Contents

Figures

Equations

Tables

Abstract

Cardiovascular disease is a general term for conditions affecting the heart or blood vessels. It is one of the most fatal problems in the whole world. Cardiovascular disease diagnosis is a complex task which requires much experience and knowledge. The traditional way of predicting it is a doctor's examination or several medical tests such as ECG, Stress Test, and Heart MRI etc. Nowadays, the Healthcare industry contains a huge amount of healthcare data, which contains hidden information. This hidden information is useful for making effective decisions. There are various data mining and machine learning techniques and tools available to extract effective knowledge from databases and to use this knowledge for more accurate diagnosis and decision making. The main objective of this project is to summarize various machine learning techniques with comparative results that have been done on CVD prediction and also make analytical conclusions.

1. Introduction

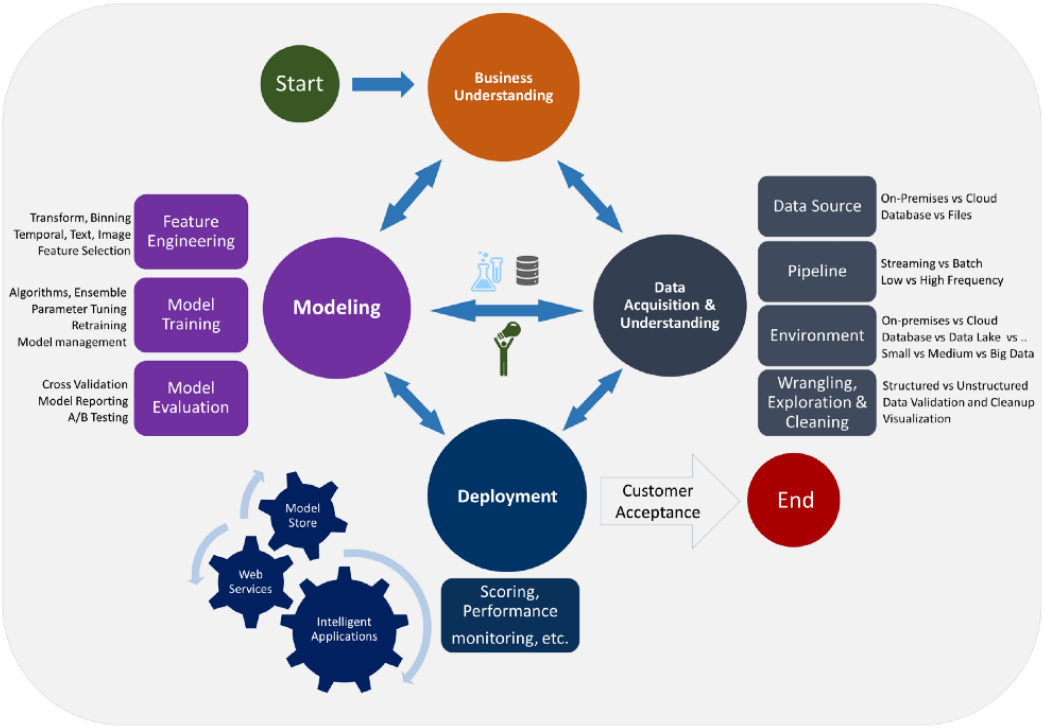
1.1 Problem Statement

People with cardiovascular disease or who are at high cardiovascular risk need early detection. A heart attack does not always have obvious symptoms, such as pain in your chest, shortness of breath and cold sweats. A heart attack can happen without a person knowing it. It is called a silent heart attack, or medically referred to as silent ischemia (lack of oxygen) to the heart muscle. Therefore, many suffer from it without prior knowledge and precautions.

1.2 Solution Statement

The overall objective of the system will be to predict with few tests and attributes the likelihood of the presence of cardiovascular disease to help the patients get special attention. Attributes considered form the primary basis for tests and give accurate results more or less. This prediction can be used to analyze data of a certain system e.g.hospital to improve the precautions taken against the CVD. Machine learning provides computer programs the ability to learn from predetermined data and improve performance from experiences without human intervention and then apply what has been learned to make an informed decision. The system can classify the patients based on risk level into two classes as Low and High.

1.3 Data Science Lifecycle



[1]

Figure.1 Data Science Lifecycle

Business Understanding: Before anyone can even start on a data science project, it is critical to understand the problems that need to be solved.

Data Exploration: is the initial step in data analysis, where users explore a large data set in an unstructured way to uncover initial patterns, characteristics, and points of interest. This process isn't meant to reveal every bit of information a dataset holds, but rather to help create a broad picture of important trends and major points to study in greater detail.

Data Cleaning: Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results.

Predictive Modelling: is where machine learning finally comes into data science projects. This phase trains the model and assesses the accuracy, but also uses comprehensive statistical methods and tests to ensure that the outcomes from the model make sense and are significant. Based on the questions asked in the business understanding stage.

2. Background

2.1 Classification

Classification is a process of categorizing a given set of data into classes. It is a supervised machine learning technique. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories. [2]

2.2 Random Forest

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. [3]

2.3 K-Nearest-Neighbor

(KNN) algorithm is an easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. KNN algorithm stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). [4]

Steps:

1. Initialize K to your chosen number of neighbors
2. For each example in the data
   2.1. Calculate the distance between the query example and the current example from the data by Euclidean distance:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Equation.1 Euclidean Distance

   2.2. Add the distance and the index of the example to an ordered collection
3. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
4. Pick the first K entries from the sorted collection.
5. Get the labels of the selected K entries and return the mode of the K labels.

2.4 Decision Trees

Decision Trees are constructed via an algorithmic approach that identifies ways to split a dataset based on different conditions. It is one of the most widely used for supervised learning for both classification and regression tasks. [5]

Steps:

1. Create the root node.
2. Generate a list of all questions which needs to be asked at that node.
3. Partition rows into subsets based on each question asked.
4. Calculate Gini impurity and partition of data from the previous step.

$$\sum_{k \neq i} p_k = 1 - p_i$$

Equation.2 Gini Impurity

5. Update best question based on Gini impurity.
6. Divide the node on the best question. Repeat from step 1 until we get a pure node (leaf nodes).[32]

2.5 Multi-Layer Perceptron Neural Network

The MLPNN involves an input layer, hidden layers and output layer. Each layer consists of several neurons. Connections between all neurons in the next layer represent the weights (*w*). The weights represent the strength of the connection between neurons. It consists of a feedforward algorithm and backpropagation algorithm. [6]

2.5.1 MLPNN Feedforward Algorithm

Each input value is submitted to the input layer in a vector form. Each neuron obtains inputs from other neurons, achieves a summation of weights, applies an activation function to the weighted sum:

$$Y = Activation(\sum(weight * input) + bias)$$

Equation.3 Activation Function

and generates results to other neurons in the network. The outputs of neurons in the output layer represent the output of the network. In classification cases, we usually use the sigmoid function in calculating the output [7]

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Equation.4 Sigmoid Function

2.5.2 Backpropagation Algorithm

Backpropagation is a short for the backward propagation of errors, since the error is computed at the output and distributed backwards throughout the network's layers. [8]

$$Error = output(1 - output)(target - output)$$

Equation.5 Error between the target and the output

$$Error = output(1 - output)\sum_{k} Error_k * W_{jk}$$

Equation.6 Error between the output and the hidden neurons

When we train neural networks, we optimize the weights and bias. Learning rate determines how quickly or how slowly the model will update the weights and bias. Learning rate should be high enough so that it won't take ages to converge, and it should be low enough so that it finds the local minima.

$$\Delta w_{ij} = l * Error_j * output_i$$

$$w_{ij=} w_{ij} + \Delta w_{ij}$$

Equation.7 Updating weight

$$\Delta \theta_j = l * Error_j$$

$$\theta_{j=} \theta_j + \Delta \theta_j$$

Equation.8 Updating bias

2.5.3 Weight VS Bias

Weights and biases are both learnable parameters inside the network. As training continues, both parameters are adjusted toward the desired values and the correct output. Simply, bias represents how far off the predictions are from their intended value. Biases make up the difference between the function's output and its intended output. Weights, on the other hand, affect the amount of influence a change in the input will have upon the output.

$$Output = \sum (weights * inputs) + bias$$

Equation.9 Calculating output by weight and bias

2.6 Cross-Validation

For a prediction problem, a model is generally provided with training and testing data sets. A round of cross-validation comprises the partitioning of data into complementary subsets, then performing analysis on one subset. After this, the analysis is validated on other subsets (testing sets). [9]

[10]

Figure.2 Cross-Validation

2.7 Model fitting

The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen.
There is a terminology used in machine learning when we talk about how well a machine-learning model learns and generalizes to new data, namely **overfitting** and **underfitting.**
Overfitting and underfitting are the two biggest causes for poor performance of machine-learning algorithms.

2.7.1 Overfitting in ML

Overfitting refers to a model that models the training data too well. it happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

2.7.2 Underfitting in ML

Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

2.8 One-Hot Encoding

Considered as one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assigns **0** or **1** to them.

2.9 Imputation Method

It fills in the missing values with a number. For instance, we can fill in the mean value along each column.

3. Medical Data Understanding

3.1 Heart Disease Dataset Description

Data preview

The dataset is publically available on the Kaggle website, and it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD). The dataset provides the patients' information. It includes over 4,000 records and 15 attributes. [11]

Variables

Each attribute is a potential risk factor. There are both demographic, behavioural and medical risk factors.

Table.1 CVD Dataset Attributes

Attribute. No	Attribute Name	Description
1	Sex	1: Male 0: Female
2	Age	Continuous
3	Education	1: No education 2: Low education 3: Medium education 4: High education
4	Current Smoker	1: Smoker 0: Not smoker
5	Cigs Per Day	Continuous

6	BP Meds	1: On blood pressure medication 0: Not on blood pressure medication
7	Prevalent Stroke	1: The patient had previously had a stroke. 0: The patient had not previously had a stroke.
8	Prevalent Hyp	1: The patient was hypertensive. 0: The patient was not hypertensive.
9	Diabetes	1: The patient had diabetes. 0: The patient didn't have diabetes.
10	Tot Chol	Total cholesterol level (Continuous)
11	Sys BP	Systolic blood pressure (Continuous)
12	Dia BP	Diastolic blood pressure (Continuous)
13	BMI	Body Mass Index (Continuous)
14	Heart Rate	Continuous
15	Glucose	Glucose level (Continuous)
16 (Target)	TenYearCHD	1: "Yes" 0: "No"

3.2 Displaying Data Types

The first step in getting to know the data is to discover the different data types it contains by info() function. It shows how many non-null values a column contains. Null values often indicate a problem in the data-gathering process. They can make several analysis techniques, like different types of machine learning difficult or even impossible.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4238 non-null   int64
 1   age              4238 non-null   int64
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64
 7   prevalentHyp     4238 non-null   int64
 8   diabetes         4238 non-null   int64
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

Figure.3 Displaying data types

We can observe that we have only numerical attributes.

3.3 Showing Basics Statistics

Now that we've seen what data types are in our dataset, it's time to get an overview of the values each column contains. The describe function in python is a function that allows analysis between the numerical values contained in the dataset. Using this function count, mean, std, min, max, 25%, 50%, 75%.

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	di
count	4238.000000	4238.000000	4133.000000	4238.000000	4209.000000	4185.000000	4238.000000	4238.000000	4238.000000	4188.000000	4238.000000	4238.000
mean	0.429212	49.584946	1.978950	0.494101	9.003089	0.029630	0.005899	0.310524	0.025720	236.721585	132.352407	82.893
std	0.495022	8.572160	1.019791	0.500024	11.920094	0.169584	0.076587	0.462763	0.158316	44.590334	22.038097	11.910
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.000000	83.500000	48.000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000	75.000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000	82.000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	263.000000	144.000000	89.875
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000	295.000000	142.500

Figure.4 Displaying basic statistics

As seen in the table above, for example in the total cholesterol column we noticed that the maximum value is not acceptable so it may have been entered by mistake. So this can tell us that this column needs to be cleaned.

3.4 Discovering Missing Values

It is very important to check if the data contains missing values in some columns. After calling isnull().sum() function. We've found 645 missing values in the data.

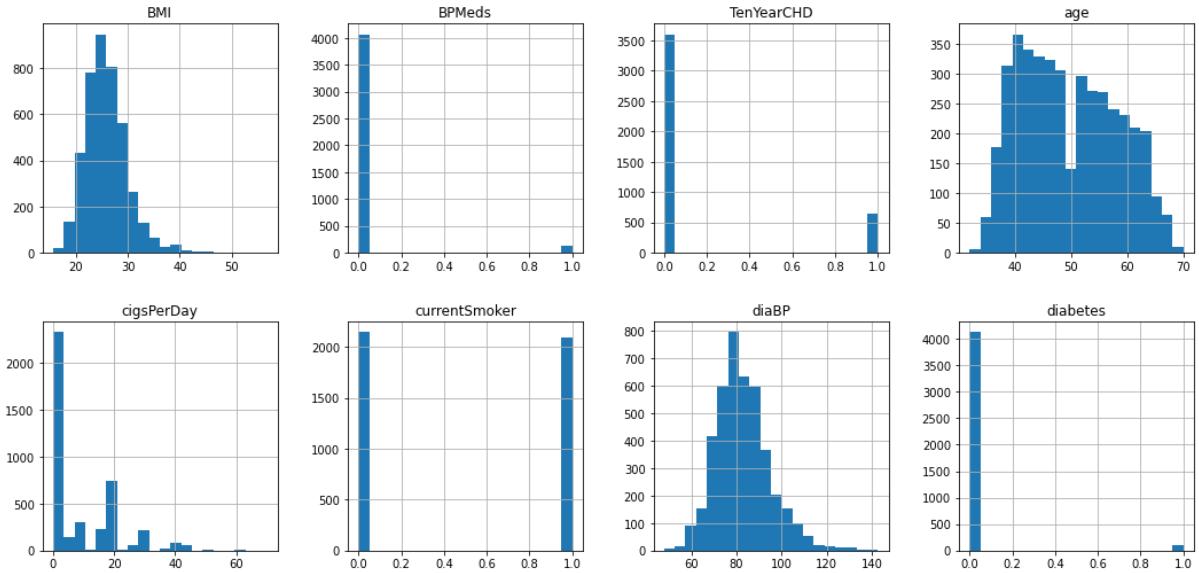
```
[ ]  heart_data.isnull().sum()
```

```
male                0
age                 0
education         105
currentSmoker       0
cigsPerDay         29
BPMeds             53
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            50
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           388
TenYearCHD          0
dtype: int64
```

Figure.5 Displaying the missing values

3.5 Exploring Our Dataset

We started by observing the distribution of all attributes by histograms.



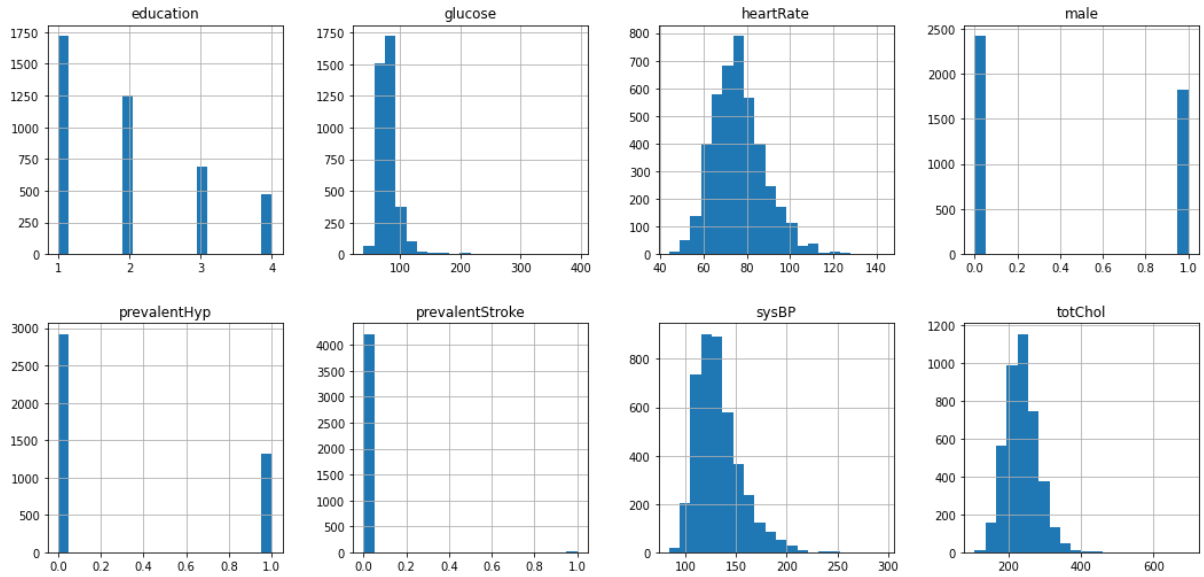


Figure.6 Displaying the distribution of our dataset attributes

It is easy to pick out the discrete and continuous attributes. Also, it can be seen that none of the respondents has a prevalent stroke, just a few of them have diabetes and also a few of them have a blood pressure medication.
Class 1 at the target attribute is much less than the class 0 so our dataset is unbalanced.
The next step is to detect outliers by box and whisker plot for continuous features that their ranges in the basic statistics don't make sense which are heart rate, systolic blood pressure and total cholesterol.

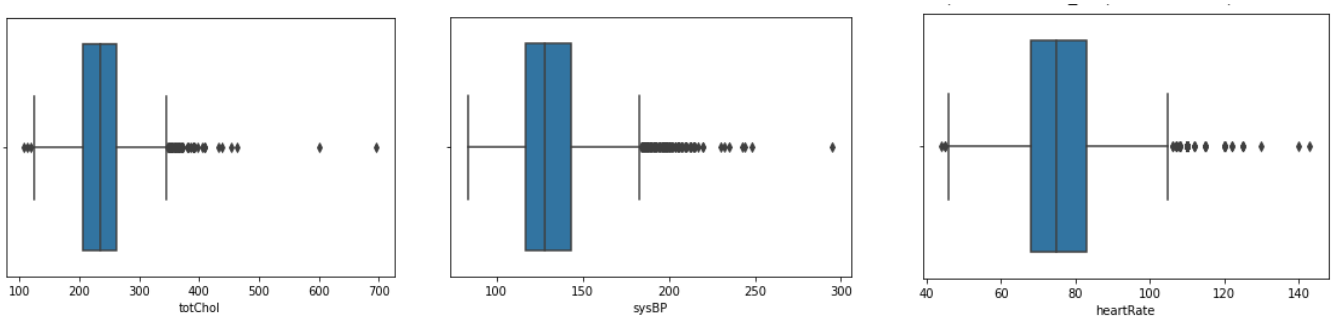


Figure.7 Box and Whisker plots of totChol, sysBP and heartRate

Many outliers have been observed at total cholesterol, sysBP and heart rate attributed.

4. Medical Data Wrangling

4.1 Missing Values Handling

We have used the Imputation method to fill in the missing values with the median value. The imputed value won't be exactly right in most cases, but it usually leads to more accurate models than you would get from dropping the column entirely or dropping the rows that contain missing values

.

```
imputed_data.isnull().sum()

male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

Figure.8 Displaying missing values after imputation

4.2 Outliers Removing

We know outliers can either be a mistake or just variance, but how to decide if they are important or not. It is pretty simple if they are the result of a mistake, then we can ignore them, but if it is just a variance in the data we would need to think a bit further like what is the abnormal level of the feature.

After removing outliers:

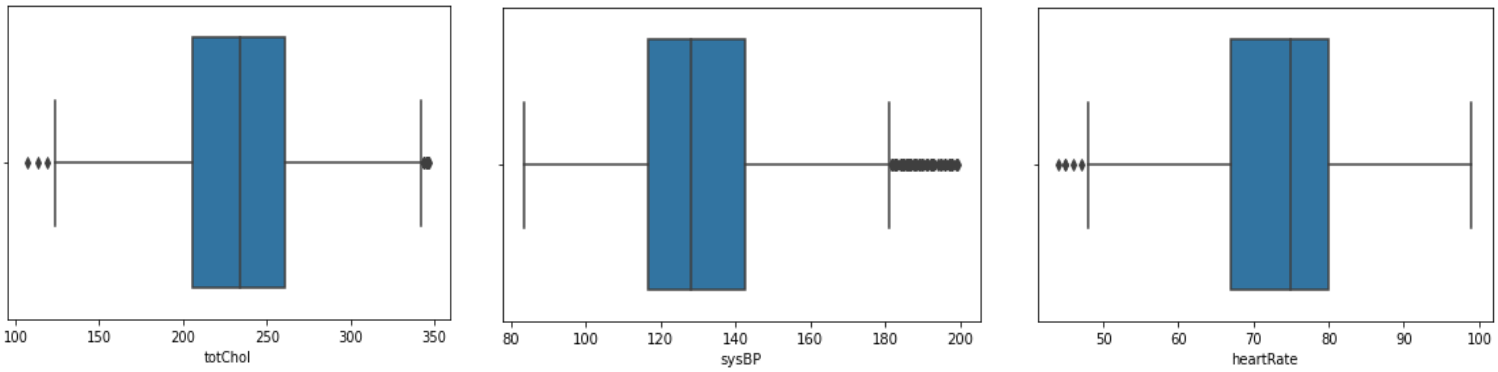


Figure.9 Box and Whisker plots after removing outliers

We checked for the distribution of the ages of the people who had CHD and the number of the sick generally increased with age with the peak at 63 years old.

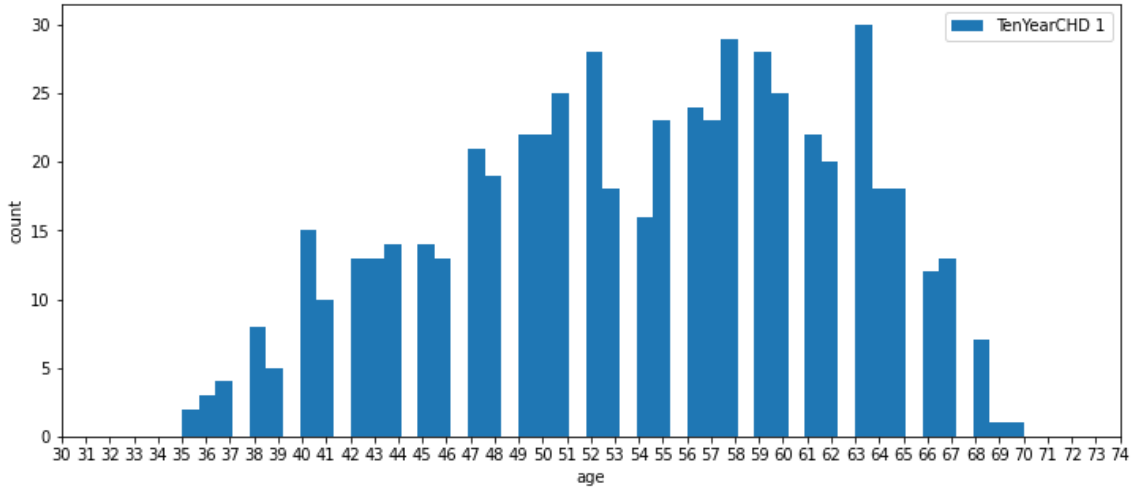


Figure.10 Distribution of age for the respondents who have a high risk

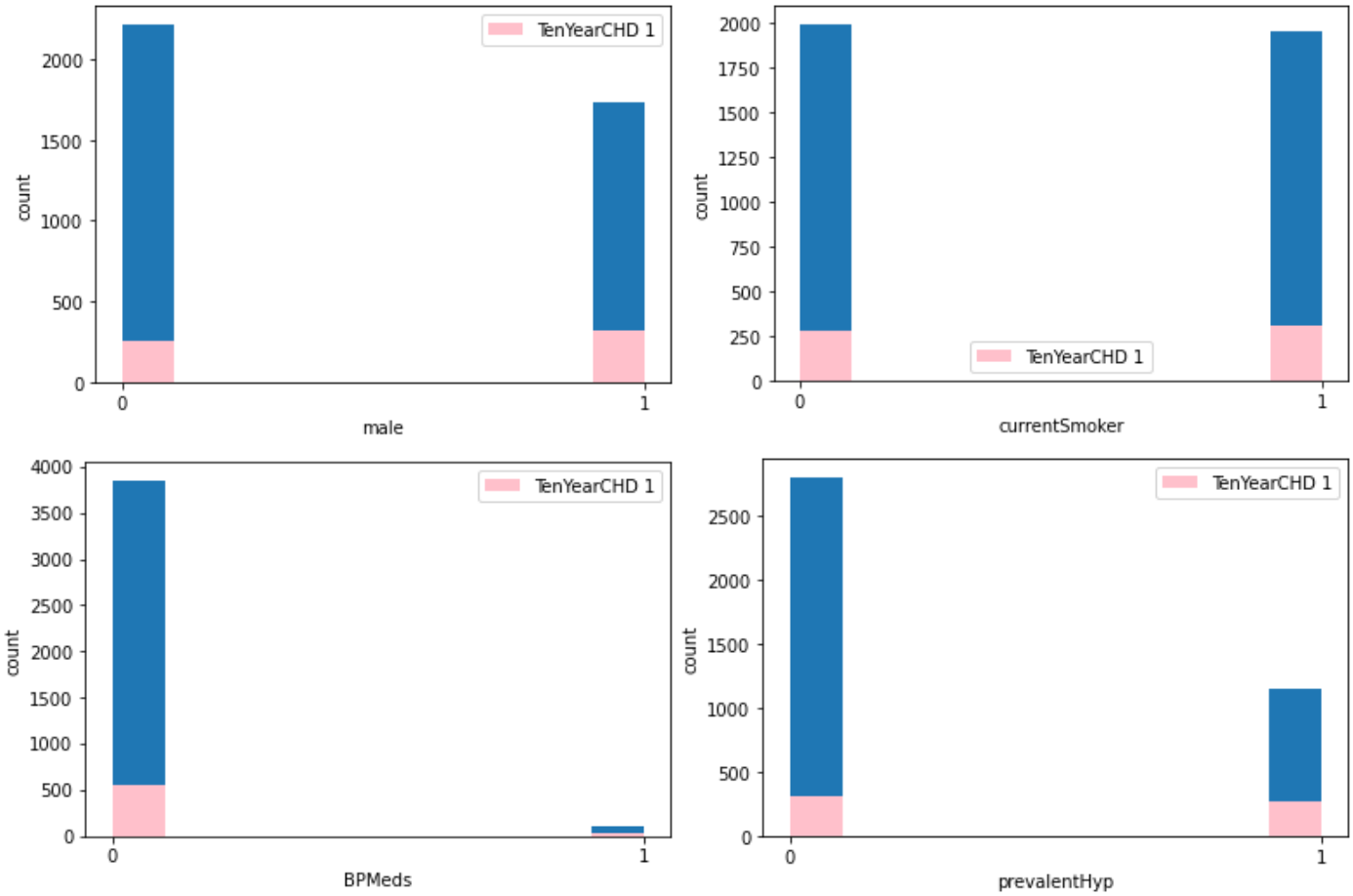


Figure.11 Categorical variables of having and not having risk

From the above results there are the conclusions:

- Males are suffering more than females.
- Current smoker attribute is not efficient in that the smoking count is almost equal to non-smoking count.
- The percentage of people who have CHD is higher with prevalent hypertension as compared to those who don't.

The final step is to check the correlation between the target and the features and between the features themselves.

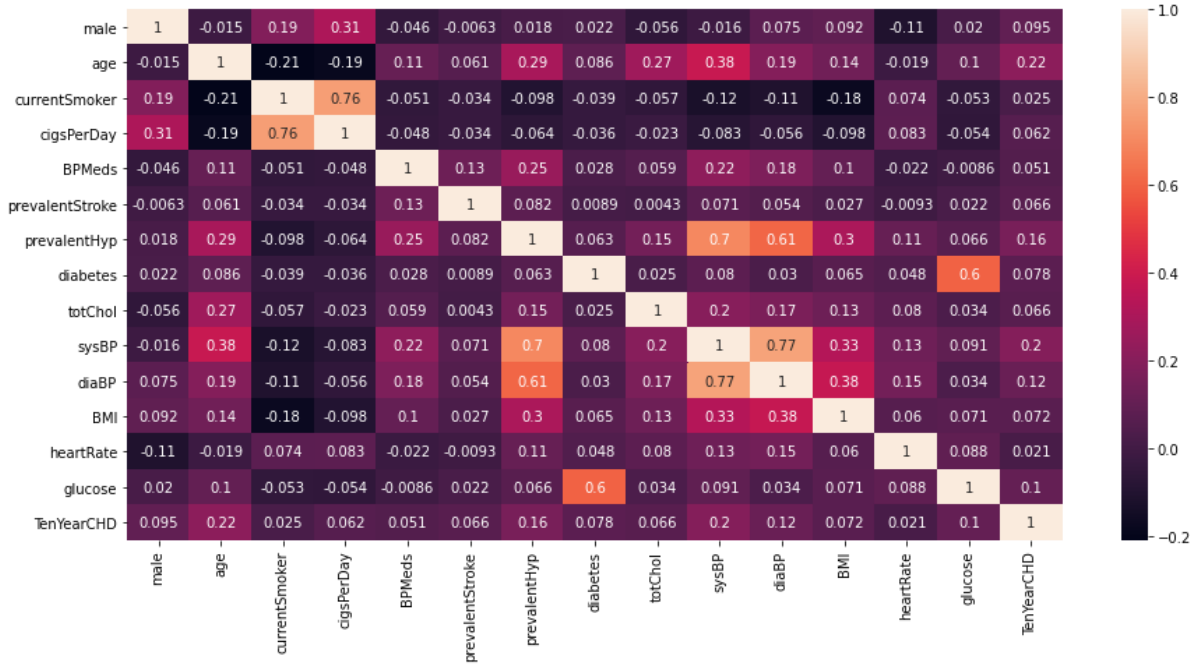


Figure.12 Correlation matrix by heatmap

We can see that there is no strong relationship between the target and any of the features (< 0.5). Which indicates that they are poor predictors. The highest correlation between are age, prevalent hypertensive, systolic BP and diastolic BP.

There is a high correlation between some features like *('currentSmoker', 'cigsPerDay'), ('prevalentHyp, sysBP'), ('prevalentHyp', 'diaBP') and ('diabetes', 'glucose')*.

5. Proposed Machine Learning Classification Algorithms for Our Medical Data

5.1 Feature Engineering On Medical Data

To Prepare the proper input dataset for, compatible with the machine learning algorithm requirements and Improving the performance of machine learning models; we used some techniques in feature engineering before building KNN, decision tree and random forest models.

5.1.1 Create Interaction Features

The first step is checking we have any categorical features that can create interactive features. In our dataset we have 5 which are *['male', 'prevalentStroke'*

,*'prevalentHyp', 'diabetes', 'BPMeds']*. We iterated through each pair of 2 features and combined them into interaction features.

Table.2 Sample of the features after feature generation

male_prevalentStroke	male_prevalentHyp	male_diabetes	male_BPMeds
1.0_0.0	1.0_0.0	1.0_0.0	1.0_0.0

5.1.2 Categorical Encoding

We have used the One-Hot *encoding* method. This is a sample of our dataset after encoding.

Table.3 Sample of the features after one-hot encoding

male_BPMeds_0.0_0.0	male_BPMeds_0.0_1.0	male_BPMeds_1.0_0.0	male_BPMeds_1.0_1.0
0	0	1	0
0	1	0	0
0	1	0	0

5.1.3 Feature Selection

Not all of the features you engineer should be important. Many of them don't improve the model's performance. These are some of the most important features and are ranked due to their importance.

Table.4 Features ranking due to the importance

columns	importance
male_prevalentHyp_1.0_1.0	0.148849
age	0.116728

male_prevalentHyp_0.0_0.0	0.113687
male_BPMeds_0.0_0.0	0.075771
male_prevalentStroke_0.0_0.0	0.075297
male_diabetes_1.0_0.0	0.063696
male_diabetes_0.0_0.0	0.062012
sysBP	0.058977
cigsPerDay	0.044662
glucose	0.042122
BMI	0.034154
totChol	0.030434
prevalentHyp_diabetes_1.0_0.0	0.030259
diaBP	0.030183
heartRate	0.026041
male_prevalentHyp_1.0_0.0	0.024215
prevalentHyp_BPMeds_1.0_0.0	0.021978

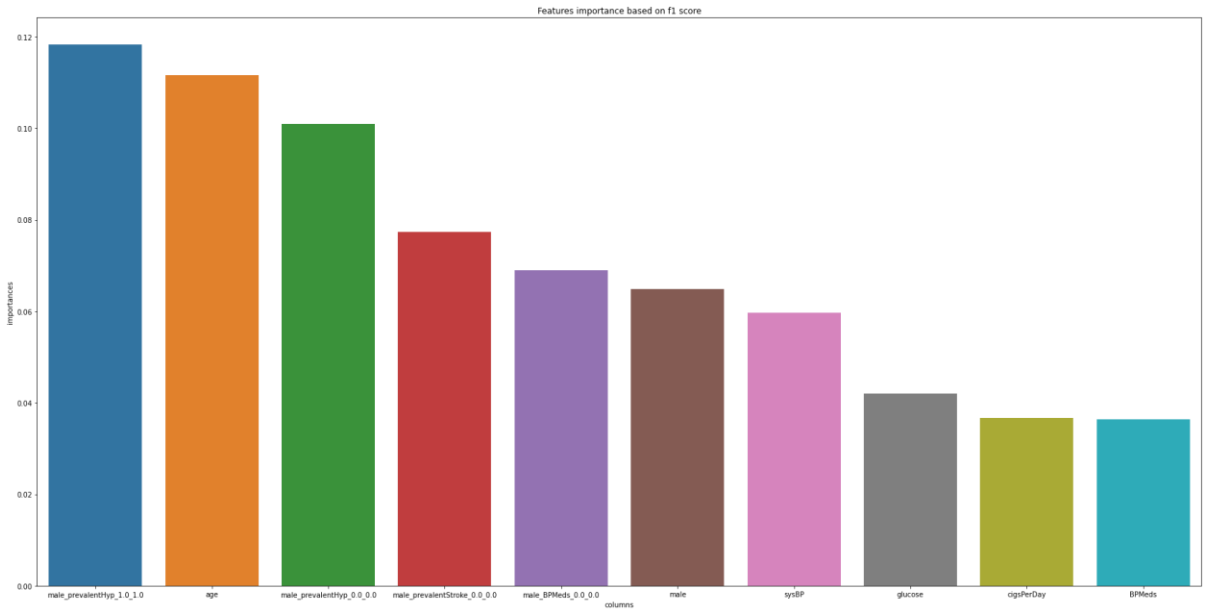


Figure.13 Features importance based on f1 score

5.2 Proposed Classical Learning Algorithms

5.2.1 K-Nearest-Neighbor

There are no predefined statistical methods to find the best value of K. So we Initialized a range for K value from 1 to 15 and started computing. Then we derived a plot between error rate and K values in the defined range.

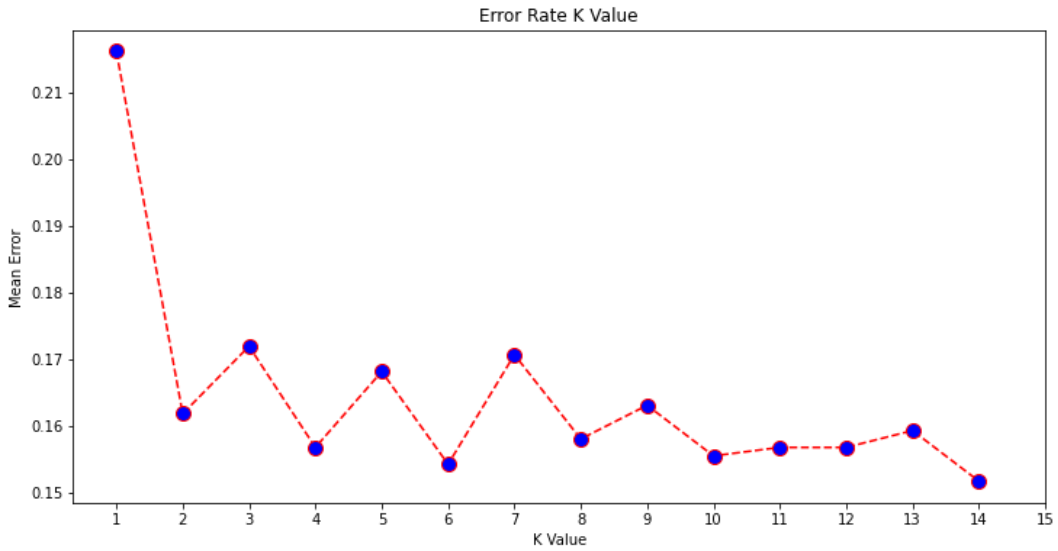


Figure.14 Error rate of different K values

We chose the K value as having the minimum error rate which is 4.
After executing the model, we've found that the accuracy is 85%.

Then we plotted the confusion matrix as shown below.

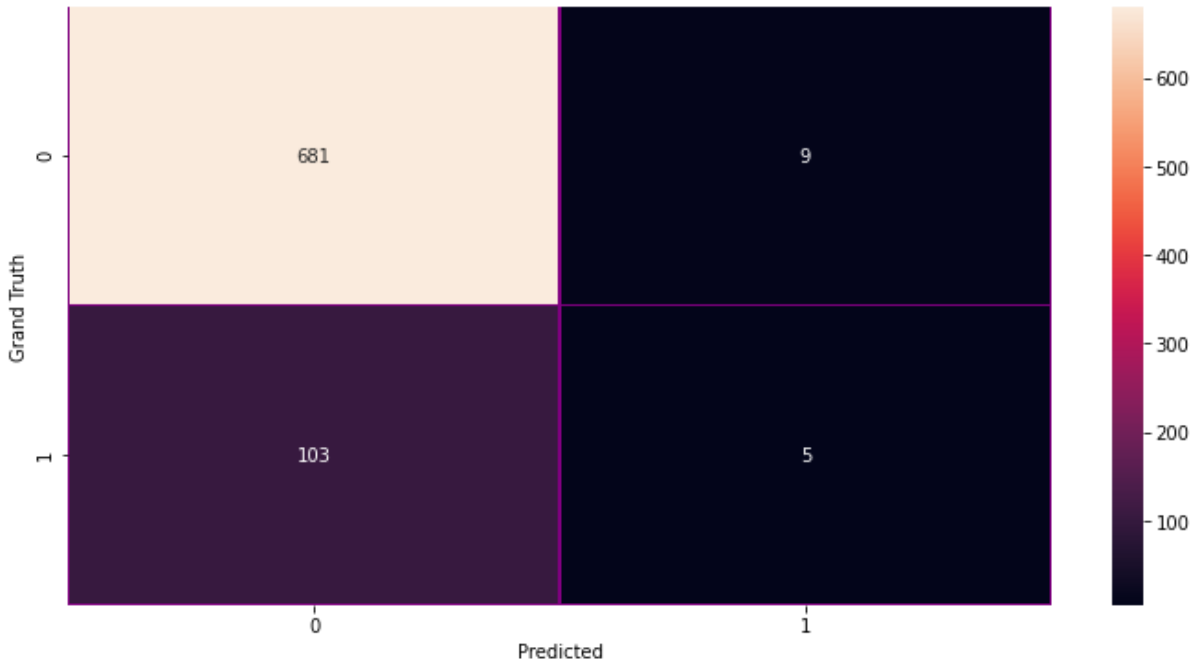


Figure.15 Confusion Matrix of KNN

5.2.2 Decision Tree

The most important parameter on building a decision tree with high performance is the maximum depth because it shouldn't be very low that makes an underfitting problem or very high that makes an overfitting problem. So, to get the optimal decision tree with highest possible accuracy we have applied a range of maximum depth between 1 and 6 on the dataset and the highest accuracy we got is with depth equals 3.

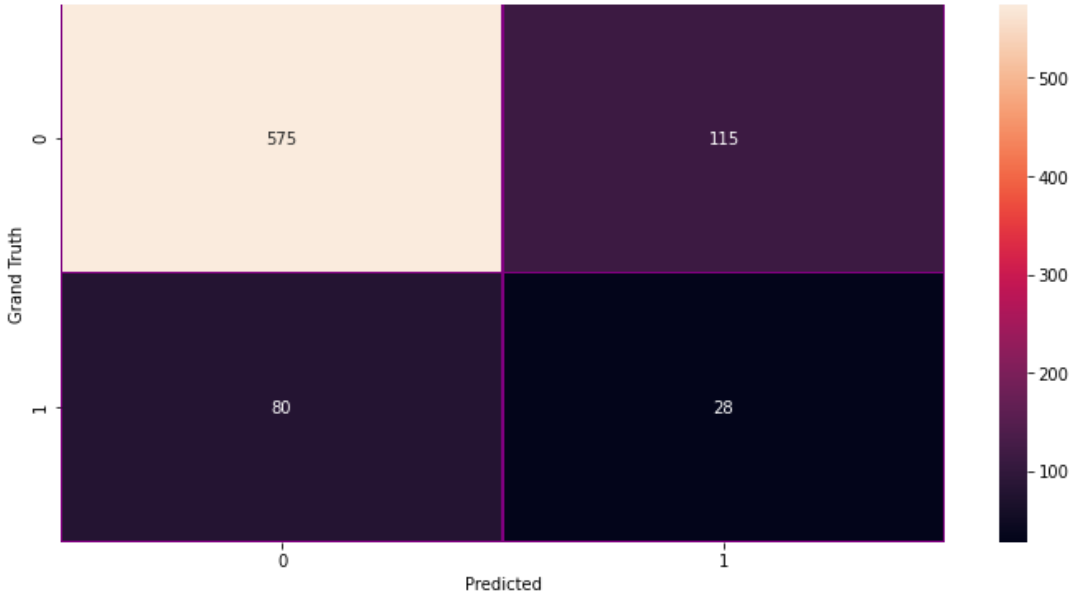


Figure.16 Confusion Matrix of Decision Tree

This is the report of the confusion matrix. We can see that the recall and the precision of the class 1 is very low and that is due to the false negative and the true positive respectively.

	precision	recall	f1-score	support
0.0	0.88	0.83	0.86	690
1.0	0.20	0.26	0.22	108
accuracy			0.76	798

5.3 Proposed Ensemble Method

Ensemble methods use multiple machine learning algorithms. One of them is a random forest that combines many decision trees.

5.3.1 Baseline Model

We started by trying a simple model and the accuracy for the model was 82% but the results weren't very good.

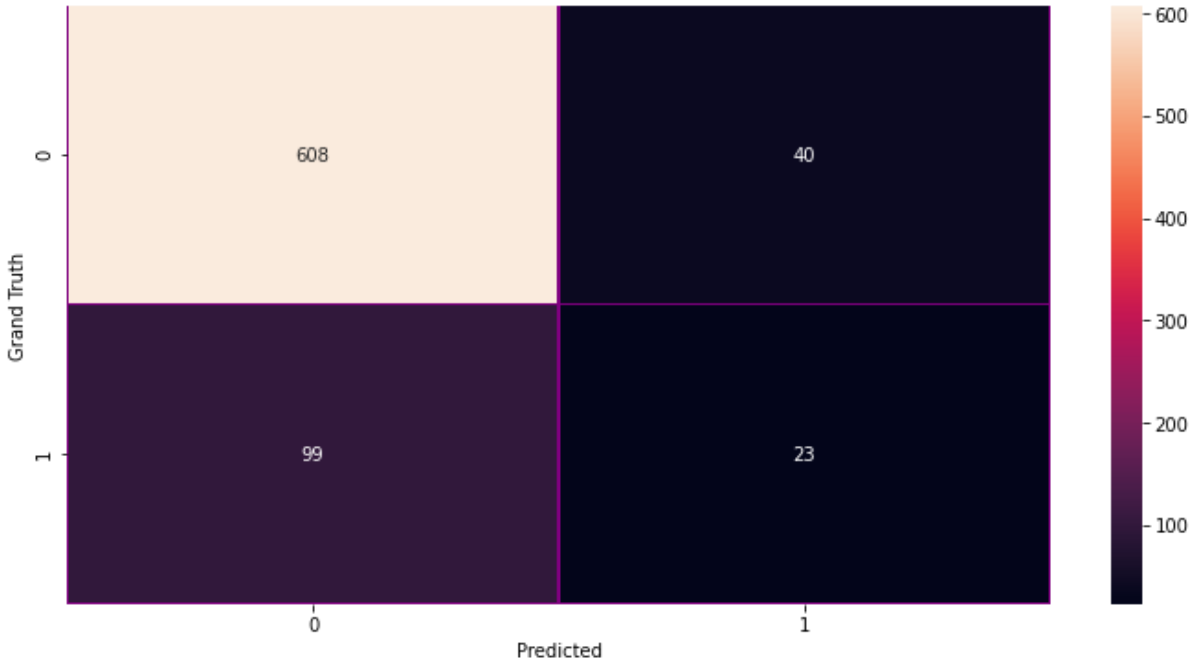


Figure.17 The Confusion Matrix of the Initial Random Forest Model

5.3.2 Random Search Training

We want to find the best hyperparameters and the best approach to narrow our search is to evaluate a wide range of values for each hyperparameter. Using Scikit-Learn's RandomizedSearchCV method, we can define a grid of hyperparameter ranges, and randomly sampled from the grid, performing K-Fold CV with each combination of values. We tried to adjust the following set of hyperparameters with the following values:

Table.5 The Random Forest Hyperparameters

Hyperparameter name	Description	Values range
n_estimators	number of trees in the forest	[50: 200]
max_features	max number of features considered for splitting a node	['auto', 'sqrt']
max_depth	max number of levels in each decision tree	[5: 35]
min_samples_split	min number of data points placed in a node before the node is split	[2, 5, 10]
min_samples_leaf	min number of data points allowed in a leaf node	[1, 2, 4]
bootstrap	method for sampling data points (with or without replacement)	[True, False]

After that, we instantiate the random search and fit it like any Scikit-Learn model. Then the random search shows the best parameters after fitting the model.

- `'bootstrap': True`
- `'max_depth': 12`
- `'max_features': 'sqrt'`
- `'min_samples_leaf': 2`
- `'min_samples_split': 10`
- `'n_estimators': 200`

5.3.3 Model Evaluation

Now the model accuracy is 85% with improvement of 0.82%.

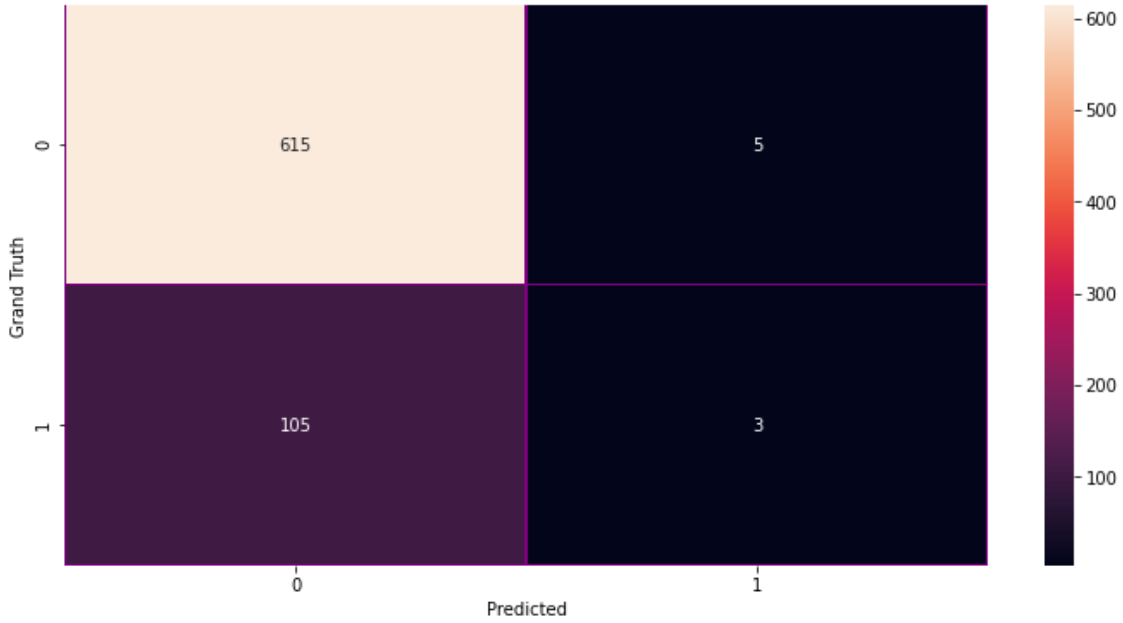


Figure.18 The Confusion Matrix of Random Forest After Hyperparameter Tuning

The false negatives didn't improve much as in the confusion matrix.

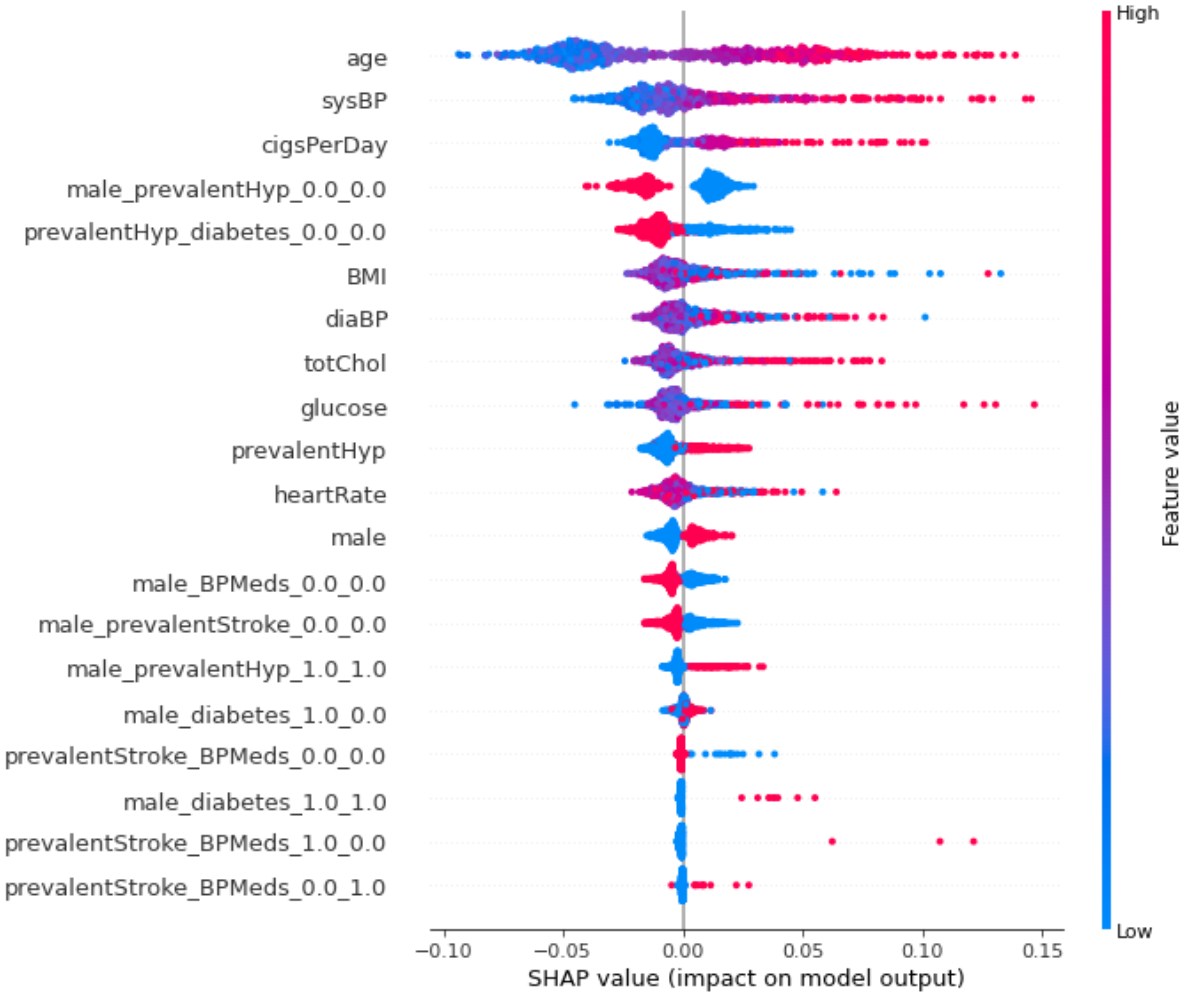


Figure.19 Features' Impact on Model Output By SHAP

From the above plot, we can observe that the most features that have an impact on the model are 'age', 'sysBP' and 'cigsPerDay'.

5.4 Proposed Neural Network algorithms

5.4.1 Baseline Model

The Neural Network has many types but the Multilayer Perceptron is used here. It is a simple algorithm intended to perform binary classification. it classifies whether the input belongs to 10-years risk class or not.

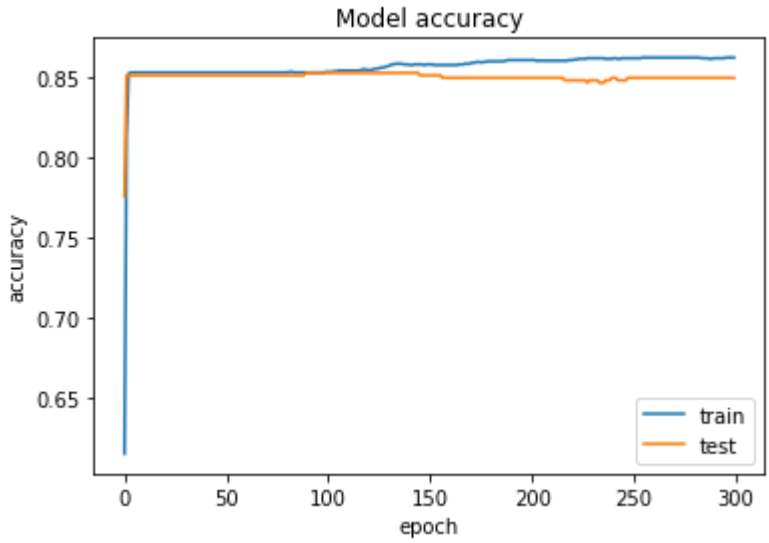


Figure.20 Train-Test Model Accuracy

The above plot is after running the base model of the neural network which has only one hidden layer of 20 nodes. The activation function used in the hidden layer is **Relu** while the output layer activation function is **Sigmoid**. The model's accuracy is 84% and has an overfitting problem.

5.4.2 Hyperparameter Tuning

Hyperparameter optimization is a big part of deep learning. The reason is that neural networks are difficult to configure and there are a lot of parameters that need to be set. On top of that, individual models can be very slow to train. In our project, we used Grid search to configure the optimal parameters for the model.

5.4.2.1 Optimizers

Keras offers a suite of different optimization algorithms. So we tune the optimization algorithm used to train the network, each with default parameters. We tried 'SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax' and 'Nadam'. The best algorithm with the highest accuracy was **Adamax**.

```
Best: 0.856195 using {'optimizer': 'Adamax'}

0.854224 (0.006786) with: {'optimizer': 'SGD'}

0.853489 (0.009874) with: {'optimizer': 'RMSprop'}

0.853486 (0.006080) with: {'optimizer': 'Adagrad'}

0.855456 (0.009463) with: {'optimizer': 'Adadelta'}

0.851025 (0.009735) with: {'optimizer': 'Adam'}

0.856195 (0.009254) with: {'optimizer': 'Adamax'}
```

```
0.851764 (0.008727) with: {'optimizer': 'Nadam'}
```

5.3.2.2 Learning Rate and Momentum

Learning rate controls how much to update the weight at the end of each batch and the momentum controls how much to let the previous update influence the current weight update. We tried a suite of small standard learning rates 0.0001, 0.001, 0.01 and 0.1 and momentum values from 0.0 to 0.9 in steps of 0.2.

```
Best: 0.857919 using {'learn_rate': 0.001, 'momentum': 0.0}

0.857919 (0.007449) with: {'learn_rate': 0.001, 'momentum': 0.0}

0.854718 (0.007043) with: {'learn_rate': 0.001, 'momentum': 0.2}

0.853980 (0.008537) with: {'learn_rate': 0.001, 'momentum': 0.4}

0.855457 (0.009978) with: {'learn_rate': 0.001, 'momentum': 0.6}

0.853241 (0.008095) with: {'learn_rate': 0.001, 'momentum': 0.8}

0.855949 (0.007852) with: {'learn_rate': 0.001, 'momentum': 0.9}
```

5.4.2.3 Weight Initialization

Neural network weight initialization used to be simple. Now there is a suite of different techniques to choose from Keras. We tune the selection of network weight initialization by evaluating all of the available techniques. We will use the same weight initialization method on each layer. We tried 'uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal' and 'he_uniform'. The best weight initialization was **lecun_uniform.**

```
Best: 0.855456 using {'init_mode': 'lecun_uniform'}

0.854719 (0.008222) with: {'init_mode': 'uniform'}

0.855456 (0.009170) with: {'init_mode': 'lecun_uniform'}

0.854965 (0.009559) with: {'init_mode': 'normal'}

0.851764 (0.007753) with: {'init_mode': 'zero'}

0.854719 (0.010311) with: {'init_mode': 'glorot_normal'}

0.854225 (0.007461) with: {'init_mode': 'glorot_uniform'}
```

```
0.851764 (0.011275) with: {'init_mode': 'he_normal'}

0.853980 (0.010273) with: {'init_mode': 'he_uniform'}
```

5.4.2.4 Number Of Neurons

The number of neurons in a layer is an important parameter to tune. Generally, the number of neurons in a layer controls the representational capacity of the network, at least at that point in the topology, A larger network requires more training and at least the batch size and number of epochs should ideally be optimized with the number of neurons.

```
Best: 0.855457 using {'neurons': 10}

0.855457 (0.008495) with: {'neurons': 10}

0.855457 (0.009825) with: {'neurons': 12}

0.853733 (0.009149) with: {'neurons': 15}

0.854226 (0.009850) with: {'neurons': 20}

0.854226 (0.008422) with: {'neurons': 25}

0.853734 (0.008770) with: {'neurons': 30}
```

5.4.3 Model Evaluation

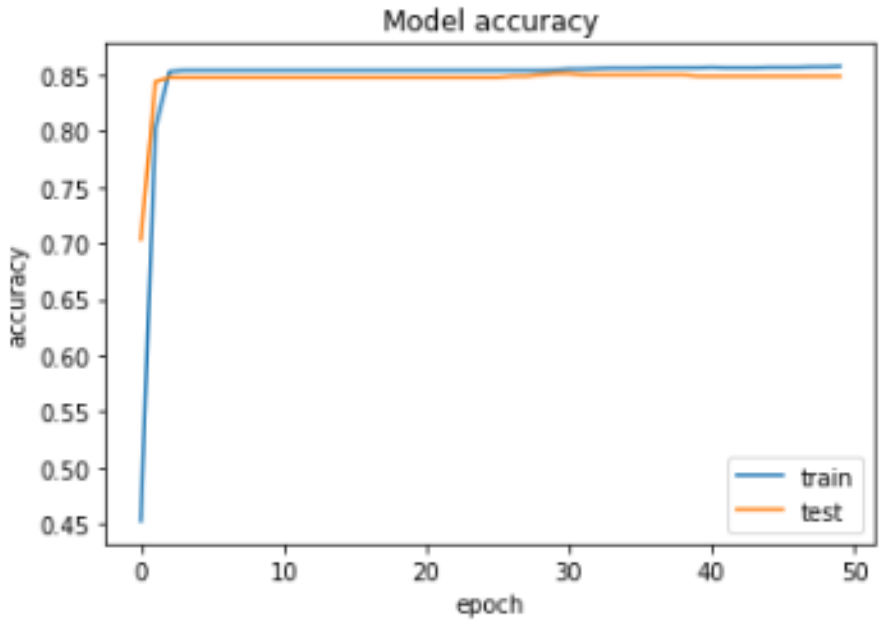


Figure.21 Train-Test Model Accuracy

The above plot is after tuning the parameters based on the grid search result and the accuracy is 85.5% with an improvement of 1%.

6. Models Comparisons

6.1 Accuracy Comparison

Table.6 Accuracy Comparison of Different Classification Techniques

Classification Technique	CVD Model Accuracy
Random Forest	87%
MLPNN	85.5%
KNN	84.14%
Decision Tree	74.20%

6.2 Medical Data Comparison

Some doctors make initial diagnostics based on the symptoms the patients give. But most of the time these diagnostics are misleading. So we have been searching for a disease dataset that has almost the same features as heart disease like liver failure. The liver failure has 20 attributes and the target. The common features between liver failure and CVD are *['gender', 'age', 'HyperTension', 'Diabetes', 'Total cholesterol', 'Maximum Blood Pressure', 'Minimum Blood Pressure', 'Body Mass Index']*.

We applied the multilayer perceptron neural network and random forest on the liver failure dataset.

Table.7 Accuracy and Medical Data Comparison of Different Classification Techniques

Classification Technique	CVD Model Accuracy	LF Model Accuracy
MLPNN	85.5%	93.4%
Random Forest	87%	93.9%

7. Implementation

7.1 Software Tools

The language used in the project is **Python 3.**

7.1.1 Python Libraries

Table.8 Python Libraries Used

Library	Function
Pandas	An open-source data manipulation tool in data science
Scikit Learn	A machine learning library that provides classification, regression and clustering algorithms
Matplot	Provides interactive animated or static visualization for data
Seaborn	A statistical graphing library based on matplot
SHAP	Shows the impact of each feature on the output
Keras	API for neural networks

7.1.2 Google Colaboratory

Google colaboratory is a **Jupyter notebook** environment that runs entirely in the cloud. It does not require a setup and can be edited by the team members. Colab supports GPU which is required for parallel operations. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Google Colab Capabilities:

CPU: Intel(R) Xeon(R) CPU @ 2.2GHz (There are 2 CPUs available).

Disk :

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         108G   31G   72G  31% /
tmpfs            64M     0   64M   0% /dev
tmpfs           6.4G     0  6.4G   0% /sys/fs/cgroup
shm             5.9G     0  5.9G   0% /dev/shm
tmpfs           6.4G   12K  6.4G   1% /var/colab
/dev/sda1       114G   33G   82G  29% /etc/hosts
tmpfs           6.4G     0  6.4G   0% /proc/acpi
tmpfs           6.4G     0  6.4G   0% /proc/scsi
tmpfs           6.4G     0  6.4G   0% /sys/firmware
```

RAM: 13 GB

7.2 Challenges

7.2.1 Finding an appropriate dataset

- At the very beginning we've found on Kaggle a huge CardioVascular Disease dataset of about 70,000 instances.[20]

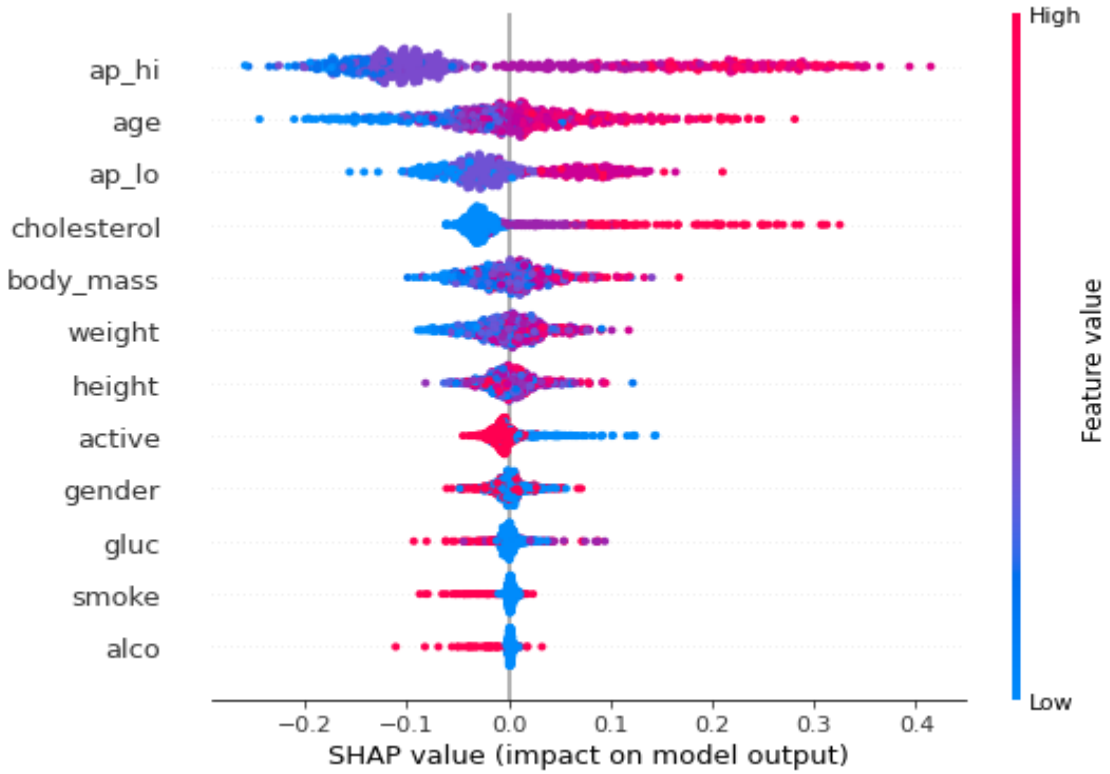


Figure.22 Features Impact on Model Output by SHAP

There are some observations from the above plot:

1. The high pressure is very effective so that at high value the model increases the probability for class 1 and vice versa which is right.
2. On the other hand, smoke variable is not important at all for the model when its value is zero as most of the values are zero for this variable so the target variable is changing while it's the same value and we see that when it's 1 the model decreases the probability for having the disease! one interpretation for this is that the number of persons smoke in and do not have a disease is bigger than who have which is misleading from the small number of smoking people in the data
3. Almost the same problem applies for glu, alco
4. We can see a huge spread for cholesterol although it has only 3 values so this is an indicator that it interacts with another variable.

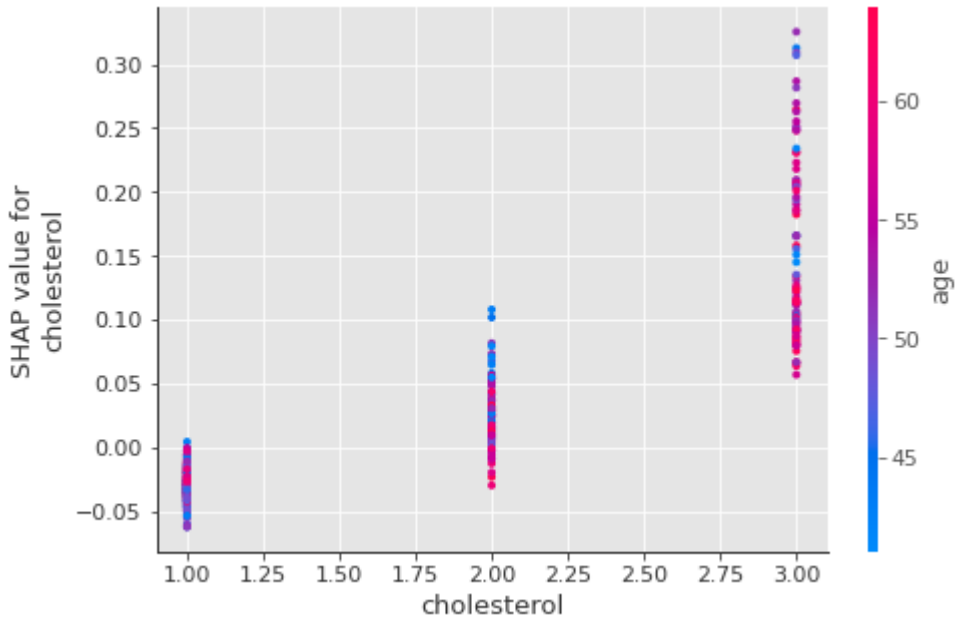


Figure.23 SHAP Value for Cholesterol

As we can see above cholesterol increases when age is small and it doesn't make any sense.

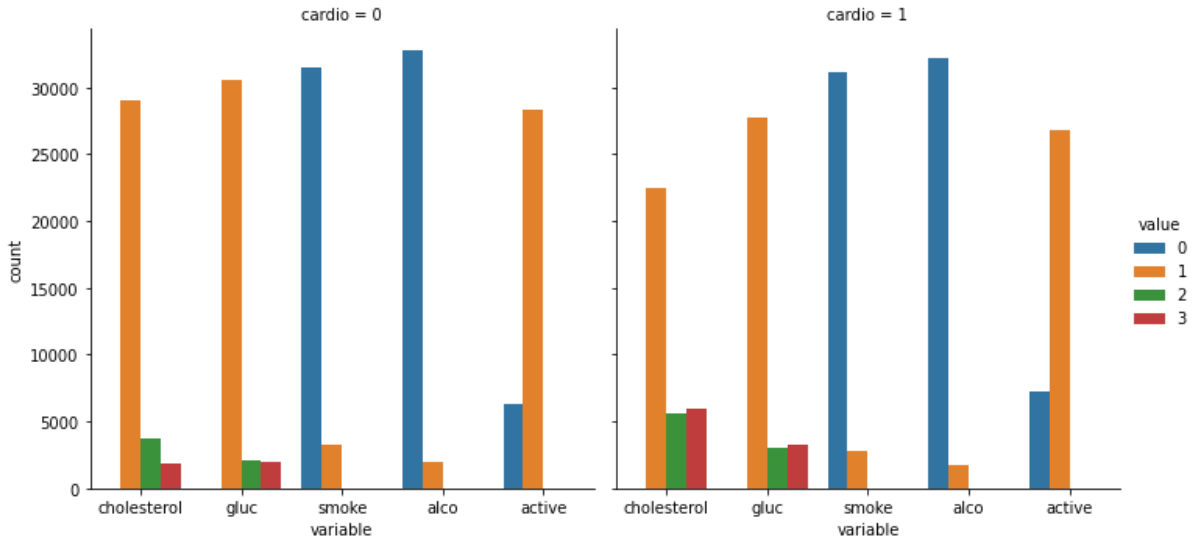


Figure.24 Count of Features in comparison with the class equals to 1 and 0

From this plot, we observe that the data is going towards being healthy when the target equals 1.

7.2.2 Hardware usage

When using Pycharm that works on our PC, we've faced a big problem. The executing performance was very bad as it took a lot to run the code. Besides, our PCs began to hang up.

8. Conclusion and Future Work

Predicting the CardioVascular Disease risk is an important step in preventing health deterioration. Through our work in the project, we've tried our best to get satisfactory accuracy. After cleaning the data and solving the inconsistencies, we've used a couple of machine learning models. Random Forest provided the best accuracy of 87% and Neural Network provided the best one following it of 85.5%.

Liver failure dataset is also used in the project to predict the Liver failure risk. Random Forest provided an accuracy of 93.9% while Neural Network provided an accuracy of 93.4%.

From this project, we came up with the following ideas that could be taken into consideration in future work. We have to take real consistent data in a large sample of patients from reputed medical institutes of our country and use that data to train and test our prediction models. We could make a complete risk assessment system with a percentage output regarding the heart and liver health. We could also add other diseases such as internal diseases. Enabling a hospital system to analyze their patients' cases could also be done.

References

[1] https://www.google.com/search?q=iterative+modeling&sxsrf=ALeKk03NauL39hG8oB-Q4WO0EPB3D3Tv2g:1595937810948&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjS87fS8_qAhVHzIUKHRevBm8Q_AUoAXoECBMQAw&biw=1366&bih=625#imgrc=V9AGVoxTF1_LiM
[2] https://www.edureka.co/blog/classification-in-machine-learning/
[3] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[4] https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
[5] https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96
[6]https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature.
[7] https://www.sciencedirect.com/science/article/pii/S2090447920300095
[8] https://deepai.org/machine-learning-glossary-and-terms/backpropagation
[9] https://www.techopedia.com/definition/32064/cross-validation
[10] https://images.app.goo.gl/smTpmWRTZCHA1yqt6
[11] https://www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression
[12] https://www.kaggle.com/sulianova/cardiovascular-disease-dataset