Comparative Analysis of Machine Learning Approaches for Heart Disease Prediction

Md Shahadat Islam
md.shahadat.cse13@gmail.com

Batch-14

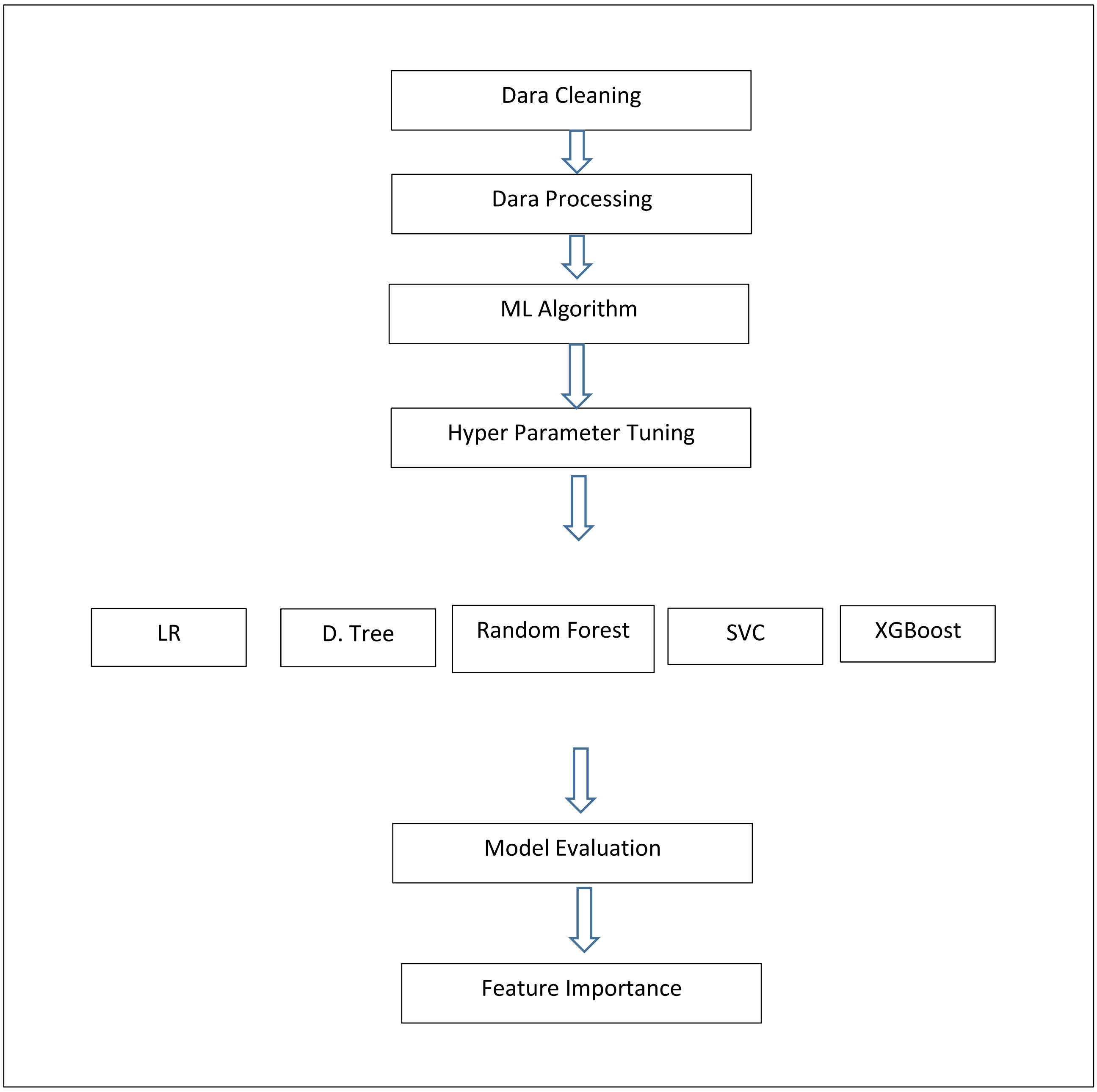
Data Science & ML with Python

AIQuest.org

Introduction:

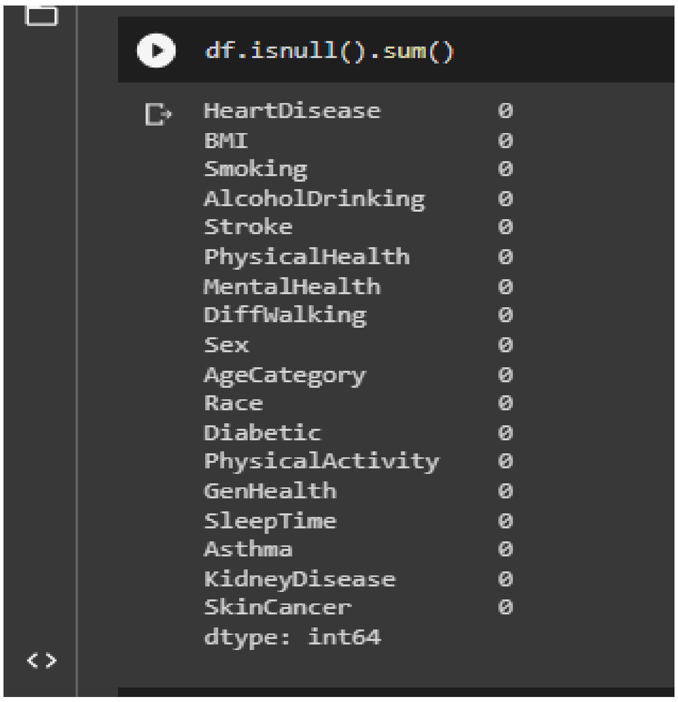
We have been given a task where we have to predict heart disease using machine learning algorithm. The dataset was provided in the project description itself where the original dataset of nearly 300 variables was reduced to just about 18 variables. We mainly focused on 5 classifier models, Decision Tree, Random Forest, Logistic Regression, XGBoost, Support Vector Machine. Now let's execute it.

Model Architecture:



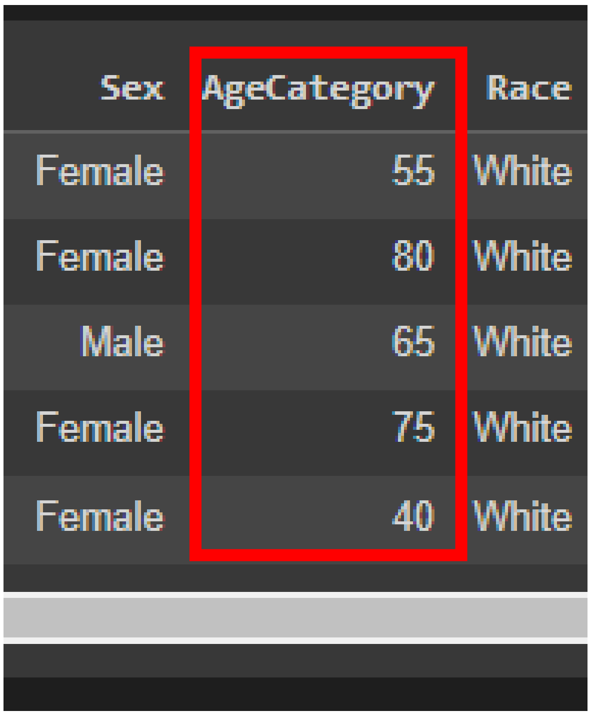
1. Data Cleaning:

Since we have to work on existing data sets, there is a strong possibility of having missing values, mixed or noisy data, biased data, etc. But thanks to our teacher, he was kind enough not to give us a noisy date set. Still we've checked!



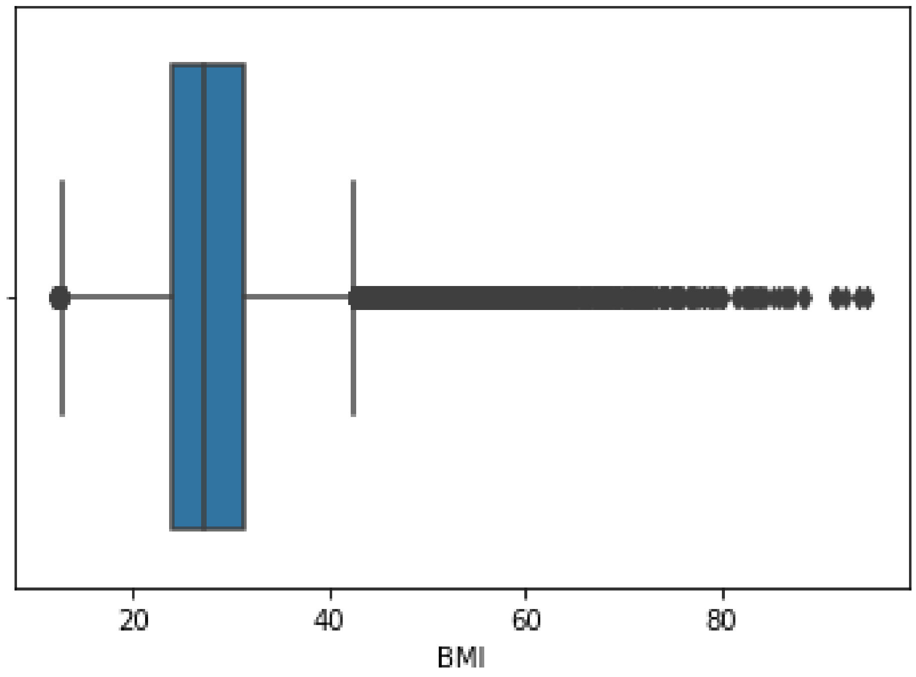
```
df.isnull().sum()

HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

But, the data in the age category column was of string type. It is better to fit numerical data rather than categorical strings to our understanding. So we extracted the minimum age from the AgeCategory column and replaced it in the Agecategory attribute.



Sex	AgeCategory	Race
Female	55	White
Female	80	White
Male	65	White
Female	75	White
Female	40	White

Next, we have focused on outliers. There is a chance to have outliers in 'BMI' column and it has. We remove outliers.

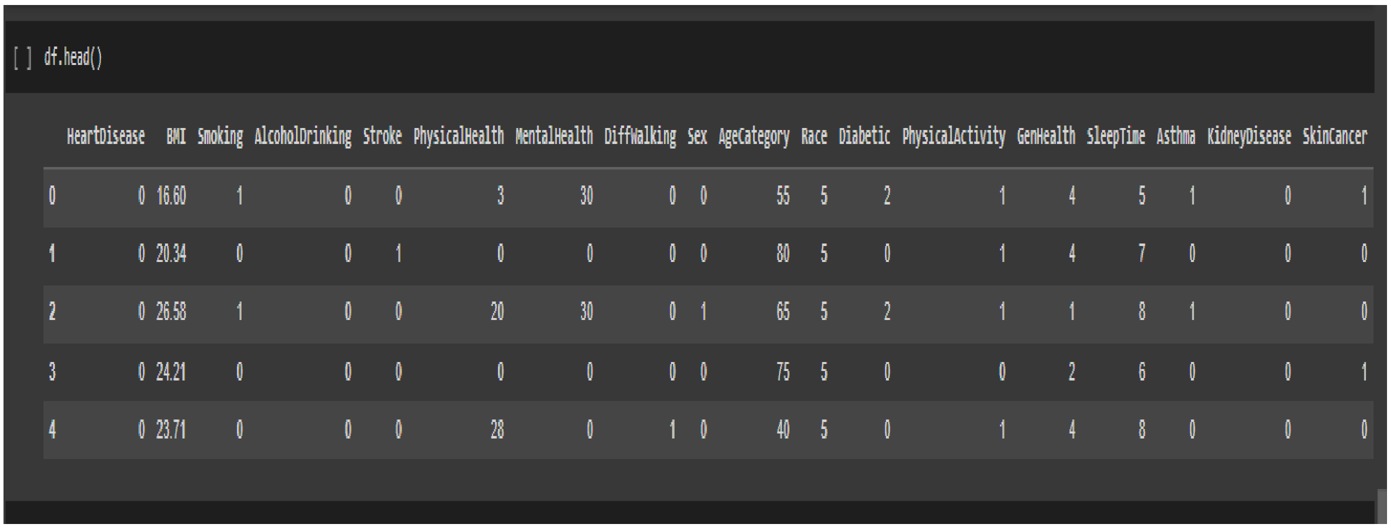


In Last we have dropped all duplicates index just to reduce our dataset.

After doing all these we have 290579 index left.

2. Data Processing:

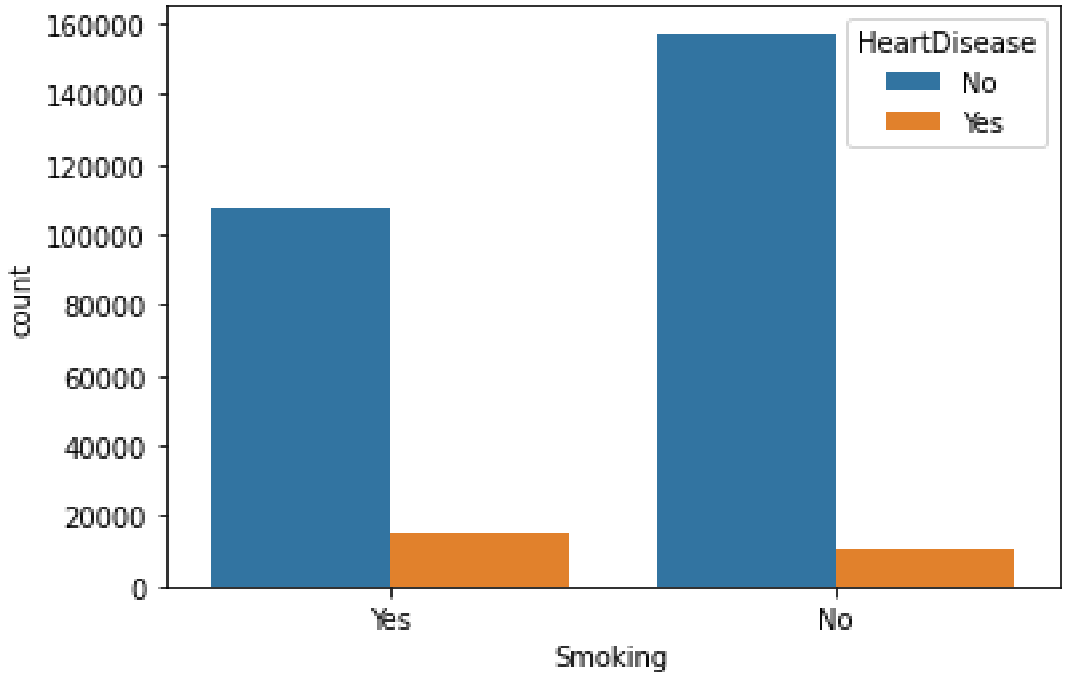
- There are 9 boolean, 5 numerical, rest 4 are categorical feature in our dataset, so it is so much obvious to use Label encoder for encoding.

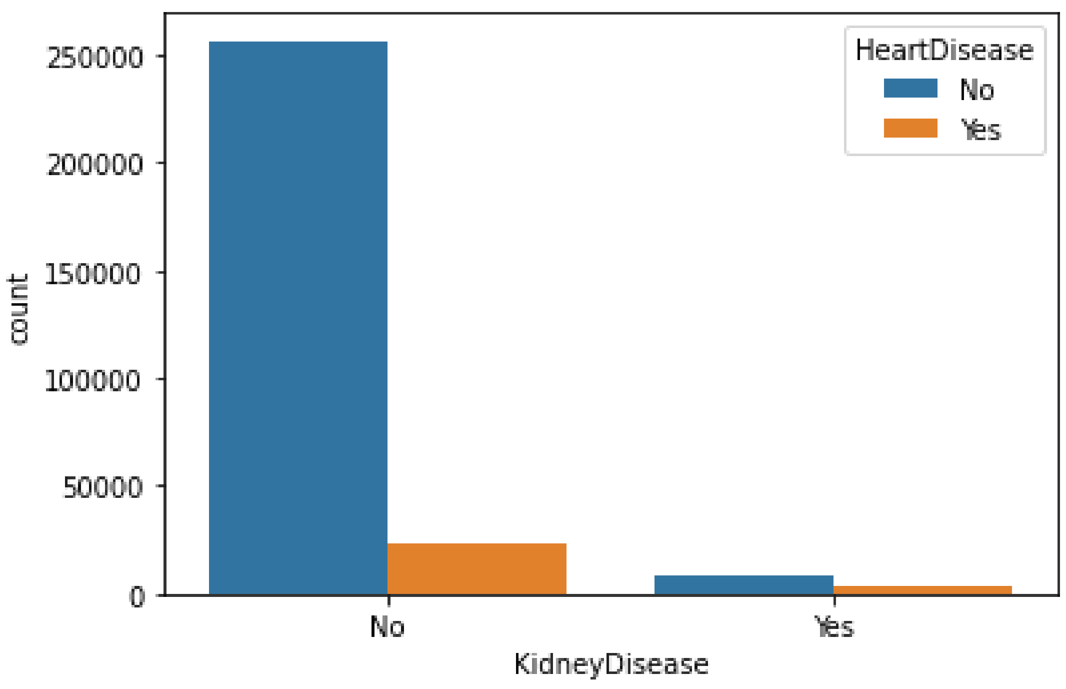


```
[ ] df.head()
```

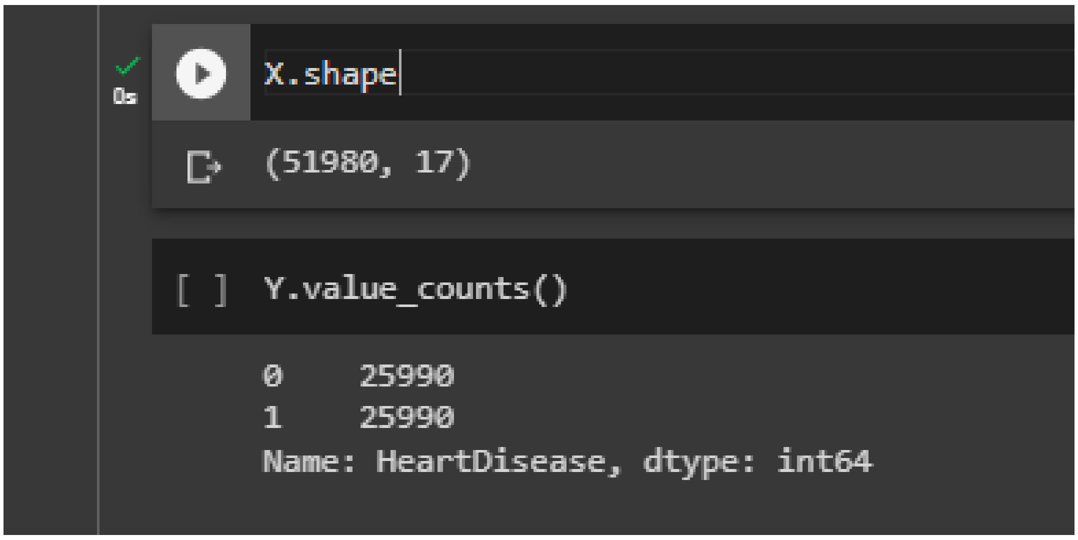
```
   HeartDisease    BMI  Smoking  AlcoholDrinking  Stroke  PhysicalHealth  MentalHealth  DiffWalking  Sex  AgeCategory  Race  Diabetic  PhysicalActivity  GenHealth  SleepTime  Asthma  KidneyDisease  SkinCancer
0             0  16.60        1                0       0               3            30            0    0           55     5         2                 1          4          5       1              0           1
1             0  20.34        0                0       1               0             0            0    0           80     5         0                 1          4          7       0              0           0
2             0  26.58        1                0       0              20            30            0    1           65     5         2                 1          1          8       1              0           0
3             0  24.21        0                0       0               0             0            0    0           75     5         0                 0          2          6       0              0           1
4             0  23.71        0                0       0              28             0            1    0           40     5         0                 1          4          8       0              0           0
```

- We have been warned about having a large amount of unbalanced data. We trust our teacher, so we checked its authenticity and our teacher was right. There was a huge imbalance set of data.





So, for handling these we have planned for using Under Sampling, there is so much benefit for using Under Sampling. As we know there is approximately 400k rows in our dataset, it is very much time consuming . So under Sampling will be best fit for handling. So, we did so. After doing Near Miss Undersampling ….



```
X.shape
(51980, 17)
```

```
Y.value_counts()
0    25990
1    25990
Name: HeartDisease, dtype: int64
```

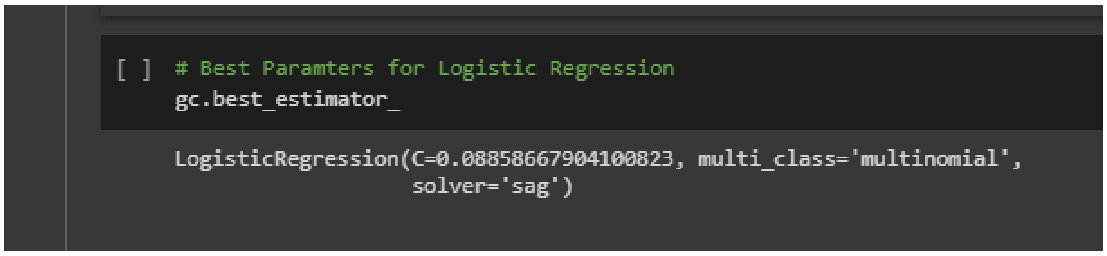
That's it! We're all done. Now we have to fit data to our ML model.

3. Hyper Parameter Tuning:

Before move on to the model training, we've to find our best parameter of the model. Though it's too much time consuming, but it will gives us best result on the testing dataset.

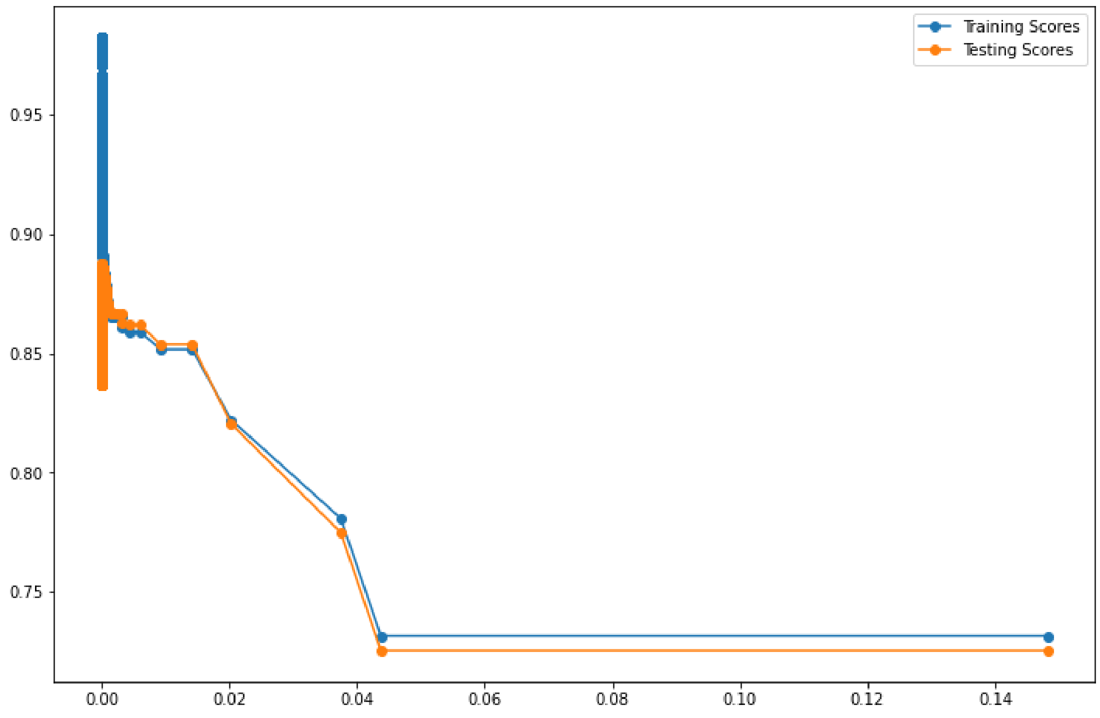
We've applied both RandomizedSearchCV and GridSearchCV on different model. And after enormous amount of time we've finally got our best parameters. Let's have a look!

- **For Logistic Regression**: We've used Grid Search for that Model.



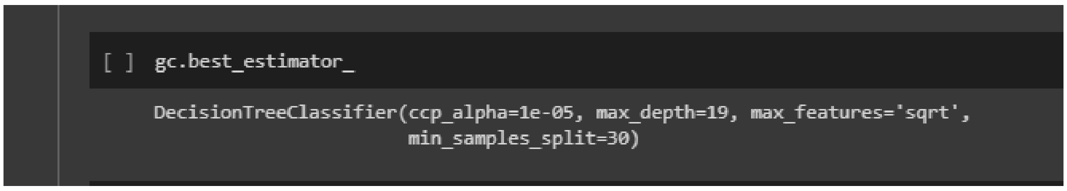
```
[ ] # Best Paramters for Logistic Regression
    gc.best_estimator_

    LogisticRegression(C=0.08858667904100823, multi_class='multinomial',
                       solver='sag')
```

- **For Decision Tree:** In decision tree let's have a look on **ccp(cost complexity pruning) alpha** value.



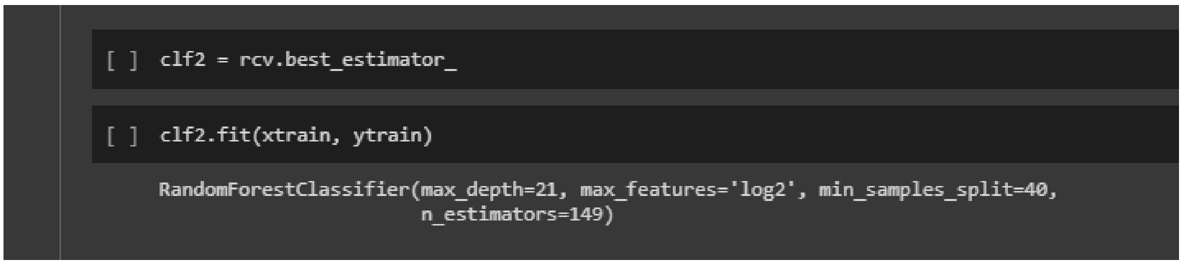
From this plot we can observed near 0.0000001 or 0.02 are the best ccp value for the tree. Now, it's time to see tuning result of Decision Tree Model.

We've used Grid Search for the Model.



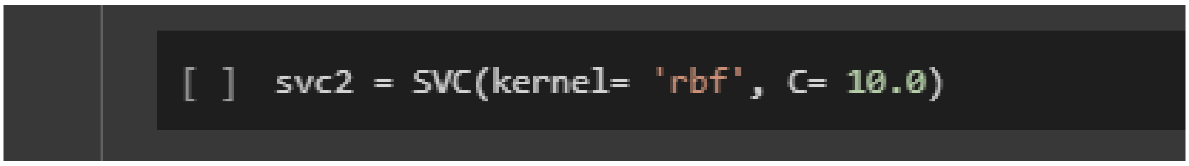
```
[ ] gc.best_estimator_

    DecisionTreeClassifier(ccp_alpha=1e-05, max_depth=19, max_features='sqrt',
                           min_samples_split=30)
```

- **For Random Forest:** We've used Randomized Search for the Model.



```
[ ] clf2 = rcv.best_estimator_

[ ] clf2.fit(xtrain, ytrain)

    RandomForestClassifier(max_depth=21, max_features='log2', min_samples_split=40,
                           n_estimators=149)
```

- **For Support Vector Classifier:** We've used Randomized Search for the Model



```
[ ] svc2 = SVC(kernel= 'rbf', C= 10.0)
```

That's it for Hyper Parameter Tuning.

4. Model Implementation and It's Analysis:

We separated the train and test data into 75 and 35 percentiles for each model train

4.1. Logistic Regression:

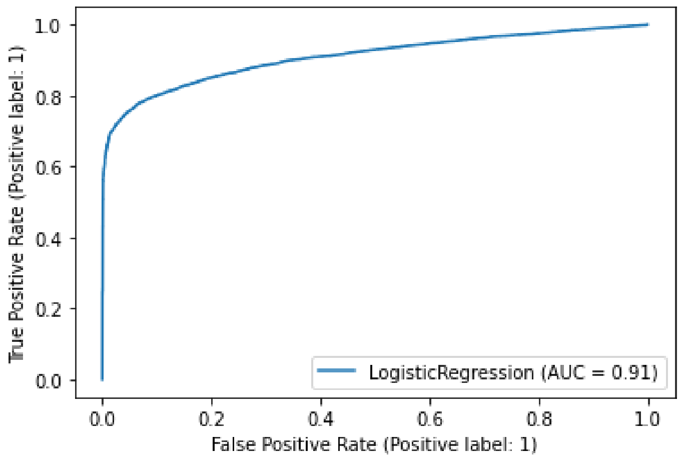
Since logistic regression estimates the probability of a binary outcome based on the value of the independent variable, this model is appropriate for our expected outcome. It is based on Activation Functions, Sigmoid and Softmax function. The softmax function is used for multiclass logistic regression whereas the sigmoid function is used for the two-class logistic regression, and that's what we need to count.

Model Evaluation:

Confusion Matrix:

$$\begin{bmatrix} 7270 & 463 \\ 1820 & 6041 \end{bmatrix}$$

ROC Curve:



4.2 XGBoost:

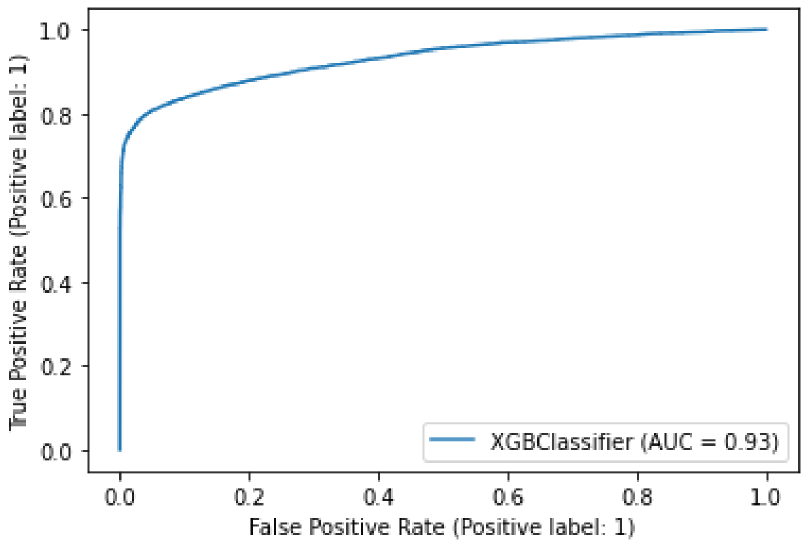
It's an ensemble Method where specific score is assigned to specific tree and it takes advantage misclassification error to produce a better model at each iteration. Initially, we used the default parameters of the algorithm. We have implemented this algorithm can give better results due to higher performance. Because the XGBoost algorithm is highly scalable and specifically designed for processing Parallel computation and its regularization techniques.

Model Evaluation:

Confusion Matrix:

$$\begin{bmatrix} 7319 & 414 \\ 1492 & 6369 \end{bmatrix}$$

ROC Curve:



4.3 Decision Tree:

Let's dive into the popular supervised learning approach-Decision Tree to predict outcome. It's tree-structured model that make decision by dividing feature into the smaller subgroups. It builds the tree by splitting the entire datasets into the subsets based on the features that makes best information gain. Information gain measures the reduction of entropy. This algorithm is so much powerful that can handle both classification and numerical data.

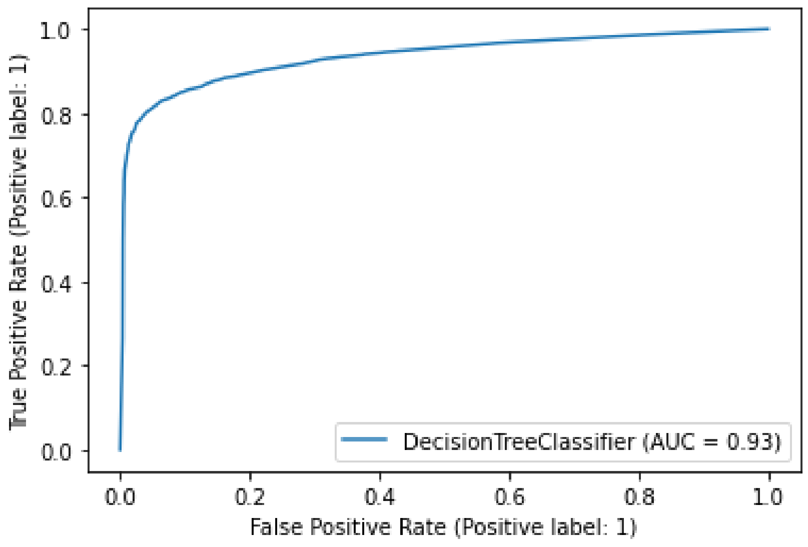
But it can suffer from Overfitting if the tree is too deep. To prevent this we've used Cost Complexity Pruning, in short **ccp.** Cost Complexity Pruning removes nodes that do not impose any significant condition on the accuracy of the model.

Model Evaluation:

Confusion Matrix:

$$\begin{bmatrix} 7403 & 394 \\ 1468 & 6397 \end{bmatrix}$$

ROC Curve:



4.4 Random Forest:

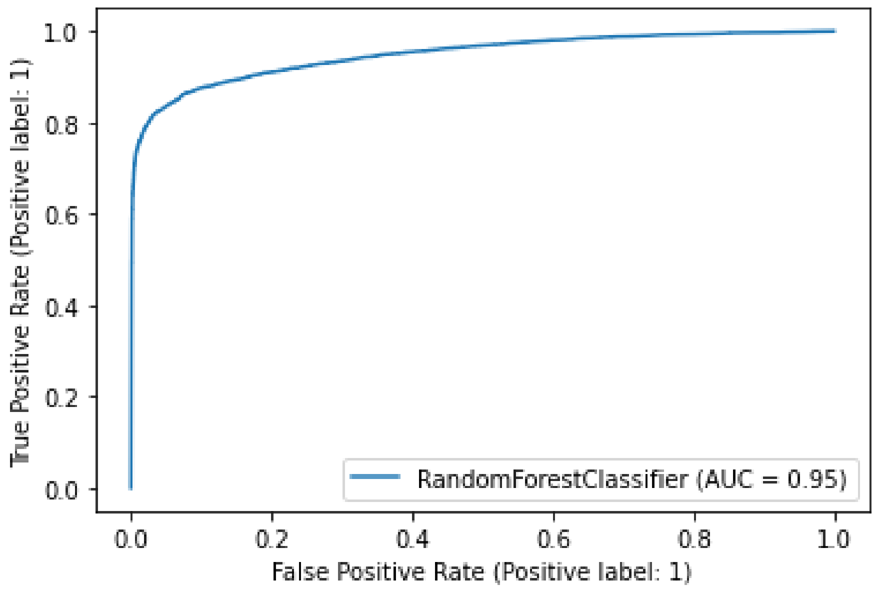
Now let's dive into another popular ensemble learning algorithm that combines multiple tree-based model (decision-tree) to improve the overall performance of the model. During training, it's selects training samples randomly from a subset of features and builds each decision-tree. That helps reducing overfitting and increase the performance of the model. Each Decision tree makes each prediction, and the final prediction is made by aggregating all predictions.

Model Evaluation:

Confusion Matrix:

$$\begin{bmatrix} 7338 & 459 \\ 1229 & 6636 \end{bmatrix}$$

ROC Curve:



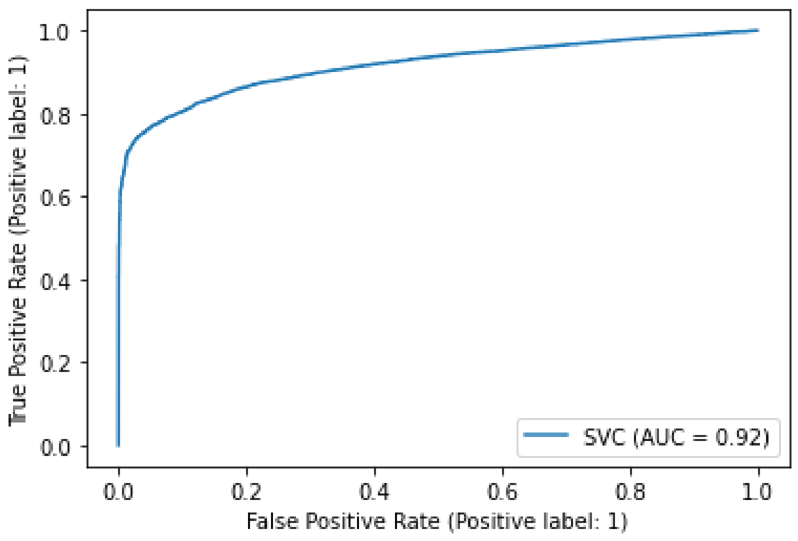
4.5 Support Vector Classifier: The objective of the support vector machine algorithm is to find a hyperplane in an n-dimensional space that neatly classifies the data points. Hyperplanes are decision boundaries that help classify data points. Data points falling on either side of the hyperplane can be attributed to different classes.

Model Evaluation:

Confusion Matrix:

$$\begin{bmatrix} 7431 & 302 \\ 1940 & 5921 \end{bmatrix}$$

ROC Curve:



Cross Validation: We performed cross-validation on three models(SVC, Logistic, Decision Tree) to check for overfitting. For this we applied k-fold cross validation and concluded that there was no overfitting of the data.

Final Decision:

MODEL NAME	TP	FP	FN	TN	ACCURACY
DECISION TREE	7403	394	1468	6397	88%
RANDOM FOREST	7338	459	1229	6636	89%
SVM	7431	302	1940	5921	86%
XGBOOST	7319	414	1492	6369	88%
LOGISTIC REGRESSION	7270	463	1820	6041	85%

By evaluating all the models, we found the random forest model to have the highest accuracy, which is 89%. This is the best fit model for the dataset. The closest accuracy is the Decision tree model, which is 88%. Accuracy on average for the rest of the models is 86%.

5. Feature Importance:

Now is the time to decide which features are most important for predicting heart disease. We have used several methods for this. We used Extra Tree Classifier and Principal Component Analysis.

- **Extra Tree Classifier:** It's works by creating a large number of decision tree. It's make importance by gaining IF(Information gain) value.

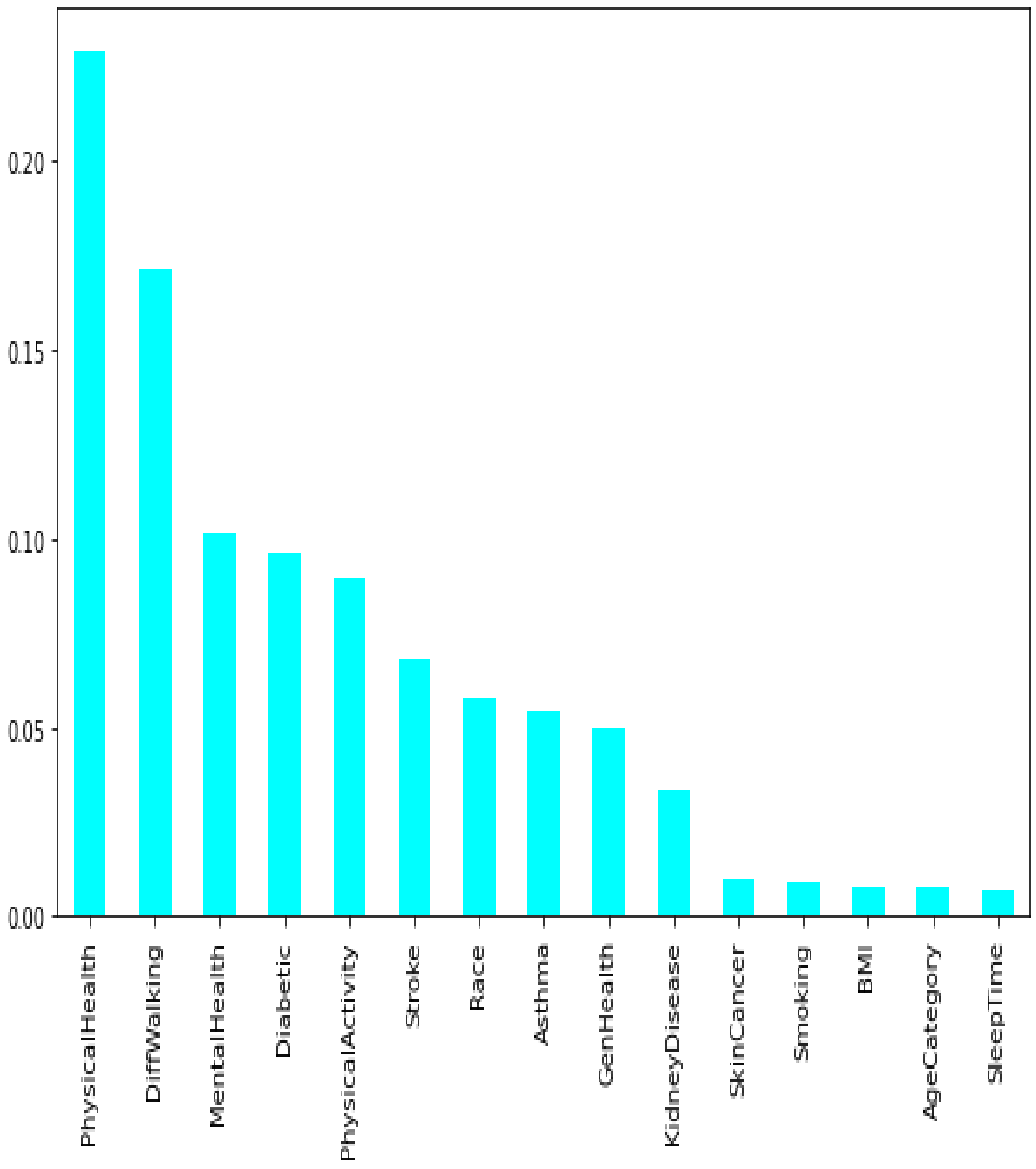
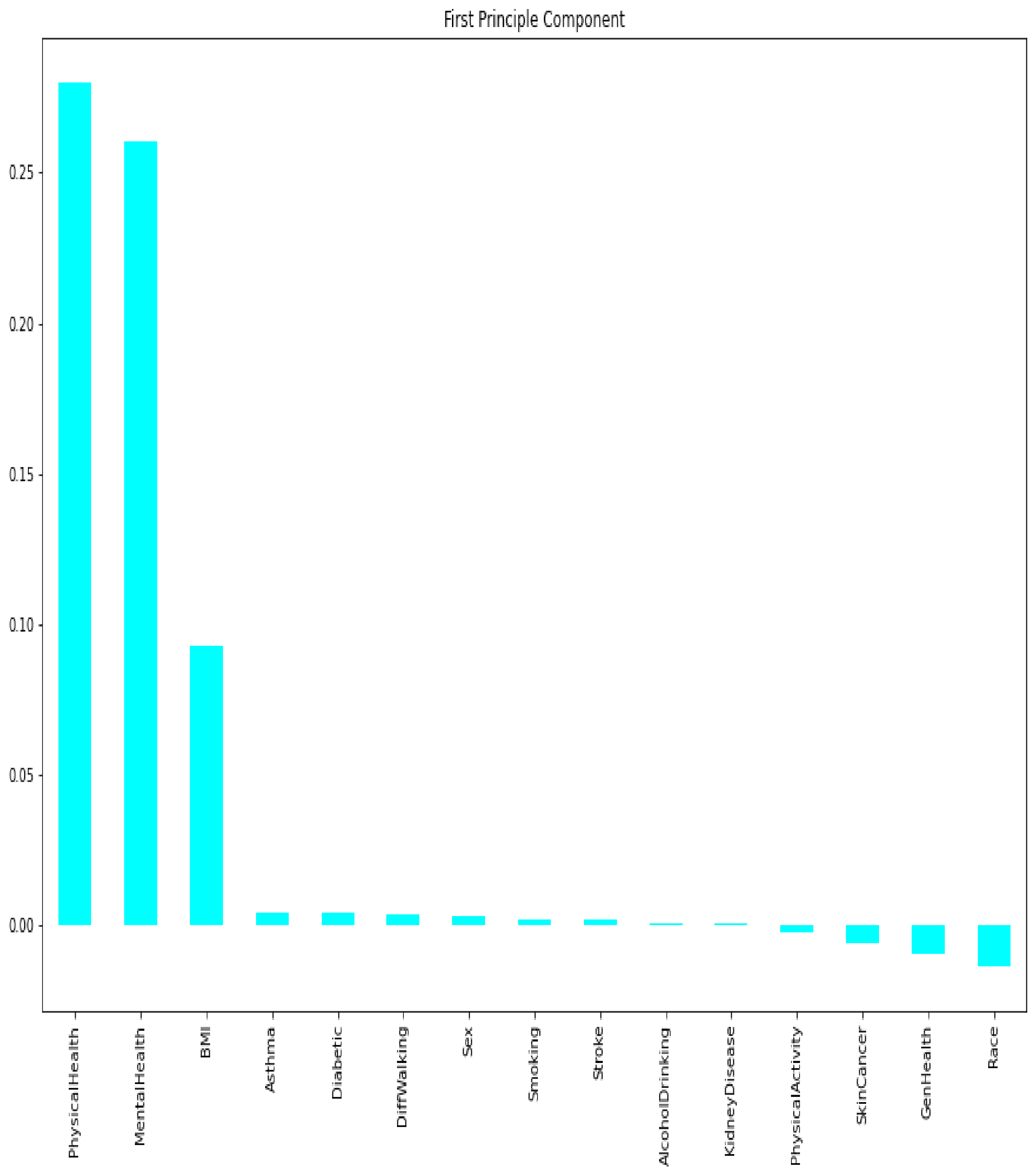


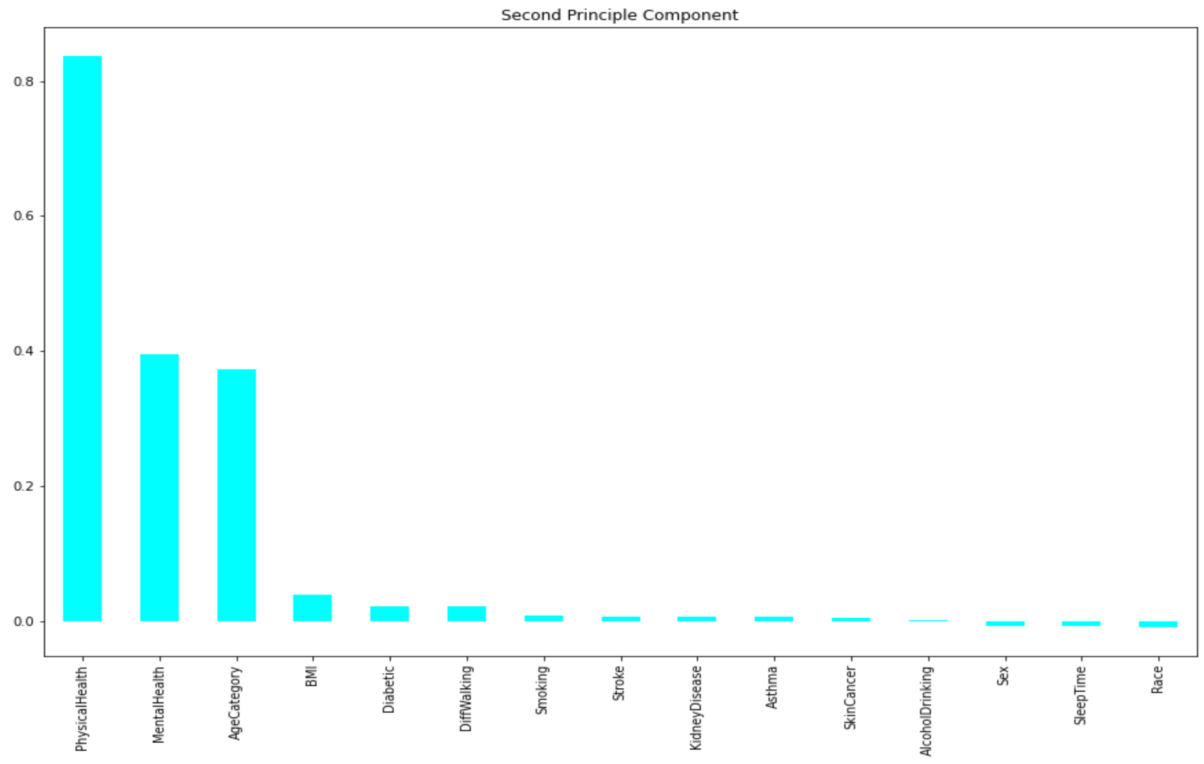
Figure: Feature Important Chart Using Extra tree classifier

- **Principle Component Analysis:** PCA works by computing the eigenvectors and eigenvalues of the covariance matrix of the input data. The eigenvectors indicate the direction of the new coordinate axes. We've used three axes, first principle component, second principle component and third principle component.

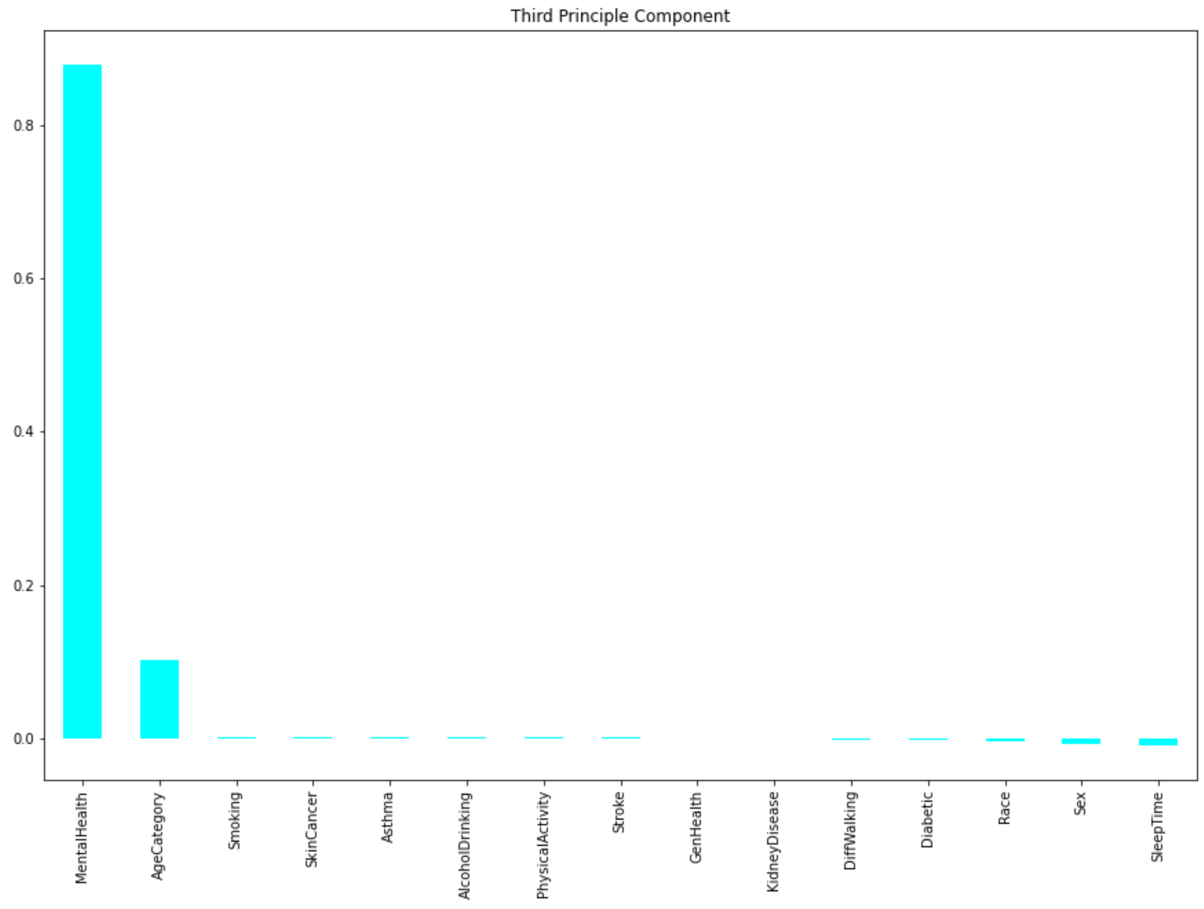
- **PCA-1:**



- **PCA-2:**


Second Principle Component
0.8
0.6
0.4
0.2
0.0
PhysicalHealth
MentalHealth
AgeCategory
BMI
Diabetic
DiffWalking
Smoking
Stroke
KidneyDisease
Asthma
SkinCancer
AlcoholDrinking
Sex
SleepTime
Race

- **PCA-3:**


Third Principle Component
0.8
0.6
0.4
0.2
0.0
MentalHealth
AgeCategory
Smoking
SkinCancer
Asthma
AlcoholDrinking
PhysicalActivity
Stroke
GenHealth
KidneyDisease
DiffWalking
Diabetic
Race
Sex
SleepTime

Conclusion:

Now we can come to a conclusion. **Physical health** plays the most effective role in heart disease. At least that's what the above method tells us. The features in the dataset were already selected so we did not have to use the feature selection method.

Above, we have done heart disease prediction using 5 classifiers Model. Among the Random Classifier Model gave the highest performance. Which gave the highest True Positive rate of 90% and true negative rate of 89%.