Heart Disease prediction using Cleveland dataset and Classification using Random Forest classifier

A PROJECT REPORT

Submitted by

Shatakshi Shukla (16BCE2119)
Vipul Goel (16BCI0115)
Daksh Chawla(16BCI0060)

Course Code: CSE 3013

Course Title: ARTIFICIAL INTELLIGENCE

In partial fulfilment for the award of the degree of

B.Tech

in

Computer Science and Engineering

Under the guidance of

Dr. S. Anto

Associate Professor, SCOPE,

VIT , Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

APRIL, 2019

ABSTRACT

In today's era, each and every human-being on earth depends on medical treatment and medicines. Every day we can hear some new person having heart disease. So, to deal with such situations, we are developing a system, a heart disease prediction model which will predict if the person has heart disease or not with utmost accuracy. One major problem in today's world is hike in Doctor's fee. So, the middle class and lower-class people are unable to afford for the fee and treatment charges. This system is developed taking this fact in mind. Using this system, one can easily find if a person has heart disease or not by entering values of various attributes. The given problem is prediction of heart disease, to classify users from 8 dataset of UCI data repository into 5 classes. Goal of the experiment is to classify users into classes which is non-zero(1, 2, 3 or 4) for severity of presence and zero for absence of heart disease and also to measure the correctness of our classifier.

Keywords: Fisher Score, Random Forest Classifiers, Randomised Search, Cross Validation, Normalization

INDEX

Sr No.	Title	Page No.
1	Introduction	1
1.1	Requirements	2
1.2	Methodology	3-4
2	Literature Survey	4-9
3	Overview of the work	10
3.1	Problem Description	10
3.2	Software Requirements	10
3.3	Hardware Requirements	10
4	System Design	11-13
5	Implementation	14-22
5.1	Description of modules	14
5.2	Source code	15-18
5.4	Execution Snapshots	18-22
6	Conclusion	22
7	References	23

1. INTRODUCTION

Heart disease is one of the prevalent disease that can lead to reduce the lifespan of human beings nowadays. Each year 17.5 million people are dying due to heart disease. Life is dependent on component functioning of heart, because heart is necessary part of our body. Heart disease is a disease that affects on the function of heart. An estimate of a person's risk for coronary heart disease is important for many aspects of health promotion and clinical medicine. A risk prediction model may be obtained through multivariate regression analysis of a longitudinal study. Due to digital technologies are rapidly growing, healthcare centres store huge amount of data in their database that is very complex and challenging to analysis. Data mining techniques and machine learning algorithms play vital roles in analysis of different data in medical centres. The techniques and algorithms can be directly used on a dataset for creating some models or to draw vital conclusions, and inferences from the dataset. Common attributes used for heart disease are Age, Sex, Fasting Blood Pressure, Chest Pain type, Resting ECG (measures the electrical activity of the heart), Number of major vessels colored by fluoroscopy, Threst Blood Pressure (high blood pressure), Serum Cholestrol (determine the risk for developing heart disease), Thalach (maximum heart rate achieved), ST depression (finding on an electrocardiogram, trace in the ST segment is abnormally low below the baseline), painloc (chest pain location), Fasting blood sugar, Exang (exercise included angina), smoke, Hypertension, Food habits, weight, height and obesity.

This report is proposing a system that allows user to predict heart disease instantly through an intelligent heart disease production system system. The system is fed with different attributes associated with heart disease and the system predicts if the person has heart disease or not with maximum accuracy.. Here we use some intelligent data mining and machine learning techniques to predict whether the person has a heart disease or not with utmost accuracy. The model will prove helpful in urgent cases where the patient is unable to reach hospital or in cases when there are no doctors available in the area.

1.1.REQUIREMENTS

Software specification :

- Jupiter Notebook (running on virtual environment) - used for our code, output and visualisation of data
- Python - the code is written in Python3
- Pip

Libraries :

- numpy
- pandas
- matplotlib.pyplot
- seaborn
- from sklearn.model_selection import train_test_split
- from sklearn.preprocessing import Normalizer
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.ensemble import RandomForestRegressor
- from sklearn.metrics import make_scorer, accuracy_score
- from sklearn.model_selection import GridSearchCV
- from sklearn.feature_selection import chi2
- from sklearn.feature_selection import SelectKBest, SelectPercentile
- from sklearn import model_selection
- from sklearn.model_selection import cross_val_score

- from sklearn.metrics import classification_report, confusion_matrix

1.2.METHODOLOGY

Our heart disease prediction model is distributed into four modules - data preprocessing, dimensionality reduction, classification before after dimensionality reduction followed by optimisation. We are using the Cleveland dataset (https://archive.ics.uci.edu/ml/datasets/heart+Disease) for training and testing our model.

Data Pre-Processing:

Cleaning the dataset by replacing the missing values with the mean of that particular feature. Categorical variables are converted to dummy variables during data pre-processing.

Normalization: Normalizing the data is done so that all the input variables have the same treatment in the model and the coefficients of a model are not scaled with respect to the units of the inputs.

Classification Technique:

Random Forest Classifiers - Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Dimensionality Reduction:

Fisher Score Algorithm - Fisher scoring is a hill-climbing algorithm for getting results. It maximizes the likelihood by getting successively closer and closer to the maximum by taking another step (an iteration). It knows when it has reached the top of the hill in that taking another step does not increase the likelihood. It is known to be an efficient procedure as not many steps are usually needed and generally converges to an answer.

Optimization Technique

Randomized Search with cross validation - RandomizedSearchCV implements a “fit” and a “score” method. It also implements “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.

2. LITERATURE SURVEY

Sr No.	Title of the Paper	Auth or s	Year of Publ ic ation	Datas et Used	Methodolog y /Technology	Perfor ma nce metrics	Advant ag es	Drawb ac ks
1	Cardiovas c ular Risk Prediction Method Based on CFS Subset Evaluatio n	Xu, S., Zhang, Z., Wang, D., Hu, J., Duan, X., &	201 7	PKU People's Hospital Cardiolo g y Inpatient Dataset and	CFS Subset Evaluation and Random Forest Classificatio n Framework	Random Forest was proved to be best classifier than others	Great significan c e in accuracy and practical use for patients’	Uses only one classifica t ion framewor k

	and Random Forest Classifica ti on Framewor k	Zhu, T.		Clevelan d Heart- Disease Database (CHDD),			treatment and doctors' diagnose.	
2	A highly accurate firefly based algorith m for heart disease predicti on	Long, N. C., Meesa d, P., & Unger , H.	2015	Heart diseas e dataset (catalo g) And SPEC TF dataset	Firefly based algorithm	Discrete chaos fire- fly algorith m is faster than the binary particle swarm optimiza ti on to achieve rough sets based attribute reductio n.	Uses a combin ati on of rough sets based attribute reductio n with interval type-2 fuzzy logic system for heart disease diagnosi s.	The propose d rough sets based 880 attribute reductio n is unmana ge able when the number of attribute s is huge or when the number of records is large. Trainin g time of interval type-2 fuzzy logic system by chaos firefly 883 and genetic hybrid algorith m s is quite slow.
3	A Scalable Solutio n for Heart Disease Prediction	Rashm i G Saboji and Prem	2017	Clevelan d database	Random Forest Algorith m and Naïve Bayes	Random forest algorith m on Spark framewor k	Implemen t ed random forest algorithm on Spark	Fails to investig at e the impact of large

	using Classification Mining Technique	Kumar Ramesh				for predicting heart disease, and shown that with as small as 600 dataset records, we are able to achieve the best accuracy.	framework for predicting heart disease, and shown that with as small as 600 dataset records, we are able to achieve 98% accuracy.	supervised datasets at a colossal scale on performance and accuracy, running on high performance clusters.

4	Intellige nt heart disease predictio n system using data mining techniqu es	Sellap a n Pallan ia pan And Rafia h Awan g	200 8	Clevelan d Heart Diseas e databa se	CRISP-DM methodolog y to build the mining models. It consists of six major phases: business understandi ng , data understandi ng , data preparation, modelling, evaluation, and deployment .T echnologies used are Naïve Bayes , Decision Tree and neural networks	Naïve Bayes fared better than Decision Trees as it could identify all the significa nt medical predictor s. The relations hi p between attribute s produce d by Neural Network is more difficult to understa nd .	Training tool to train nurses and medical students to diagnose patients with heart disease.	Limitati on is that it only uses categori ca l data. Another limitati on is that attribute s should be expande d to provide a more compre he nsive diagnos is system. It only uses three data mining techniq ue s
5	Predicting and Diagnos in g of Heart Disease Using Machin e Learnin g Algorith m s	Sanjay Kum ar Sen	2017	UCI Machin e Learni n g Reposit or y	Support vector machine ,K N N, Naïve Base classifier and Decision Tree.	Naïve base classifie r was proved to be the best algorith m of heart disease predicti on.	Implemen t ed Naïve base classifie r , Support Vector Machine , Decisio n Tree and K-Nearest Neighbo ur on a data set of 303, with 83.4%ac ur acyof naïve base classifie r.	Attribute s should be expande d to provide a more compre he nsive diagnos is system.

6	Effective heart disease prediction system using data mining techniques	Poorni ma Singh ,S anjay Singh ,a nd Gayatri S Pan di-Jain	2018	Carried out on a publicly available database for heart disease	Multilayer perceptron neural network (MLPNN) with back propagation (BP) was used as the training algorithm.	BP network calculates the difference between real and predicted values, which is circulated from output nodes backwards to nodes in previous layer. In MLPNN, the input nodes pass values to the first hidden layer, then nodes of first hidden layer pass values to the second and so on producing outputs	Implement ed BP and MLPNN algorithm predicting heart disease, and shown that with as small as 303 dataset records.	Limitation is that it uses only categorica l data.

7	Heart disease predictio n using machine learning techniqu es : a survey	V.V. Ramal in gam*, Ayant an Danda p ath, M Karthi k Raja	2018	Clevelan d database	Support vector machine ,K N N, Naïve Base classifier, Decision Tree and Random Forest	Random Forest was proved to be best classifier than others algorith ms.	Implemen t ation of random forest along with other algorith ms predictin g heart disease, and shown that we are able to achieve 97.7% accuracy	Models based on Naïve Bayes classifie r were comput ati onally very fast A lot of researc h can also be done on the correct ensemb le of algorith m s for a particul ar type of data
8	Heart Disease Predictio n System using Hybrid Techniq ue of Data Mining Algorith m s	Navd ee p Singh Sonik a Jinda l	2018	UCI Machin e Learni n g Reposit or y	Naïve Bayes and genetic algorithm. Fuzzy Rules	GA for Feature Selecti on Cardiova sc ular Disease Classific ati on Using Naive Bayes	The most effective algorith ms of Naïve Bayes and Genetic Algorith m for their perform an ce analysis on the heart disease predictio n. with 97.1% accuracy, for a data set of 303 with 14 attribut es.	Fails to investig at e the impact of large supervi se d datasets

9	Efficient Heart Disease Predicti on System	Purush o ttama,c * , Prof. (Dr.) Kanak Saxen ab , R i c h a S h a r ma c	2016	Clevelan d Clinic Foundat io n	Decision Tree	Decision Trees as it could identify all the significa nt medical predictor s.	This algorith m can help medical practitio ne r in efficient decision making based on the given paramet er. We have train and test the system using 10 fold method and find the accurac y of 86.3 % in testing phase and 87.3 %	Decision tree follows a natural course of events by tracing relation shi ps between events, it may not be possible to plan for all conting en cies that arise from a decision , and such oversig hts can lead to bad decision s.
10	An Analysis of Heart Disease Predicti on using Differen t Data Mining Techniq ue s	Nid hi Bhat la Kira n Jyot i	2012	.	Fuzzy logic; Decision tree; Naive bayes; Classificati on via clustering; Neural networks; Weka tool; Genetic algorithm	Neural network is emulatio n of biologic al neural system, and was found to be best among all other algorith ms	Impleme nt ed Fuzzy logic; Decision tree; Naive bayes; Classific at ion via clusterin g; Neural networks ; Weka tool; Genetic algorithm	Trainin g time of interval type-2 fuzzy logic and genetic hybrid algorit hm s is quite slow.

TITLE	METHODOLOGY	ACCURACY
Heart disease prediction using machine learning techniques : a survey	Support vector machine ,KNN, Naïve Base classifier, Decision Tree and Random Forest	97.7%(random forest)
A Scalable Solution for Heart Disease Prediction using Classification Mining Technique	Random Forest Algorithm and Naïve Bayes	98%(random forest)
Cardiovascular Risk Prediction Method Based on CFS Subset Evaluation and Random Forest Classification Framework	CFS Subset Evaluation and Random Forest Classification Framework	86%(random forest)
Efficient Heart Disease Prediction System	Decision Tree	86.3 % in testing phase and 87.3 % in training phase
Intelligent heart disease prediction system using data mining techniques	Naïve Bayes , Decision Tree and neural networks	86.12(naïve bayes)
Heart Disease prediction using Cleveland dataset and Classification using Random Forest classifier	Random forest classifier , dimensionality reduction using fisher score algorithm, and optimisation using RandomForestCV	98%

3. OVERVIEW OF THE WORK

3.1.- Problem Description

In rural areas, there is a glaring lack of awareness about heart diseases. People ignore the various symptoms they experience until it's too late. The fact that there are inadequate facilities present in remote areas does not help. So, we have come up with a model that aid people in the diagnosis of heart diseases by developing a framework that uses Machine Learning to predict heart disease based on the input of various attributes without going to the doctor.

3.2.Software Requirements

- Anaconda (On a virtual environment) with Python version 3.6 and above.

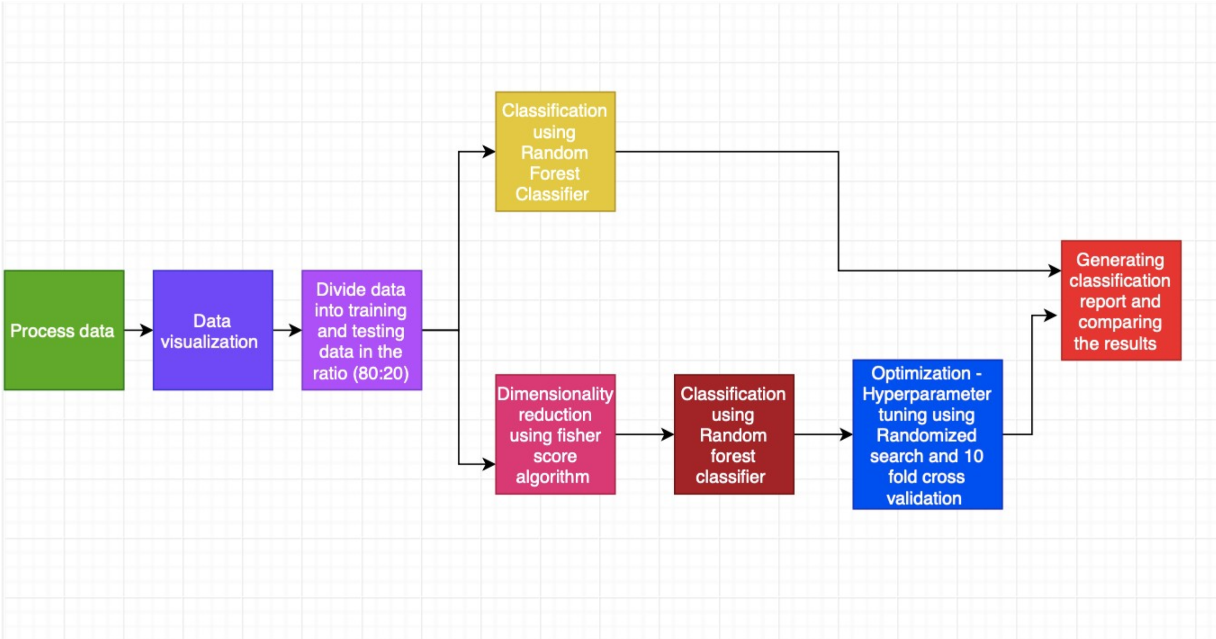
- Libraries for visualisation and implementation of various algorithms.

3.3.Hardware Requirements

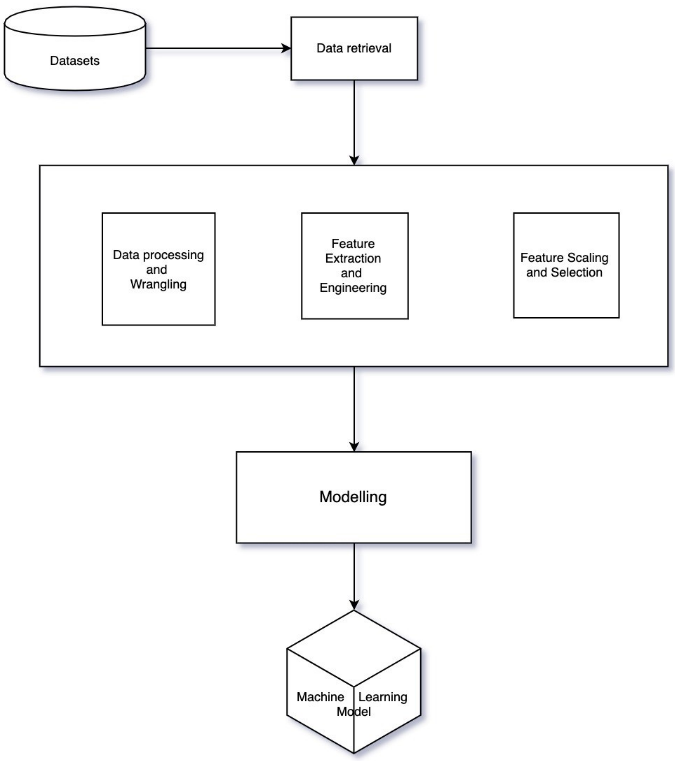
- CPU: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs
- RAM: 32 GB (or 16 GB of 1600 MHz DDR3 RAM)
- Storage: 300 GB. (600 GB for air-gapped deployments)

4. SYSTEM DESIGN

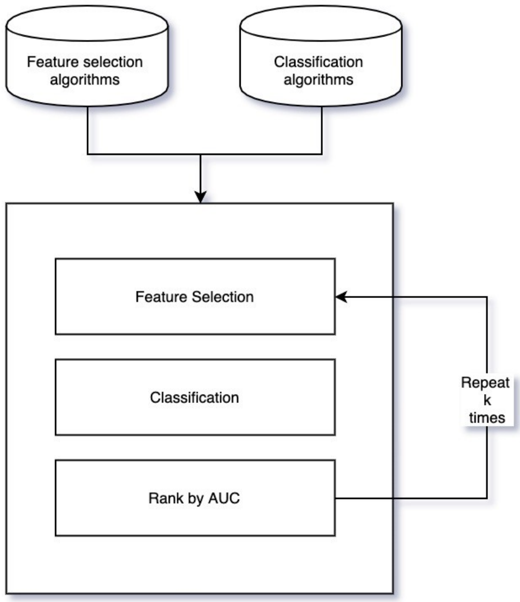
Model design



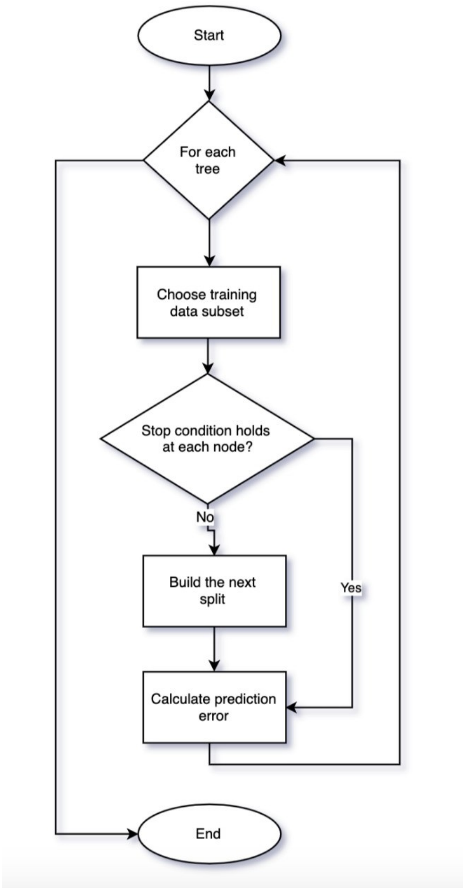
Data Pre-Processing



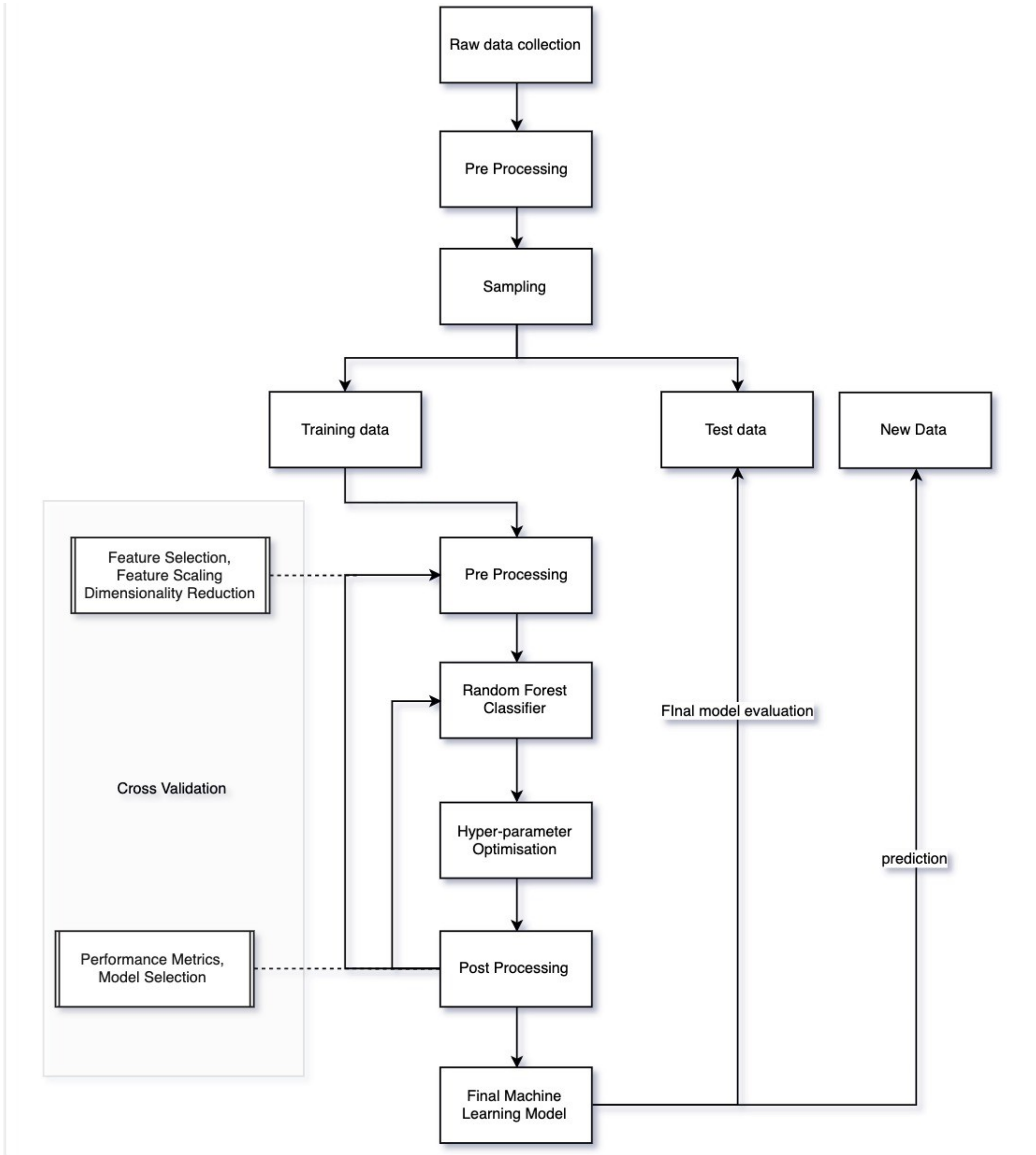
K- Fold Cross Validation



Random Forest Classifier (Working)



System Model



5. IMPLEMENTATION

5.1.Description of Modules/Programs

Dataset used : https://archive.ics.uci.edu/ml/datasets/heart+Disease

1. Data pre-processing : Categorising multi class target variable to binary. Cleaning data by removing all null and missing values and replacing them with the mean of that particular feature . Normalizing the data is done to standardize so that all the input variables have the same treatment in the model and the coefficients of a model are not scaled with respect to the units of the inputs.
2. Training and testing data : Data is split in the ratio 80:20 of which 80 per cent data is used for training purpose and 20 per cent is used for testing.
3. Classifying the pre-processed dataset using Random forest classifier : The data is then classified using Random forest classifier.It creates a set of decision trees from the selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. Various performance measures such as confusion matrix, classification report and accuracy score are printed which is used for comparison later after feature extraction and optimization using Randomized Search with cross validation.
4. Important feature extraction using Fisher Score Algorithm : We use the library chi2 for calculation of fisher score , after which the p values are calculated. According to fisher algorithm , features with smaller p value are more important.
5. Classifying the reduced dataset using random forest classifier to verify increase in accuracy and processing time.
6. We use K-Fold Cross-Validation for performance evaluation where K is any number. The process of K-Fold Cross-Validation is straightforward. You divide the data into K folds. Out of the K folds, K-1 sets are used for training while the remaining set is used for testing. The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.
7. RandomizedSearchCV implements a “fit” and a “score” method. It also implements “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used. The parameters of the estimator used to apply

these methods are optimized by cross-validated search over parameter settings.If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

5.2.Source Code

```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import DataFrame
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer

# Data Pre-Processing
df = pd.read_csv("processed.cleveland.csv", header=None)
df[13] = df[13].replace([1, 2, 3, 4, 5, 6], 1)
df = df[df[12]!='?']
df = df[df[11]!='?']
df.dtypes
X = df.iloc[:,:-1]
y = df.iloc[:,-1:]
scaler = Normalizer().fit(X)
normalized = scaler.transform(X)

# Data Visualization

from matplotlib import rcParams
df.hist(figsize=(18, 16))
rcParams['figure.figsize'] = 8,6
plt.bar(df[13].unique(), df[13].value_counts(), color = ['blue', 'green'])
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class’)

#Splitting data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

# Random Forest classifier without dimensionality reduction

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import make_scorer, accuracy_score
```

```
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn import model_selection

rfc = RandomForestClassifier(n_estimators = 10, oob_score = False)
rfc.fit(X_train,y_train.values.ravel())
rfc_predict = rfc.predict(X_test)

from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix

rfc_cv_score = cross_val_score(rfc, X, y, cv=2)

print("=== All AUC Scores ===")
print(rfc_cv_score)
print('\n')
print("=== Mean AUC Score ===")
print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
print('Accuracy : %.2f' %metrics.accuracy_score(y_test,rfc_predict))

# Dimensionality reduction using Fisher Score Algorithm

fisher_score = chi2(X_train.fillna(0), y_train)
p_values = pd.Series(fisher_score[1])
p_values.index = X_train.columns
p_values.sort_values(ascending=True)


fisher_index = (p_values.sort_values(ascending=True).index)
fisher_index

limit = 6
from pandas import DataFrame
index = []
for i in range(0,limit):
   index.append(fisher_index[i])

X_train_fisher = X_train[index]
X_test_fisher = X_test[index]

# Classification after dimensionality reduction

rfc = RandomForestClassifier(n_estimators = 10, oob_score = True)
rfc.fit(X_train_fisher,y_trainr.values.ravel())
rfc_predict = rfc.predict(X_test_fisher)
```

```
rfc_cv_score = cross_val_score(rfc, X, y, cv=2)
print("=== All AUC Scores ===")
print(rfc_cv_score)
print('\n')
print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
print('Accuracy : %.2f' %metrics.accuracy_score(y_test,rfc_predict))
importances = list(rfc.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(index,
importances)]
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];


# Hyper - Parameter tuning

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {'n_estimators': n_estimators,
            'max_features': max_features,
            'max_depth': max_depth,
            'min_samples_split': min_samples_split,
            'min_samples_leaf': min_samples_leaf,
            'bootstrap': bootstrap}

print(random_grid)


# Randomized Search

f = RandomForestRegressor(random_state = 42)
rf_random = RandomizedSearchCV(estimator=rf, param_distributions=random_grid,
                        n_iter = 100, scoring='neg_mean_absolute_error',
                        cv = 10, verbose=2, random_state=42, n_jobs=-1,
                        return_train_score=True)

rf_random.fit(X_train_fisher, y_train);
rf_random.best_params_
rf_random.cv_results_
```

```
# Classification after hyper- parameter tuning

def evaluate(model, test_features, test_labels):
    predictions = model.predict(test_features)
    errors = abs(predictions - test_labels)

    print('Model Performance')
    print('Average Error: {:0.4f} degrees.'.format(np.mean(errors)))


best_random = rf_random.best_estimator_
evaluate(best_random, X_test_fisher, y_test.values.ravel())

rfc = RandomForestClassifier(n_estimators = 1000,min_samples_split =
2,min_samples_leaf= 1,max_depth= 40,max_features = 'auto', bootstrap = 'false')
rfc.fit(X_train_fisher,y_train.values.ravel())
rfc_predict = rfc.predict(X_test_fisher)
print("=== All AUC Scores ===")
print(rfc_cv_score)
print('\n')
print("=== Mean AUC Score ===")
print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
print('Accuracy : %.2f' %metrics.accuracy_score(y_test,rfc_predict))
```

5.4 Execution Snapshots

#Dataset head, data types

```
In [2]: df.head()
Out[2]:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	1
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0

```
In [3]: df.dtypes
Out[3]: 0      float64
        1      float64
        2      float64
        3      float64
        4      float64
        5      float64
        6      float64
        7      float64
        8      float64
        9      float64
        10     float64
        11      object
        12      object
        13       int64
        dtype: object
```

Parameter Information

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
0     297 non-null float64
1     297 non-null float64
2     297 non-null float64
3     297 non-null float64
4     297 non-null float64
5     297 non-null float64
6     297 non-null float64
7     297 non-null float64
8     297 non-null float64
9     297 non-null float64
10    297 non-null float64
11    297 non-null object
12    297 non-null object
13    297 non-null int64
dtypes: float64(11), int64(1), object(2)
memory usage: 34.8+ KB
```

Dataset Description

In [6]:
```
df.describe()
```

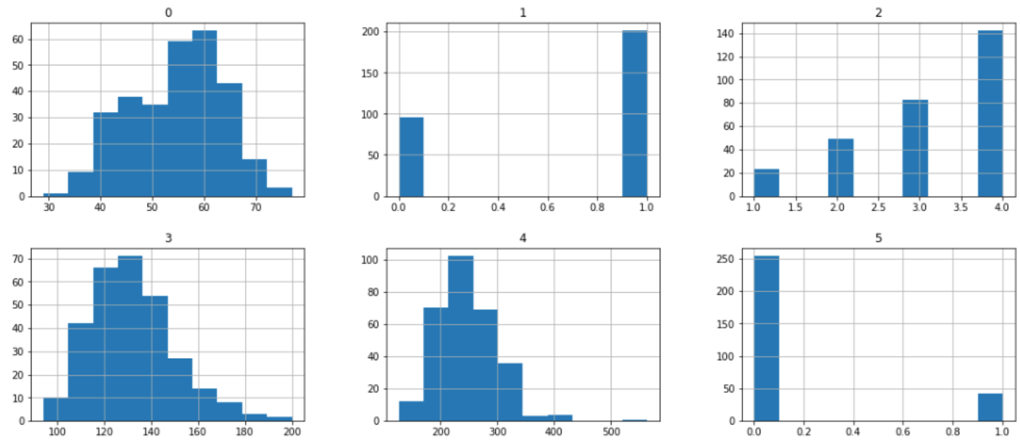
Out[6]:

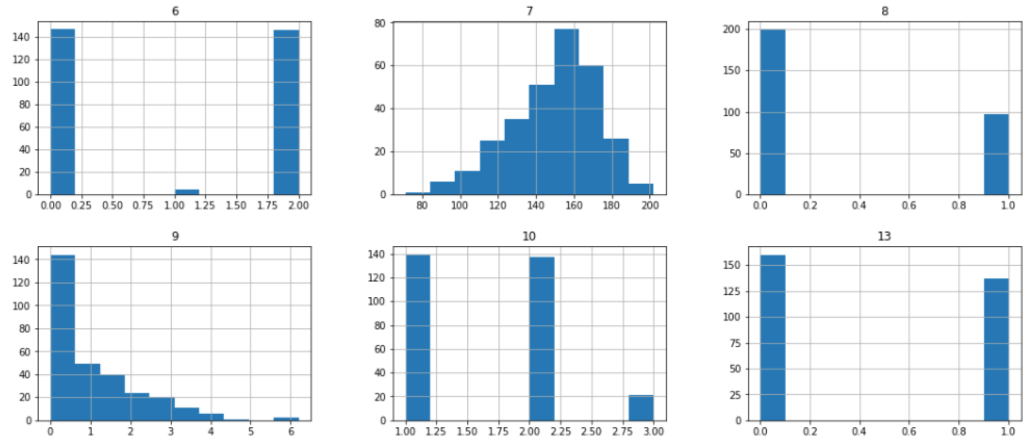
	0	1	2	3	4	5	6	7	8	9	10	13
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000
mean	54.542088	0.676768	3.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	1.602694	0.461279
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187	0.499340
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000
50%	56.000000	1.000000	3.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	1.000000

Data Visualization

In [7]:
```
df.hist(figsize=(18, 16))
```

Out[7]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x10528de48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17ea4d68>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17ecdfd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a17eff278>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17f284e0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17f51710>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a17f79978>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17fa1c18>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17fa1c50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a17ffd0f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a18025358>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a180505c0>]],
      dtype=object)
```

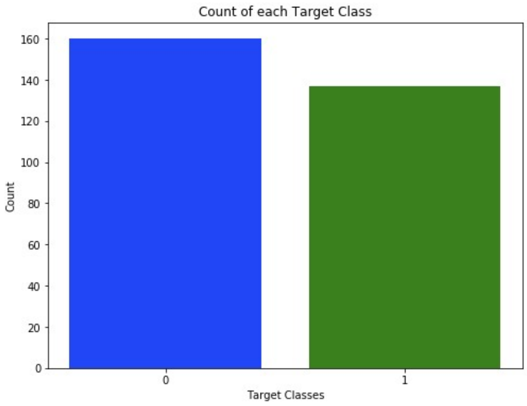




Target class Histogram

```
In [9]: from matplotlib import rcParams
        rcParams['figure.figsize'] = 8,6
        plt.bar(df[13].unique(), df[13].value_counts(), color = ['blue', 'green'])
        plt.xticks([0, 1])
        plt.xlabel('Target Classes')
        plt.ylabel('Count')
        plt.title('Count of each Target Class')

Out[9]: Text(0.5, 1.0, 'Count of each Target Class')
```



Data Pre-Processing

```
In [4]: normalized

Out[4]: array([[0.19741527, 0.00313358, 0.00313358, ..., 0.00940073, 0.        ,
         0.01880145],
        [0.1905785 , 0.00284446, 0.01137782, ..., 0.00568891, 0.00853337,
         0.00853337],
        [0.22578415, 0.00336991, 0.01347965, ..., 0.00673983, 0.00673983,
         0.02358939],
        ...,
        [0.23663584, 0.00347994, 0.01391976, ..., 0.00695988, 0.00695988,
         0.02435957],
        [0.25337478, 0.00444517, 0.01778069, ..., 0.00889034, 0.00444517,
         0.0311162 ],
        [0.17495614, 0.        , 0.00613881, ..., 0.00613881, 0.00306941,
         0.00920822]])
```

Splitting data into training and testing

255	42.0	0.0	3.0	120.0	209.0	0.0	0.0	173.0	0.0	0.0	2.0	0.0	3.0
21	58.0	0.0	1.0	150.0	283.0	1.0	2.0	162.0	0.0	1.0	1.0	0.0	3.0
161	77.0	1.0	4.0	125.0	304.0	0.0	2.0	162.0	1.0	0.0	1.0	3.0	3.0
280	57.0	1.0	4.0	110.0	335.0	0.0	0.0	143.0	1.0	3.0	2.0	1.0	7.0
194	68.0	0.0	3.0	120.0	211.0	0.0	2.0	115.0	0.0	1.5	2.0	0.0	3.0
260	44.0	0.0	3.0	118.0	242.0	0.0	0.0	149.0	0.0	0.3	2.0	1.0	3.0
150	52.0	1.0	1.0	152.0	298.0	1.0	0.0	178.0	0.0	1.2	2.0	0.0	7.0
131	51.0	1.0	3.0	94.0	227.0	0.0	0.0	154.0	1.0	0.0	1.0	1.0	7.0
152	67.0	0.0	3.0	115.0	564.0	0.0	2.0	160.0	0.0	1.6	2.0	0.0	7.0
100	45.0	1.0	4.0	115.0	260.0	0.0	2.0	185.0	0.0	0.0	1.0	0.0	3.0
88	53.0	0.0	4.0	138.0	234.0	0.0	2.0	160.0	0.0	0.0	1.0	0.0	3.0
217	46.0	0.0	4.0	138.0	243.0	0.0	2.0	152.0	1.0	0.0	2.0	0.0	3.0
122	51.0	1.0	3.0	100.0	222.0	0.0	0.0	143.0	1.0	1.2	2.0	0.0	3.0
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0
20	64.0	1.0	1.0	110.0	211.0	0.0	2.0	144.0	1.0	1.8	2.0	0.0	3.0
190	50.0	1.0	3.0	129.0	196.0	0.0	0.0	163.0	0.0	0.0	1.0	0.0	3.0
71	67.0	1.0	4.0	125.0	254.0	1.0	0.0	163.0	0.0	0.2	2.0	2.0	7.0
107	57.0	1.0	3.0	128.0	229.0	0.0	2.0	150.0	0.0	0.4	2.0	1.0	7.0
274	59.0	1.0	1.0	134.0	204.0	0.0	0.0	162.0	0.0	0.8	1.0	2.0	3.0
103	71.0	0.0	3.0	110.0	265.0	1.0	2.0	130.0	0.0	0.0	1.0	1.0	3.0

237 rows × 13 columns

Random forest classifier without dimensionality reduction

```
=== Confusion Matrix ===
[[30  6]
 [ 2 22]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.94      0.83      0.88        36
           1       0.79      0.92      0.85        24

   micro avg       0.87      0.87      0.87        60
   macro avg       0.86      0.88      0.86        60
weighted avg       0.88      0.87      0.87        60


=== All AUC Scores ===
[0.83333333 0.73333333 0.86666667 0.86666667 0.83333333 0.7
 0.66666667 0.86206897 0.72413793 0.79310345]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.7879310344827586
Accuracy : 0.87
```

Random forest classifier with dimensionality reduction (after apply fisher score)

```
=== Confusion Matrix ===
[[30  6]
 [ 4 20]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.88      0.83      0.86        36
           1       0.77      0.83      0.80        24

   micro avg       0.83      0.83      0.83        60
   macro avg       0.83      0.83      0.83        60
weighted avg       0.84      0.83      0.83        60


=== All AUC Scores ===
[0.86666667 0.83333333 0.93333333 0.96666667 0.86666667 0.73333333
 0.7        0.75862069 0.75862069 0.75862069]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.8175862068965518
Accuracy : 0.97
```

 # Random forest classifier with hyper parameter tuning on the reduced data

```
Model Performance
Average Error: 0.2862 degrees.
=== All AUC Scores ===
[0.86666667 0.83333333 0.93333333 0.96666667 0.86666667 0.73333333
 0.7        0.75862069 0.75862069 0.75862069]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.8175862068965518
Accuracy : 0.98
```

 # Chosen set of hyper parameters

```
In [23]: rf_random.best_params_

Out[23]: {'n_estimators': 200,
          'min_samples_split': 2,
          'min_samples_leaf': 1,
          'max_features': 'sqrt',
          'max_depth': None,
          'bootstrap': False}
```

6. CONCLUSION AND FUTURE DIRECTIONS

Based on the results shown above and experiments performed, it is evident that input data plays an important role in prediction along with machine learning techniques. As is seen in the dataset, provided, we have labels from 0 to 4 where the labels of 4 are hardly 13 and when we split the data into train and test, the number become very less which is nothing but noise and can be totally removed from the dataset by using filtering techniques and hence the linear model will be available to predict the outcome much better with absence of noise. Moreover, Fisher Score has again proven that we can get rid of similar feature set and still obtain predictions with great efficiency. Random forest classifiers are used along with a randomised search with cross validation to optimise the working of the classifier and provide results with greater accuracy and precision. Most importantly, the above experiment not only helped us in predicting the outcome but also gave us valuable insights about the nature of data, which can be used in future to train our classifiers in a much better way.

Random Forest classifier is an ensemble technique which is comparable in performance with bagging and boosting and used especially for high dimensional data. Reducing number of trees in Random Forest will enhance the performance both for learning and classification. The intention of this paper was to present survey of pruning efforts for Random Forest along with the necessary theoretical foundations. There is comparatively less work done in this area, specifically for dynamic pruning of Random Forest.

7. REFERENCES

1. Xu, S., Zhang, Z., Wang, D., Hu, J., Duan, X., & Zhu, T. (2017) “Cardiovascular risk prediction method based on CFS subset evaluation and random forest classification framework”

2. Long, N. C., Meesad, P., & Unger, H. (2015) “A highly accurate firefly based algorithm for heart disease prediction.”

3. Saboji, R. G. (2017) “A scalable solution for heart disease prediction using classification mining technique”

4. Palaniappan, S., & Awang, R. (2008) “Intelligent heart disease prediction system using data mining techniques.”

5. V. Ramalingam, V., Dandapath, A., & Karthik Raja, M. (2018) “Heart disease prediction using machine learning techniques : a survey.”
6. Andy Liaw and Matthew Wiener “Classification and Regression by randomForest”

7. Quanquan Gu, Zhenhui Li & Jiawei Han “Generalized Fisher Score for Feature Selection”