Project Report

On

# “Implementing Machine Learning Models For Accurate Disease Prediction”

Submitted in Fulfillment of the Formal Project

Under the Guidance and Supervision of

**Dr L Rajya Lakshmi**

Faculty, Department of Computer Science & Information Systems

Birla Institute of Technology & Science, Pilani, 333031

Rajasthan, INDIA


**Submitted by**

| | |
|---|---|
| **Hrishabh Sehrawat** | **2019A7PS0136P** |
| **Sanskar Jhajharia** | **2019A7PS0148P** |

# Acknowledgements

We would like to highly acknowledge the guidance provided by our supervisor, professor Dr L Rajya Lakshmi under whose supervision this project has become a reality. We would also like to acknowledge the exemplary research work done by the fast.ai founders Jeremy Howard, and Rachel Thomas, along with the research work done in the field of heart disease detection by M.Ganesan, Dr N.Sivakumar among others.

Our heartfelt gratitude to the Administration and Teaching and Non Teaching staff of Birla Institute of Technology and Science, Pilani, Pilani Campus, for constantly motivating us to work towards attaining the seemingly unattainable.

We would take this opportunity to thank our parents and family for supporting us throughout the journey and instilling us with the requisite strength and courage to stay motivated through the entire duration.

We would also like to thank our friends and peers who helped us with our doubts and believed in us when we ourselves didn’t.

# Abstract

With the rapid changes and latest advancements in the field of IoT and sensing technologies, the domain of health care has been presented with an infinite amount of data available at their disposal in almost no time. To properly utilise the same, it is important to monitor such devices and come up with methods to diagnose serious diseases. In our study, we have tried to work on an efficient framework that processes on the UCI Repository dataset to predict people suffering from heart diseases. To properly analyse the dataset, extensive data preprocessing is carried out which includes correlation analysis, the introduction of dummy variables, data normalisation and descriptive analysis of cross dependence of variables. An array of classification algorithms are then used to classify the patient data for identification of the heart disease (target).

For testing and training purposes the same database is used after having split it in a 70:30 ratio. The seed of randomness is varied gradually to test the same on different test and train data. For experimentation, the benchmark dataset is tested on a set of classifiers namely Logistic Regression, Multilayer perceptron, Support Vector Machine, K-Nearest Neighbour and Extreme Gradient Boosting (XGBoost).

An important aspect to be noted for the same is that the referenced paper had worked on a trimmed dataset post removal of some outliers (the method of whose detection was not informed). We, however, performed only pre-processing on the data without compromising on any observation points that were available. The simulation result ensured that the SVM and the MLP models showed better performance than the other classifiers. The test accuracy and precision of our trained models (namely Logistic Regression, SVM and MLP) were significantly better than the accuracy and precision of the referenced work.

# Table of Contents

# Introduction

The healthcare industry has historically been an early user of technology advancements and has reaped significant benefits. Machine learning is now used in a variety of health-related fields, including the development of new medical procedures, the management of patient data and records, the treatment of chronic diseases, the ability to predict a patient's likelihood of readmission to the hospital, and the prediction of the need for end-of-life care, to name a few.

Machine learning is a crucial part of the rapidly expanding discipline of data science. Algorithms are trained to generate classifications or predictions using statistical approaches, revealing crucial insights in data mining initiatives. These statistical methods often find their use in multitudes of scenarios, such as finance, healthcare, computer science, agriculture, big data analysis, to name a few.

In recent years, due to industrial developments, and a modernised way of living, which often has a side effect of a sedentary life, we are being pushed towards a future of severe chronic health-related issues, including but not limited to heart, kidney and diabetic diseases. Due to this, the healthcare sector can be severely impacted and overloaded with new patients without a way to diagnose them accurately and with an appreciable speed. To solve these issues, we have to look at the big data analysis provided by the IoT devices, which can then be analysed by state-of-the-art machine learning algorithms to predict the occurrences of diseases in patients.

In this project, we focus our attention on diabetes and cardiac diseases using the datasets used in the 2019 paper “IoT based heart disease prediction and diagnosis model for healthcare using machine learning models”[3] by M. Ganesan and N. Sivakumar, and the 2019 paper by Ed-daoudy, Abderrahmane & Maalmi, Khalil. “A new Internet of Things architecture for real‑time prediction of various diseases using machine learning on big data environment”[5] to train machine learning algorithms and compare our models’ accuracy to those mentioned in the paper. We trained our models on popular machine learning algorithms such as Decision Trees, XGBClassifier, Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Random Forest (RF), Multi-Layer Perceptron (MLP), Logistic Regression (LR) to name a few. While doing the same, we were able to improve our models to such an accuracy that we were able to exceed the accuracy of the first paper.

# Why Machine Learning in Health Care?

Machine Learning is a powerful tool that helps deliver insights hidden in the data extracted using Internet of Things (IoT) devices. They work smartly to help improve the decision-making process in different areas like education, security, business and the healthcare industry. With the help of ML, we can easily demystify the hidden patterns in bulk data for optimal prediction and recommendation systems. Modern healthcare fields have embraced IoT and ML so as to automate the machines with the maintenance of medical records, predicting disease diagnoses based on some underlying data pattern and most importantly real-time monitoring of patients. Individual ML algorithms have different results on different datasets and depend heavily on how well is the dataset in consideration processed.

Accurate health prediction systems are important as it helps hospitals promptly reassign outpatients to less congested treatment facilities. The current pandemic has proven that we as a society lack a number of integral health care facilities ranging from beds, oxygen cylinders to even men forced to take care of the patients. The Machine Learning algorithms aim to address the common issue of a sudden change in the inflow and outflow of patients. The IoT creates a bridge between virtual computers and physical things to facilitate this communication/ It enables the immediate gathering of information through innovative microprocessor chips and relays the same as inputs to machine learning paradigms. The surge in population and the erratic spread of chronic conditions has strained modern health care.

Besides the direct application in the hospitals for maintenance of the patient flow, Machine Learning can as well help in tracking the spread of disease. The smart devices in IoT have enabled researchers from across the globe to evaluate patient samples immediately and exchange them in real-time with thousands of disease tracking devices. Applications such as Health Map and Epic Aster merge IoT data with demographics, GIS data, land use information, social network alerts and other Simulation Lab sources to track the evolving threats to public health during an outbreak of a disease.

One recent use case of Machine Learning in Healthcare is seen in Microsoft's Project InnerEye which employs Machine Learning based algorithms to differentiate between tumours and healthy anatomy using 3D radiological images. This has significantly helped the medical experts in radiotherapy and surgical planning. Pfizer is currently using Machine Learning for immuno-oncology research to study how the body's immune system can fight against cancer naturally.

Decision Support Systems in Healthcare extensively use Machine Learning. It helps the finance branches of the hospitals and clinics to keep track of charging accounts, receivables, outlays and accounts payable by the patients. This method additionally helps to better predict the patient's insurance policy and different refund choices. It also helps the medical practitioners in arranging

a perception of health issues by exposing the background knowledge of the individual patients. This includes computer-aided diagnosis, image registration, image annotation. Image-guided medical aid and image database retrieval, multimodal image fusion, medical image segmentation among others. A bulk of medical knowledge is unstructured information within the variety of numerous completely unrelated notes, images, audios, reports and discharge summaries.

An analysis of the same is shown in the table below:

| Application Name | Description | Issues Addressed | Challenges |
|---|---|---|---|
| Medical imaging | Medical imaging is largely manual today as it entails a health professional examining images to determine abnormalities. However, machine learning algorithms can be used to automate this process and enhance the accuracy of the imaging process. | The use of machine learning addresses the issues of accuracy and efficiency when imaging is done manually. | High dependency on the quality and amount of training sets. Ethical and legal issues concerning the use of ML in healthcare. It is often difficult to explain the outputs of deep learning techniques logically. |
| Diagnosis of diseases | Clinical diagnosis can benefit from machine learning by improving the quality and efficiency of decision-making. | Wrong patient diagnoses result in inappropriate interventions and adverse outcomes. | The lack of sound laws and regulations defining the utilization of ML in healthcare. Obtaining well-annotated data for supervised learning is challenging. |
| Heart disease prediction | AI can be utilized to predict heart disease, hence enabling patients and health providers to implement preventive measures. | There is a need for accurate prediction of cardiovascular diseases, as well as the implementation of effective treatments to improve patient outcomes. | The lack of ethical guidelines to direct the adoption of heart disease prediction algorithms. Machine learning algorithms cannot solve highly abstract reasoning problems. |

# Dataset

The dataset used for the training and testing of the algorithms is the standard Heart Disease UCI dataset available freely for research and analysis purposes.[4]
The dataset comprises 303 data entries corresponding to 13 features and one target variable.

The target variable is denoted by header 'target' which takes values either 1 or 0 depending on whether the patient in consideration has heart disease or not. The distribution of same is as shown below:

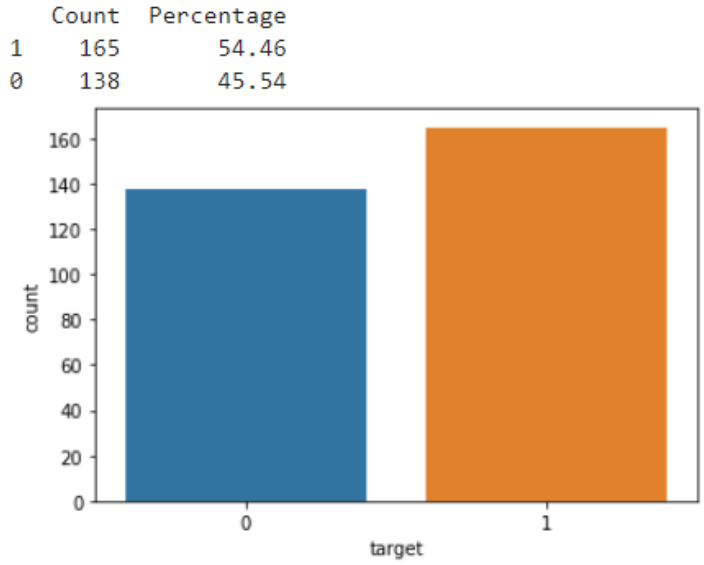


Fig 1: Graph showing the distribution of the target variable

The 13 predictor or independent variables are as noted below:

1) age: Inputs the age of the patient in years.
2) sex: Used to represent the sex of the patient (1: male, 0:female)
3) cp: Chest Pain type detected by the patient. (Can take 4 values: 0,1,2,3)
4) trestbps: Discrete value denoting resting blood pressure (in mm Hg on admission to the hospital)
5) chol: Serum cholesterol in mg/dl
6) fbs: Fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
7) restecg: Resting electrocardiographic results (Can take values 0,1,2)
8) thalach: Maximum heart rate achieved
9) exang: Denotes exercise induced angina (1 = yes; 0 = no)
10) oldpeak: ST depression induced by exercise relative to rest
11) slope: The slope of the peak exercise ST segment (can take 3 distinct values: 0,1,2)
12) ca: Denotes number of major vessels colored by fluoroscopy (Can take values: 0,1,2,3,4)
13) thal: Categorical variable with: 3 = normal; 6 = fixed defect; 7 = reversible defect

A graphical view of each variable is attached below:

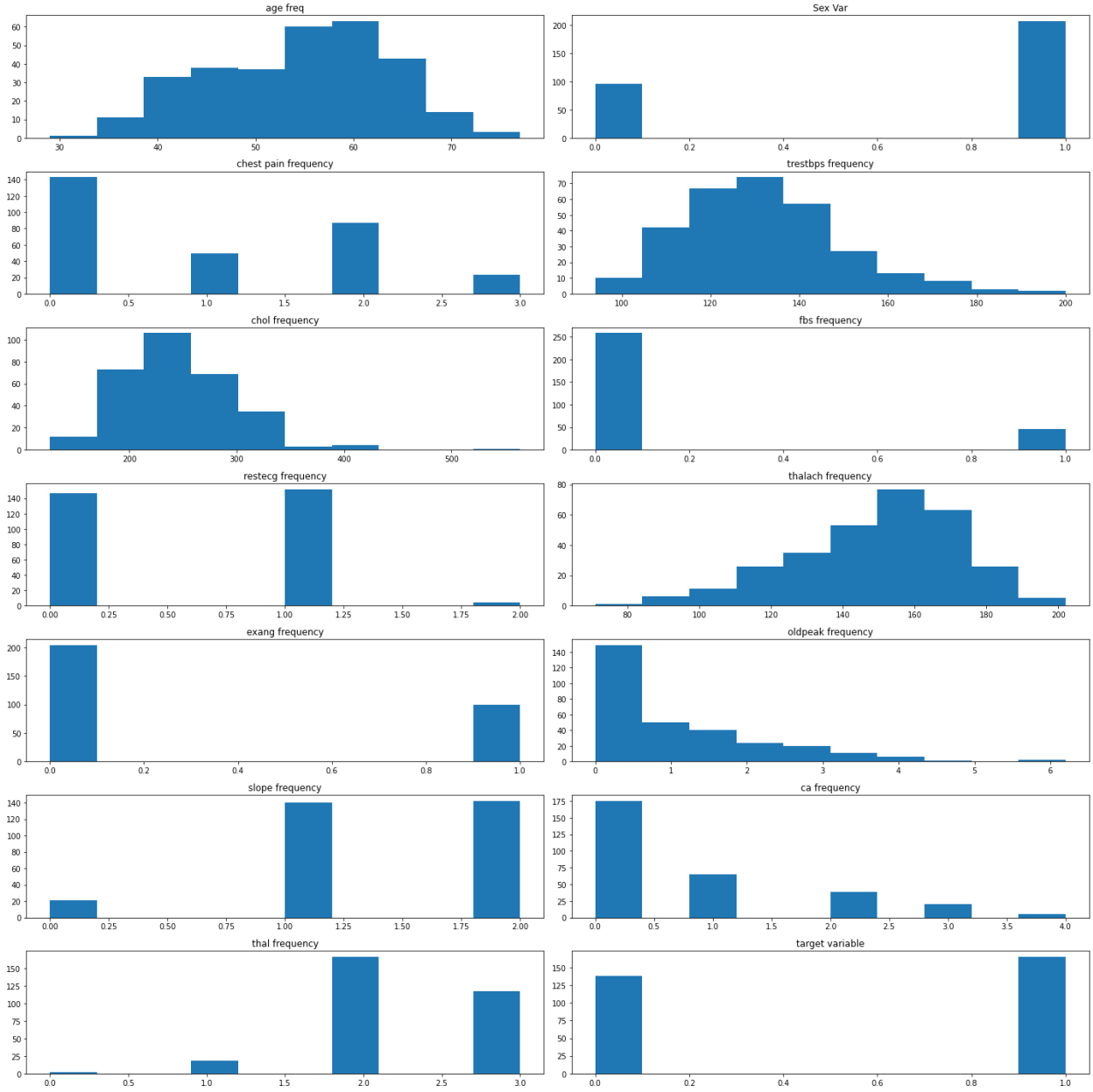


Fig 2: Set of graphs representing the various parameters of our dataset

To understand the variation in the data better, we find the descriptive statistical parameters of the 14 data heads. The results of the same are attached below:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **age** | 303.000000 | 54.366337 | 9.082101 | 29.000000 | 47.500000 | 55.000000 | 61.000000 | 77.000000 |
| **sex** | 303.000000 | 0.683168 | 0.466011 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| **cp** | 303.000000 | 0.966997 | 1.032052 | 0.000000 | 0.000000 | 1.000000 | 2.000000 | 3.000000 |
| **trestbps** | 303.000000 | 131.623762 | 17.538143 | 94.000000 | 120.000000 | 130.000000 | 140.000000 | 200.000000 |
| **chol** | 303.000000 | 246.264026 | 51.830751 | 126.000000 | 211.000000 | 240.000000 | 274.500000 | 564.000000 |
| **fbs** | 303.000000 | 0.148515 | 0.356198 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **restecg** | 303.000000 | 0.528053 | 0.525860 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 2.000000 |
| **thalach** | 303.000000 | 149.646865 | 22.905161 | 71.000000 | 133.500000 | 153.000000 | 166.000000 | 202.000000 |
| **exang** | 303.000000 | 0.326733 | 0.469794 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| **oldpeak** | 303.000000 | 1.039604 | 1.161075 | 0.000000 | 0.000000 | 0.800000 | 1.600000 | 6.200000 |
| **slope** | 303.000000 | 1.399340 | 0.616226 | 0.000000 | 1.000000 | 1.000000 | 2.000000 | 2.000000 |
| **ca** | 303.000000 | 0.729373 | 1.022606 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 4.000000 |
| **thal** | 303.000000 | 2.313531 | 0.612277 | 0.000000 | 2.000000 | 2.000000 | 3.000000 | 3.000000 |
| **target** | 303.000000 | 0.544554 | 0.498835 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |

Fig 3: A table focussing on the descriptive statistics of the parameters

The dataset is thus found to comprise both categorical variables and metric variables. One fundamental step during data pre-processing would be to convert all such categorical variables to dummy variables using the suitable number as specified by the levels the categorical variable can take.

From the values of *age*, *trestbps*, *chol,* and *thalach*, we can analyse that the best model can be trained only after the normalisation of these 4 fields. In the absence of proper normalisation, the contribution or level of significance of them can't be compared first-hand with those of the other variables.

# Data Pre-Processing

## Correlation Analysis

Correlation is used by us on a regular basis to denote some form of association. In statistics, correlation means an association between two quantitative variables. For all practical purposes, we consider this relation to a linear one and proceed with the analysis from there. This degree of association is called the correlation coefficient (denoted by r). It is also called the Pearson correlation coefficient.

The correlation coefficient is measured on a scale that varies from +1 through 0 to -1. A complete correlation between two variables is expressed by either +1 (Complete positive relation denoting that an increase in one causes an equal increase in the other variable) or -1 (denoting an equal measure of decrement in the value of one variable with the increase in another). The complete absence of correlation is represented by r=0.

The formula for the calculation of the same is as given below:

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{[\sum(x - \bar{x})^2(y - \bar{y})^2]}}$$

To demonstrate the correlation of the various independent variables in our dataset (13 in total) with the 'target' variable, a heatmap is generated. The same is as shown:

The more the absolute value of the correlation coefficient, the higher is the effect of that variable on the target. This heat map suggests that *'cp'* and *'exang'* have the most effect on the target.

The analysis of the correlation between the target and the independent variables is followed by correlation analysis between the independent variables. The same is necessary to detect any multicollinearity in the dataset. Multicollinearity occurs when there is a significant correlation between the independent variables of the model. This is a sensitive condition as by default the independent variables are considered to be uncorrelated to each other.

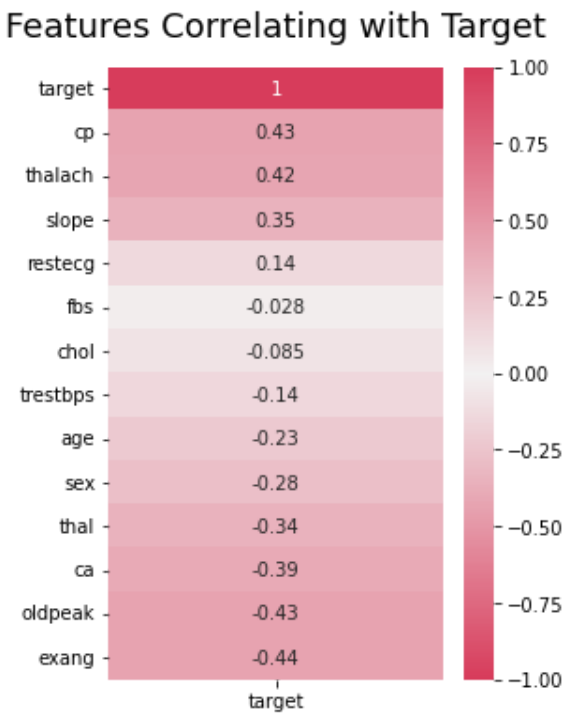


Fig 4: Representation of Correlation of features with target

A key goal of all forms of regression analysis is to isolate the relationship between the dependant variable (target) and each independent variable (13 in our case). For impartial interpretation of the regression coefficient, it should be possible to change the value of only one independent variable withholding all other variables constant. Multicollinearity compromises this aspect of regression analysis. The change in one variable will assert a change in the value of the other variable too and hence the estimation of the relationship is difficult.

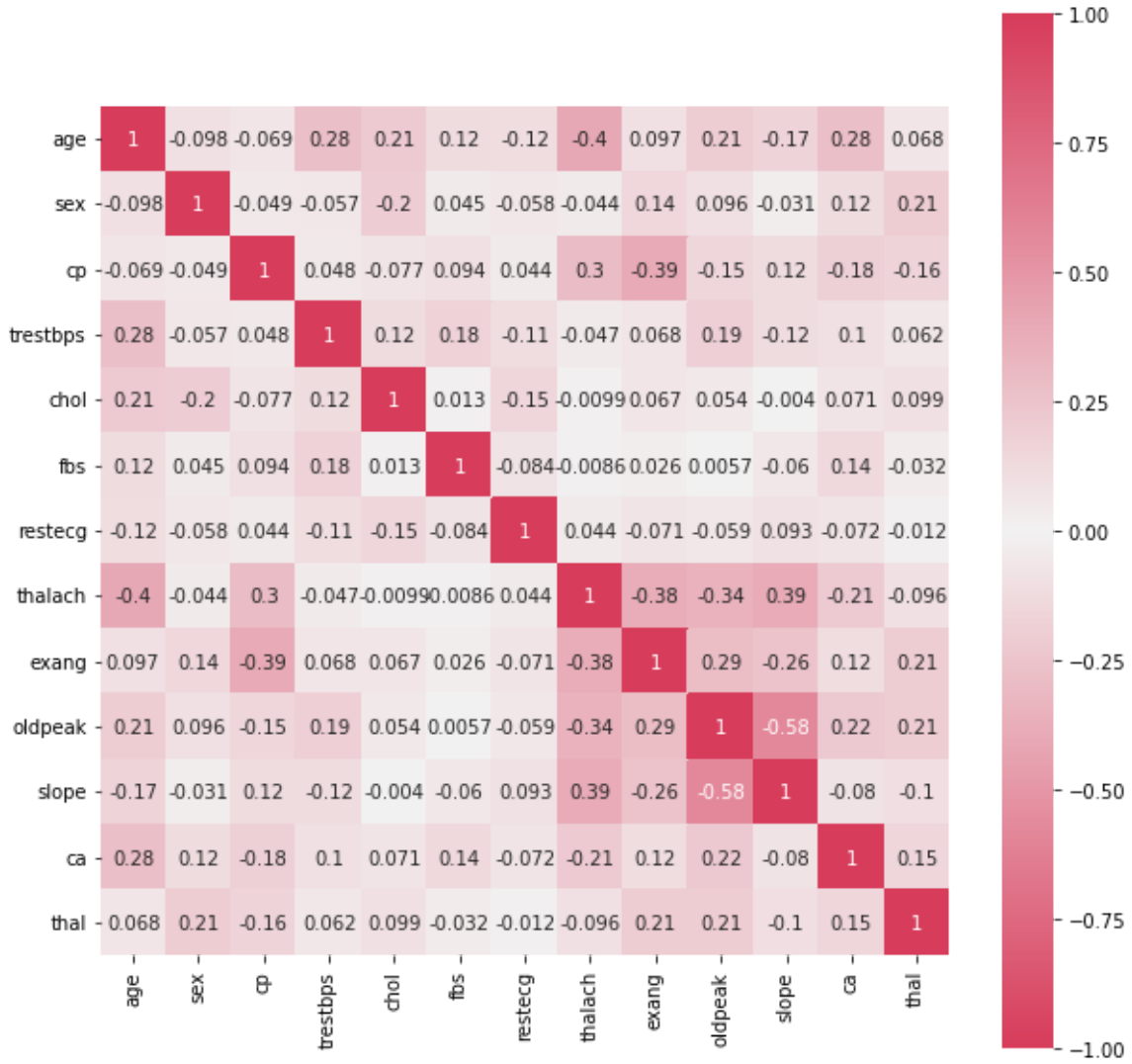


Fig 5: Correlation matrix of the dataset parameters

The results above show the correlation between every parameter in the dataset available. The principal diagonal of the matrix indicates the correlation between the same variables is always 1.

As a threshold for identifying the multicollinearity between any two variables, we use the judgement stick of $|r| >= 0.7$. Most of the literature[1] indicates an $r > 0.7$ or $r < -0.7$ to be used as a

yardstick for multicollinearity. Multicollinearity prevents us from moving any further with our known models. Our correlation matrix fails to record any variable pair with a correlation coefficient lying in the specified region. Hence we can safely conclude that there is no multicollinearity in the independent variables.

## Data Reduction

When dealing with a vast database like this, it is necessary to check if the entire data is worthy enough to be carried forward for training purposes. We apply multiple data reduction techniques in our dataset.

### Factor Analysis

Factor analysis is a statistical technique that is responsible for reducing a large number of variables into fewer factors. It extracts the maximum common variance from all the variables and puts them together. There are multiple possible ways to apply the same factor analysis to the data. We shall focus mostly on the Principal Component Analysis method.

PCA is the most used method wherein we extract the maximum variance and put them into the first factor. After this, from the remaining unexplained part of the variance, the same process is repeated to get the second factor with maximum variance explaining capability. This can continue till we have completely explained the 100% of the variance of the sample. In some cases, the Kaiser Guttman condition is applied which limits the use of only those factors with Eigenvalues greater than 1.

Before performing factor analysis on the data provided, we must check if factor analysis will actually have an important effect on the data. For the same, Bartlett's Test of Sphericity is performed. Bartlett's test compares the observed correlation matrix with an identity matrix. It is used primarily to check if there is significant redundancy between the variables that we summarize with a fewer number of factors that can explain the same.

**Null Hypothesis:** The Variables are orthogonal or uncorrelated.
**Alternative Hypothesis:** The correlation matrix diverges significantly from the identity matrix.

This uses a chi-squared statistic and a chi-squared test case with pre-specified degrees of freedom dependant on k (number of the factors used in analysis)

In our case, the p-value did not lie in the rejection region (p-value > **α**). Hence we fail to reject the Null hypothesis that means that factor analysis is extremely redundant in the same dataset.

## Variable Reduction based on Correlation Values

Most of the common correlation-based filtering methods have a likelihood to drop more features than required. This problem gets amplified as datasets become larger and more pairwise correlations are seen above a specified threshold value. However, this comes with an innate shortcoming. If we drop more variables, less information will be available and this may lead to suboptimal model performance.

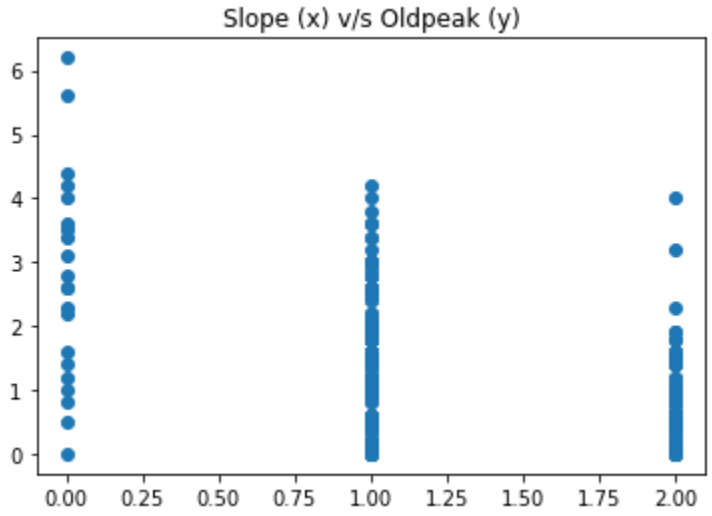


Fig 6: Plot of Slope v/s Oldpeak →

Thus before dropping any variable, it is necessary to descriptively analyse the dataset for any strong correlation which is not intrinsic. For the same, most of the literature suggests a threshold of |r|>=0.4 for a strong self-correlation. In our dataset, there are two pairs of values that conform to the same. One is (slope, oldpeak) pair with a correlation coefficient of -0.58 and the second is (thalach, age) pair with a correlation coefficient of -0.4

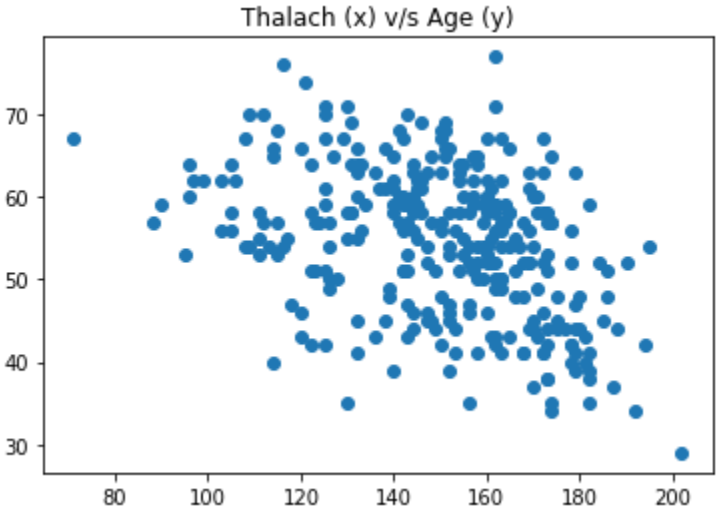


Fig 7: Plot of Thalach v/s Age →

As is seen from the above graphs, the slope v/s oldpeak graph is almost a categorical graph. Hence dropping the pair just based on the correlation coefficient would be erroneous. We cannot drop the pair because of the absence of a metric-metric relation between them.

For the *thalach* v/s *age* graph, we plot the scatter plot. As the correlation coefficient lies right on the borderline, the decision to accept or reject the drop would have been based on the presence of a strong visible relation between the two parameters. However, the plot is almost uniformly scattered over a wide circular region and detection of a clear fit line is impossible.

Hence, for analysis further, we cannot drop any of the variables. We need to proceed with all 13 independent variables.

## Dummy Variables

A dummy variable is one that takes values of 0 or 1 where the value represents the presence or absence of some particular trait. When we have categorical variables in our dataset, it can be represented by a set of dummy variables with one variable for each category of the original categorical imperative defined.

For transforming categorical attributes to numerical attributes, we can use the label encoding procedure (wherein we assign a unique integer to every suitable category of data). In most Machine Learning models, One hot encoding is used. This enables us to create new variables as per the number of levels in a categorical variable. (for example, if there are n number of categories in a categorical attribute, n new dummy variables are generated).

To split our dataset, we first find a vector holding all the categorical headers. The same is found by counting the number of unique values under a particular header. The threshold being 10, any variable with less than unique values, has been put in as a categorical variable.

```
[18] categorical_val = []
     continous_val = []
     for column in heart_data.columns:
         if len(heart_data[column].unique()) <= 10:
             categorical_val.append(column)
         else:
             continous_val.append(column)
     categorical_val.remove('target')
```

```
[19] dataset = pd.get_dummies(heart_data, columns = categorical_val)
```

| sex_0 | sex_1 | cp_0 | cp_1 | cp_2 | cp_3 | fbs_0 | fbs_1 | restecg_0 | restecg_1 | restecg_2 | exang_0 | exang_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Fig 8: The code and result of creating dummy variables from the categorical variables

In the table above, we have a snippet of the dummy variables and their values in our dataset for the categorical vectors. The split into the dummy variables is shown in the table below.

| **Original Variable in Dataset** | **Dummy Variables Introduced** |
|---|---|
| sex | sex_0, sex_1 |
| cp | cp_0, cp_1, cp_2, cp_3 |
| fbs | fbs_0, fbs_1 |
| restecg | restecg_0, restecg_1, restecg_2 |
| exang | exang_0, exang_1 |
| slope | slope_0, slope_1, slope_2 |
| ca | ca_0, ca_1, ca_2, ca_3, ca_4 |
| thal | thal_0, thal_1, thal_2 |

## Normalisation of the data

The plot of the various variables as mentioned in the dataset is non normalised. For variables like age, thalach and oldpeak, the spread of the data is way too much. This creates an inefficient model as it may have a considerable amount of effect on the correlation coefficients. Hence we normalise five of our independent variables: age, thalach, oldpeak, trestbps and chol.

The corresponding formula used is:

$$age\ (normalised) = \frac{age\ (original) - sample\ mean\ of\ age}{sample\ standard\ deviation\ of\ age}$$

$$thalach\ (normalised) = \frac{thalach\ (original) - sample\ mean\ of\ thalach}{sample\ standard\ deviation\ of\ thalach}$$

$$oldpeak\ (normalised) = \frac{oldpeak\ (original) - sample\ mean\ of\ oldpeak}{sample\ standard\ deviation\ of\ oldpeak}$$

$$trestbps\ (normalised) = \frac{trestbps\ (original) - sample\ mean\ of\ trestbps}{sample\ standard\ deviation\ of\ trestbps}$$

$$chol\ (normalised) = \frac{chol\ (original) - sample\ mean\ of\ chol}{sample\ standard\ deviation\ of\ chol}$$

The post normalisation plots of the five variables are as given below:

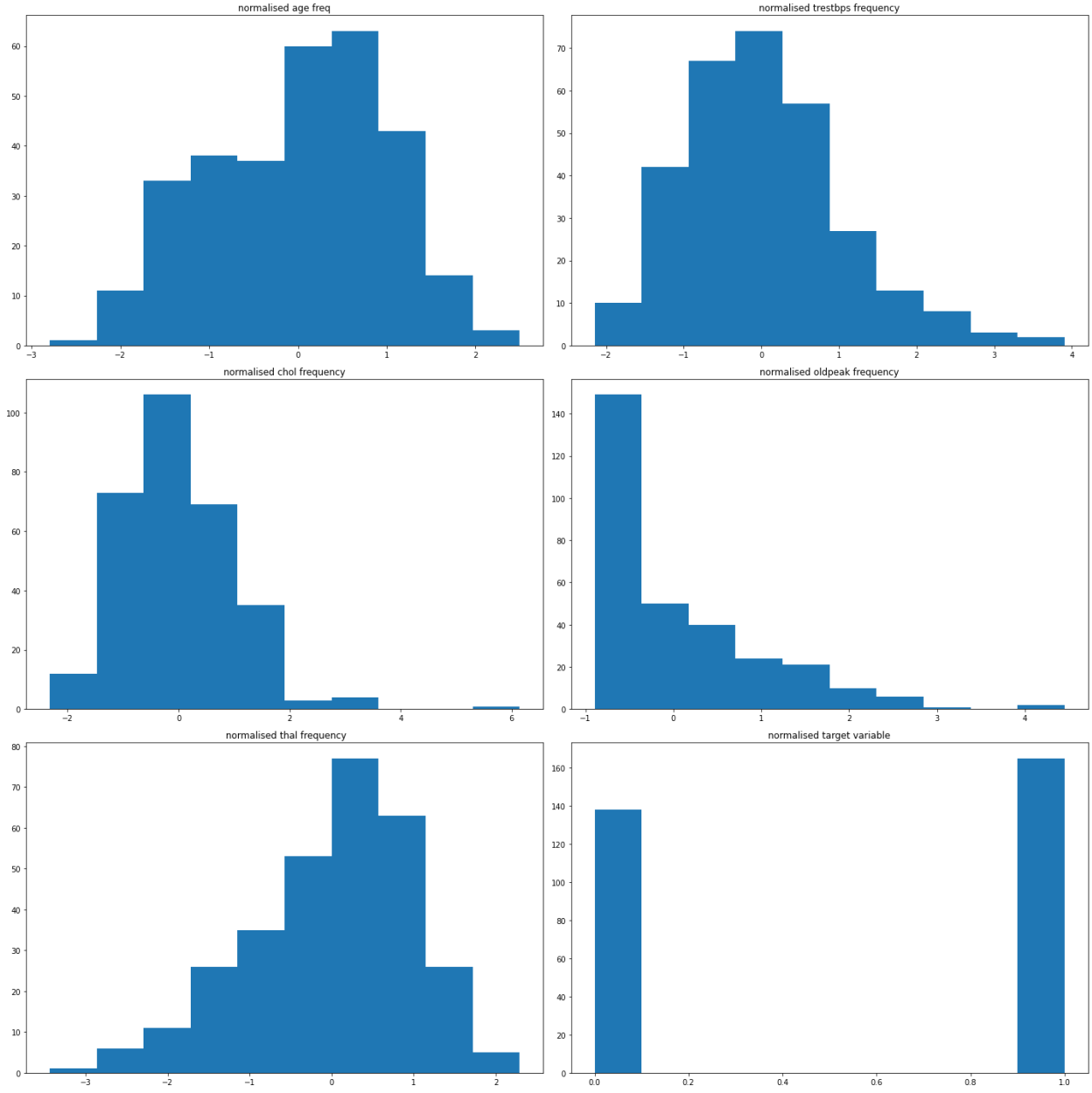


Fig 9: Plot of the dataset variables post normalization

After all the above-mentioned data processing, the next step is to fix the approximation level of the data. For the same, we shall use a 2 floating-point approximation on the complete dataset.

```
[23] pd.set_option("display.float", "{:.2f}".format)
     dataset.head()
```

Fig 10: Code Snippet for setting precision to 2

The updated descriptive statistics of the dataset are as follows:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| age | 303.000000 | 0.000000 | 1.001654 | -2.797624 | -0.757280 | 0.069886 | 0.731619 | 2.496240 |
| trestbps | 303.000000 | -0.000000 | 1.001654 | -2.148802 | -0.663867 | -0.092738 | 0.478391 | 3.905165 |
| chol | 303.000000 | -0.000000 | 1.001654 | -2.324160 | -0.681494 | -0.121055 | 0.545674 | 6.140401 |
| thalach | 303.000000 | -0.000000 | 1.001654 | -3.439267 | -0.706111 | 0.146634 | 0.715131 | 2.289429 |
| oldpeak | 303.000000 | -0.000000 | 1.001654 | -0.896862 | -0.896862 | -0.206705 | 0.483451 | 4.451851 |
| target | 303.000000 | 0.544554 | 0.498835 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| sex_0 | 303.000000 | 0.316832 | 0.466011 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| sex_1 | 303.000000 | 0.683168 | 0.466011 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| cp_0 | 303.000000 | 0.471947 | 0.500038 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| cp_1 | 303.000000 | 0.165017 | 0.371809 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| cp_2 | 303.000000 | 0.287129 | 0.453171 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| cp_3 | 303.000000 | 0.075908 | 0.265288 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| fbs_0 | 303.000000 | 0.851485 | 0.356198 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| fbs_1 | 303.000000 | 0.148515 | 0.356198 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| restecg_0 | 303.000000 | 0.485149 | 0.500606 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| restecg_1 | 303.000000 | 0.501650 | 0.500824 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| restecg_2 | 303.000000 | 0.013201 | 0.114325 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| exang_0 | 303.000000 | 0.673267 | 0.469794 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| exang_1 | 303.000000 | 0.326733 | 0.469794 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| slope_0 | 303.000000 | 0.069307 | 0.254395 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| slope_1 | 303.000000 | 0.462046 | 0.499382 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| slope_2 | 303.000000 | 0.468647 | 0.499842 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| ca_0 | 303.000000 | 0.577558 | 0.494765 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| ca_1 | 303.000000 | 0.214521 | 0.411169 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| ca_2 | 303.000000 | 0.125413 | 0.331734 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| ca_3 | 303.000000 | 0.066007 | 0.248704 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| ca_4 | 303.000000 | 0.016502 | 0.127605 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| thal_0 | 303.000000 | 0.006601 | 0.081110 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| thal_1 | 303.000000 | 0.059406 | 0.236774 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| thal_2 | 303.000000 | 0.547855 | 0.498528 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 |
| thal_3 | 303.000000 | 0.386139 | 0.487668 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |

Fig 11: Descriptive Statistics of the dataset parameters post-processing

The updated correlation of each variable (after all preprocessing) with our target is shown below:

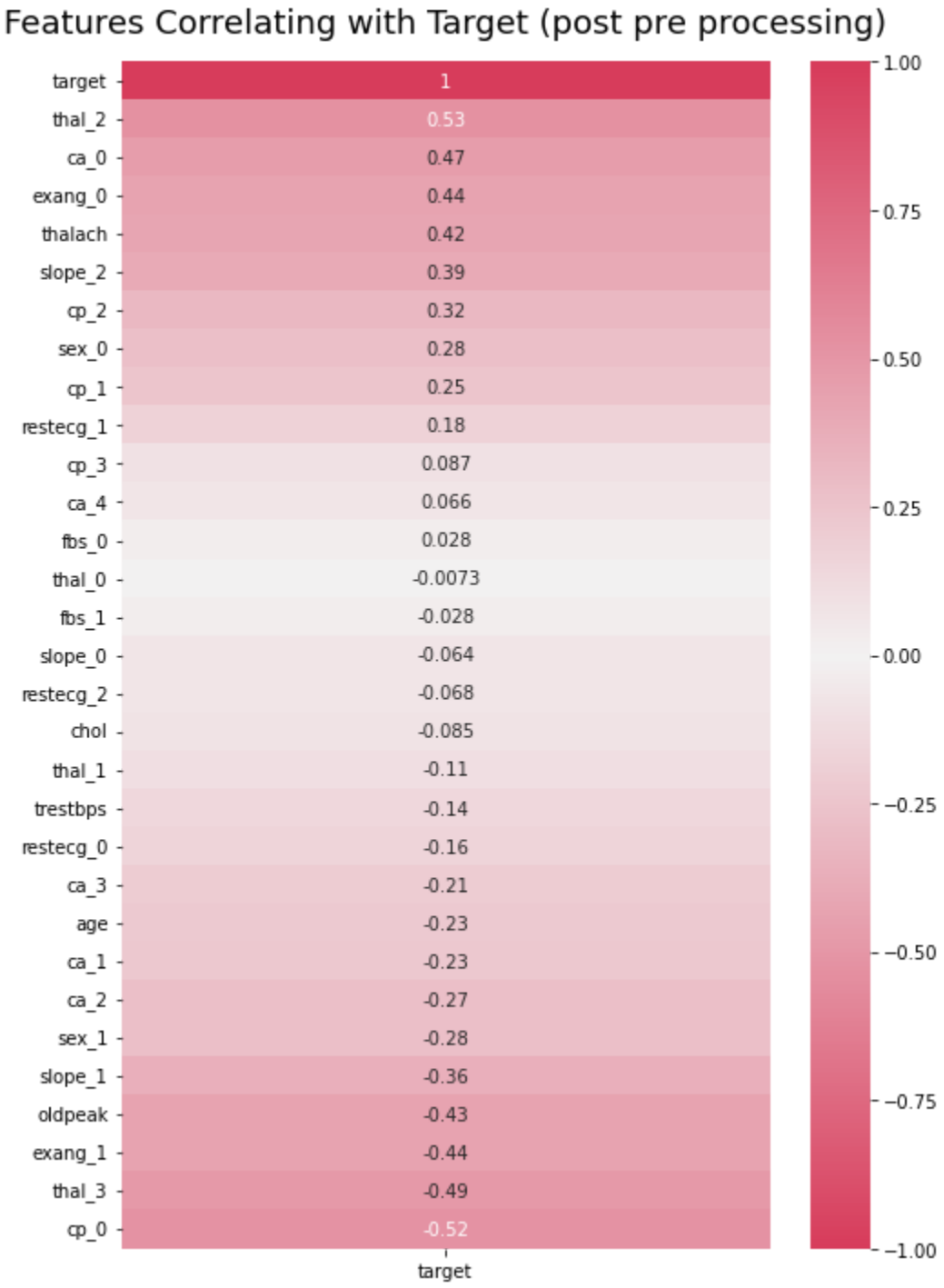


Fig 12: Heat Map analysing the correlation between the post-processed parameters and target

# Seed of Randomness

All Machine Learning models are based on training the models on certain parts of a dataset and then testing it on the remaining data points of the same dataset. For the same, we ought to split our data into a train set and a test set. For the same, we use the train_test_split() function of the sklearn library.

```
sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None,
random_state=None, shuffle=True, stratify=None)
```

Fig 13: The function syntax of train_test_split

- **arrays*: comprises the input to the split function. This is the master dataset and must be a sequence of indexable.
- *test_size*: It is a floating-point value that represents the proportion of the dataset to be included in the test split. The default value of the same is 0.25
- *random_state*: Controls the shuffling that is to be applied to the data before applying the split. We pass a fixed integer value to the same.
- *shuffle* and *stratify* are used to determine the fashion of splitting (random stratified or normal).

For our machine learning model, we do a 70-30 split of the dataset as follows:

```
# Splitting into predictor and target variables
x_full = dataset.drop('target', axis = 1)
y_full = dataset['target']

# Train test split
x_train, x_test, y_train, y_test = train_test_split(
    x_full, y_full, test_size = 0.3, random_state = 42)

print('Train data records: %d \nTest data records: %d'
      % (x_train.shape[0], x_test.shape[0]))
```

```
Train data records: 212
Test data records: 91
```

Fig 14: The results of splitting data on our dataset

Out of the total 303 data points, 91 of them are randomly allotted to the test dataset and 212 are used for training purposes of the models we have used.

Choosing an optimal value of the random_state is necessary for our analysis. For all practical purposes, random_state can take any integral value. However, it should be the same value if we want to validate the processing over multiple runs of the code. It is a parameter that is used for initialising the internal random number generator which decides whether a particular data point enters the training dataset or the test dataset. If the random_state variable is not defined, it shall be using a randomly generated value.

For our dataset, we had tried to initially train the model on a random_state of 1. However, the results weren't very conducive. A thorough literature review highlighted the fact that the referenced research paper, though using the same dataset, was only using 270 data points for training and testing purposes. The authors had mentioned the use of some pre-processing through which they had got rid of 33 observations (approximately 10% of the dataset). It was this 10% of observations in our complete dataset that was pushing our test accuracy down significantly.

On deliberation, we found that our training accuracy was significantly better than the training frequencies on the same model by the research paper. A prominent reason for the same was thus analysed to be an unfair splitting of the data set. It was concluded that the random_seed =1 was splitting our dataset in a fashion where all the 'good' observations were being allocated to the 70% train data (which pushed the training accuracy to a better value than expected) while the 30% test data comprised the outliers and the 'bad' values which didn't perform well on the models trained on 'good' data.

No literature suggested an 'optimal value of random_state' that fits all models. Most of the online resources available and codes of machine learning algorithms suggested a value of random_state between 40 and 45 be used mostly. Hence we performed a grid search for the same.

We trained the Logistic Regression Model on our dataset with a 70:30 split on different random_state values ranging from 0 to 50 and noted the corresponding test accuracies. The random_state of 42 gave the best results on the logistic regression model with a marginal difference for random_states between 41 to 46.

The same grid search was then repeated for the SVM model and MLP classifier models, however this time for only the random_state values between 41 to 46. In both these models, 42 gave a marginally better test accuracy and thus we finalised on the random_seed value of 42 for our further model training and analysis.

# Machine Learning Algorithms

We focussed our aim on standard state-of-the-art algorithms to train our dataset and compare them to the original used in the 2019 paper by M Ganesan[3], which are as follows:

## Logistic Regression

Logistic regression is a classification algorithm, deriving the term "regression" from its close relative, linear regression. Because the classes in supervised classification issues are discrete, the algorithms' purpose is to determine the decision boundaries between them. Examples of one class are separated from those of another by decision boundaries. Decision boundaries can be complicated and nonlinear in geometric shape depending on the issue instance. Different machine learning algorithms make various assumptions about the geometry of decision boundaries in general. The assumption in logistic regression is that decision boundary are linear. That is, they are hyperplanes in a high-dimensional feature space, the dimension of which is defined only by the number of elements in a training example's feature vector.

Instead of fitting a line to the data, LR tries to fit an S-Shaped logistic function to the graph. The curve goes from 0-1. Therefore the chance of it overfitting is very low.

The figure below shows the S-shaped curve of Logistic Regression:

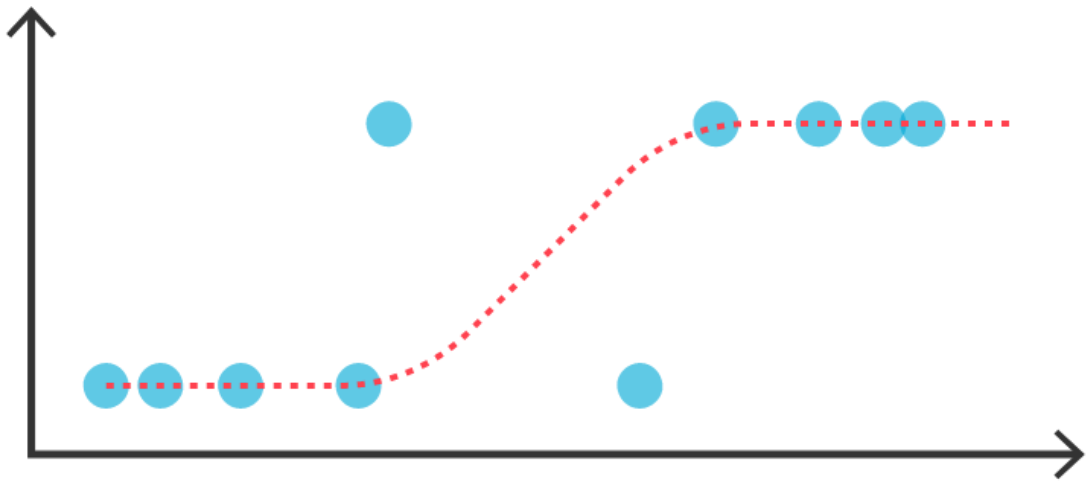

Fig 15: The Logistic Regression Plot (red) based on two distinct targets (in blue)

## XGBoost

"XGBoost stands for e**X**treme **G**radient **B**oosting", according to Tianqi Chen, the mind behind both the Gradient Boosting algorithm and XGBoost[8].

XGBoost library relies on a multiple additive regression trees model to implement its machine learning architecture. This is also called gradient boosting which is an ensemble technique where several models of the same kind are trained and voting is done to weed out errors made by some of the models. More models are added till no change occurs in terms of improvement. For example, as used in the diabetes dataset, we used the AdaBoost algorithm which is also one of the ensemble ML models. Gradient Boosting helps in ensembling as it makes new models with the sole purpose of identifying and predicting the errors of previous models and using them to make final predictions in the model. XGBoost uses parallelization to build the multiple trees it needs for its implementation.

The figure attached shows the Tree Ensemble Model as shown by Tianqi Chen in his revolutionary paper about XGBoost.

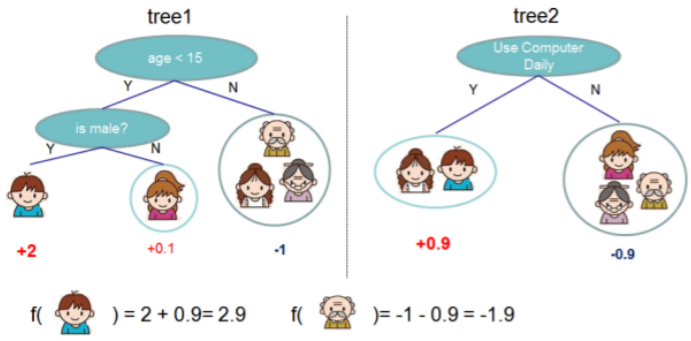


Fig 16: A graphical representation of how the XGB classifier creates multiple decision trees

It is one of the most used machine learning models in "competitive machine learning" due to its versatile data types optionality, and the extreme control of hyperparameters it allows for us to fine-tune. It can even be used for regression, classification, and rank problems.

To use the XGBoost classifier in Python, we need to execute the following code:

```
[ ]  from xgboost import XGBClassifier
```

```
[ ]  XGB_clf =  XGBClassifier()
     XGB_clf.fit(X_train, Y_train)
     XGB_clf.score(X_test,Y_test)
```

Fig 17: The code for implementation of XGBoost classifier in Python

## Multi-Layer Perceptron

Multilayer Perceptron is a deep learning model. Deep Learning is the subset of Machine Learning inspired by the structure of the human brain wherein it tries to form “neural networks” which solves a problem layer by layer. They are non-linear decision-making tools that help us classify information. Used primarily to group unlabeled data by identifying similarities between them or classify and then predict labelled data, MLP is a feedforward artificial neural network. It can be used to take in multiple inputs to generate multiple outputs in just one processing unit as it removes the boundary of linearly separable problems. Any MLP algorithm has input, hidden, and output nodes, where all nodes are interconnected to one another.

MLP is built on the principle of Perceptron which is a binary classification linear decision boundary model. It works by finding the hyperplane that divides the points into two such halves that they are clubbed together as much as possible. It used Stochastic Gradient Descent to learn the weights that help in minimizing the cost and increasing accuracy. Now due to its linearly separable ideology, it drew criticism as it could not solve non-linear situations such as the representation of the XOR gate. Therefore perceptron was virtually useless for non-linear data.

To solve this glaring issue, MLP was developed where input and output layers were connected non-linearly. MLP is a feed-forward algorithm with backpropagation ability. The same is illustrated in the diagram below[9]:

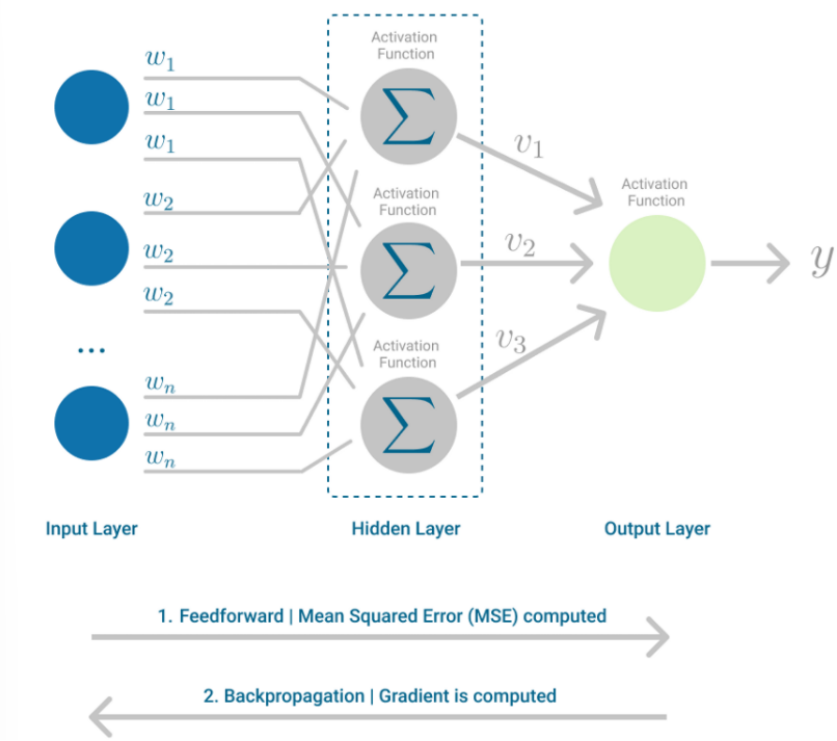


Fig 18: Inner working of the Multi Layer Perceptron Model

## K Nearest Neighbours

K-nearest neighbours is an algorithm used in statistics that can be used for classification as well as for regression. It is a supervised machine learning algorithm. The algorithm assumes that similar things exist in close proximity.

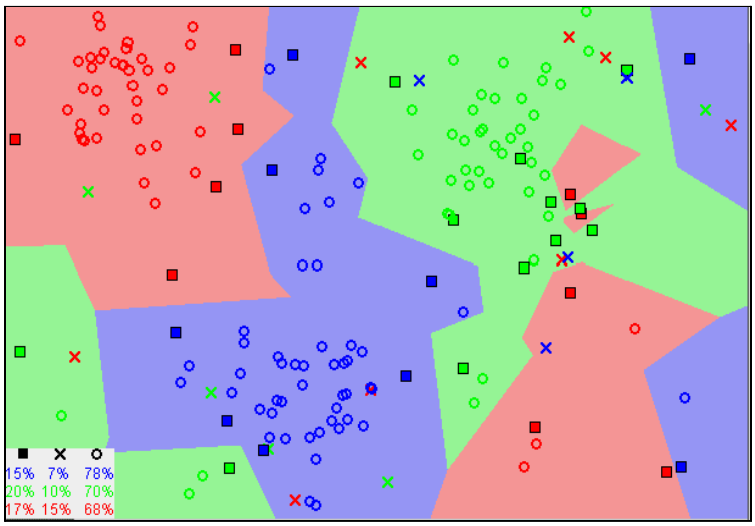


Fig 19: Image showing how similar data points typically exist close to each other

To calculate the distance between two points, the Euclidean distance between them is taken, though Hamming distance can be taken for discrete variables like text classification. Both for classification and regression, assigning weights to the contributions of the neighbours becomes useful. The best choice of 'k' depends upon the data. The larger the value of *k*, the less is the effect of the noise on the classification, at the expense of less distinct boundaries between classes. Several hyperparameter optimization techniques are used to select a good *k*. The accuracy of the algorithm can be severely hampered by the presence of noisy or irrelevant features, or if the scales of the features are not consistent with their importance to the point.

For high-dimensional data, dimension reduction is usually performed prior to applying the k-NN algorithm. This is done to ensure that the curse of dimensionality is avoided. It basically means that the Euclidean distance is unhelpful in higher dimensions, as most vectors are almost equidistant to the search query vector.

Decision boundaries are implicitly computed while using the k-nearest neighbour algorithm. Even so, it is possible to compute the decision boundary explicitly, so that the computational complexity is a function of the boundary complexity.

The algorithm is extremely simple and easy to implement. There's no need to build a model, fine-tune several parameters, or make additional adjustments. The algorithm is versatile, as it can be used for classification and regression. It is widely used. In fact, popular recommendation systems are based on this algorithm. The model analyzes your preferences with respect to several other peoples', and recommends the next content based on the preference of your neighbours.

## Random Forests

A decision tree is a sort of model that may be used for classification as well as regression. Trees respond to consecutive questions by leading us down a certain branch of the tree based on the response. The model responds to "if this, then that" circumstances, resulting in a certain outcome. Below figure is a great example of how decision trees function

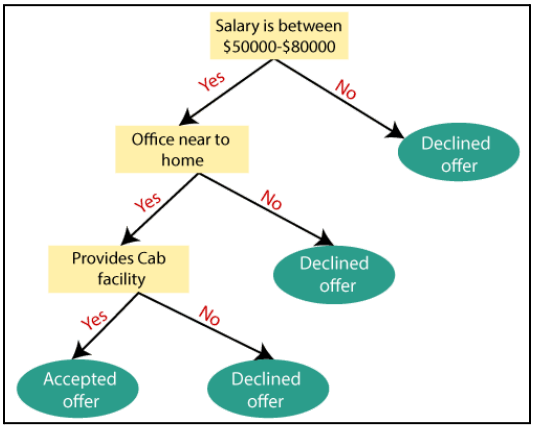


Fig 20: Example of a Decision Tree algorithm

Decision trees have several advantages, such as they're easy to interpret, can handle both categorical data as well as numerical data, perform well on large datasets and are extremely fast. But they also have limitations of their own; namely overfitting if the tree is too deep and the fact that even the most optimum decision tree algorithm does not take into account the factor of global optimum and only considers the best possible outcome in the next step. Hence there can be cases when we don't reach our optimum answer.

To solve these disadvantages, Random Forests were introduced. It's basically a collection of decision trees whose individual results are pooled together. They are so strong because of their ability to reduce overfitting without significantly increasing error due to bias.
Random Forests can minimize variance by training on several samples of data. Another way they do the same is to use a random subset of characteristics in each decision tree. Since Random Forest is just a collection of decision trees, it ends up studying all the characteristics in totality. Error due to bias and error due to variance will be reduced as a result of the addition of so many characteristics. Our forest's foundation trees might become extremely linked if characteristics weren't picked at random. Because a few features may be especially predictive, several of the foundation trees will employ the same features.

## Support Vector Machine

Support Vector Machine is a type of supervised machine learning model that is used for regression and classification. It peaked in its popularity because of its high accuracy and fast computational speed. Like KNN, it too divides the dataset according to a hyperplane but takes the help of support vectors to do so. Support vectors are the vectors parallel to the hyperplane that pass through the closest points to the hyperplane that is not on the hyperplane. The attached figure illustrates the same.

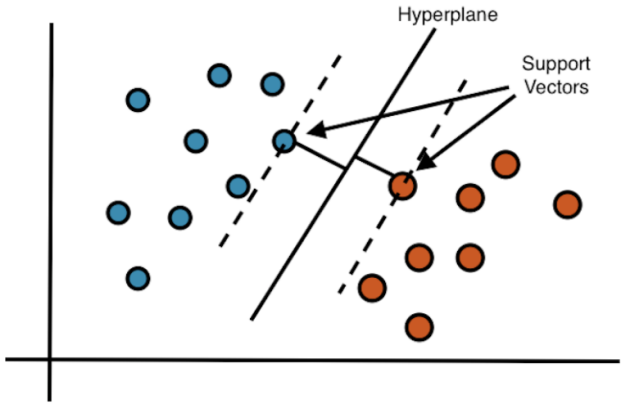


Fig 21: An illustration of the SVM Model

# Metrics

After training our machine, we saw an uptake in the accuracy and precision of the models. To measure the accuracy of our model, we employed several metrics which are as follows:

1. *Accuracy*: The percentage of accurately identified scenarios is known as accuracy. It is one of the most often used classification performance measures, with a value closer to 100 indicating better classification results.

$$Accuracy = \frac{True\ Positive+True\ Negative}{True\ Positive+True\ Negative+False\ Positive+False\ Negative}$$

2. *Precision*: Precision is the inherent property of the classifier to not detect any false cases. It is formulated as:

$$Precision = \frac{True\ Positive}{True\ Positive+False\ Positive}$$

3. *Recall*: This is the property of the machine to find all true samples. Defined as:

$$Recall = \frac{True\ Positive}{True\ Positive+False\ Negative}$$

4. *F-Score:* Defined as the harmonic mean of the recall and the precision

$$F - Score = 2 * \frac{(precision * recall)}{(precision + recall)}$$

# Observations

Using the aforementioned metrics, we trained our dataset on the set of 212 data points with about 91 data points as our test cases. We did a random split of 70-30 and made sure to use the knowledge learnt about random split to divide our dataset in such a way that all the erroneous dataset is divided in equal proportion in both the datasets. We trained our dataset on the standard state-of-the-art machine learning models like RF, KNN, SVM, LR, as well as new and revolutionary methodologies in the field of machine learning, such as XGB Classifier which took the world by storm in 2016. After fine-tuning our models, we were able to get appreciable accuracy, precision, recall and F - Score for all our models. As one can see from figure 22, we were able to reach an accuracy of>87.9 per cent which is a plausible feat.

| Model | Accuracy | F Score | Precision | Recall |
|---|---|---|---|---|
| XGB Classifier | 83.52 | 84.85 | 85.71 | 84 |
| RF Classifier | 81.32 | 83.17 | 82.35 | 84 |
| KNN Classifier | 86.81 | 85.81 | 87.54 | 88 |
| MLP Classifier | 87.91 | 88.89 | 88.55 | 88 |
| SVM Classifier | 87.91 | 89.72 | 89.8 | 87 |
| Logistic Regression | 85.71 | 87.38 | 84.91 | 90 |

Fig 22: A tabular representation of our models, and their scoring against the metrics as defined earlier

As stated earlier, the best results were achieved with SVM Classifier and MLP Classifier, with little to no difference between them.

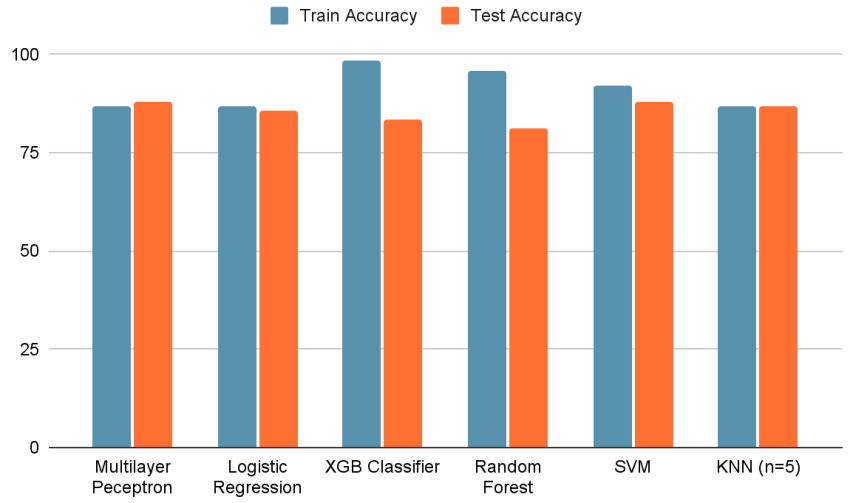


Fig 23: Representation of the test and train accuracies of various ML models on our revised dataset

# Conclusion and Discussion

When we compare the accuracy achieved by our models, that have been trained on the original dataset without any random trimming to suit our needs, we can clearly see that by introducing randomness, normalization, correlation analysis, the introduction of dummy variables, data normalisation, and descriptive analysis of cross dependence of variables, we were able to exponentially improve the accuracy of the models by 9.77% for MLP, 2.01% for Logistic Regression, 3.84% for SVM, which is a substantial increase in itself.

| Algorithm | Accuracy | F-score | Precision | Recall |
|---|---|---|---|---|
| SVM | 84.07 | 84.1 | 84.1 | 84.1 |
| SVM (revised) | 87.91 | 88.89 | 89.8 | 88 |
| LR | 83.7 | 83.7 | 83.7 | 83.7 |
| LR (revised) | 85.71 | 87.38 | 84.91 | 90 |
| MLP | 78.14 | 78.2 | 78.4 | 78.1 |
| MLP (*revised*) | 87.91 | 88.89 | 89.8 | 88 |

Fig 24: Comparison between the metrics in referenced paper and our models

Precision confirms better classification performance. Our test results show that SVM and MLP show the best precision in all metrics that have been employed. This proves the improvement we have done on the paper "IoT based heart disease prediction and diagnosis model for healthcare using machine learning models,"[3] which relied on random removing of data to reach the published result.

Fig 25 depicts our revised models, in comparison to the models used by the paper; as is visible from the bar graph, the models that we created stand true to their worth and improve upon already available results.

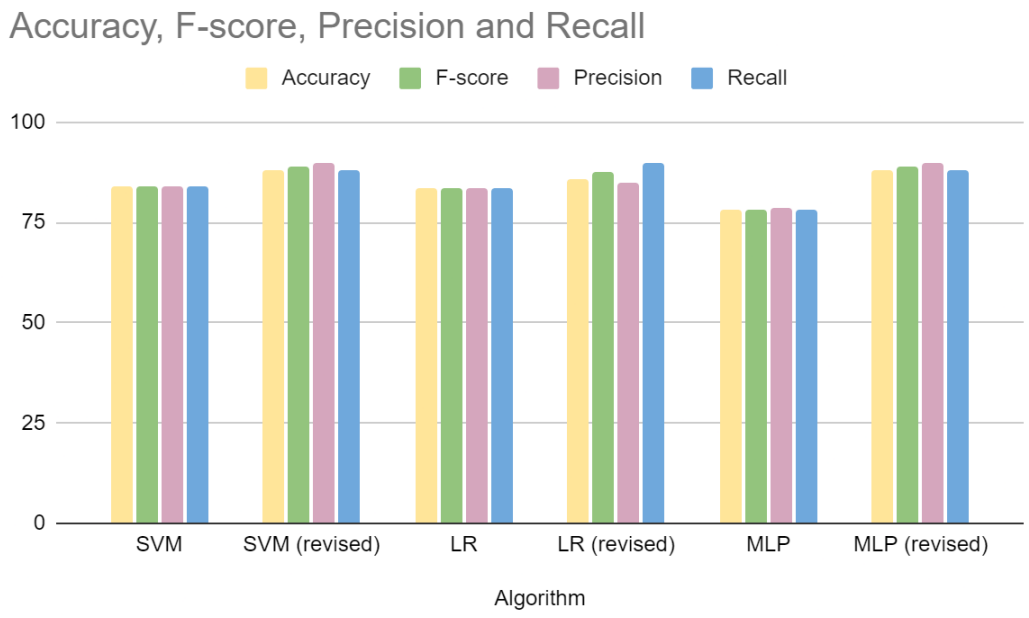


Fig 25: Comparison between the metrics in referenced paper and our models

# Future Works

The above-mentioned models have significantly improved the test-train efficiency of the data set and provide a new avenue for data scientists to implement better models for higher precision and accuracy for disease prediction. One avenue that still remains unexplored is the J48 classifier. The work under reference showed significantly good results when the J48 classifier was used for the same.

J48 is a decision tree based classifier based on the Iterative Dichotomiser 3. This algorithm builds decision trees based on a set of training data in the same way the ID3 algorithm does, by using the concept of information entropy. The C4.5 method finds the data characteristic that most effectively divides its set of samples into subsets, enriched in one class or the other, at each node of the tree. The normalised information gain, which is determined from the difference in entropy, is used as the splitting criterion. To make the decision, the attribute with the highest normalised information gain is picked.

We weren't able to implement the same for our research owing to the lack of a versatile and open-sourced library that satisfies the precision and accuracy of the model. The only available and reliable source code for the same could be found in the Java environment which wasn't a viable option for implementation in our Python-based repository. In the future, one might develop on the same and come up with a more reliable J48 Python library which can be used to test the classifier further.

The dataset we have been using through the entire discussion (UCI Repository) comprises only 303 observation points. The model we use is trained on 13 independent parameters which have been further subdivided (to introduce dummy variables) into 31 parameters. For such a vast model, a dataset of just 303 data points is not very appreciable. It further partitions into 212 training data points and 91 test observation points. In the future, we can work on better datasets that are of equal authenticity. We tried to search for other datasets too but found them to be heavily filled with outliers. At this stage, where outlier detection is not directly possible, we weren't able to proceed further. Hence, the avenue for a more refined dataset with a larger observation set would be a good metric to test the models

The domain of IoT base sensing devices is prone to its own set of faults and faulty reporting of observations. It is thus important that we train our models on more real-time datasets acquired from authentic IoT sensors to ensure that our models are fault-tolerant and practical.

# References


1) Ratner, B. The correlation coefficient: Its values range between +1/−1, or do they?. J Target Meas Anal Mark 17, 139–142 (2009). https://doi.org/10.1057/jt.2009.5
2) Pett, M. A., Lackey, N. R., & Sullivan, J. J. (2003). Making sense of factor analysis: The use of factor analysis for instrument development in health care research. Thousand Oaks, CA: Sage Publications.
3) M. Ganesan and N. Sivakumar, "IoT based heart disease prediction and diagnosis model for healthcare using machine learning models," 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 2019, pp. 1-5, DOI: 10.1109/ICSCAN.2019.8878850.
4) UCI Dataset (https://archive.ics.uci.edu/ml/datasets/Heart+Disease)
5) Ed-daoudy, Abderrahmane & Maalmi, Khalil. (2019). A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment. Journal of Big Data. 6. 10.1186/s40537-019-0271-7.
6) T. C. Quinn, “The Johns Hopkins Center for Global Health: transcending borders for world health,” Academic Medicine, vol. 83, no.2, pp. 134-142, 2008.
7) K. Shailaja, B. Seetharamulu and M. A. Jabbar, "Machine Learning in Healthcare: A Review," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 910-914, DOI: 10.1109/ICECA.2018.8474918.
8) Chen, T., & Guestrin, C. (2016). XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
9) Multilayer Perceptron- Towards Data Science
10) Kakria P, Tripathi N.K, Kitipawang P, “A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors”, Int. J. Telemed. Appl. (2015).
11) N. Sharma, & A. Singh, “Diabetes detection and prediction using machine learning/IoT: A survey,” In International Conference on Advanced Informatics for Computing Research, pp. 471-479, 2018.
12) Yahaya, L.; Oye, N.D.; Garba, E.J. A Comprehensive Review on Heart Disease Prediction Using Data Mining and Machine Learning Techniques. Am. J. Artif. Intell. 2020, 4, 20.
13) Shinozaki, A. Electronic Medical Records and Machine Learning in Approaches to Drug Development. In Artificial Intelligence in Oncology Drug Discovery and Development; IntechOpen: London, UK, 2020; p. 51.
14) Manogaran G, Lopez D. A survey of big data architectures and machine learning algorithms in healthcare. Int J Biomed Eng Technol. 2017;25(2–4):182–211.
15) Mocnik, F.-B.; Raposo, P.; Feringa, W.; Kraak, M.-J.; Köbben, B. Epidemics and pandemics in maps–the case of COVID-19. J. Maps 2020, 16, 144–152.
16) Ibrahim, N.; Akhir, N.S.M.; Hassan, F.H. Predictive analysis effectiveness in determining the epidemic disease infected area. AIP Conf. Proc. 2017, 1891, 20064.