

1. INTRODUCTION

1.1 Introduction

Heart failure is a term covering any disorder of the heart. Heart failure have become a major concern to deal with as studies show that the number of deaths due to heart failure have increased significantly over the past few decades in India, in fact it has become the leading cause of death in India.

A study shows that from 1990 to 2016 the death rate due to heart failures have increased around 34 per cent from 155.7 to 209.1 deaths per one lakh population in India.

Thus preventing Heart failures has become more than necessary. Good data-driven systems for predicting heart failures can improve the entire research and prevention process, making sure that more people can live healthy lives. This is where Machine Learning comes into play. Machine Learning helps in predicting the Heart failures, and the predictions made are quite accurate.

Classification algorithms are very important category of supervised machine learning algorithms. These algorithms require a very large training set. These training data sets are consisting of many features or attributes which describe the individual sample. Since we are doing supervised learning algorithm. All the training set are labelled correctly. The classification algorithms such as decision trees, Gaussian Naïve Bayes, Random Forest, K-nearest Neighbours, K-Means and support vector machines (SVM), develop model with these data with many different parameters. When we have a new unlabelled sample, we can use the model to predict the label of the new sample. These techniques are used for disease diagnosis to help doctor to effectively label the new case.

1.2 Existing System

Nowadays, decisions are made by doctors based on their experience and knowledge. This practice may lead to errors, consume a lot of time and excessive medical costs which affects the quality of service provided to patients.

Disadvantages:

1. Doesn't generate accurate and efficient results
2. Computation time is very high
3. Difficulty in maintenance of patient records
4. Lacking of accuracy may result in lack of efficient further treatment

1.3 Proposed System

We proposed to develop a system which will help practitioners to predict heart failure based on some attributes like smoke, high blood pressure, diabetes and so on. So, there is a need for developing a decision system which will help practitioners to predict the heart failure in an easier way, which can offer prediction about the heart condition of patient so that further treatment can be made effectively. This proposed system not only accurately predicts heart disease but also reduces time for prediction. The machine learning algorithms like decision tree, random forest, Naive Bayes, K Nearest Neighbours have proven to be most accurate & reliable and hence, used in this project.

Advantages:

1. Generates accurate and efficient results
2. Computation time is greatly reduced
3. Reduces manual work
4. Efficient further treatment
5. Automated prediction

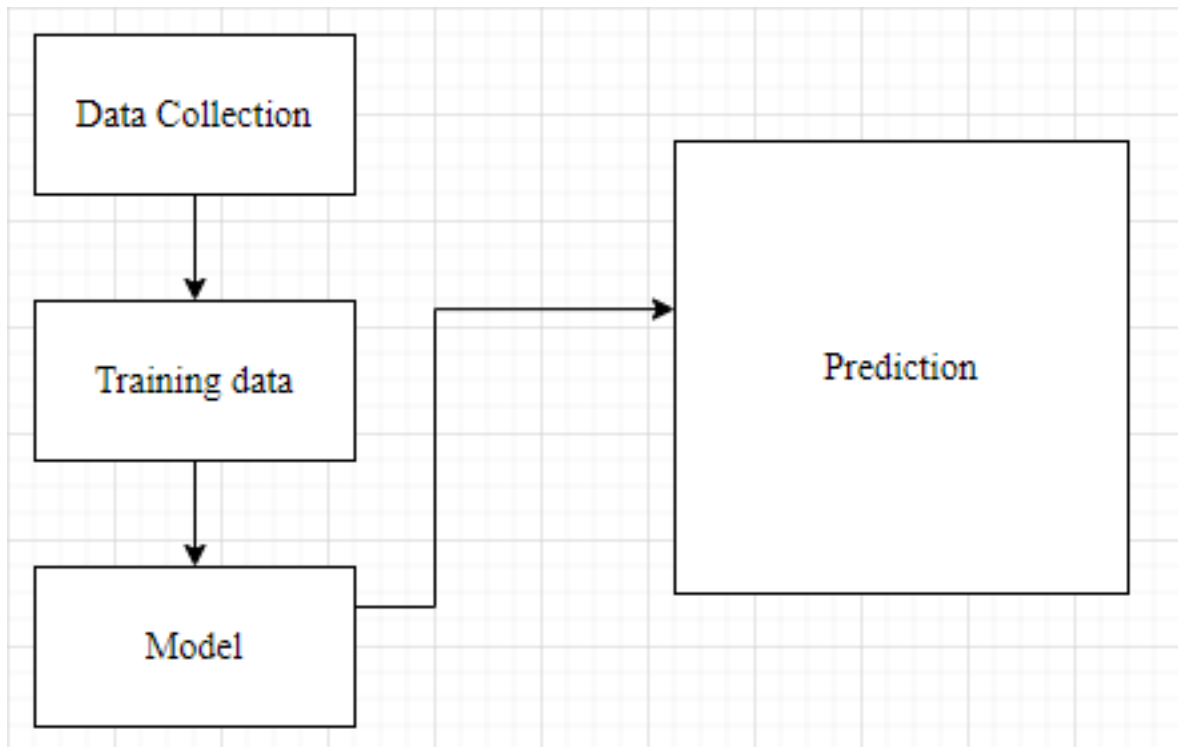


Fig 1.3.1: Proposed System

1.4. System Requirements

1.4.1 Hardware Requirements:

- System Type : intel®core™i7-7500UCPU@2.70gh
- Cache memory : 4MB(Megabyte)
- RAM : 8 gigabyte (GB)

1.4.2 Software Requirements:

- Operating System : Windows 10 Home, 64 bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, IDLE

2. LITERATURE SURVEY

2.1 Machine Learning

Machine learning is one of the applications of artificial intelligence (AI) that provides computers, the ability to learn automatically and improve from experience instead of explicitly programmed. It focuses on developing computer programs that can access data and use it to learn from themselves. The main aim is to allow computers to learn automatically without human intervention and also adjust actions accordingly.

2.2 Some machine learning methods

Machine learning algorithms are often categorized as supervised and unsupervised.

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data.
- **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training – typically a small amount of labelled data and a large amount of unlabelled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labelled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabelled data generally doesn't require additional resources.

- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best. This is known as the reinforcement signal.

2.3 Applications of machine learning

1. Virtual Personal Assistants
2. Predictions while Commuting
3. Videos Surveillance
4. Social Media Services
5. Email Spam and Malware Filtering
6. Online Customer Support
7. Search Engine Result Refining
8. Product Recommendations
9. Online Fraud Detection

2.4 Prevalence of Heart Failure

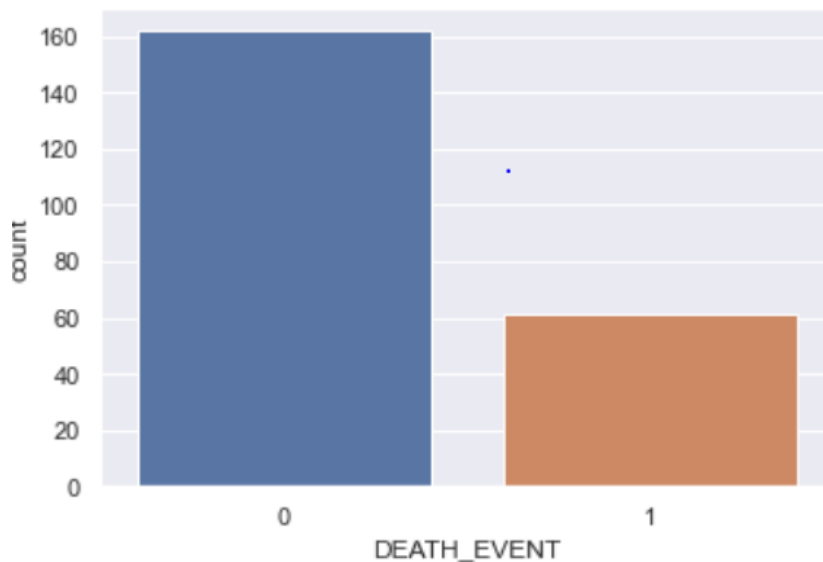


Fig:2.4.1 Death Event

2.5 Importance of machine learning in healthcare

The importance of machine learning in healthcare is increasing because of its ability to process huge datasets efficiently beyond the range of human capability, and then dependably convert analysis of that data into clinical insights that assist physicians in planning and providing care, which ultimately leads to better outcomes, reduces the costs of care, and increases patients satisfaction. Using these types of advanced analytics, we can provide better information to doctors at the point of patient care.

2.6 Implementation of machine learning using Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse, Anaconda which are particularly useful when managing larger collections of Python files.

Python was designed for its readability. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

In the older days, people used to perform Machine Learning tasks manually by coding all the algorithms and mathematical and statistical formula. This made the process time consuming,

tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Machine Learning are:

- Numpy
- Scipy
- Scikit-learn
- Theano
- TensorFlow
- Keras
- PyTorch
- Pandas
- Matplotlib

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

Scikit-learn is one of the most popular Machine Learning libraries for classical Machine Learning algorithms. It is built on top of two basic Python libraries, NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with Machine Learning.

Theano is a popular python library that is used to define, evaluate and optimize mathematical expressions involving multi-dimensional arrays in an efficient manner. It is achieved by

optimizing the utilization of CPU and GPU. It is extensively used for unit-testing and self-verification to detect and diagnose different types of errors. Theano is a very powerful library that has been used in large-scale computationally intensive scientific projects for a long time but is simple and approachable enough to be used by individuals for their own projects.

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library which is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that supports on Computer Vision, Natural Language Processing(NLP) and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, histogram, error charts, bar chats, etc.

2.7 Machine learning products

Madni, H. A., Anwar, Z., & Shah, Used many Data mining techniques and applications for predicting chronic kidney disease. Highest accuracy is 97.72% for roughsets and back propogation.

Radha, N., and S. Ramya compared may algorithms like support vector machine, Naïve Bayes, k- nearest neighbors, decision tree and the highest accuracy is 99.45 for Decision Tree.

3. SYSTEM ANALYSIS

3.1 Scope of the project

This project mainly focuses on the how the heart failure can be predicted by using some related information. It mainly depends on the symptoms. The scope of this site is the user can able to predicting the stage in which he/she lies.

3.2 Analysis

The dataset used in this project is Heart failure prediction dataset which is obtained from Kaggle.com. The dataset contains 13 attributes which are used to predict the heart failure such as:

age – age

hbp - high_blood_pressure

sex - sex

smoke - smoking

anaemia - anaemia

creatinine - 'creatinine_phosphokinase

dia - diabetes'

eject_fraction - 'ejection_fraction'

platelets- platele

serum_c-'serum_creatinine'

serum_s-'serum_sodium'

'time'-time

DEATH_EVENT'- DEATH_EVENT'

| age | anaemia | creatinine | diabetes | ejection_f | high_bloo | platelets | serum_cr | serum_so | sex | smoking | time | DEATH_EVENT |
|-----|---------|------------|----------|------------|-----------|-----------|----------|----------|-----|---------|------|-------------|
| 75 | 0 | 582 | 0 | 20 | 1 | 265000 | 1.9 | 130 | 1 | 0 | 4 | 1 |
| 55 | 0 | 7861 | 0 | 38 | 0 | 263358 | 1.1 | 136 | 1 | 0 | 6 | 1 |
| 65 | 0 | 146 | 0 | 20 | 0 | 162000 | 1.3 | 129 | 1 | 1 | 7 | 1 |
| 50 | 1 | 111 | 0 | 20 | 0 | 210000 | 1.9 | 137 | 1 | 0 | 7 | 1 |
| 65 | 1 | 160 | 1 | 20 | 0 | 327000 | 2.7 | 116 | 0 | 0 | 8 | 1 |
| 90 | 1 | 47 | 0 | 40 | 1 | 204000 | 2.1 | 132 | 1 | 1 | 8 | 1 |
| 75 | 1 | 246 | 0 | 15 | 0 | 127000 | 1.2 | 137 | 1 | 0 | 10 | 1 |
| 60 | 1 | 315 | 1 | 60 | 0 | 454000 | 1.1 | 131 | 1 | 1 | 10 | 1 |
| 65 | 0 | 157 | 0 | 65 | 0 | 263358 | 1.5 | 138 | 0 | 0 | 10 | 1 |
| 80 | 1 | 123 | 0 | 35 | 1 | 388000 | 9.4 | 133 | 1 | 1 | 10 | 1 |
| 75 | 1 | 81 | 0 | 38 | 1 | 368000 | 4 | 131 | 1 | 1 | 10 | 1 |
| 62 | 0 | 231 | 0 | 25 | 1 | 253000 | 0.9 | 140 | 1 | 1 | 10 | 1 |
| 45 | 1 | 981 | 0 | 30 | 0 | 136000 | 1.1 | 137 | 1 | 0 | 11 | 1 |
| 50 | 1 | 168 | 0 | 38 | 1 | 276000 | 1.1 | 137 | 1 | 0 | 11 | 1 |
| 49 | 1 | 80 | 0 | 30 | 1 | 427000 | 1 | 138 | 0 | 0 | 12 | 0 |
| 82 | 1 | 379 | 0 | 50 | 0 | 47000 | 1.3 | 136 | 1 | 0 | 13 | 1 |
| 87 | 1 | 149 | 0 | 38 | 0 | 262000 | 0.9 | 140 | 1 | 0 | 14 | 1 |
| 45 | 0 | 582 | 0 | 14 | 0 | 166000 | 0.8 | 127 | 1 | 0 | 14 | 1 |

Fig:3.1 Dataset

3.3 Data Pre-processing

Before feeding data to an algorithm we have to apply transformations to our data which is referred as pre-processing. By performing pre-processing the raw data which is not feasible for analysis is converted into clean data. In-order to achieve better results using a model in Machine Learning, data format has to be in a proper manner. The data should be in a particular format for different algorithms. For example, if we consider Random Forest algorithm it does not support null values. So that those null values have to be managed using raw data.

3.3.1 Missing values

Filling missing values is one of the pre-processing techniques. The missing values in the dataset is represented as '?' but it a non-standard missing value and it has to be converted into a standard missing value NaN. So that pandas can detect the missing values. The fig1 below is a heatmap representing the missing values. In these graph missing values are present in all the

features. We have filled that missing values using the median of the features. After filling the missing values our heat map looks like below figure.

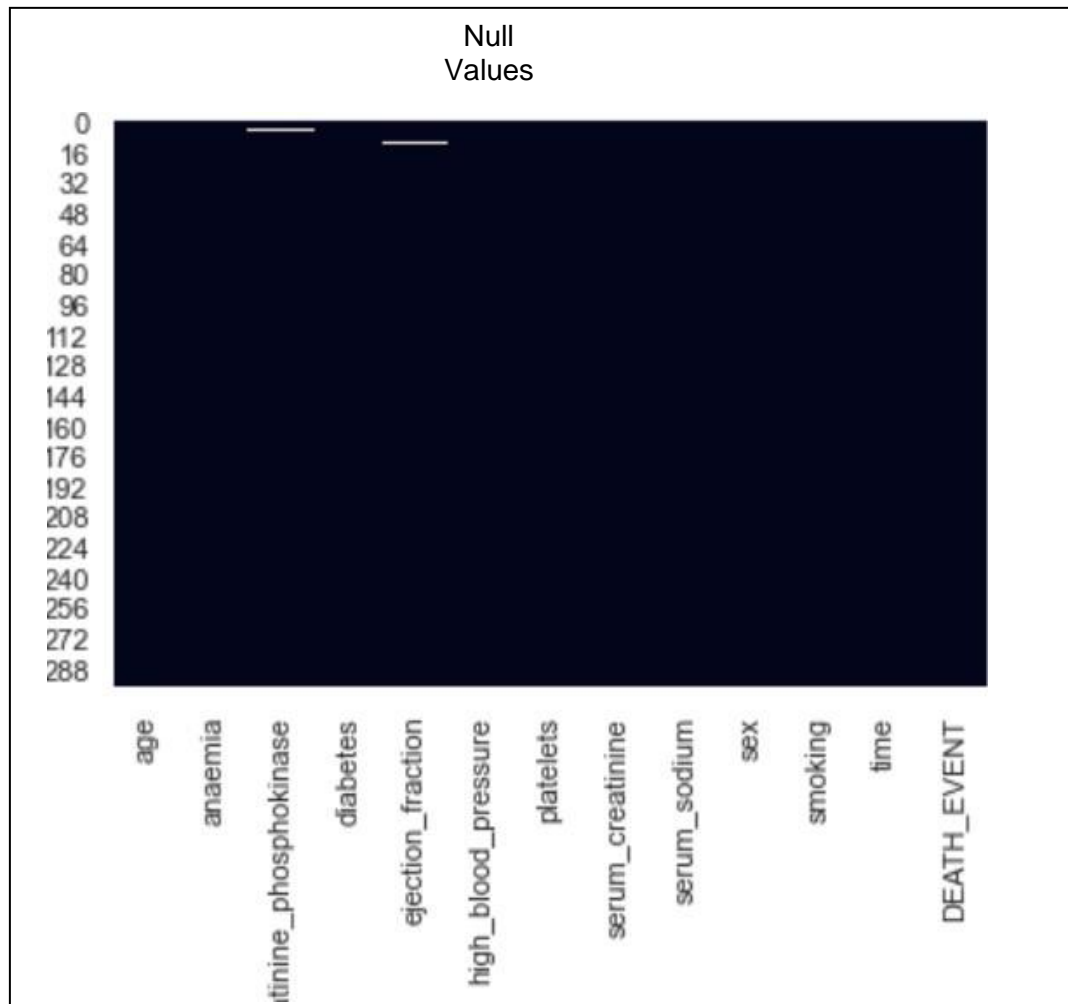


Figure:3.2 Missing data visualisation with NULL values

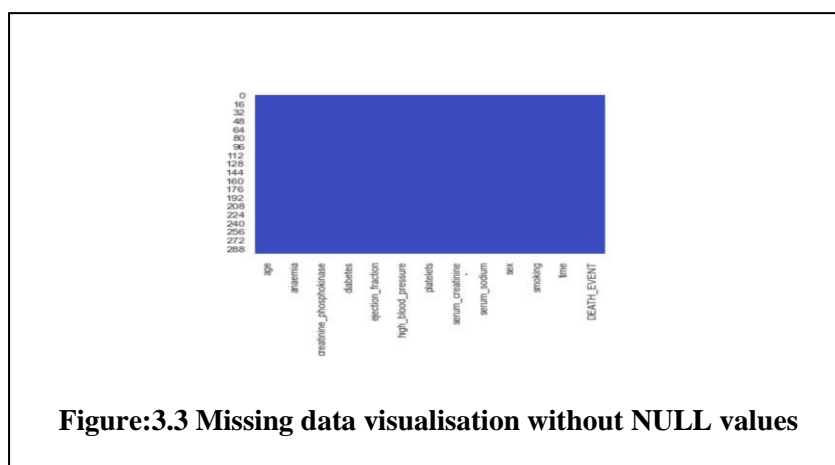


Figure:3.3 Missing data visualisation without NULL values

3.4 Removing of outliers:

An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining.

Why outlier analysis?

Most data mining methods discard outliers; noise or exceptions, however, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring one and hence, the outlier analysis becomes important in such case.

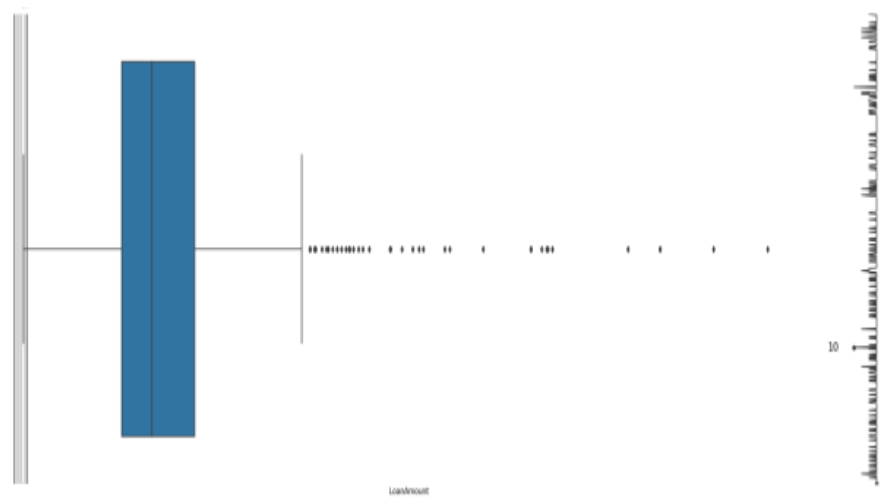


Fig 3.4 Outliers in Dataset

As there are outliers in the data better to remove those

3.5 FEATURE SELECTION: Feature Selection is process of reducing the number of input variables when developing a predictive model[10]. To reduce the computational cost of modelling and in some of the cases to increase or improve the performance of the model it is desirable to reduce the number of input variables. Feature Selection method used in this project is correlation.

3.5.1 CORRELATION: Correlation describes the linear relationship between the two continuous variables. Correlation is used when there is no identified response variable.

statistical measure that indicates the extent to which two or more variables fluctuate together. A positive correlation indicates extent to which those variables increase or decrease in parallel. A negative correlation indicates extent to which one variable increases as the other decreases.

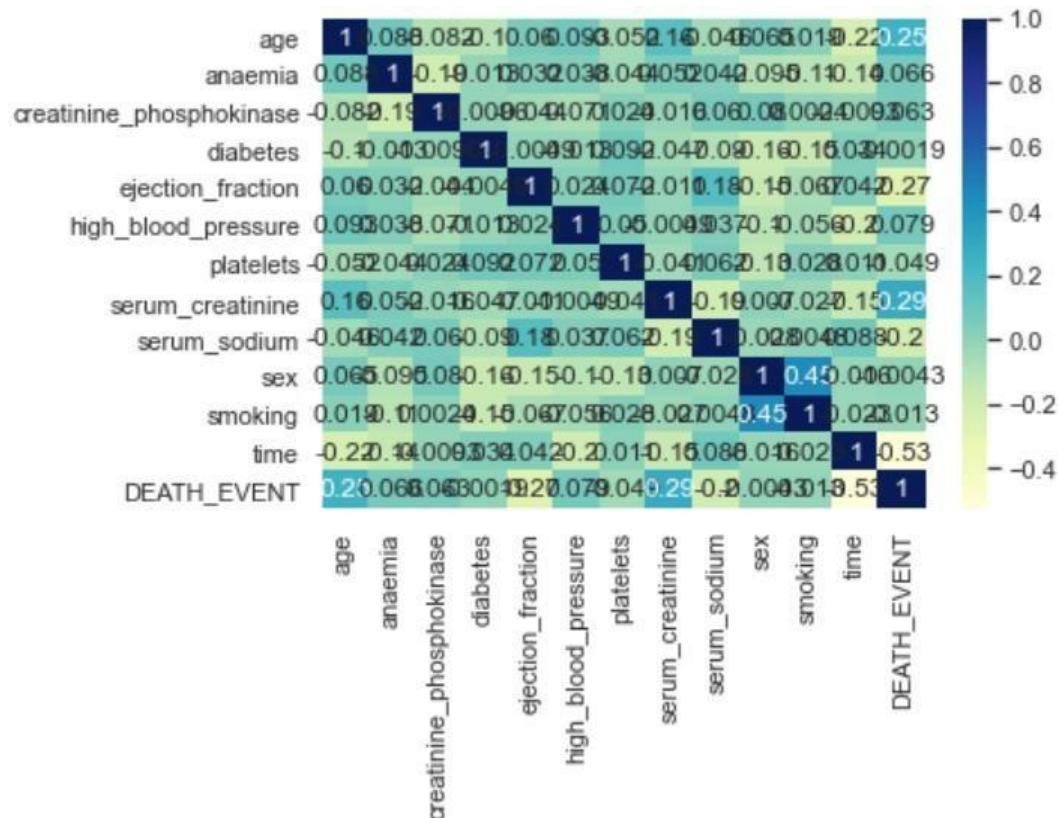


Figure:3.5 Correlation for heart failure prediction Dataset

Figure displays the correlation applied on the WBCD.

3.6 Classification

It is a process of categorising data into given classes. Its primary goal is to identify the class of our new data.

3.6.1 Machine learning algorithms for classification

Research on data mining has led to the formulation of several data mining algorithms. These algorithms can be directly used on a dataset for creating some models or to draw vital conclusions and inferences from that dataset. Some popular data mining

algorithms are Decision tree, Naïve Bayes, Logistic Regression, Random Forest Classifier, Support Vector Machine, K-Nearest Neighbours.

1. Decision Tree: Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model.

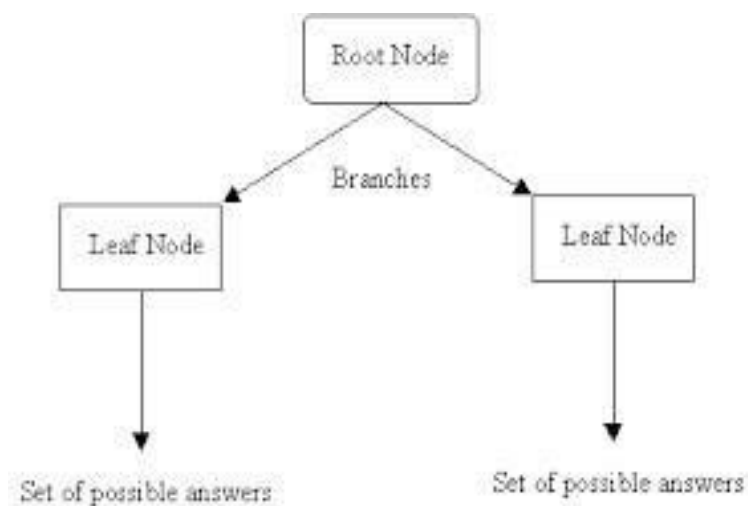


Figure:3.6 Decision Tree classifier

2. Naive Bayes (NB): It is a simple technique for constructing classifiers. It is a probabilistic classifier based on Bayes' theorem. All Naive Bayes classifiers assume that the value of any particular feature is independent of the value of any other feature, given the class variable. Bayes theorem is given as follows: $P(C|X) = P(X|C) * P(C)/P(X)$, where X is the data tuple and C is the class such that $P(X)$ is constant for all classes. Though it assumes an unrealistic condition that attribute values are conditionally independent, it performs surprisingly well on large datasets where this condition is assumed and holds.

3. Random Forest Classifier: Random Forest Classifier are an ensemble learning method (also thought of as a form of nearest neighbour predictor) for classification and regression techniques. It builds multiple decision trees and then merges them together in-order to get more accurate and stable predictions. It constructs a number of Decision trees at training time and outputs the class that is the mode of the classes output by individual trees. It also tries to minimize the problems of high variance and high bias by averaging to find a natural balance between the two extremes. Both R and Python have robust packages to implement this algorithm.

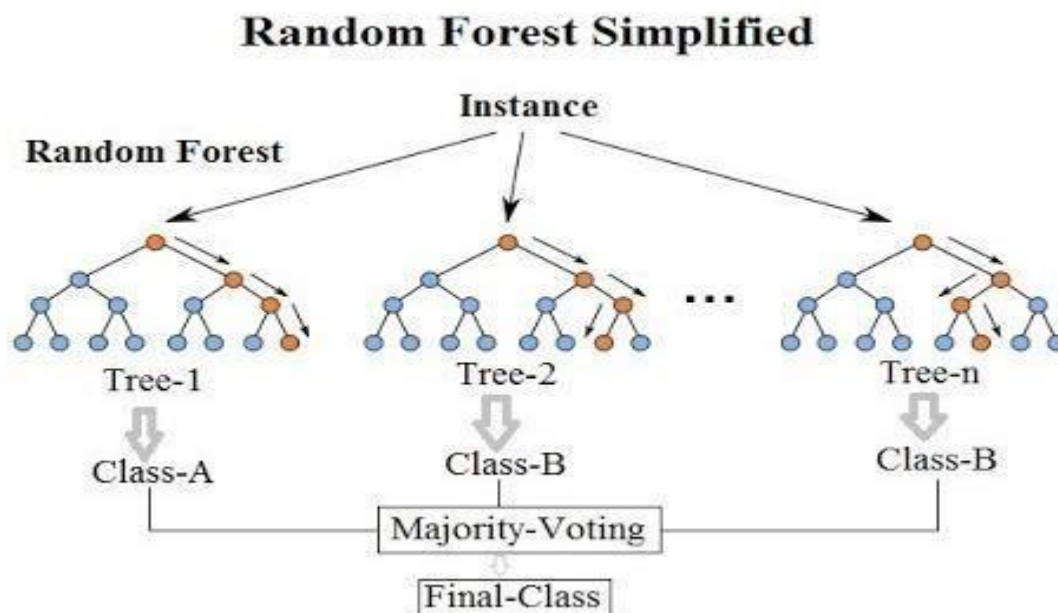


Figure: 3.7 Random Forest Classifier

4. KNN: KNN algorithm is one of the simplest classification algorithms and it is one of the most used learning algorithms. KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a dataset in which the data points are separated into several classes to predict the classification of a new sample point. A KNN algorithm uses a data and classifies new data points based on a similarity measures (e.g. distance function, error rate). Classification is done by a majority vote to its neighbours. The data is assigned to the class which has the most nearest neighbours. As we increase the number of nearest neighbours, the value of k , accuracy may increase.

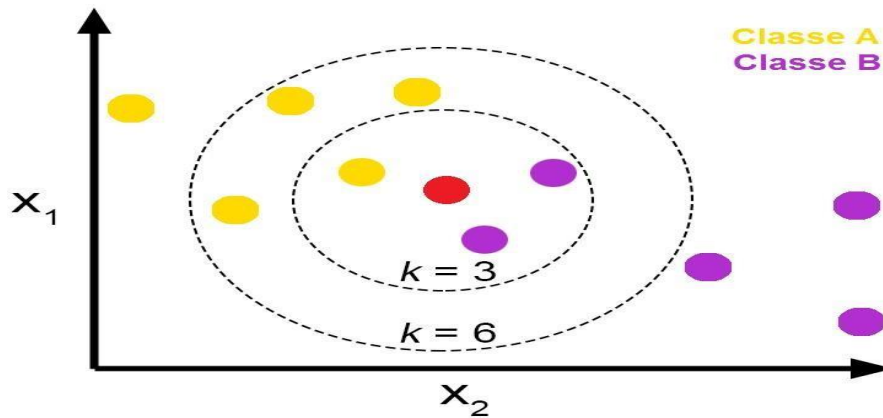


Figure: 3.8 K-Nearest Neighbours

When we say a technique is non-parametric, it means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. If you think about it, it's pretty useful, because in the "real world", most of the data does not obey the typical theoretical assumptions made (as in linear regression models, for example). Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data.

5. Logistic Regression: Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

6. Support Vector Machine: Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the

extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

3.7 CONFUSION MATRIX

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

| | | Predicted class | |
|--------------|----------|----------------------|----------------------|
| | | <i>P</i> | <i>N</i> |
| Actual Class | <i>P</i> | True Positives (TP) | False Negatives (FN) |
| | <i>N</i> | False Positives (FP) | True Negatives (TN) |

Fig:3.9 Confusion Matrix

A true positive (tp) is a result where the model predicts the positive class correctly. Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class.

A false positive (fp) is an outcome where the model incorrectly predicts the positive class. And a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

Sensitivity or Recall or hit rate or true positive rate (TPR)

It is the proportion of individuals who actually have the disease were identified as having the disease.

$$\text{TPR} = \text{tp} / (\text{tp} + \text{fn})$$

Specificity, selectivity or true negative rate (TNR)

It is the proportion of individuals who actually do not have the disease were identified as not having the disease.

$$\text{TNR} = \text{tn} / (\text{tn} + \text{fp}) = 1 - \text{FPR}$$

Precision or positive predictive value (PPV)

If the test result is positive what is the probability that the patient actually has the disease.

$$\text{PPV} = \text{tp} / (\text{tp} + \text{fp})$$

Negative predictive value (NPV)

If the test result is negative what is the probability that the patient does not have disease.

$$\text{NPV} = \text{tn} / (\text{tn} + \text{fn})$$

Miss rate or false negative rate (FNR)

It is the proportion of the individuals with a known positive condition for which the test result is negative.

$$\text{FNR} = \text{fn} / (\text{fp} + \text{tn})$$

Fall-out or false positive rate (FPR)

It is the proportion of all the people who do not have the disease who will be identified as having the disease.

$$\text{FPR} = \text{fp} / (\text{fp} + \text{tn})$$

False discovery rate (FDR)

It is the proportion of all the people identified as having the disease who do not have the disease.

$$\text{FDR} = \text{fp} / \text{fp} + \text{tp}$$

False omission rate (FOR)

It is the proportion of the individuals with a negative test result for which the true condition is positive.

$$\text{FOR} = \text{fn} / (\text{fn} + \text{tn})$$

Accuracy

The accuracy reflects the total proportion of individuals that are correctly classified.

$$ACC = (tp + tn) / (tp + tn + fp + fn)$$

F1 score

It is the harmonic mean of precision and sensitivity

$$F1 = 2tp / (2tp + fp + fn)$$

RMSE Score

Root mean square of the error that has occurred between the test values and the predicted values.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

3.8 SMOTE (Synthetic Minority Oversampling Technique) :

SMOTE (Synthetic Minority Oversampling Technique) is commonly used oversampling methods to solve imbalance problem. It balance class distribution by indiscriminately increasing minority. New minority instances are synthesized by using existing minority instances. It generates virtual training records by linear interpolation for minority class. These training records are generated by randomly selecting one or more of the k-nearest neighbors. After the process, the data is reconstructed and classification can be applied for the processed data.

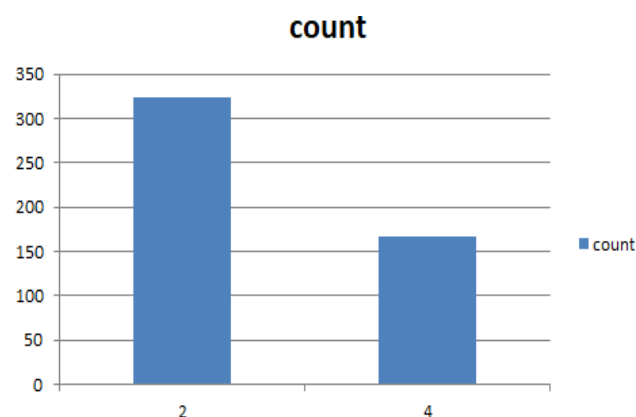


Figure: 3.10 Class distribution before SMOTE technique

Figure shows the number of instances for class-2 and class-4 before applying the SMOTE technique. class-2 has more instances than class-4. So, SMOTE technique is applied to balance the class.

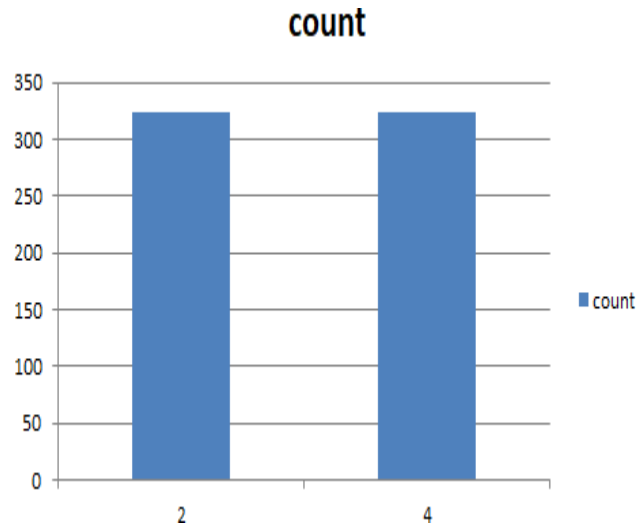


Figure:3.11 Class distribution after SMOTE technique

Figure shows the number of instances for class-2 and class-4 after applying the SMOTE technique. class-2 and class-4 contains same number of instances after applying SMOTE technique..

4.Implementation code

```
import numpy as np #used for arrays

import pandas as pd #for analyzing the data

import seaborn as sns;sns.set()

from sklearn.model_selection import train_test_split #to split the data into test and training

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

df=pd.read_csv("D:\project 4.2\heart_failure_clinical_records_dataset.csv")

df.head()

df.dtypes

df.info()

df.shape

categorical=['sex','anaemia','high_blood_pressure','smoking','DEATH_EVENT']

numerical=['age','creatinine_phosphokinase',    'diabetes',    'ejection_fraction','platelets',
'serum_creatinine', 'serum_sodium','time']

cols=['sex','anaemia','high_blood_pressure','smoking','age','creatinine_phosphokinase',
'diabetes', 'ejection_fraction','platelets'

      , 'serum_creatinine', 'serum_sodium','time','DEATH_EVENT']

sns.heatmap(df.isnull(),cbar=False)

cleanup_nums = {"sex":    {"male": 0, "female": 1},

                "anaemia":  {"yes": 0, "no": 1},

                "high_blood_pressure":    {"yes": 0, "no": 1},

                "smoking":    {"yes": 0, "no": 1},

                "DEATH_EVENT":{"Die": 0, "notDie": 1}}
```

```

#filling missing values with mean

for i in numerical:

    df[i].fillna(df[i].mean(),inplace=True)

df.info()

sns.heatmap(df.isnull(),cmap='coolwarm',cbar=False)

corrmat=df.corr()

plt.figure(figsize=(20,20))

plt.show()

heat_map=sns.heatmap(corrmat,annot=True,cmap='YlGnBu',          vmin=None,
vmax=None,linewidths=0)

plt.show()

# Select upper triangle of correlation matrix

corr_matrix = df.corr().abs()

upper = np.triu(np.ones_like(corr_matrix,dtype=bool))

k=corr_matrix.mask(upper)

# Find index of feature columns with correlation greater than 0.4

to_drop = [column for column in k.columns if any(k[column] > 0.7)]

# Drop features

after_dropped=df.drop(df[to_drop], axis=1)

print(to_drop)

print(len(after_dropped.columns))

X = after_dropped.iloc[:, :-1].values # attributes to determine dependent variable / Class

y = after_dropped.iloc[:, -1].values# dependent variable / Class

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.50, random_state=42)

X1_train, X1_test, y1_train, y1_test = train_test_split(X, y, test_size=0.20, random_state=42)

```

```

plt.figure(figsize = (19,19))

sns.heatmap(after_dropped.corr(), annot = True, cmap = 'coolwarm') # looking for strong
correlations with "class" row

plt.title("Correlation Heatmap after Removing columns", fontsize=12)

plt.show()

#splitting dataset into training and test data

y=df.DEATH_EVENT

x=df.drop('DEATH_EVENT',axis=1)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)

print(x_train)

#boxplot

def remove_outlier(df, col_name):

    q1 = df[col_name].quantile(0.25)

    q3 = df[col_name].quantile(0.75)

    iqr = q3-q1 #Interquartile range

    fence_low = q1-1.5*iqr

    fence_high = q3+1.5*iqr

    df_out = df.loc[(df[col_name] > fence_low) & (df[col_name] < fence_high)]

    return df_out

for i in cols:

    df=remove_outlier(df,i)

print(df)

```



```

df.shape

df.info()

#splitting data

y=df.DEATH_EVENT

x=df.drop('DEATH_EVENT',axis=1)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)

print(x_train)


sns.countplot(df.DEATH_EVENT)

plt.show()

#Logistic regression algorithm

classifier = LogisticRegression(random_state = 0)

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix


#from sklearn. import check_targets

cm1 = confusion_matrix(y_test, y_pred)

print(cm1)

#print("Accuracy score of train LogisticRegression")

```

```

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test LogisticRegression")

a1=accuracy_score(y_test, y_pred)*100

print(a1)

#decision tree algorithm

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(random_state = 0)

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix

cm2= confusion_matrix(y_test, y_pred)

print(cm2)

#print("Accuracy score of train Decision tree")

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test Decision tree")

a2=accuracy_score(y_test, y_pred)*100

print(a2)


#RandomForestClassifier algorithm

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(random_state = 0)

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

```

```

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix

cm3 = confusion_matrix(y_test, y_pred)

print(cm3)

#print("Accuracy score of train RandomForestClassifier")

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test RandomForestClassifier")

a3=accuracy_score(y_test, y_pred)*100

print(a3)

#KNeighborsClassifier algorithm

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier()

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix

cm4 = confusion_matrix(y_test, y_pred)

print(cm4)

#print("Accuracy score of train KNeighborsClassifier")

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test KNeighborsClassifier")

a4=accuracy_score(y_test, y_pred)*100

print(a4)

```

```

#support vector machine algorithm

from sklearn.svm import SVC

classifier = SVC()

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix

cm5 = confusion_matrix(y_test, y_pred)

print(cm5)

#print("Accuracy score of train support vector machine(svm)")

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test support vector machine(svm)")

a5=accuracy_score(y_test, y_pred)*100

print(a5)

#naive bayes algorithm

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

trained_model=classifier.fit(x_train,y_train)

trained_model.fit(x_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(x_test)

# Making the Confusion Matrix

cm6 = confusion_matrix(y_test, y_pred)

print(cm6)

```

```

#print("Accuracy score of train Naive bayes")

#print(accuracy_score(y_train, trained_model.predict(x_train))*100)

print("Accuracy score of test Naive bayes")

a6=accuracy_score(y_test, y_pred)*100

print(a6)

# x-coordinates of left sides of bars

left = [0,20,40,60,80,100]

# heights of bars

height = [a1,a2,a3,a4,a5,a6]

# labels for bars

tick_label = ['LogisticRegression', 'DecisionTreeClassifier', 'RandomForestClassifier',
'KNeighborsClassifier', 'SupportVectorMachine', 'NaiveBayes']

plt.figure(figsize=(15, 8))

# plotting a bar chart

plt.bar(left, height, tick_label = tick_label,

        width = 15, color = ['lightpink', 'lightblue', 'yellow', 'orange', 'lightgreen', 'purple'])

# naming the x-axis

plt.xlabel('algorithms')

# naming the y-axis

plt.ylabel('accuracy')

# plot title

plt.title('comparision of algorithms')

# function to show the plot

plt.show()

```

```
RF=RandomForestClassifier()

#Train the model

RF.fit(x_train,y_train)

# Saving model to disk

pickle.dump(RF, open('model.pkl','wb'))

# Loading model to compare the results

model = pickle.load(open('model.pkl','rb'))

#print(model.predict([[2, 9, 6]]))
```

5.Result Analysis

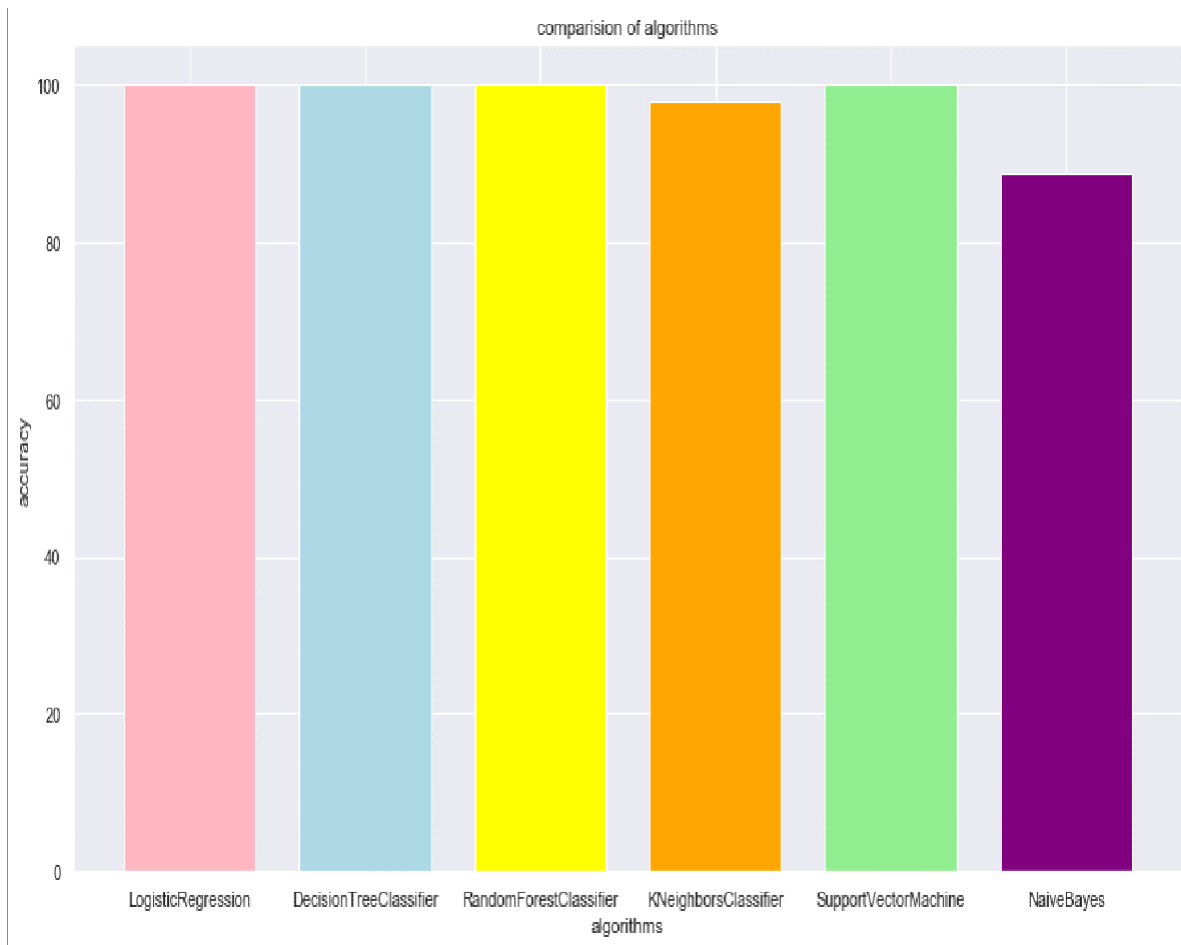
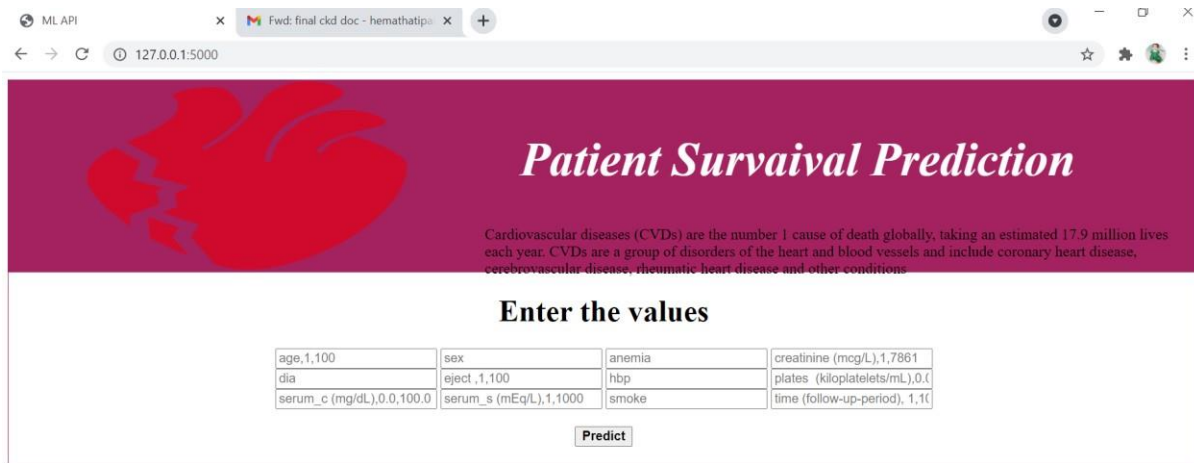


Figure:5.1 Comparison of accuracy of algorithms

Figure shows the comparison of accuracy of six classifiers (Random Forest Classifier, Logistic Regression, Gaussian Naïve Bayes, Decision SVM, KNN, Tree). The highest accuracy is 100% for Random Forest Classifier.

6. SCREEN SHOTS

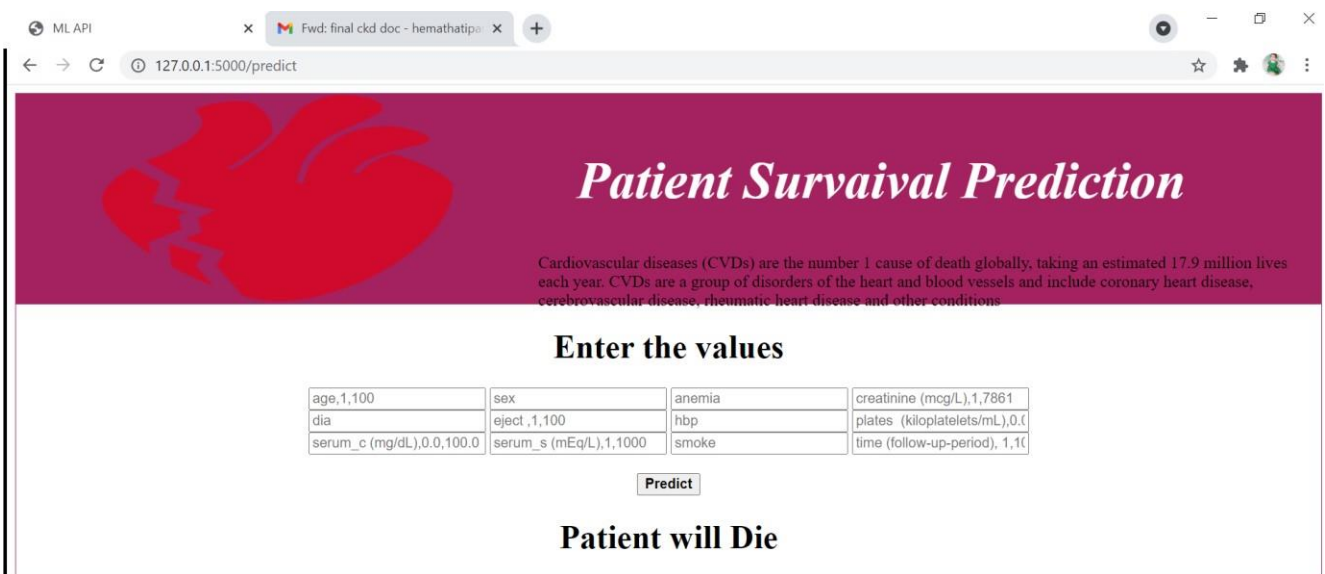


The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page has a dark blue header with a heart icon and the title "Patient Survival Prediction". Below the header, there is a paragraph about Cardiovascular diseases (CVDs). The main content area is titled "Enter the values" and contains a form with eight input fields arranged in a 2x4 grid. The fields are labeled: age, sex, anemia, creatinine (mcg/L), dia, eject, hbp, plates (kiloplatelets/mL), serum_c (mg/dL), serum_s (mEq/L), smoke, and time (follow-up-period). A "Predict" button is located below the form.

| | | | |
|---------------------------|------------------------|--------|--------------------------------|
| age,1,100 | sex | anemia | creatinine (mcg/L),1,7861 |
| dia | eject ,1,100 | hbp | plates (kiloplatelets/mL),0.0 |
| serum_c (mg/dL),0.0,100.0 | serum_s (mEq/L),1,1000 | smoke | time (follow-up-period), 1,100 |

Predict

Fig 6.1: Main page



The screenshot shows the same web browser window as Fig 6.1, but the URL is now 127.0.0.1:5000/predict. The page layout is identical, but the "Predict" button is now disabled. Below the form, the text "Patient will Die" is displayed in a large, bold font.

| | | | |
|---------------------------|------------------------|--------|--------------------------------|
| age,1,100 | sex | anemia | creatinine (mcg/L),1,7861 |
| dia | eject ,1,100 | hbp | plates (kiloplatelets/mL),0.0 |
| serum_c (mg/dL),0.0,100.0 | serum_s (mEq/L),1,1000 | smoke | time (follow-up-period), 1,100 |

Predict

Patient will Die

Fig 6.2: Prediction Yes

Patient Survaival Prediction

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year. CVDs are a group of disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions.

Enter the values

| | | | |
|---------------------------|------------------------|--------|-------------------------------|
| age,1,100 | sex | anemia | creatinine (mcg/L),1,7861 |
| dia | eject ,1,100 | hbp | plates (kiloplatelets/mL),0.0 |
| serum_c (mg/dL),0.0,100.0 | serum_s (mEq/L),1,1000 | smoke | time (follow-up-period), 1,10 |

Predict

Patient will not Die

Fig 6.3: Prediction No

7.CONCLUSION

We have used 6 algorithms like Decision Trees, Random Forests, Naive Bayes, SVM, KNN and Logistic Regression in-order to predict presence or absence of heart failure. The accuracy varies for different algorithms. The accuracy for Decision tree algorithm is 93.33%. The accuracy for Random Forest algorithm is 96.19%. The accuracy for Naive Bayes algorithm is 93.80%. The accuracy for KNN algorithm is 93.80%. The accuracy for Logistic Regression algorithm is 94.28%. The accuracy for SVM algorithm is 93.80%. The highest accuracy is given when we have used Random Forest algorithm is nearly 100.00%.

8.FUTURE SCOPE

This project further can be developed as Android app and also suggest precautions to be taken by the person.

9.REFERENCES

1. Jemal A, Murray T, Ward E, Samuels A, Tiwari RC, Ghafoor A, Feuer EJ, Thun MJ. Cancer statistics, 2005. CA: a cancer journal for clinicians. 2005 Jan 1;55(1):10-30.
2. U.S. Cancer Statistics Working Group. United States Cancer Statistics: 1999–2008 Incidence and Mortality Web-based Report. Atlanta (GA): Department of Health and Human Services, Centers for Disease Control
3. Madhu Kumaria, Vijendra Singhb Department of Computer Science and Engineering, The NorthCap University, Sector 23A, Gurugram, Haryana, 2017, India.
4. Hiba Asria, Hajar mousannifb, Hassaan Al Moata ssimec, Thomas Noeld. The 6th International Symposium on Frontiers in Ambient and Mobile System (FAMS 2016).
5. “UCI Machine Learning Repository: Breast Cancer Wisconsin (Original) Data Set.” [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29>.
6. Step 7 - Science Fair 2012: Project *Sites.google.com*, 2020. [Online].
7. Data Preprocessing- an overview | ScienceDirect Topics, Sciencedirect.com
8. Handling Missing Values in machine learning: part1, @george.drakos62.
9. Machine Learning | Handling Imbalanced Data with SMOTE and Near Miss Algorithm in Python - GeeksforGeeks
10. D.Lavanya, Dr.K.Usha Rani, Analysis of feature selection with classification: Breast cancer datasets”, Indian Journal of Computer Science and Engineering (IJCSE), October 2011.
11. In-Database Machine Learning 2: Calculate a correlation Matrix – A Data Exploration Post | Vertica
12. D.Lavanya, Dr.K.Usha Rani,...,” Analysis of feature selection with classification: Breast cancer datasets”, Indian Journal of Computer Science and Engineering (IJCSE), October 2011
13. Logistic Regression for Machine Learning, machinelearningmastery.com
14. Zhang, J.; Zulkernine, M.; Haque, A. Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. 2008, 38, 649–659.

15. Rish I. An empirical study of the naive Bayes classifier. *IJCAI Work Empir methods Artif Intell.* 2001;3(November):41-46.
16. Noble WS. What is a support vector machine? *Nat Biotechnol.* 2006;24(12):1565-1567. doi:10.1038/nbt1206-1565.
17. Introduction to KNN, K-Nearest Neighbors : Simplified, analyticsvidhya.com
18. Quinlan JR. C4.5: Programs for Machine Learning.; 2014:302.
<https://books.google.com/books?hl=fr&lr=&id=b3ujBQAAQBAJ&pgis=1>. Accessed January 5, 2016.
- 19.3.1 Cross-validation: evaluating estimator performance – scikit-learn 0.22.2
20. Han and M. Kamber, "Data Mining Concepts and Techniques", Morgan Kauffman Publishers, 2000
21. M. Sireesha, S. N. Tirumala Rao, Srikanth Vemuru, Frequent Itemset Mining Algorithms: A Survey *Journal of Theoretical and Applied Information Technology* Vol - 96, No .3, Feb – 2018 ISSN - 1992-8645, Pages – 744 – 755.
22. M. Sireesha, Srikanth Vemuru and S. N. Tirumala Rao, "Coalesce based binary table: an enhanced algorithm for mining frequent patterns", *International Journal of Engineering and Technology*, vol. 7, no. 1.5, pp. 51-55, 2018.
23. M. Sireesha, S. N. Tirumala Rao, Srikanth Vemuru, Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction *International Journal of Recent Technology and Engineering* Vol - 7, No 6, Mar - 2019 ISSN - 2277-3878, Pages – 1754-1772.
24. M. Sireesha, Srikanth Vemuru, S.N. Tirumala Rao "Classification Model for Prediction Of Heart Disease Using Correlation Coefficient Technique" *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 9, No. 2, March - April 2020, Pages- 2116 – 2123.
25. Sireesha Moturi , Dr. S. N. Tirumala Rao, Dr. Srikanth Vemuru,. (2020). Predictive Analysis of Imbalanced Cardiovascular Disease Using SMOTE. *International Journal of Advanced Science and Technology*, 29(05), 6301 - 6311.
26. Moturi S., Tirumala Rao S.N., Vemuru S. (2021) Risk Prediction-Based Breast Cancer Diagnosis Using Personal Health Records and Machine Learning Models. In: Bhattacharyya D., Thirupathi Rao

N. (eds) Machine Intelligence and Soft Computing. Advances in Intelligent Systems and Computing, vol 1280. Springer, Singapore. https://doi.org/10.1007/978-981-15-9516-5_37

27. Moturi S., Srikanth Vamuru, Tirumala Rao S.N. (2021) ECG based Decision Support System for Clinical Management using Machine Learning Techniques. IOP Conference Series: Materials Science and Engineering. [Volume 1085, Annual International Conference on Emerging Research Areas on "COMPUTING & COMMUNICATION SYSTEMS FOR A FOURTH INDUSTRIAL REVOLUTION" \(AICERA 2020\) 14th-16th December 2020, Kanjirapally, India](#)