Heart disease prediction using Machine Learning

Machine Learning is used across many spheres around the world. The healthcare industry is no exception. Machine Learning can play an essential role in predicting presence/absence of Locomotor disorders, Heart diseases and more. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis.

Heart disease could mean range of different conditions that could affect your heart. It is one of the most complex disease to predict given number of factors in your body that can potentially lead to it. Identifying and predicting it poses a great deal of challenge for doctors and researchers alike.

In this article, I’ll discuss a project where I worked on predicting potential Heart Diseases in people using Machine Learning algorithms. The algorithms included K Neighbors Classifier, Support Vector Classifier, Decision Tree Classifier and Random Forest Classifier. The dataset has been taken from [Kaggle](https://www.kaggle.com/ronitf/heart-disease-uci" \t "_blank)

There are 303 records in the dataset and contains 14 continuous attributes. The goal is to predict the presence of heart disease in the patient.Here are the 14 attributes from the dataset along with their descriptions. These attributes have been narrowed down to total of 14 in the dataset from the original set of 76 .

**age**: The person’s age in years  
**sex**: The person’s sex (1 = male, 0 = female)  
**cp**: The chest pain experienced (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)  
**trestbps**: The person’s resting blood pressure  
**chol**: The person’s cholesterol measurement in mg/dl  
**fbs**: The person’s fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)  
**restecg**: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes’ criteria)  
**thalach**: The person’s maximum heart rate achieved

**exang**: Exercise induced angina (1 = yes; 0 = no)  
**oldpeak**: ST depression induced by exercise relative to rest (‘ST’ relates to positions on the ECG plot)  
**slope**: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)  
**ca**: The number of major vessels (0–3)  
**thal**: A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversible defect)  
**target**: Heart disease (0 = no, 1 = yes)

**Exploratory Analysis**

Before we start the detailed data analysis, let’s begin with the exploratory analysis to understand how data is distributed and extract the preliminary knowledge.

First things first, download the data and import the dataset to Pandas data frame.

**Import libraries:**

I imported several libraries for the project:

**1)numpy**: To work with arrays

**2)pandas**: To work with csv files and dataframes

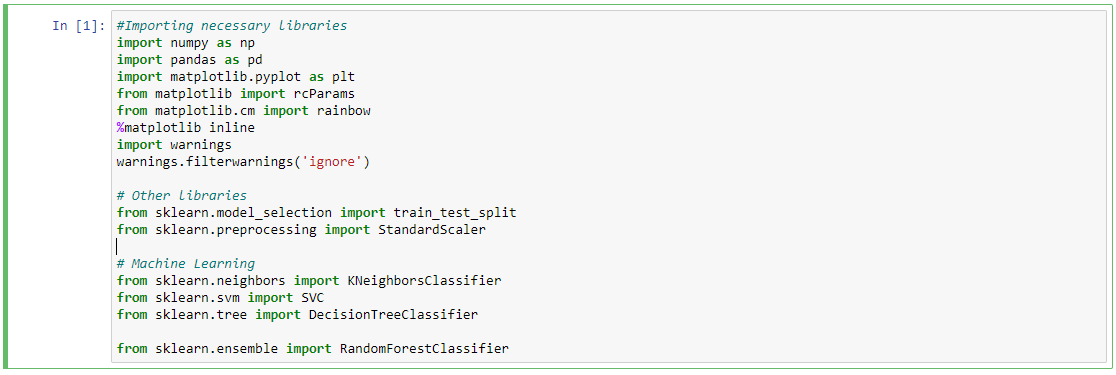
**3)matplotlib**: To create charts using pyplot, define parameters using rcParams and color them with cm.rainbow

**4)warnings**: To ignore all warnings which might be showing up in the notebook due to past/future depreciation of a feature

**5)train\_test\_split**: To split the dataset into training and testing data

**6)StandardScaler**: To scale all the features, so that the Machine Learning model better adapts to the dataset

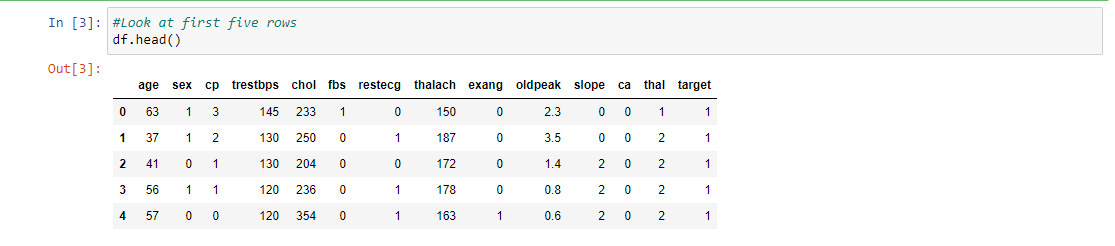
Next, I imported all the necessary Machine Learning algorithms.



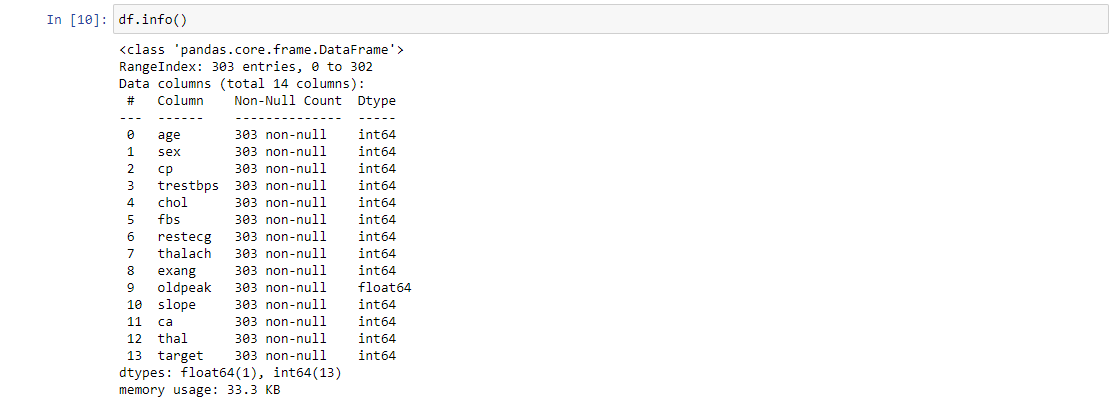
Import the dataset to Pandas data frame:



Let’s have a look at first few rows with the head()



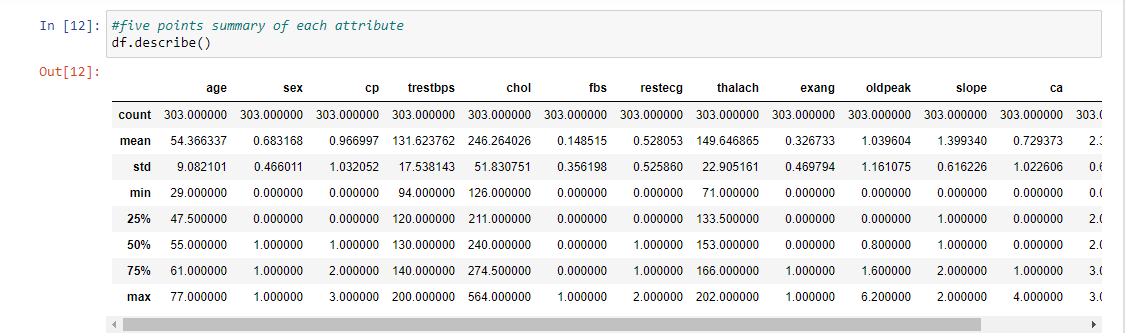
Before any analysis, I just wanted to take a look at the data. So, I used the info() method.



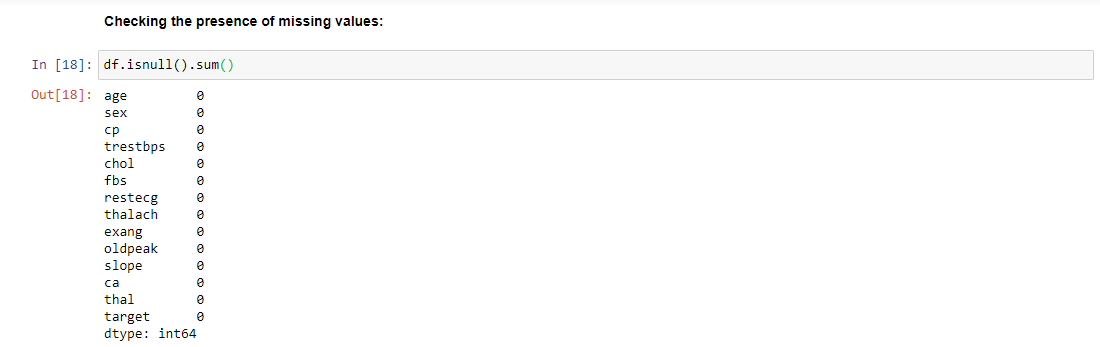
Shape method is used to check the number of rows and columns in the dataset.As we can see there are 303 rows and 14 columns.



Next, I used describe() method for five point summary for each numeric variable.



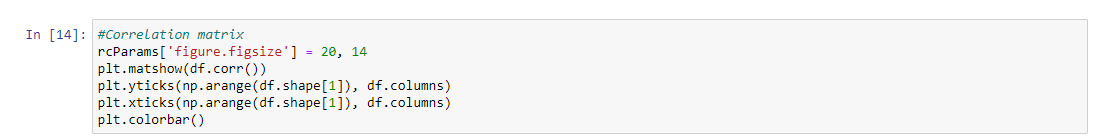
Next ,I used isnull().sum() for checking the missing values.

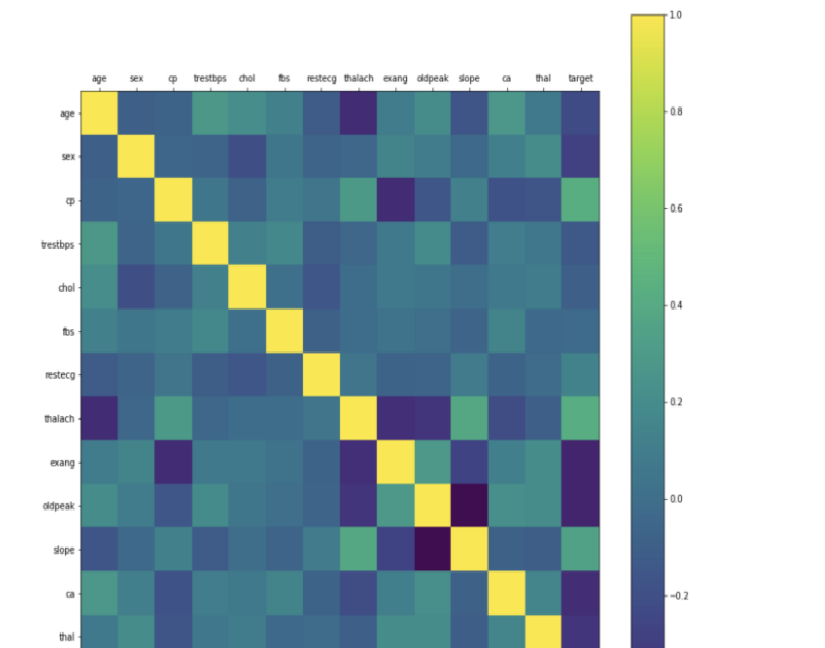


As we can see there is no missing values.

**Correlation Matrix:**

To begin with, let’s see the correlation matrix of features and try to analyse it. The figure size is defined to 12 x 8 by using rcParams. Then, I used pyplot to show the correlation matrix. Using xticks and yticks, I’ve added names to the correlation matrix. colorbar() shows the colorbar for the matrix.



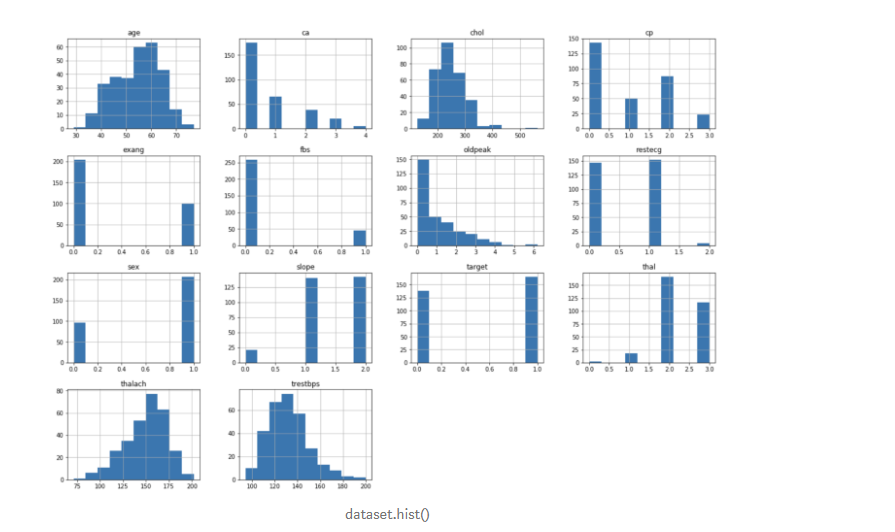


It is very useful to highlight the most correlated variables in a data table. In this plot, correlation coefficients are coloured according to the value.

As you can observe there is no strong correlation between any of the 14 attributes.

**Histogram:**

The best part about this type of plot is that it just takes a single command to draw the plots and it provides so much information in return. Just use dataset.hist().

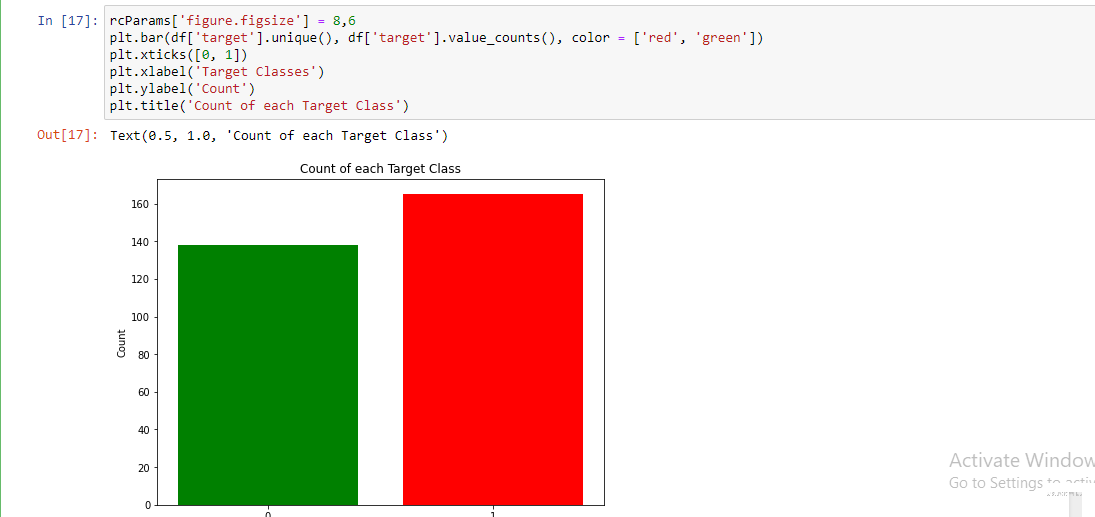


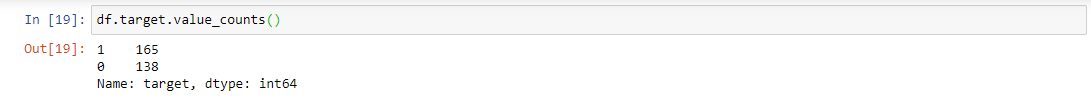
Let’s take a look at the plots. It shows how each feature and label is distributed along different ranges, which further confirms the need for scaling. Next, wherever you see discrete bars, it basically means that each of these is actually a categorical variable. We will need to handle these categorical variables before applying Machine Learning. Our target labels have two classes, 0 for no disease and 1 for disease.

Bar Plot for Target Class :

It’s really essential that the dataset we are working on should be approximately balanced. An extremely imbalanced dataset can render the whole model training useless and thus, will be of no use. Let’s understand it with an example.

Let’s say we have a dataset of 100 people with 99 non-patients and 1 patient. Without even training and learning anything, the model can always say that any new **person** would be a non-patient and have an accuracy of 99%. However, as we are more interested in identifying the 1 person who is a patient, we need balanced datasets so that our model actually learn.





From the plot, we can see that the classes are almost balanced and we are good to proceed with data processing.As there is no class imbalance in the data.

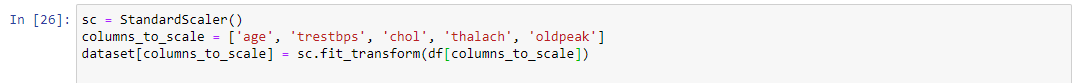
**Data Processing:**

To work with categorical variables, we should break each categorical column into dummy columns with 1s and 0s.

Let’s say we have a column Gender, with values 1 for Male and 0 for Female. It needs to be converted into two columns with the value 1 where the column would be true and 0 where it will be false.

To get this done, we use the get\_dummies() method from pandas. Next, we need to scale the dataset for which we will use the StandardScaler. The fit\_transform() method of the scaler scales the data and we update the columns.

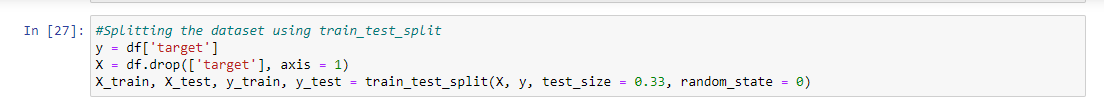




The dataset is now ready. We can begin with training our models.

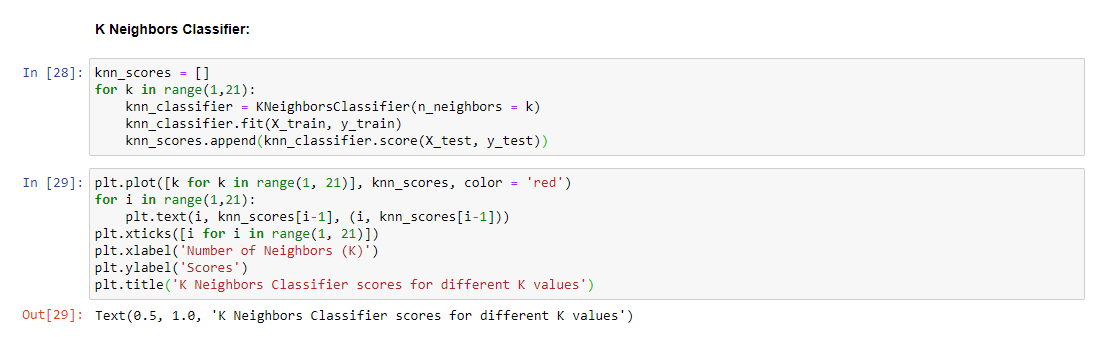
**Machine Learning:**

In this project, I took 4 algorithms and varied their various parameters and compared the final models. I split the dataset into 67% training data and 33% testing data.

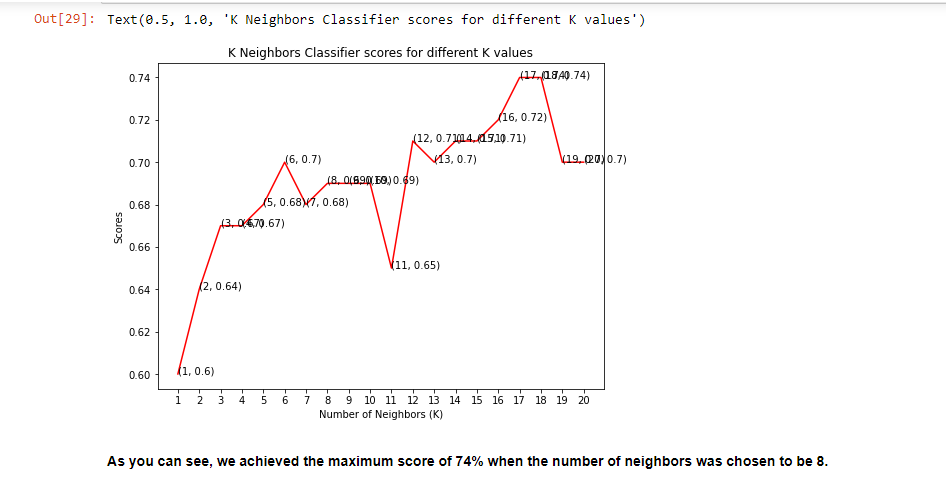


**KNearest Neighbors Classifier :**

This classifier looks for the classes of K nearest neighbors of a given data point and based on the majority class, it assigns a class to this data point. However, the number of neighbors can be varied. I varied them from 1 to 20 neighbors and calculated the test score in each case.

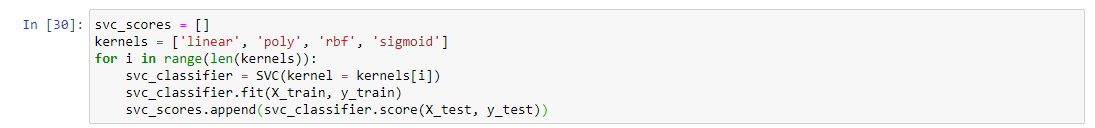


Then, I plot a line graph of the number of neighbors and the test score achieved in each case.

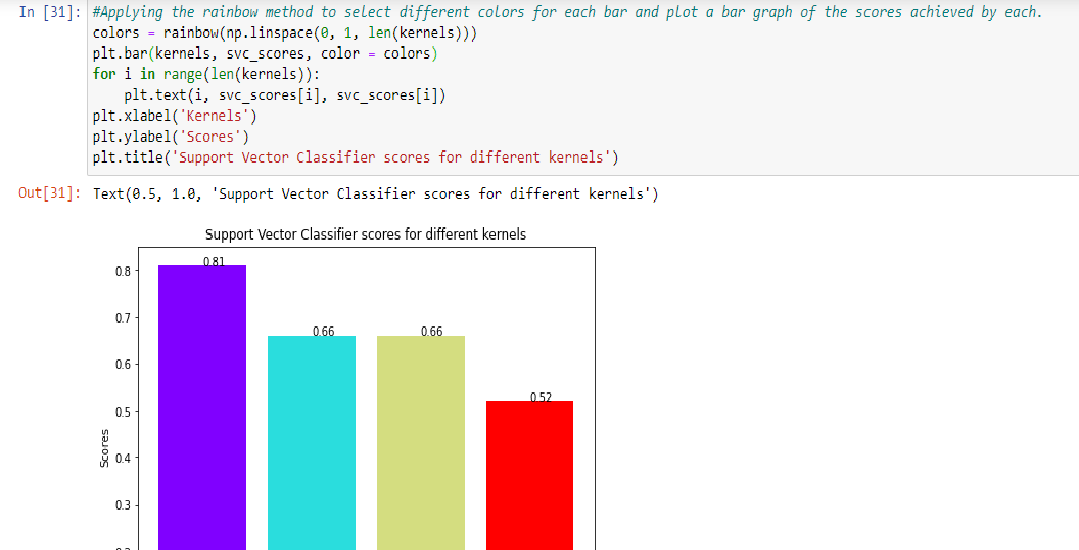
 **As you can see, we achieved the maximum score of 74% when the number of neighbors was chosen to be 8.**

**Support Vector Classifier:**

This classifier aims at forming a hyperplane that can separate the classes as much as possible by adjusting the distance between the data points and the hyperplane. There are several kernels based on which the hyperplane is decided. I tried four kernels namely, linear, poly, rbf, and sigmoid.



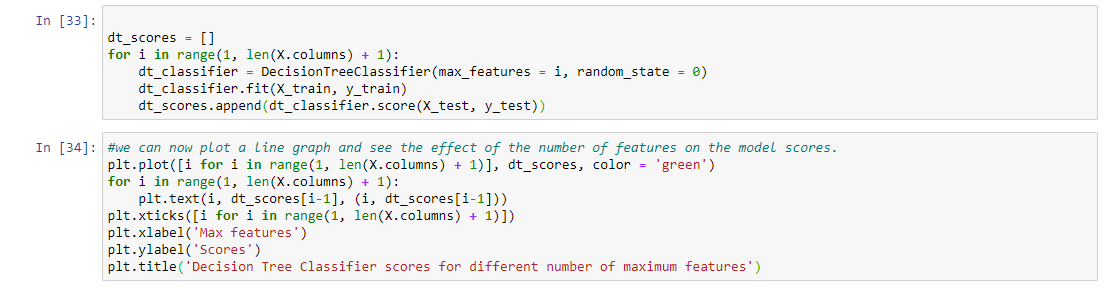
Once I had the scores for each, I used the rainbow method to select different colors for each bar and plot a bar graph of the scores achieved by each.



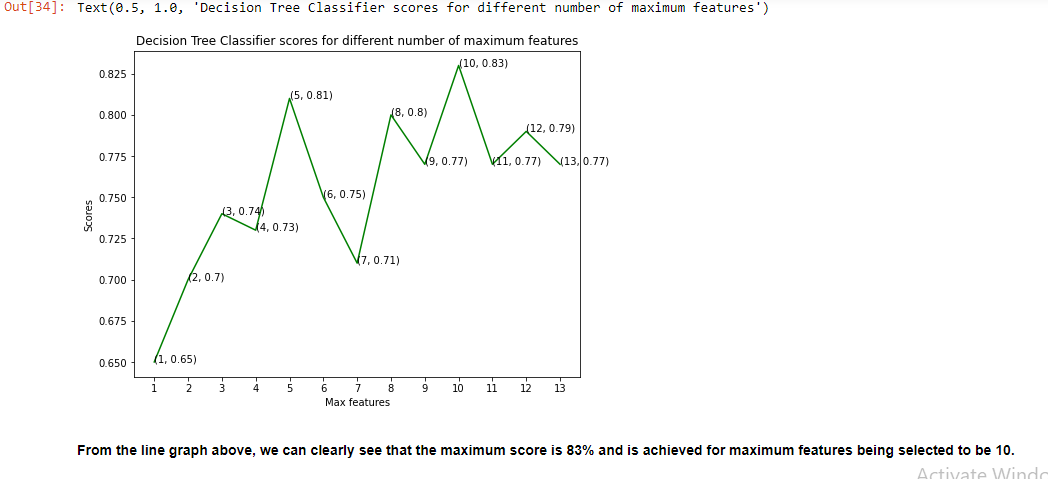
**As can be seen from the plot above, the linear kernel performed the best for this dataset and achieved a score of 83%.**

**Decision Tree Classifier:**

This classifier creates a decision tree based on which, it assigns the class values to each data point. Here, we can vary the maximum number of features to be considered while creating the model. I range features from 1 to 30 (the total features in the dataset after dummy columns were added).

****

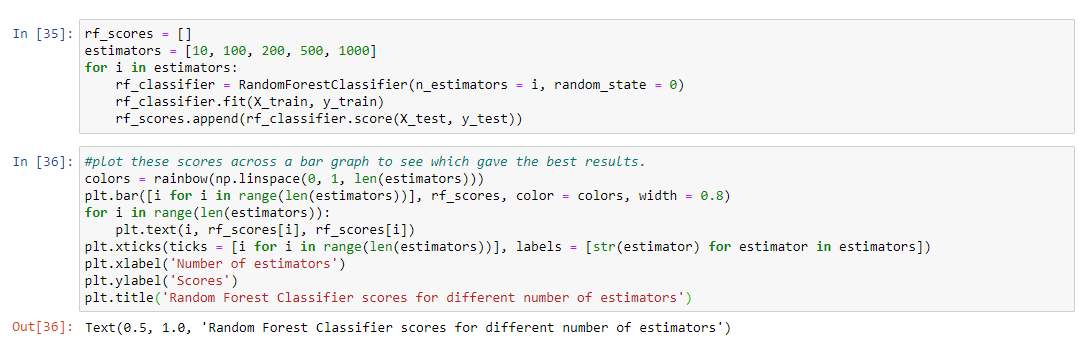
Once we have the scores, we can then plot a line graph and see the effect of the number of features on the model scores.

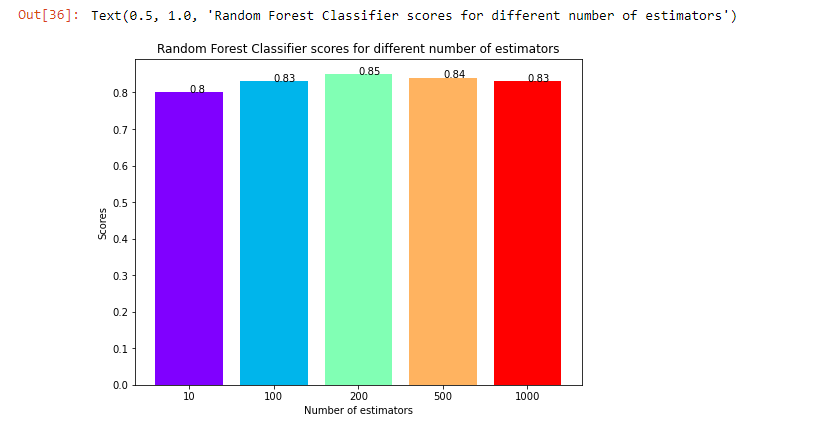


**From the line graph above, we can clearly see that the maximum score is 83% and is achieved for maximum features being selected to be 10.**

**Random Forest Classifier:**

This classifier takes the concept of decision trees to the next level. It creates a forest of trees where each tree is formed by a random selection of features from the total features. Here, we can vary the number of trees that will be used to predict the class. I calculate test scores over 10, 100, 200, 500 and 1000 trees.





**Taking a look at the bar graph, we can see that the maximum score of 85% was achieved for 200 trees**.

**Conclusion** The project involved analysis of the heart disease patient dataset with proper data processing. Then, 4 models were trained and tested with maximum scores as follows:

K Neighbors Classifier: 74%

Support Vector Classifier: 81%

Decision Tree Classifier: 83%

Random Forest Classifier: 85%

**RandomForest Classifier scored the best score of 85% with 200 trees.**