# HEART FAILURE PREDICTION USING MACHINE LEARNING

## A Major Project

Submitted in partial fulfillment of the requirements
for the degree of

## BACHELOR OF TECHNOLOGY

**(Computer Science & Engineering)**

*by*

**Shivam Vishwakarma**
**0225CS181049**

*Under the guidance of*

**Prof. Vishal Paranjape**
**Department of Computer Science & Engineering**


**Global Nature Care Sangathan's Group of Institutions, Jabalpur (M.P.)**

*under*

**RAJIV GANDHI PRODYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)**

**APR-2022**

**Global Nature Care Sangathan's Group of Institutions, Jabalpur (M.P.)**

Department of Computer Science & Engineering

# Certificate

This is to certify that the Major Project report entitled **Heart Failure Prediction** submitted by **Shivam Vishwakarma** has been carried out under my guidance & supervision. The project report is approved for submission towards partial fulfillment of the requirement for the award of degree of **Bachelor of Engineering** in **Computer Science & Engineering** from "**Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P).**

**Prof. Vishal Paranjape**
Project Incharge

**Prof. Saurabh Sharma**
HOD
Dept of CSE


**Global Nature Care Sangathan's Group of Institutions, Jabalpur (M.P.)**

Department of Computer Science & Engineering

# Certificate

This is to certify that the Major Project report entitled "**Heart Failure Prediction**" is submitted by **Shivam Vishwakarma** for the partial fulfillment of the requirement for the award of degree of **Bachelor of Engineering** in **Computer Science & Engineering** from **Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P).**

Internal Examiner
Date:

External Examiner
Date:

# Abstract

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Four out of 5CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. Heart failure (HF) occurs when the heart cannot pump enough blood to meet the needs of the body. Available electronic medical records of patients quantify symptoms, body features, and clinical laboratory test values, which can be used to perform biostatistics analysis aimed at highlighting patterns and correlations otherwise undetectable by medical doctors. Machine learning, in particular, can predict patient's survival from their data and can individuate the most important features among those included in their medical records.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

# Declaration

I hereby declare that the project entitled **"Heart Failure Prediction"** which is being submitted in partial fulfillment of the requirement for award of the Degree of Bachelor of Engineering in Computer Science and Engineering to **"RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)"** is an authentic record of my own work done under the guidance of **Prof. Vishal Paranjape**, Department Computer Science & Engineering, **GLOBAL ENGINEERING COLLEGE, JABALPUR**..

The matter reported in this Project has not been submitted earlier for the award of any other degree.

**Dated :** **Shivam Vishwakarma**

**Place : Jabalpur** **0225CS181049**

# Acknowledgment

I sincerely express indebtedness to esteemed and revered guide “Prof. Vishal Paranjape”, in department of CSE for his invaluable guidance, supervision and encouragement throughout the work. Without his kind patronage and guidance the project would not have taken shape.

I take this opportunity to express deep sense of gratitude to “Prof. Saurabh Sharma”, Head of “Department of Computer Science & Engineering” for his encouragement and kind approval. Also I thank him in providing the computer lab facility. I would like to express our sincere regards to him for advice and counseling from time to time.

I owe sincere thanks to all the lecturers in “Department of Computer Science & Engineering*”* for their advice and counseling time to time.

**Dated :** **Shivam Vishwakarma**

**Place : Jabalpur** **0225CS181049**

# Table of Contents

# Chapter 1

# *Introduction*

# BACKGROUND

Cardiovascular diseases (CVDs) are disorders of the heart and blood vessels including, coronary heart disease (heart attacks), cerebrovascular diseases (strokes), heart failure (HF), and other types of pathology. Altogether, cardiovascular diseases cause the death of approximately 17.9 million people worldwide annually, with fatalities figures on the rise for first time in 50 years the United Kingdom. In particular, heart failure occurs when the heart is unable to pump enough blood to the body, and it is usually caused by diabetes, high blood pressure, or other heart conditions or diseases.

The clinical community groups heart failure into two types based on the ejection fraction value, that is the proportion of blood pumped out of the heart during a single contraction, given as a percentage with physiological values ranging between 50% and 75%. The former is heart failure due to reduced ejection fraction (HFrEF), previously known as heart failure due to left ventricular (LV) systolic dysfunction or systolic heart failure and characterized by an ejection fraction smaller than 40% . The latter is heart failure with preserved ejection fraction (HFpEF), formerly called diastolic heart failure or heart failure with normal ejection fraction. In this case, the left ventricle contracts normally during systole, but the ventricle is stiff and fails to relax normally during diastole, thus impairing filling.

Many researcher have created similar kind of solution but they have used a single dataset that contains observation between 100 and 300. Here we are using a dataset which is made using 5 different datasets so total observation we get is approximate equal to 1000.

# OBJECTIVE

The objective is to predict the chances of getting a heart failure of particular person based on their health reports. We will use various algorithms for training the model and we will find the best suitable algorithm for our project.

We have to make sure that our model works fine with high amount of accuracy. We have to train our models using different algorithms that are available. After we select our best suitable model we will tune it and then check it accuracy. Once we get satisfied with model performance we will make a web interface of general user to interact with model and get reults.

# PURPOSE, SCOPE AND APPLICABILITY

## Purpose

The main purpose of this project is to predict the chances of heart failure in humans and cure it before it's too late. The project is designed in such a way that anyone can use it with their browser by filling the form with necessary information and they will get the result that they are safe or not.

## Scope

We are going to train our model with different algorithms and the will tune the models with 'catboost' after that we will select the best model. Since the dataset dose not contains data for children we may not be able to predict very correct information about children. This could be a limitation for our project.

## Applicability

The project can be used by individuals or by hospitals to get predictions for their patients. The project provides a simple web interface which will be easy to use. In future when we get more data we can simple train new model and apply it to same project. The project can also be used as an API service to integrate with another application.

# ACHIEVEMENTS

- First, I've made the detailed exploratory analysis.
- I have decided which metric to use.
- I have analyzed both target and features in detail.

- I’ve transformed categorical variables into numeric so we can use them in the model.
- I’ve  used pipeline to avoid data leakage.
- I’ve looked at the results of the each model and selected the best one for the problem on hand.
- Looked in detail Catboost.
- I’ve done hyper parameter tuning of the Catboost with Optuna to see the improvement
- Looked at the feature importance.
- I’ve successfully created a web application for user interface using streamlit library.

# Chapter 2

# *Survey of Technologies*

# PYTHON

Machine Learning projects differ from traditional software projects. The differences lie in the technology stack, the skills required for a machine learning based project, and the necessity of deep research. To implement machine learning aspirations, one should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python AI projects today.

From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefits that make Python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

## Simple and consistent

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language.

Additionally, Python is appealing to many developers as it's easy to learn. Python code is understandable by humans, which makes it easier to build models for machine learning.

Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes.

# MACHINE LEARNING

Machine learning is a form of AI that enables a system to learn from data rather than through explicit programming. However, machine learning is not a simple process. As the algorithms ingest training data, it is then possible to produce more precise models based on that data. A machine-learning model is the output generated when you train your machine-learning algorithm with data. After training, when you provide a model with an input, you will be given an output. For example, a predictive algorithm will create a predictive model. Then, when you provide the predictive model with data, you will receive a prediction based on the data that trained the model.

The advantage of machine learning is that it is possible to leverage algorithms and models to predict outcomes. The trick is to ensure that the data scientists doing the work are using the right algorithms, ingesting the most appropriate data (that is accurate and clean) and using the best performing models. If all these elements come together it's possible to continuously train the model and learn from the outcomes by learning from the data. The automation of this process of modeling, training the model and testing leads to accurate predictions to support business change.

## Scikit-Learn

Scikit-learn is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

# LIBRARIES

## NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

## Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

## Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

## Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

## Sklearn

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

## Optuna

Optuna is an automatic hyperparameter optimization software framework, particularly designed for machine learning. It features an imperative, define-by-run style user API. Thanks to our define-by-run API, the code written with Optuna enjoys high modularity, and the user of Optuna can dynamically construct the search spaces for the hyperparameters.

## Xgboost

XGBoost is an open-source software library which provides a regularizing gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting Library"

## Lightgbm

LightGBM, short for Light Gradient Boosting Machine, is a free and open source distributed gradient boosting framework for machine learning originally developed by Microsoft. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks.

## Cufflinks

Cufflink is also a python library that connects plotly with pandas so that we can create charts directly on data frames.

## Catboost

CatBoost is an open-source software library developed by Yandex. It provides a gradient boosting framework which attempts to solve for Categorical features using a permutation driven alternative compared to the classical algorithm.

## Plotly

Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST.

## StreamLit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

### Pickle

The pickle module can transform a complex object into a byte stream and it can transform the byte stream into an object with the same internal structure. We will use pickle to save our model as a pickle file.

# Chapter 3
# *Requirement and Analysis*

# PROBLEM DEFINITION

There are 4 major problems to be solved:

1. Analysis of data
2. Cleaning the data
3. Training the model
4. Finding best suitable model
5. Making a web application for user interface.

## Analysis of Data

Here, we have to analyze our dataset and find the parameters and features which are important to us. Also, we have to check for duplicate rows if any, also we have to check for any column with empty data, we don't want such incomplete observations.

After that we need to analyze the importance of each and every parameter and make heatmaps to find which features affect the results most.

Our dataset must be having two kinds of features, numerical and categorical, we have to check the impact of each feature.

## Cleaning the Data

In this step have to clean our dataset i.e. if there is any missing value then either we have to remove it or replace it with some other value like mean or median value for the column.

## Training the model

In model training we have to use various models and then compare their accuracy. We will train around 17 models and will use the best model for our project. We also have to tune our model to get more accurate results.

## Finding best suitable model

This is a very crucial step as we will be having a list of models and their accuracy score on test data. Also we have to tune our models with multiple libraries and then compare them.

## Making a web application

Once our model is ready then we have to make a web application to make it useable for users. In this particular project I will be using Streamlit library for web interface. I am using Streamlit because it is easy to use with python and can easily integrate with other machine learning libraries.

# SOFTWARE REQUIREMENT SPECIFICATION (SRS)

## Server side specifications

### Software Specification:

The server needs to configure with the following:

1. Python 3.9 (or higher)
2. Webserver
3. Jupyter Notebook

The following python libraries needs to be installed in server:

1. Numpy
2. Matplotlib
3. Seaborn
4. Pandas
5. Optuna
6. XGBoost
7. LightGBM
8. CatBoost
9. Scikit-Learn
10. Plotly
11. Cufflinks
12. Streamlit

### Hardware Specifications:

The server should have these minimum hardware requirements

1. CPU intel i5 or equivalent.
2. GPU ( for better performance but not mandatory).
3. RAM 4GB minimum.
4. OS can be any linux, Windows or Mac.
5. Storage Space 16GB minimum.

# Client side specifications

The client/user of the product should have following specification:

1. Device with any of the following OS
    a. IOS
    b. Android
    c. Windows
    d. Linux
    e. Mac
2. Browser with javascript support
    a. Chrome
    b. Firefox
    c. Edge
    d. Safari
3. Internet Conectivity

# Chapter 4
# *Dataset Analysis*

# DATA DICTIONARY

The dataset I am using is created by combining different datasets already available independently but not combined before. In this dataset, 5 heart datasets are combined over 11 common features which make it the largest heart disease dataset available so far for research purposes.

The five data sets that are used to make this complete dataset are:

1. Cleveland: 303 observations
2. Hungarian: 294 observations
3. Switzerland: 123 observations
4. Long Beach VA: 200 observations
5. Stalog (Heart) Data Set: 270 observations

On combining all the above dataset we get:

1. Total: 1190 observations
2. Duplicate: 272 observations
3. Final dataset: 918 observations

## Attribute Information

The Dataset have the following attributes:

1. Age: age of the patient [years]
2. Sex: sex of the patient [M: Male, F: Female]
3. ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. RestingBP: resting blood pressure [mm Hg]
5. Cholesterol: serum cholesterol [mm/dl]
6. FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]

8. MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
9. ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
10. Oldpeak: oldpeak = ST [Numeric value measured in depression]
11. ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12. HeartDisease: output class [1: heart disease, 0: Normal]

# WHAT PROBLEM WE HAVE AND WHICH METRIC TO USE

1. Based on data and data dictionary, we have classification problem.
2. We will make classification on the target variable Heart Disease
3. And we will build a model to get best classification possible on the target variable.
4. For that we will look at the balance of the target variable.
5. As we will see later, our target variable has balanced or balanced like data
6. For that reason we will use Accuracy score

# EXPLORATORY DATA ANALYSIS

**Data Columns**

| # | Attribute Name | Rows | Non- null | Data type |
|---|---|---|---|---|
| 1 | Age | 918 | non-null | Int64 |
| 2 | Sex | 918 | non-null | Object |
| 3 | Chest Pain Type | 918 | non-null | Object |
| 4 | Resting BP | 918 | non-null | Int64 |
| 5 | Cholesterol | 918 | non-null | Int64 |
| 6 | Fasting BS | 918 | non-null | Int64 |
| 7 | Resting ECG | 918 | non-null | Object |
| 8 | Max Heart rate | 918 | non-null | Int64 |
| 9 | Exercise Angina | 918 | non-null | Object |
| 10 | OldPeak | 918 | non-null | Float64 |
| 11 | ST_Slope | 918 | non-null | object |
| 12 | Heart Disease | 918 | non-null | Int64 |

Overall data types seems ok.

| # Age | A Sex | A ChestPain... | # RestingBP | # Cholesterol | # FastingBS | A RestingECG | # MaxHR | ✓ ExerciseA... |
|---|---|---|---|---|---|---|---|---|
| 40 | M | ATA | 140 | 289 | 0 | Normal | 172 | N |
| 49 | F | NAP | 160 | 180 | 0 | Normal | 156 | N |
| 37 | M | ATA | 130 | 283 | 0 | ST | 98 | N |
| 48 | F | ASY | 138 | 214 | 0 | Normal | 108 | Y |
| 54 | M | NAP | 150 | 195 | 0 | Normal | 122 | N |
| 39 | M | NAP | 120 | 339 | 0 | Normal | 170 | N |
| 45 | F | ATA | 130 | 237 | 0 | Normal | 170 | N |
| 54 | M | ATA | 110 | 208 | 0 | Normal | 142 | N |
| 37 | M | ASY | 140 | 207 | 0 | Normal | 130 | Y |
| 48 | F | ATA | 120 | 284 | 0 | Normal | 120 | N |
| 37 | F | NAP | 130 | 211 | 0 | Normal | 142 | N |
| 58 | M | ATA | 136 | 164 | 0 | ST | 99 | Y |
| 39 | M | ATA | 120 | 204 | 0 | Normal | 145 | N |
| 49 | M | ASY | 140 | 234 | 0 | Normal | 140 | Y |
| 42 | F | NAP | 115 | 211 | 0 | ST | 137 | N |
| 54 | F | ATA | 120 | 273 | 0 | Normal | 150 | N |
| 38 | M | ASY | 110 | 196 | 0 | Normal | 166 | N |

## Target Variable

The target variable is ‘HeartDisease’. We have taking following information by plotting the chart for target variable.

1. Almost 55% of the patients had a heart disease.
2. 508 patients had a heart disease.
3. Almost 45% of patients didn't have a heart disease.
4. 410 patients didn't have a heart disease.
5. There is a little imbalance but nothing in the disturbing level.
6. We can use 'accuracy' metric as our evaluation metric.

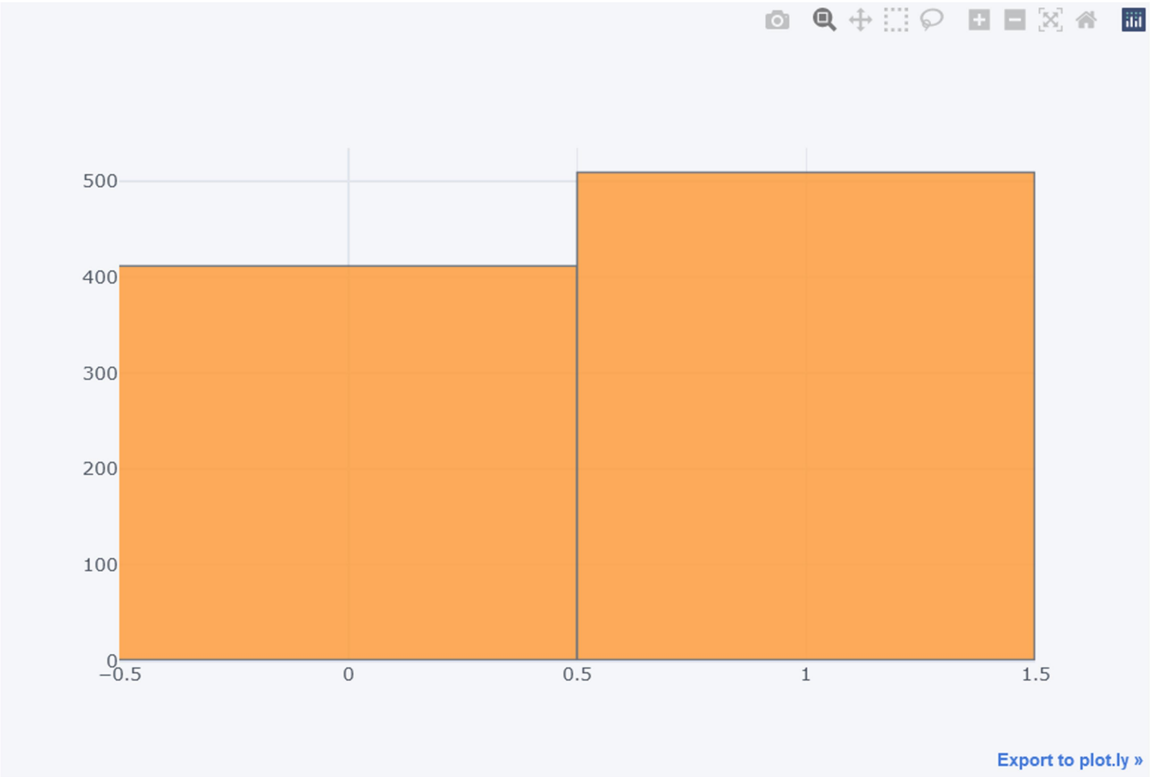



Chart for Heart Disease Attribute

In the above chart 0 represents patients not having heart disease and 1 represents patients having heart disease.

## Numerical Features

The dataset have following numerical features:

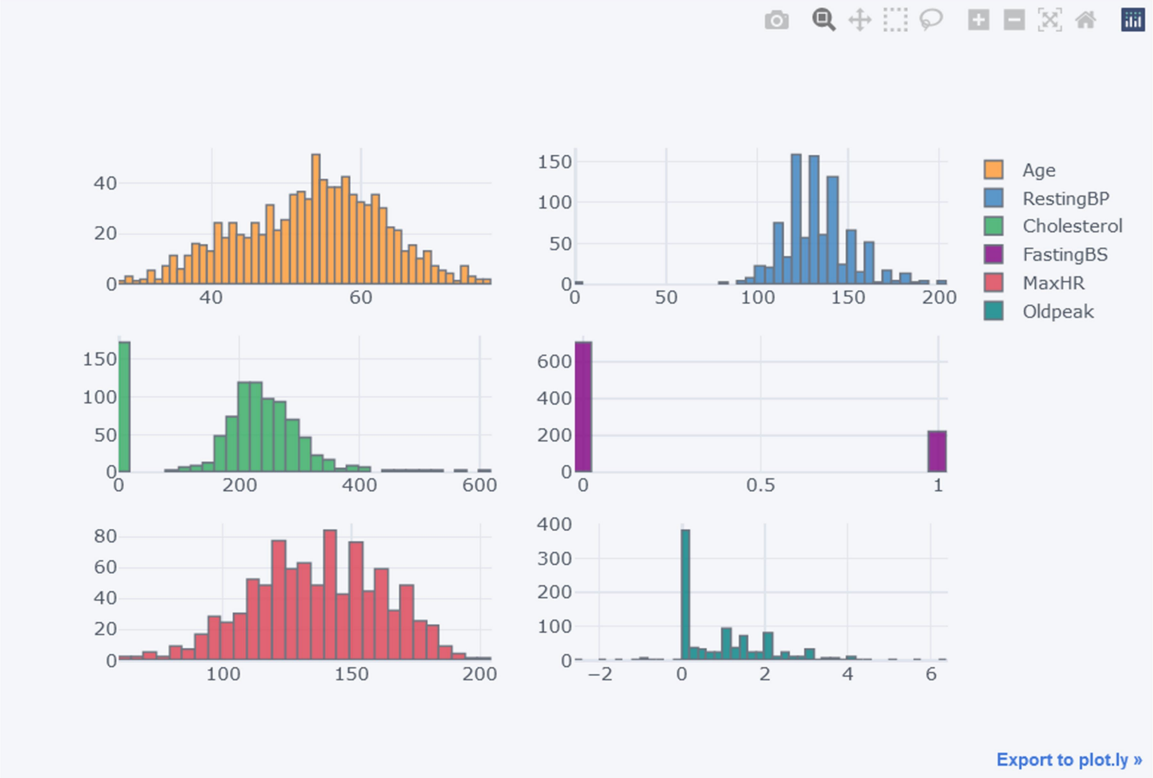
1. Age
2. Resting BP
3. Cholesterol

4. Fasting BS
5. MaxHR
6. Oldpeak

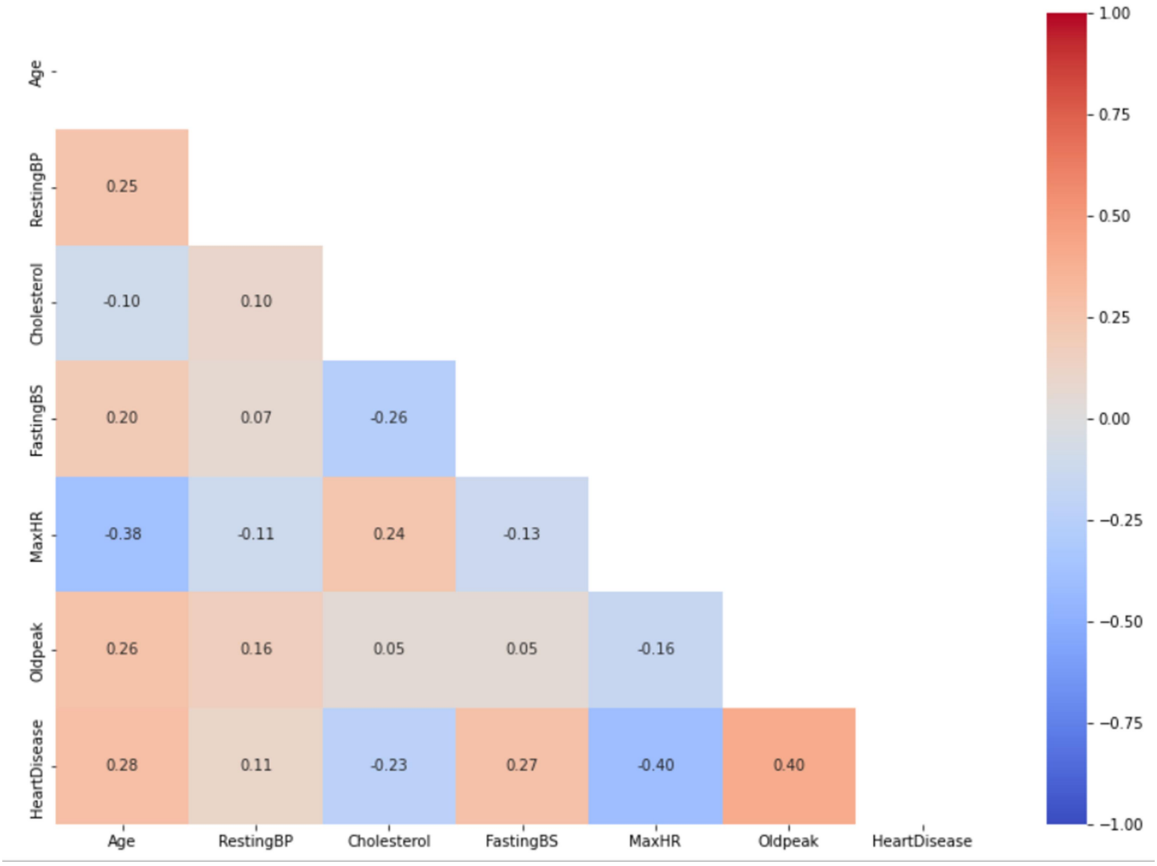
The table below describes these features briefly.

| | Age | RestingBP | Cholesterol | FastingBS | MaxHR | Oldpeak |
|---|---|---|---|---|---|---|
| count | 918.000000 | 918.000000 | 918.000000 | 918.000000 | 918.000000 | 918.000000 |
| mean | 53.510893 | 132.396514 | 198.799564 | 0.233115 | 136.809368 | 0.887364 |
| std | 9.432617 | 18.514154 | 109.384145 | 0.423046 | 25.460334 | 1.066570 |
| min | 28.000000 | 0.000000 | 0.000000 | 0.000000 | 60.000000 | -2.600000 |
| 25% | 47.000000 | 120.000000 | 173.250000 | 0.000000 | 120.000000 | 0.000000 |
| 50% | 54.000000 | 130.000000 | 223.000000 | 0.000000 | 138.000000 | 0.600000 |
| 75% | 60.000000 | 140.000000 | 267.000000 | 0.000000 | 156.000000 | 1.500000 |
| max | 77.000000 | 200.000000 | 603.000000 | 1.000000 | 202.000000 | 6.200000 |

The following histogram shows value ranges for each numerical feature:



The heatmap below given for numerical feature represents relationships among attributes.




## Categorical Features

The Dataset have following categorical features:

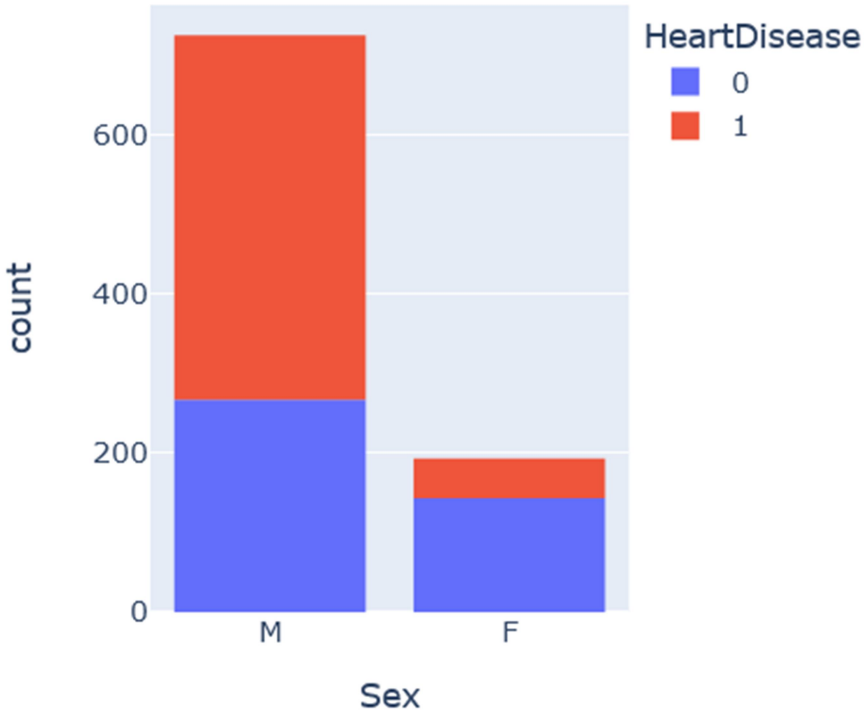
1. Sex
2. ChestPainType
3. RestingECG
4. ExerciseAngina
5. ST_Slope

The table below describes these features briefly.

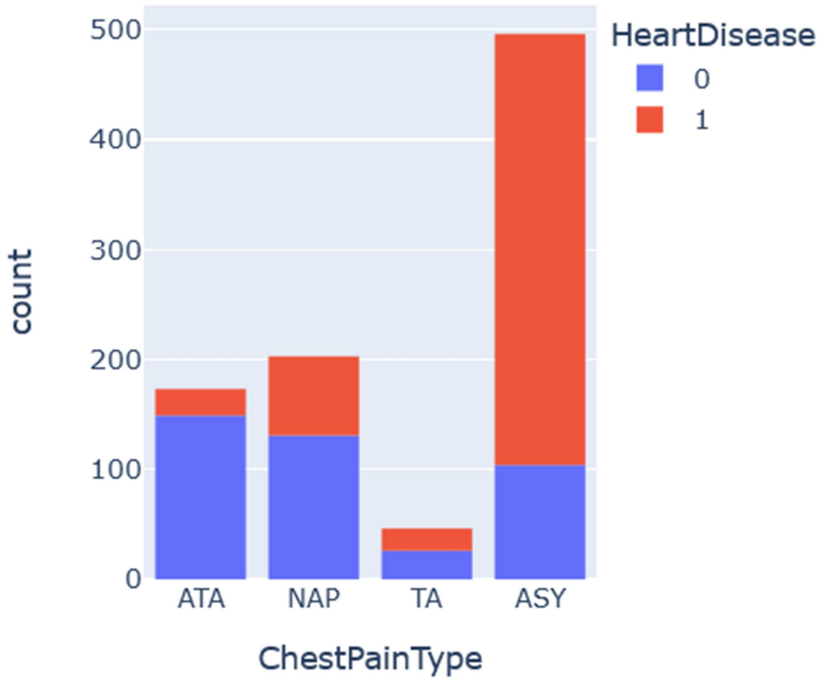
| | Sex | ChestPainType | RestingECG | ExerciseAngina | ST_Slope |
|---|---|---|---|---|---|
| count | 918 | 918 | 918 | 918 | 918 |
| unique | 2 | 4 | 3 | 2 | 3 |
| top | M | ASY | Normal | N | Flat |
| freq | 725 | 496 | 552 | 547 | 460 |

The following histogram shows value ranges for each categorical feature.

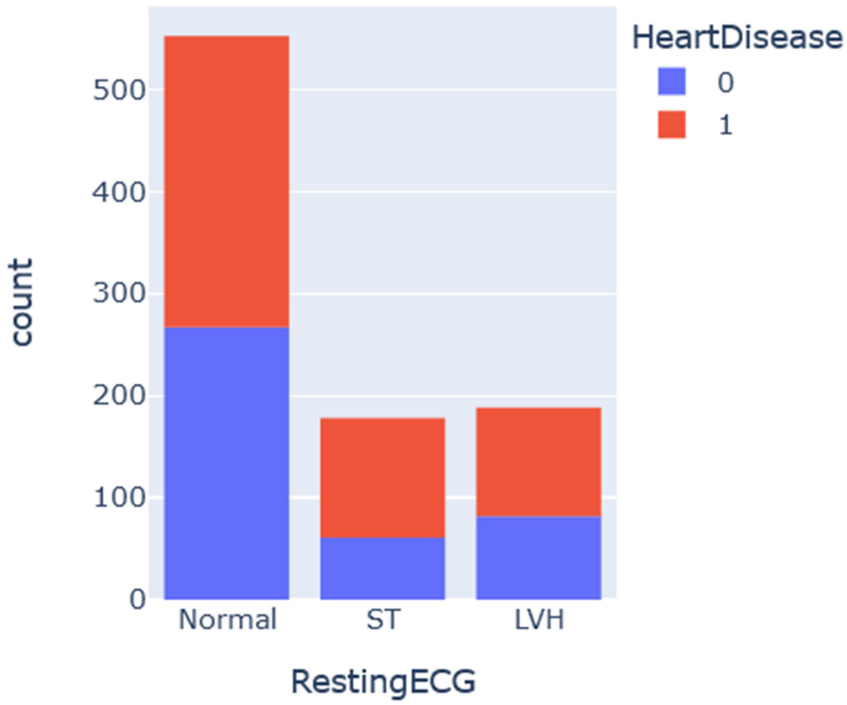

The relationship charts between heart disease and categorical features re shown in next page.



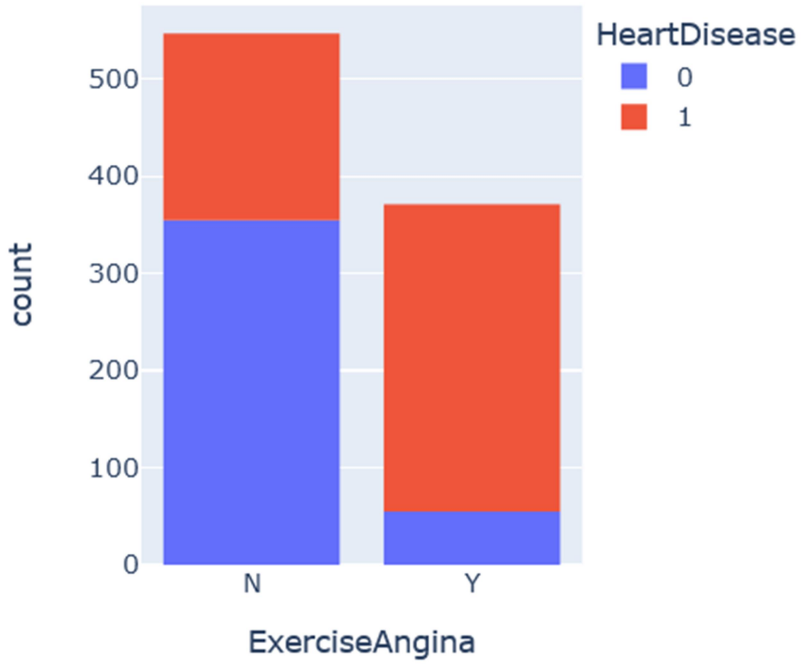

As the above figure suggests males are almost 2.44 times likely to have heart disease than women.



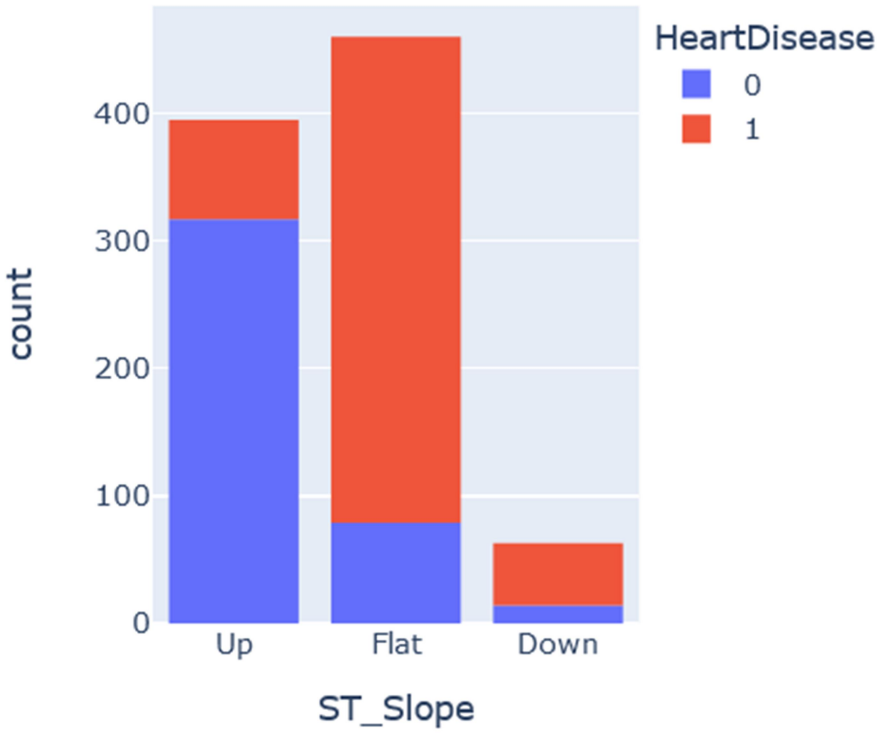

The above chart suggest that most heart patients have Asymptotic Chest pain type.



The above chart suggest that people with normal ECG have 50% chances of getting heart disease, but people with ST have 70% chances and people with LVH have around 60% chnaces.




The above chart suggests that people with Exercise Angina have almost 90% chances of getting a heart disease. Hence this is very important feature for our model.




The above chart suggests that people with flat or down ST_Slope have very high chances of getting a heart disease.

# Chapter 5
# *Training the Model*

# MODEL SELECTION

Model selection is a very important step we have to train various models. I will be training models that are very simple and I will also train models which are very complex in nature.

First we will start with dummy classifier as a base model. Then we will use Logistic & Linear Discriminant & K-Neighbors and Support Vector Machine models with and without scalar. Then we will use ensemble models, Adaboost, Randomforest, Gradient Boosting and Extra Trees. We will see famous trio, XGBoost, LightGBM & Catboost. Finally we will look in detail to hyperparameter tuning for Catboost.

## Baseline Model

First of all we will drop the 'HeartDisease' column from dataset and store it in a different data frame. Then we will spilt data into two parts training data and testing data. Here I'm using 70% data for training and 30% data for testing. After that we will apply one hot encoder to encode categorical columns into transformed numerical values. After that we will train our baseline model as Dummy Classifier model and then test its accuracy. The code to do this is given below.

```
accuracy =[]
model_names =[]
X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
ohe= OneHotEncoder()
ct=
make_column_transformer((ohe,categorical),remainder='passthrough')
model = DummyClassifier(strategy='constant', constant=1)
pipe = make_pipeline(ct, model)
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
accuracy.append(round(accuracy_score(y_test, y_pred),4))
print (f'model : {model} and  accuracy score is :
{round(accuracy_score(y_test, y_pred),4)}')
model_names = ['DummyClassifier']
dummy_result_df = pd.DataFrame({'Accuracy':accuracy},
index=model_names)
dummy_result_df
```

After testing the data we get following result as output:

```
model : DummyClassifier(constant=1, strategy='constant') and accuracy
score is : 0.5942
```

| | Accuracy |
|---|---|
| DummyClassifier | 0.5942 |

As one can see the accuracy is very low as expected. Now we will move towards comple models and train them.

## Logistic & Linear Discriminant & SVC & KNN (Without Scaler)

This time we will train 4 models together and check their accuracy. The basic preprocessing of spiting the test and train data is same. We will train Logistic & Linear Discriminant & SVC & KNN one after the other and will check their accuracy. The code for this is given below.

```
accuracy =[]
model_names =[]

X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3,
random_state=42)
ohe= OneHotEncoder()

ct=
make_column_transformer((ohe,categorical),remainder='passthrough')
lr = LogisticRegression(solver='liblinear')
lda= LinearDiscriminantAnalysis()
svm = SVC(gamma='scale')
knn = KNeighborsClassifier()

models = [lr,lda,svm,knn]

for model in models:
    pipe = make_pipeline(ct, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
    print (f'model : {model} and  accuracy score is :
{round(accuracy_score(y_test, y_pred),4)}')

model_names = ['Logistic','LinearDiscriminant','SVM','KNeighbors']
result_df1 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df1
```

After training and testing our models we get following output showing accuracy of the models.

```
model : LogisticRegression(solver='liblinear') and accuracy score is :
0.8841 model : LinearDiscriminantAnalysis() and accuracy score is :
0.8696
model : SVC() and accuracy score is : 0.7246
model : KNeighborsClassifier() and accuracy score is : 0.7174
```

As we can see this time accuracy is comparatively very higher than Dummy Classifier model. We get, 0.8841, 0.8696, 0.7246 and 0.7174 for models Logistic, Linear Discriminant, SVC and KNN respectively which is very good improvement.

But we will not stop here, now we will train same models again but this time we will use a standard scaler.

## Logistic & Linear Discriminant & SVC & KNN (With Scaler)

In this training we will use a standard scaler and scale the data while transforming categorical features into numerical features. We will train Logistic & Linear Discriminant & SVC & KNN one after the other and will check their accuracy. The code for this is given below.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

ohe= OneHotEncoder()
s= StandardScaler()
ct1= make_column_transformer((ohe,categorical),(s,numerical))


lr = LogisticRegression(solver='liblinear')
lda= LinearDiscriminantAnalysis()
svm = SVC(gamma='scale')
knn = KNeighborsClassifier()

models = [lr,lda,svm,knn]

for model in models:
    pipe = make_pipeline(ct1, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
```

```
    print (f'model : {model} and  accuracy score is :
{round(accuracy_score(y_test, y_pred),4)}')

model_names =
['Logistic_scl','LinearDiscriminant_scl','SVM_scl','KNeighbors_scl']
result_df2 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df2
```

After training and testing our models we get following output showing accuracy of the models.

```
model : LogisticRegression(solver='liblinear') and accuracy score is :
0.8804 model : LinearDiscriminantAnalysis() and accuracy score is :
0.8696
model : SVC() and accuracy score is : 0.8841
model : KNeighborsClassifier() and accuracy score is : 0.8841
```

If we compare the results with previous models we see that there is change in accuracy for all the models some of the models gives improved result like SVC and KNN, while some model give deteriorate result like Logistic Regression model. If we talk about Linear discriminant model the results are same.

## Ensemble Models (AdaBoost & Gradient Boosting & Random Forest & Extra Trees)

Now it's time to use famous ensemble models i.e AdaBoost & Gradient Boosting & Random Forest & Extra Trees. Again we will train these models one after the other and check their accuracy on test data. The code is given below.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

ohe= OneHotEncoder()
ct=
make_column_transformer((ohe,categorical),remainder='passthrough')

ada = AdaBoostClassifier(random_state=0)
gb = GradientBoostingClassifier(random_state=0)
rf = RandomForestClassifier(random_state=0)
et=  ExtraTreesClassifier(random_state=0)



models = [ada,gb,rf,et]

for model in models:
    pipe = make_pipeline(ct, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
    print (f'model : {model} and  accuracy score is :
{round(accuracy_score(y_test, y_pred),4)}')

model_names = ['Ada','Gradient','Random','ExtraTree']
result_df3 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df3
```

After training and testing our models we get following output showing accuracy of the models.

```
model : AdaBoostClassifier(random_state=0) and accuracy score is : 0.8659 model : GradientBoostingClassifier(random_state=0) and accuracy score is : 0.8768
model : RandomForestClassifier(random_state=0) and accuracy score is : 0.8877 model : ExtraTreesClassifier(random_state=0) and accuracy score is : 0.8804
```

As we can see this time all the models performed way better that all the previous models. We got highest accuracy score of 0.8877 which belongs to random forest model. We observe following points.

- Accuracy scores are very close to each other.
- Both Random Forest and Extra tree got similar accuracy scores.
- Both model can be improved by hyperparameter tuning.

Now we will check our famous trio models.

## XGBoost & LightGBM & Catboost

I will be using Catboost alone by using its capability to handle categorical variables without any preprocessing. So let's look at XGBoost and LightGBM first. Again I will train these models one after the other and check their accuracy on test data. The code is given below.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

ohe= OneHotEncoder()
ct=
make_column_transformer((ohe,categorical),remainder='passthrough')

xgbc = XGBClassifier(random_state=0)
lgbmc=LGBMClassifier(random_state=0)


models = [xgbc,lgbmc]

for model in models:
    pipe = make_pipeline(ct, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
```

```
model_names = ['XGBoost','LightGBM']
result_df4 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df4
```

After training and testing our models we get following output showing accuracy of the models.

```
model : XGBClassifier(random_state=0) and accuracy score is : 0.8297
model : LGBMClassifier(random_state=0) and accuracy score is : 0.8732
```

As we can see in default cases LightGBM gives higher accuracy score. Now we will train our catboost model but before that lets see why we using catboost and what catboost is.

Catboost is made to solve classification problems with high accuracy. The default optimized objective depends on various conditions:

1. Logless: The target has only two different values or the target_border parameter is not None.
2. Multi Class: The target has more than two different values and the border_count parameter is None.

The code below shows how we can easily use catboost without doing any preprocessing for categorical features.

```
accuracy =[]
model_names =[]

X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']

categorical_features_indices = np.where(X.dtypes != np.float)[0]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

model = CatBoostClassifier(verbose=False,random_state=0)
model.fit(X_train,
y_train,cat_features=categorical_features_indices,eval_set=(X_test,
y_test))
y_pred = model.predict(X_test)

accuracy.append(round(accuracy_score(y_test, y_pred),4))

model_names = ['Catboost_default']
result_df5 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df5
```

After training and testing the catboost model we get model accuracy score as **0.8804** which is one of the highest score we got from our models.

Now, let's make some adjustments on the catbost model to see its' peak performance on the problem.

## Catboost HyperParameter Tuning with Optuna

We will now train our catboost model and will train it with optuna the code for it is given below.

```
def objective(trial):
    X= df.drop('HeartDisease', axis=1)
    y= df['HeartDisease']
    categorical_features_indices = np.where(X.dtypes != np.float)[0]

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

    param = {
        "objective": trial.suggest_categorical("objective", ["Logloss",
"CrossEntropy"]),
        "colsample_bylevel": trial.suggest_float("colsample_bylevel",
0.01, 0.1),
        "depth": trial.suggest_int("depth", 1, 12),
        "boosting_type": trial.suggest_categorical("boosting_type",
["Ordered", "Plain"]),
        "bootstrap_type": trial.suggest_categorical(
            "bootstrap_type", ["Bayesian", "Bernoulli", "MVS"]
        ),
        "used_ram_limit": "3gb",
    }

    if param["bootstrap_type"] == "Bayesian":
        param["bagging_temperature"] =
trial.suggest_float("bagging_temperature", 0, 10)
    elif param["bootstrap_type"] == "Bernoulli":
        param["subsample"] = trial.suggest_float("subsample", 0.1, 1)

    cat_cls = CatBoostClassifier(**param)

    cat_cls.fit(X_train, y_train, eval_set=[(X_test, y_test)],
cat_features=categorical_features_indices,verbose=0,
early_stopping_rounds=100)
```

```
    preds = cat_cls.predict(X_test)
    pred_labels = np.rint(preds)
    accuracy = accuracy_score(y_test, pred_labels)
    return accuracy


if __name__ == "__main__":
    study = optuna.create_study(direction="maximize")
    study.optimize(objective, n_trials=50, timeout=600)

    print("Number of finished trials: {}".format(len(study.trials)))

    print("Best trial:")
    trial = study.best_trial

    print("  Value: {}".format(trial.value))

    print("  Params: ")
    for key, value in trial.params.items():
        print("    {}: {}".format(key, value))
```

The above code produces following output.

```
Number of finished trials: 50
Best trial:
     Value: 0.9021739130434783
     Params:
        objective: CrossEntropy
        colsample_bylevel: 0.07461412258635804
        depth: 12
        boosting_type: Plain
        bootstrap_type: MVS
```

The brief description of result parameters is given below:

- **Objective:** Supported metrics for overfitting detection and best model selection.
- **colsample_bylevel:** this parameter speeds up the training and usually does not affect the quality
- **depth:** Depth of the tree
- **boosting_type :** By default, the boosting type is set to for small datasets. This prevents overfitting but it is expensive in terms of computation. Try to set the value of this parameter to  to speed up the training.
- **bootstrap_type :** By default, the method for sampling the weights of objects is set to . The training is performed faster if the method is set and the value for the sample rate for bagging is smaller than 1.

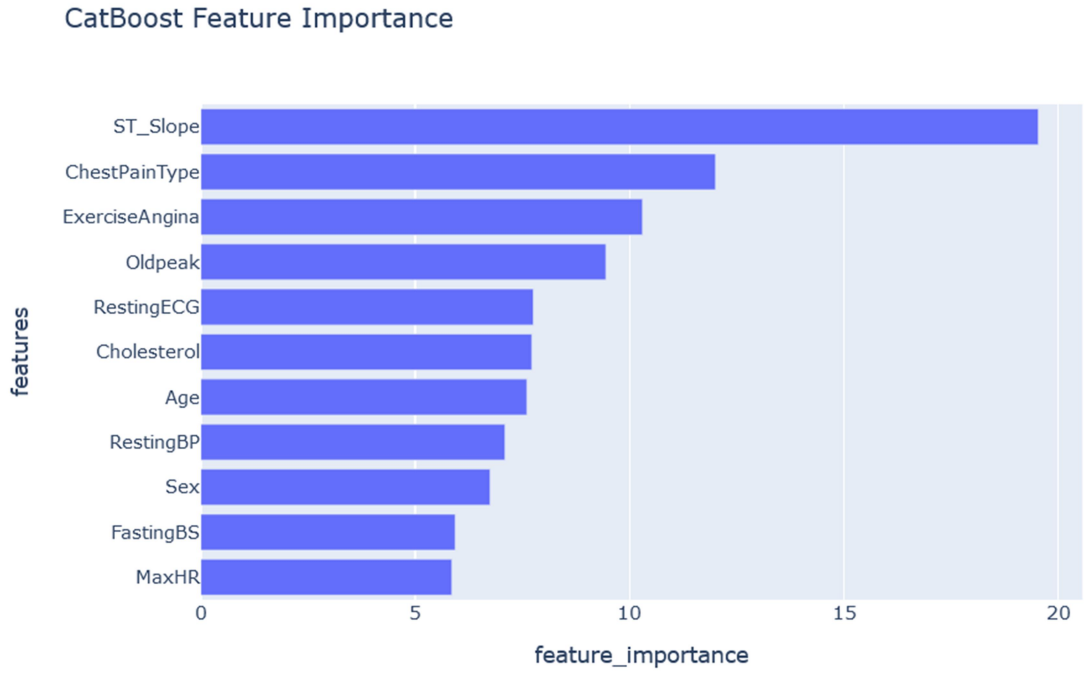
Now, let's use our best model with new parameters.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
categorical_features_indices = np.where(X.dtypes != np.float)[0]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

model = CatBoostClassifier(verbose=False,random_state=0,
                           objective= 'CrossEntropy',
    colsample_bylevel= 0.04292240490294766,
    depth= 10,
    boosting_type= 'Plain',
    bootstrap_type= 'MVS')

model.fit(X_train,
y_train,cat_features=categorical_features_indices,eval_set=(X_test,
y_test))
y_pred = model.predict(X_test)
accuracy.append(round(accuracy_score(y_test, y_pred),4))
```

```
print(classification_report(y_test, y_pred))

model_names = ['Catboost_tuned']
result_df6 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df6
```

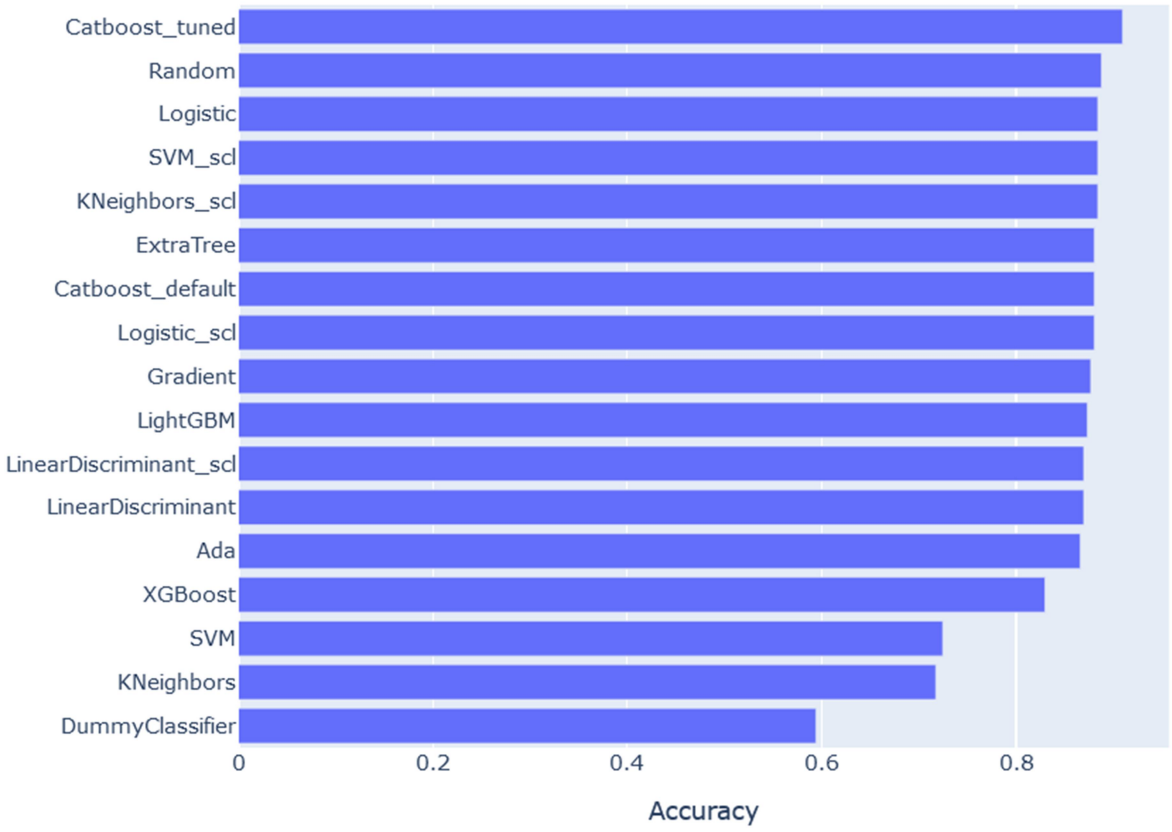
After training and tuning our model successfully we got accuracy result of **0.9094** which is **highest** among all models**.**

The figure below shows feature importance for catboost model.



# MODEL COMPARISON

I have compared all the models and created a chart for their accuracy which is shown below.




## Why Catboost is tuned?

The default Catboost model was getting an accuracy score of 0.8804, and after tuning Catboost HyperParameter with Optuna we got the accuracy score of 0.9094 which shows significant increase in accuracy.

## Saving Model as Pickle File

After training and tesing we will save our best model as a pickle file using pickle library.

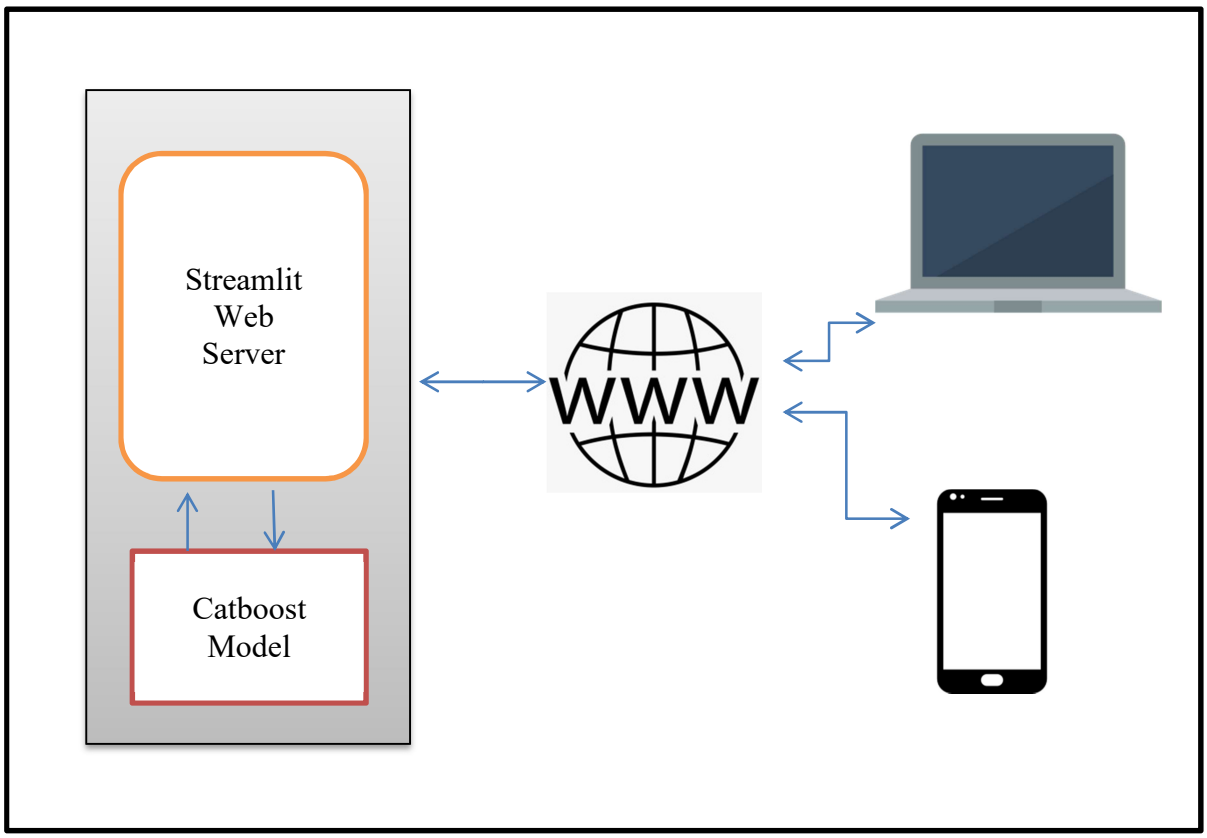
# Chapter 6
# *Web Application*

# INTRODUCTION

For web application there are various python frameworks available, but for this project I am going to use Streamlit as it's easy to use and Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

I will be running my website on localhost but we can get a domain and host it on internet too.

A simple web flow diagram is described below.



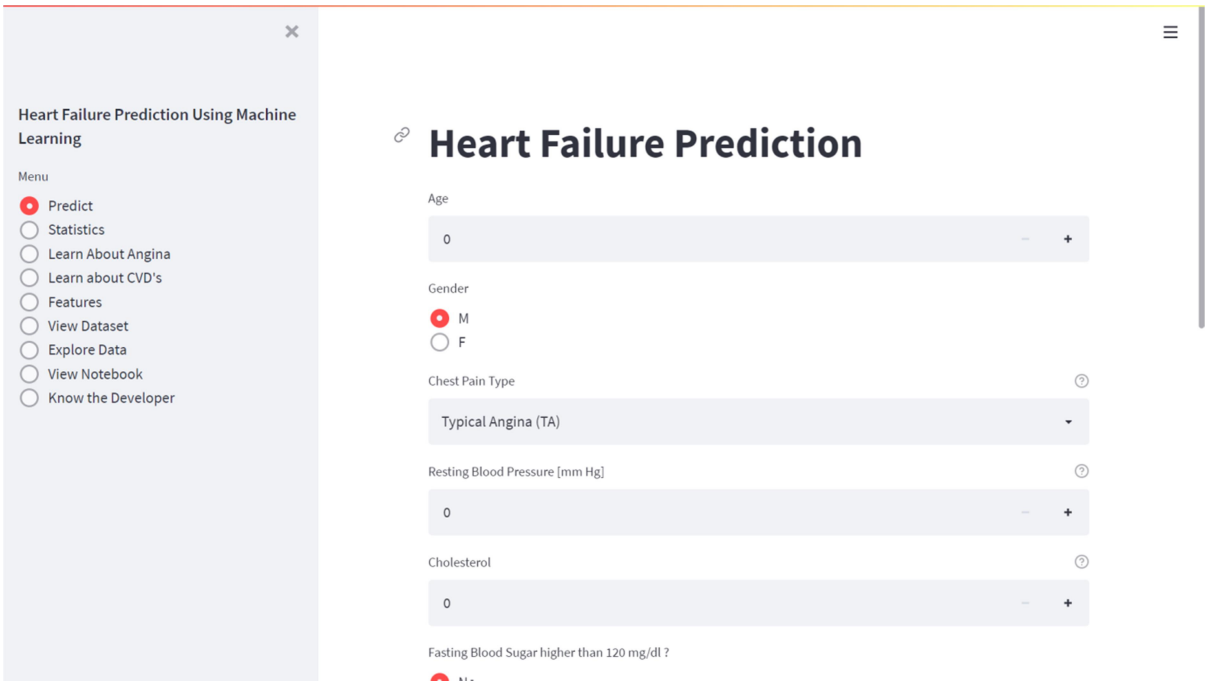
To use streamlit we will simply import Streamlit to our application as follows.

```
import streamlit as st
```

And for running the code we will use following commands.

```
streamlit run app.py
```

After coding and running our application looks like this:



# USING PICKEL

We can simply import pickle to our web app file and then load module. I have used following snippet of code to open my model. I have saved my model as 'saved_steps.pkl'.

```
def load_model():
    with open('saved_steps.pkl', 'rb') as file:
        data = pickle.load(file)
    return data

data = load_model()

model = data['model']
```

After getting our model successfully we can ask our model to predict result as follows.

```
trial =
np.array([[age,gender,chestPainType,restingBP,cholesterol,fastingBS,res
tingECG,maxHR,exerciseAngina,oldpeak,stSlope]])

result = model.predict(trial)
```

If result variable is 0 it means the patient is safe and if its 1 then there are chances of heart disease.

# Chapter 7
# *Conclusion*

# CONCLUSION

- First, I've made the detailed exploratory analysis.
- I have decided which metric to use.
- I have analyzed both target and features in detail.
- I transformed categorical variables into numeric so we can use them in the model.
- I used pipeline to avoid data leakage.
- I've looked at the results of the each model and selected the best one for the problem on hand.
- Looked in detail Catboost.
- Made hyper parameter tuning of the Catboost with Optuna to see the improvement
- Looked at the feature importance.
- Web application is made for user interface using streamlit library.

## Limitation of the System:

1. The Dataset used contains 918 observations.
2. We don't have enough data for people under age of 30 and over age of 80.
3. The project should also explain why the patient is safe or not safe.
4. The project can be easily used by doctors but normal people may find it a little hard to use.

## Future scope and further enhancement of the project:

1. More dataset can be added to get more accurate results.
2. Although we have used 17 training algorithms we can use more algorithms to train our project.
3. Web interface can become more user-friendly.

# REFRENCES


- Dataset - https://www.kaggle.com/fedesoriano/heart-failure-prediction
- I have learned Machine Learning from - https://www.kaggle.com