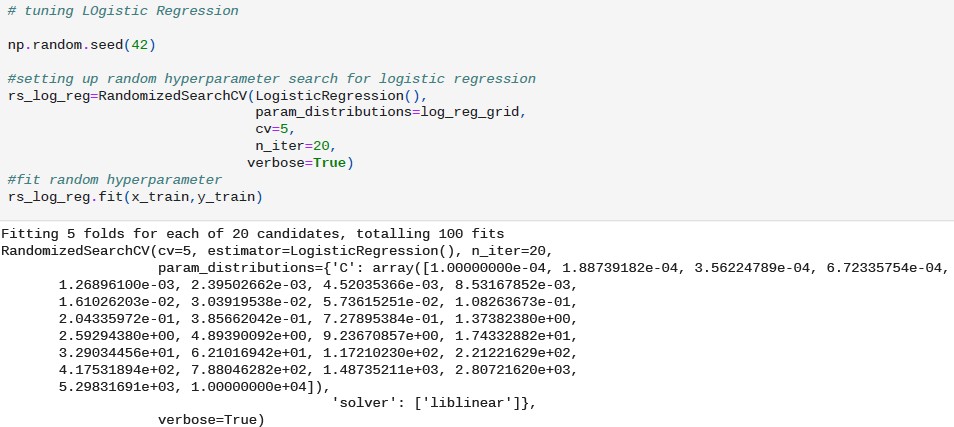
We could clearly see that the accuracy of data in KNN model is increased from 68.85% to 75.41%, but we need to consider one thing that the accuracy is still less than the logistic regression model.

We created the hyperparameter tuning of Logistic Regression and Random Forest Model using Randomized Search cv (cross validation)



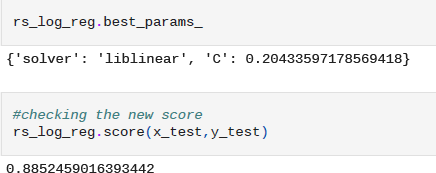
**Figure 29**

I used numpy functions that are used to calculate the mathematical functions.



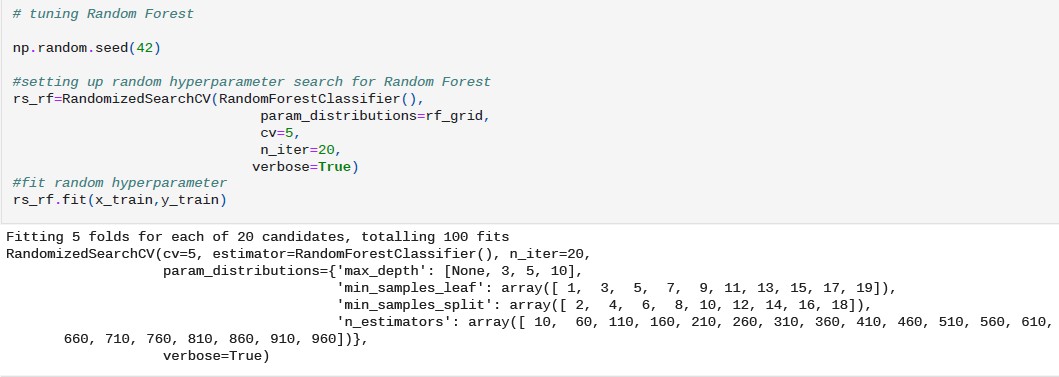
**Figure 30**

I used the **rs\_log\_reg** variable to store the accuracy values of the new model.



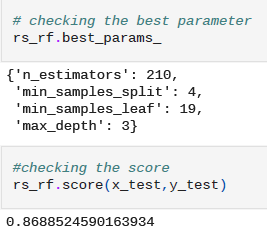
**Figure 31**

This shows the same accuracy that I got first time. Please note that the distribution is based upon the logarithmic function.

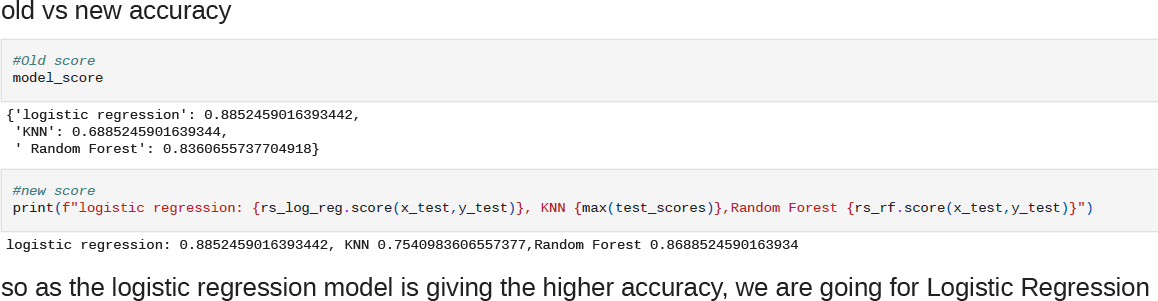


**Figure 32**

Here the distribution is based upon the grid function.

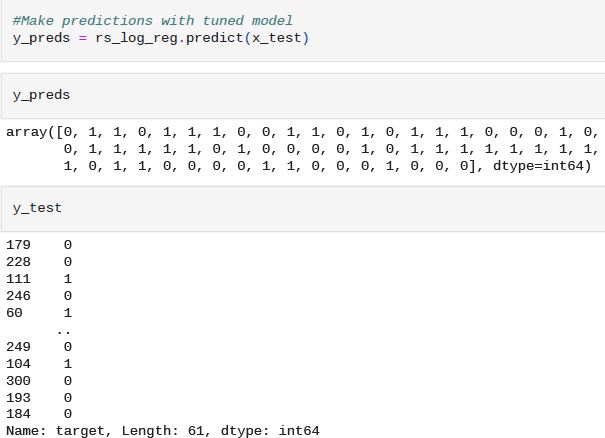


Comparing the old and the new accuracy values:

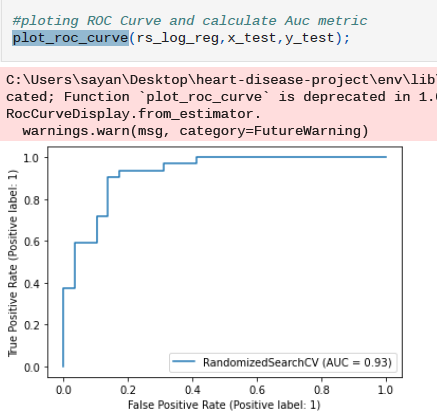


**Figure 33**

Evaluating the tuned model:



**Figure 34**

 Now plotting the values into a graph:

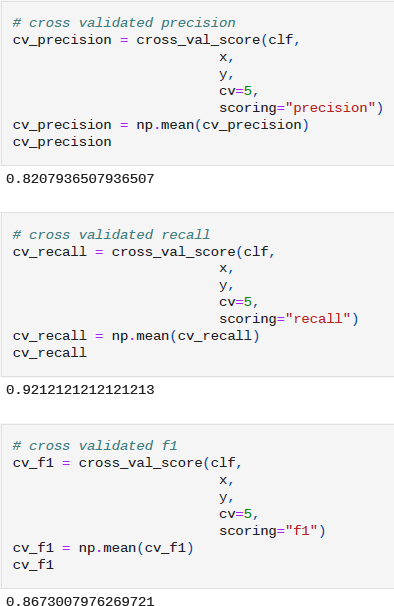
**Figure 35**

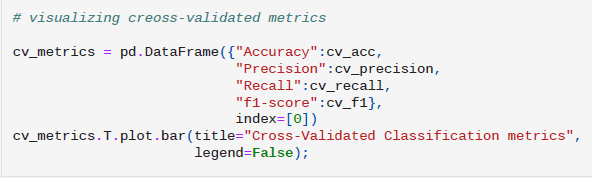


We plotted the data generated into maps. We calculated metrics using crosstabulation:



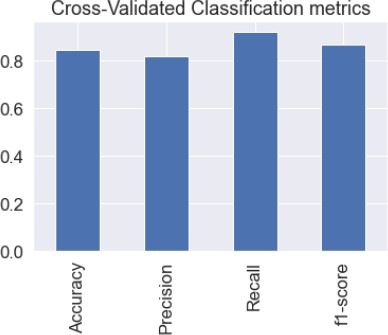
**Figure 36**



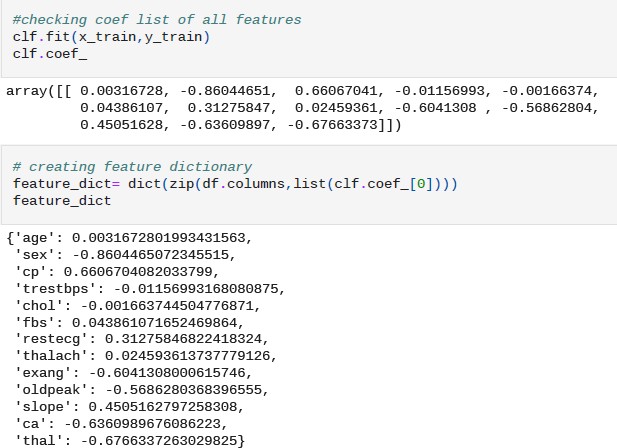


**Figure 37**

Plotting the value into the graph:

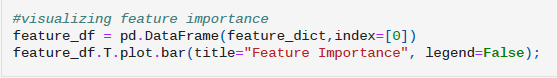


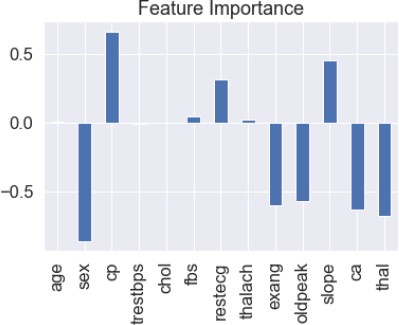
**Figure 38**



**Figure 39**

The final plotting:





**Figure 40**

Hence, the most important feature in our dataset is cp which denotes 'chest pain'.

**Result:**

I found that in the given dataset, the number of persons of males having the heart disease are more than the number of females having the heart disease.

Then it is found that, most people suffer from Type 2 category of chest pain.

I trained the data with different machine learning models and out of it, it is found that Logistic regression is the suitable model which gives high accuracy for the selected dataset and then performed some turing operations in order to check whether the accuracy will increase or not.

|  |  |
| --- | --- |
| Logistic Regression | 0.8852459016393442 |
| K-Nearest Neighbor | 0.6885245901639344 |
| Random Forest | 0.8360655737704918 |

After performing hyperparameter tuning, the accuracy score is:

|  |  |
| --- | --- |
| Logistic Regression | 0.8852459016393442 |
| K-Nearest Neighbor | 0.7540983606557377 |
| Random Forest | 0.8688524590163934 |

Even though the accuracy of K-Nearest Neighbor and Random Forest increased than the previous score but they are still less than the Logistic Regression accuracy score. So, for the prediction of heart disease, we could prefer Logistic Regression for most accuracy.

**Conclusion**

In the physical organism, the heart plays a crucial role. Heart disease necessitates greater precision and accuracy in diagnosis and analysis. Heart disorders may not be detectable in their early stages in real time. This has to be looked at more. Using a data set of heart disorders, the suggested work presents a reliable and early heart disease prediction. The provided methodology necessitates the use of a variety of machine learning methods. The analysis is based on the Confusion matrix, and the accuracy of the algorithms is compared, and Logistic Regression is shown to be the best. As a result, the efficacy of the work given has been confirmed. This method could be utilized to aid in the early and precise detection of cardiac disease. There are numerous machine learning techniques that can be employed for finer exploration and early prediction of heart illnesses in the future. This has to be looked at more.

# REFERENCES

[1] https://[www.cancer.gov/publications/dictionaries/cancer-terms/def/heart-](http://www.cancer.gov/publications/dictionaries/cancer-terms/def/heart-) disease

[2] https://[www.w3schools.com/python/numpy/numpy\_intro.asp](http://www.w3schools.com/python/numpy/numpy_intro.asp)

[3] https://[www.mygreatlearning.com/blog/python-numpy-tutorial/](http://www.mygreatlearning.com/blog/python-numpy-tutorial/)

[4] https://[www.w3schools.com/python/numpy/numpy\_array\_indexing.asp](http://www.w3schools.com/python/numpy/numpy_array_indexing.asp)

[5] https://pandas.pydata.org/about/index.html

[6] https://[www.geeksforgeeks.org/introduction-to-seaborn-python/](http://www.geeksforgeeks.org/introduction-to-seaborn-python/)