Heart Failure Prediction

Machine Learning

Tools used: -

- Python Language
  - ✓ numpy
  - ✓ pandas
  - ✓ sklearn
  - ✓ matplotlib
  - ✓ seaborn
  - ✓ jupyter notebook
  - ✓ kivy
  - ✓ pyinstaller

- Inno Setup Compiler
  - ✓ To Generate Setup file

DECLARATION

I the undersigned solemnly declare that the project report entitled ***'Heart Failure Prediction'*** *is based on my own work carried out during the course of B.TECH computer Science .*

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that

i. **The work contained in the report is original and has been done by me.**

ii. **The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.**

ACKNOWLEDGEMENT

*I **Jyothi Prakash Muddana** student of **Tirumala Engineering College** pursuing **Computer Science and Engineering** prepared a Mini Project titled **"2048 Game".** I sincerely express my gratitude towards **Srikanth Yadav .M** (Associate Professor Dept of CSE) . I also thankful to all the teachers who helped me to handle critical topics in the project ,I thank my friends who boasted me to complete the project successfully with out breaking it in middle*

~Jyothi Prakash Muddana

Table of Contents

1.Introduction

In this project we are going to develop a machine learning model which can predict the **Death_event of a heart failure person** by using previously stored data of different hearth failure patients and deploying the model , using it Graphical User Interface (GUI) which can be installed on windows platform , Title of the project is **"Heart Failure prediction "**

We are going to develop a machine learning model by **using python with some packages like pandas , numpy, sklearn, matplotlib** which are commonly used for analysis of data and developing Machine learning model

To Develop Graphical User Interface (GUI) we are using **kivy which is one of package in python** because of its easiness and we can use the same developed code for generating executable file for different OS like Windows, Linux, Unix, Android

To generate an executable file out of the code we are going to use a package named **pyinstaller** which is specially used to generate a specification for code and by using that specification it will generate our executable file

To make it a perfect windows application we have to generate an executable installer (Setup File) and it must show our Application in control panel, this is done by using **Inno Setup Compiler**

2. Dataset

We analyze a dataset of 299 patients with heart failure collected in year 2015. This dataset contains records of 105 Female , 194 Male and there ages are in between 40 to 95 and all the patients had previous heart failure that keeps them in class III or IV of New York Heart Association.

We downloaded this dataset from kaggle.com

Link : https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

Totally we have 12 feature of the patients as follow

Features:	Description
Age:	Age of the person
Anaemia:	Decrease of red blood cells
Creatinine Phosphokinase:	Level of CPK enzyme in blood
Diabetes:	Patient has diabetes
Ejection Fraction:	Percentage of blood leaving heart for a contraction
High Blood Pressure:	Patient has hypertension
Platelets:	Platelets in the blood
Serum Creatinine:	Level of creatinine in blood
Serum Sodium:	Level of Sodium in blood
Gender:	Male \ Female
Smoking:	Patient smokes or not
Time:	Follow Up Period

As we are going to develop a supervised Machine Learning model we have output data which is classified as (1,0) .

DEATH EVENT

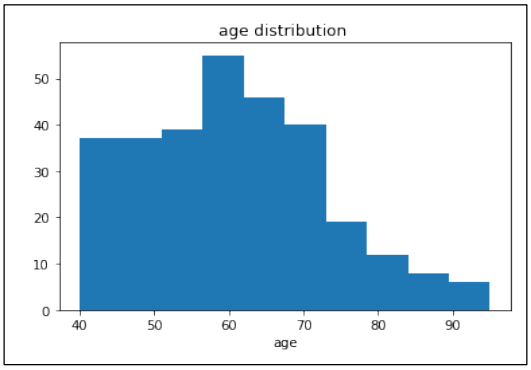
Death event is 1 ⟶ The patient died in the follow up period
Death event is 0 ⟶ The patient is live

3. Description of Dataset

3.1 Contentious valued features

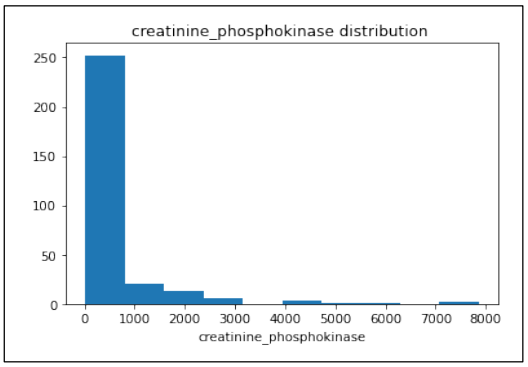
3.1.1 Age

Mean	68.83
Standard Deviation	11.90
Minimum	40
First Quartile	51
(Median)Second Quartile	60
Third Quartile	70
Maximum	95



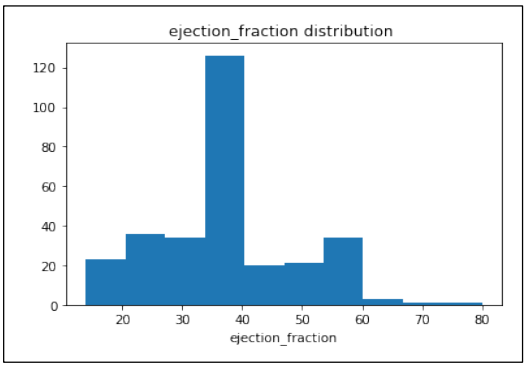
3.1.2 Creatinine Phosphokinase

Mean	581.9
Standard Deviation	570.28
Minimum	23
First Quartile	116.5
(Median)Second Quartile	250
Third Quartile	582
Maximum	7861



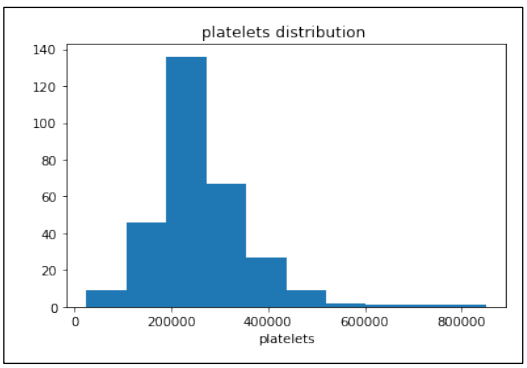
3.1.3 Ejection Fraction

Mean	38.08
Standard Deviation	11.83
Minimum	14
First Quartile	30
(Median)Second Quartile	38
Third Quartile	45
Maximum	80



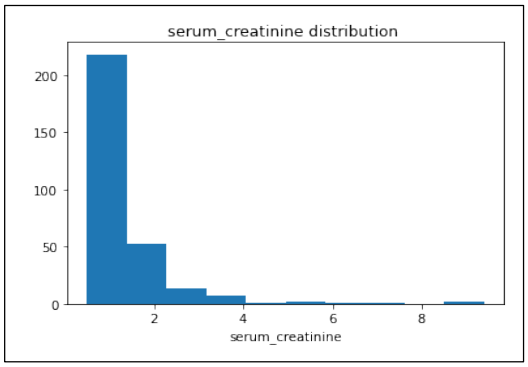
3.1.4 Platelats

Mean	263358.03
Standard Deviation	97804.24
Minimum	25100
First Quartile	212500
(Median)Second Quartile	262000
Third Quartile	303500
Maximum	850000



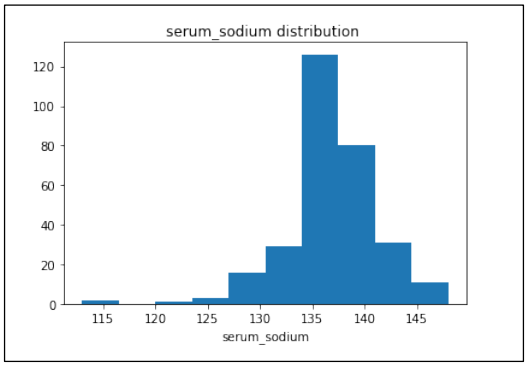
3.1.5 Serum Creatinine

Mean	1.40
Standard Deviation	1.03
Minimum	0.50
First Quartile	0.90
(Median)Second Quartile	1.10
Third Quartile	1.40
Maximum	9.40



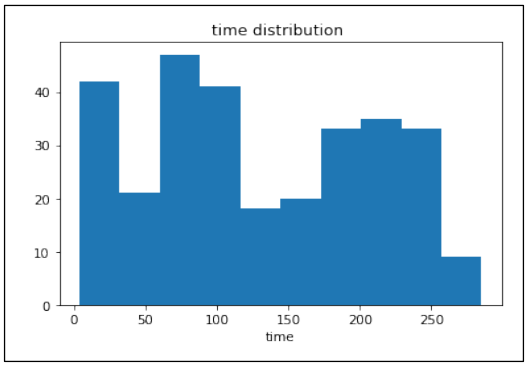
3.1.6 Serum Sodium

Mean	136.62
Standard Deviation	4.42
Minimum	113.0
First Quartile	134.0
(Median)Second Quartile	137.0
Third Quartile	140.0
Maximum	148.0



3.1.7 Time

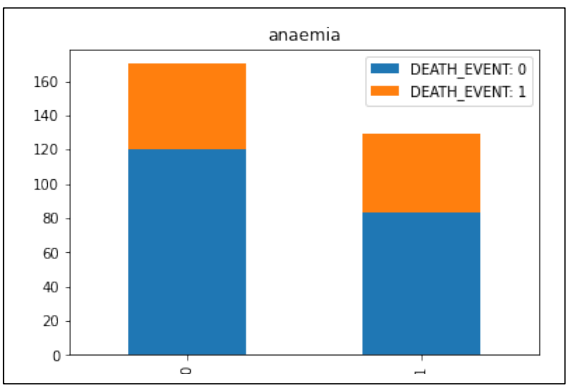
Mean	130.26
Standard Deviation	77.61
Minimum	4.0
First Quartile	73.0
(Median)Second Quartile	115.0
Third Quartile	203.0
Maximum	285.0



3.2Classification valued features

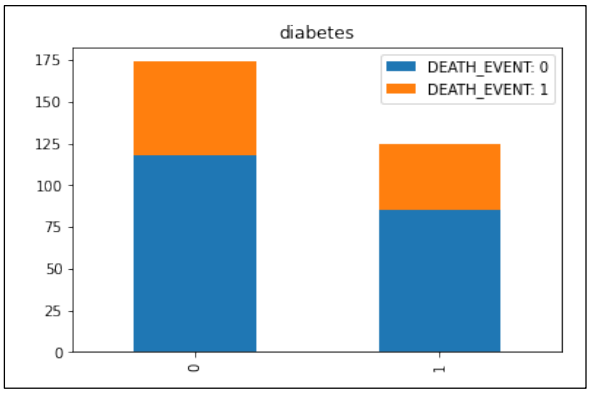
3.2.1 Anaemia

	Anaemia:0	Anaemia:1
Death_event:0	120	83
Death_event:1	50	46



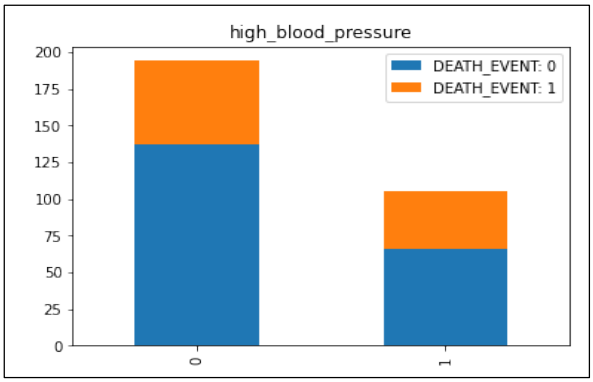
3.2.2 Diabetes

	diabetes:0	diabetes:1
Death_event:0	120	83
Death_event:1	50	46



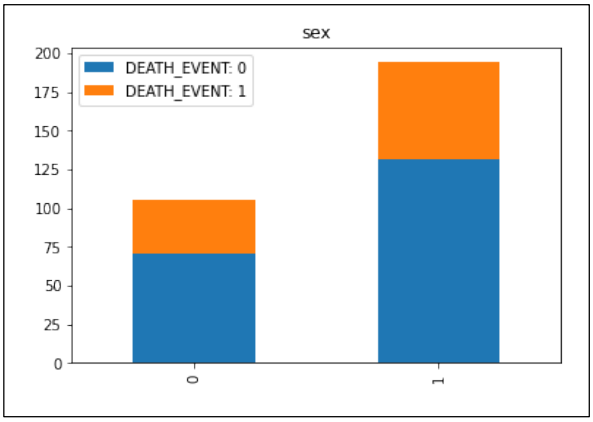
3.2.3 High Blood Pressure

	High BP:0	High BP:1
Death_event:0	118	85
Death_event:1	57	39



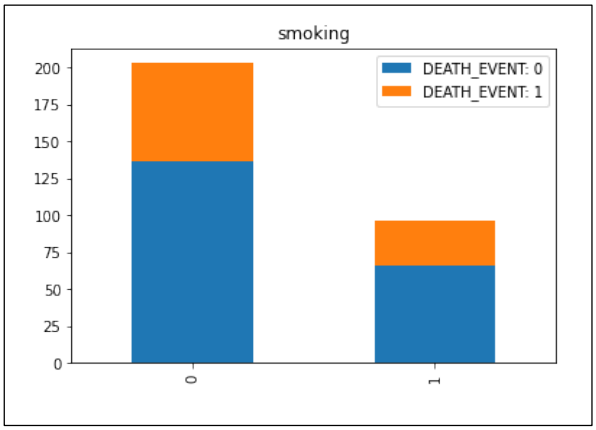
3.2.4 Gender

	Male:0	Female:1
Death_event:0	137	66
Death_event:1	57	62



3.2.5 Smoking

	Smoking:0	Smoking:1
Death_event:0	137	66
Death_event:1	66	30



4. System Requirements

System requirements are the required specifications of a device must have in order to use certain **hardware** or **Software**

- Hardware

Computer hardware refers to the physical parts of a computer and related devices. Internal hardware devices include motherboards, hard drives, and RAM. External hardware devices include monitors, keyboards, mice, printers, and scanners.

- Software

Computer software is a general term that describes computer programs. Related terms such as software programs, applications, scripts, and instruction set all fall under the category of computer software.

4.1 Hardware requirements

- ✓ Processor (minimum 2GH or More)
- ✓ Hard disk (minimum 100GB or More)
- ✓ Ram (minimum 4GB or More)
- ✓ 5 GB free diskspace
- ✓ All other Basic requirements like keyboard, Mouse etc... for a normal PC requires

4.2 Software requirements

- ✓ Windows operating system (7 or 10)
- ✓ openGL (open Graphic Language)
- ✓ Inno setup Compiler
- ✓ Python 3.6 or above version
  - ❖ Required packages
    - numpy
    - pandas
    - sklearn
    - matplotlib
    - seaborn
    - jupyter notebook
    - kivy
    - pyinstaller

5. Theoretical background

5.1 Machine Learning

The Following steps are involved in development of Machine Learning Model

- Variable Identification
- Uni-variate Analysis
- Bi-variate Analysis
- Missing Values Treatment
- Outlier Treatment
- Feature Ranking
- Different Model Generation
- Selecting best model

5.1.1 Variable Identification

- ✓ Understand the variables and the type of data for each variable
- ✓ We need to identify predictor variables,target variable,data types and category of variables

  Dependent Variable : DEATH_EVENT

  Independent Variables are :

  - ✧ Age
  - ✧ Anaemia
  - ✧ Creatinine Phosphokinase
  - ✧ Diabetes
  - ✧ Ejection Fraction
  - ✧ High Blood Pressure
  - ✧ Platelets
  - ✧ Serum Creatinine
  - ✧ Serum Sodium
  - ✧ Gender
  - ✧ Smoking
  - ✧ Time

Data Types

All the variables are numeric in the given data set

Variable Category

Continuous:

1. Age
2. Creatinine Phosphokinase
3. Ejection Fraction
4. Platelets
5. Serum Creatinine
6. Serum Sodium
7. Time

Categorical :

1. Anaemia
2. Diabetes
3. High Blood Pressure
4. Gender
5. Smoking
6. DEATH EVENT

5.1.2 Uni-variate Analysis

1. Uni-variate analysis is the simplest form of analyzing data. **"Uni" means "one"**, so we analyze one variable at one time

2. It doesn't deal with causes or relationships among variables but mostly to describe and summarize and find patterns in the data

3. Used to highlight missing and outlier values

4. Method to perform univariate analysis depends on whether the variable type is categorical or continuous
   - ✓ For Categorical Variables we use bar plots
   - ✓ For Continuous Variable we use hist plot

5.1.3 Bi-variate Analysis

1. In Uni-variate Analysis, we study one variable at a time, like we did in earlier slides, but if we want to find if there is any relation between two variables we need to perform bivariate analysis.

2. Bi-variate analysis, can be performed Categorical & Categorical for any combination of categorical and continuous variables

3. Different methods are used to tackle different combinations during analysis process.

4. Possible Combinations are:

- Continuous & Continuous
- Continuous & Categorical
- Categorical & Categorical

5.1.4 Missing Values Treatment

1. There may be situations where there could be missing values in your data.

2. Missing Data will not make any impact on the result if its percentage is less 1%, if missing data range within the range of 1-5% then it is somehow manageable; however in case of 5-15% complex techniques are used for handling the problems of missing data but if it exceeds from 15% then it will surely hinder the result achieved after applying data mining techniques

3. Handling such values is very important as this could lead to wrong results

4. Missing values could occur due to several reasons like,

- During data extraction i.e. while fetching the data required for the analysis
- During data collection itself there could be some fields for which the values may not have been collected.

5. But there are ways to handle these problems

- Replace the missing values with nearest possible value by using the other features of the record
- Drop those records which has missing values

5.1.5 Outlier Treatment

1. Outlier is an observation that appears far away and diverges from an overall pattern in a sample.

2. Outliers can drastically change the results of the data analysis and statistical modeling. There are numerous unfavorable impacts of outliers in the data set:

- It increases the error variance and reduces the power of statistical tests
- If the outliers are non-randomly distributed, they can decrease normality
- They can bias or influence estimates that may be of substantive interest

Causes of Outliers:

1. **Data Entry Errors** - Human errors such as errors caused during data collection, recording, or entry can cause outliers in data.
2. **Measurement Error** - When the measurement instrument used turns out to be faulty.
3. **Intentional Error** - This is commonly found in self-reported measures that involves sensitive data.
4. **Data Processing Error** - When data is collected from different sources
5. **Sampling Error** - Data considered which is not part of the sample
6. **Natural Outlier** - When an outlier is not artificial (due to error), it is a natural outlier.

Outlier Detection

Outliers can be detected using box plots and scatter plots

Other than the plots, Outliers can also be detected by using certain thumb rules

- Any value, which is beyond the range of -1.5x IQR to 1.5 x IQR where IQR = Q3-Q1
- Any value which out of range of 5th and 95th percentile can be considered as outlier
- Data points, three or more standard deviation away from mean are considered outlier

Handle Outliers

1. We could remove the outliers from the data if they are due to data entry or data processing errors
2. Based on business understanding you could also replace the outliers with mean or median
3. If there is a pattern of interest in the outliers then they could be handled separately. For example if the outliers are like in groups then treat both groups as two different groups and build individual model for both groups and then combine the output
4. Also the outliers can be capped with 5th or 95th percentile

5.1.6 Feature Ranking

When building a machine learning model in real-life, it's almost rare that all the variables in the dataset are useful to build a model. Adding redundant variables reduces the generalization capability of the model and may also reduce the overall accuracy of a classifier. Furthermore adding more and more variables to a model increases the overall complexity of the model.

Feature Selection in machine learning are classified as follow :

- Filter methods
- Wrapper methods
- Embedded methods
- Hybrid methods

In our Machine Learning model we will go for Filter methods

Some Filter methods are :

- **Information Gain**

  It calculates the reduction in entropy from the transformation of dataset .we can use this for feature selection by evaluating each variable in the context of the output

- **Chi - Square Text**

  In this we calculate the chi-square between each feature with the targeted feature

  1. **The variables has to be categorical, sampled independently**
  2. **It must have expected frequency greater than 5**

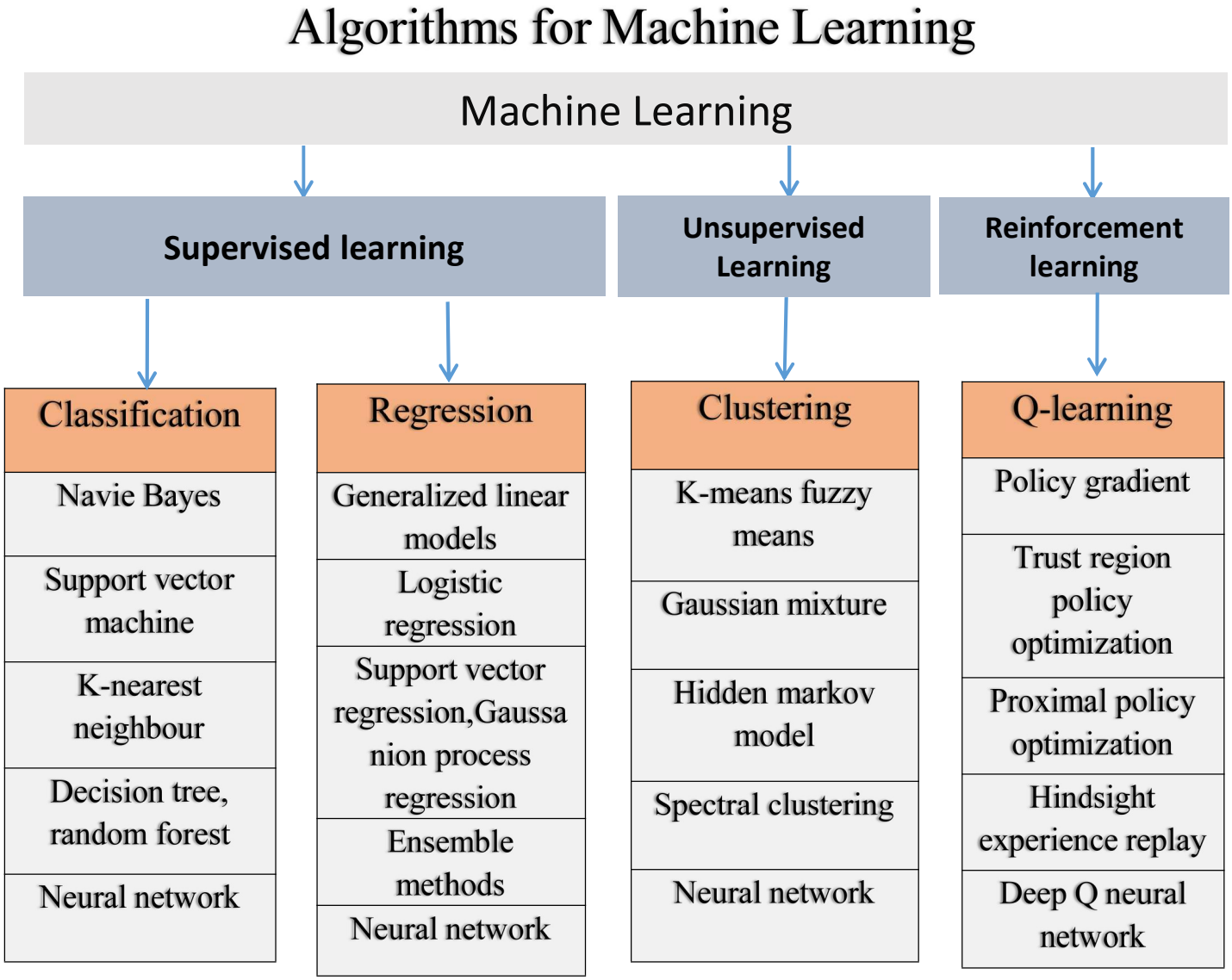
- **Fisher's Score**

  Fisher Score is one of the most widely used supervised feature selection methods. The rank of the variable is depends up on the fisher's score descending order

- **Correlation Coefficient**

  Correlation is a measure of the linear relationship of 2 or more variables. We can find one variable fomr the other by using correlation for features selection which has high correlation withe the target

5.1.7 Different Model Generation



We can use different algorithms to develop a model but in our case we need to develop a Classification model

So we can use following algorithms to develop our model

- ✓Navie Bayes
- ✓Support vector machine
- ✓K-nearest neighbour
- ✓Decision tree
- ✓Random forest

5.1.8 Selecting best model

Basically we need to select the model which give result more accurately ,We need to compare models with each other

We are working on a model which targets the classification variable (DEATH EVENT) to compare the values we need to generate the confusion matrix for the real and predicted values of the target variable

Confusion matrix really helps us to find how the model is working such as finding how frequently the model making mistakes on a specific categorical value in the targeted out put and accuracy is calculated by using following formula

$$\text{Accuracy score} = (f_{00}+f_{11}) / (f_{00}+f_{10}+f_{01}+f_{11})$$

$$\text{Where confusion matrix} = \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix}$$

As we have only two categorical values in our target feature so we have only 2x2 matrix

5.2 Python

What is python, its importance and usage

Python is a popular programming language. It was Developed by ***Guido van Rossum***, and released in 1991.

Now a days we can see its foot prints in web development (server side), software development, Data Science, Artificial Intelligence, etc…

Most import features of python

- Readable and Maintainable Code
- Multiple Programming Paradigms
- Compatible with major platforms and systems
- Robust standard library
- Many open source frameworks and tools
- Simplify complex software development
- Adopt text driven development

How to Install python

We can Install python on many Operating Systems such as Windows, Linux/Unix, Mac OS and others

To download python installers, use below link
https://www.python.org/downloads/

This is official python website and it will detect he operating system and based on that it will recommend you to download python

You can install python in u computer just like an application but make sure to give a checkmark to add paths to the system while installing itself will help you to install other required package using pip in command prompt

Packages that are required for this project

- numpy
- pandas
- sklearn
- matplotlib
- seaborn
- jupyter notebook
- kivy
- pyinstaller

5.2.1 numpy

numpy is a library from python programming language, adding support for large, Multidimensional arrays and matrices, along with the large collection of high-level mathematical function to operate on these arrays

It is a project by a Community release in 1995 as Numeric later in 2006 rename it as numpy and written in python, c It contains all prep-compiled functions makes its speeder than other basic functions of python

To install this library, use the following command in CMD

>pip install numpy

5.2.2 Pandas

Pandas is an open source python package that is most widely used for data science / data analysis and machine learning tasks . it is built on top of another package named numpy, which provides support for multi-dimensional arrays.

By using pandas we can do following

- ✓ Data cleaning
- ✓ Data normalization
- ✓ Data visualization
- ✓ Statistical analysis
- ✓ Data inspection
- ✓ Loading and saving data

To install this library, use the following command in CMD

>pip install pandas

5.2.3 Matplotlib

Matplotlib is a cross-platform, data virtualization and graphical plotting library for python and its numerical extension numpy.

A python matplotlib scripts is structured so that a few lines of code are all that is required in most instances to generate a visual data plot.

We are going to use this module to develop some graphic which are required to analyse the data

Some of the plots are :

- ✓Bar plot
- ✓Hist plot
- ✓Heat map
- ✓Scatter plot

To install this package use the below command in CMD
>pip install matplotlib

5.2.4 Seaborn

Seaborn is a python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics . Seaborn helps us to explore and understand your data . It plotting functions operate on dataframes and arrays containing whole datasets and internally performs the necessary semantic mapping and statistical aggregation to produce informative plots.

To install thie packages use this command in CMD
>pip install seaborn

5.2.5 Sklearn

The sklearn is well know as SciKit-learn python library , It is commonly used in solving the machine learning and data science problems form begining to the end .

It is built on the bases of libraries like numpy, scipy, matplotlib

Sklearn offers and extensive range of built-in algorithms that make the most of the data science project

Classification algorithms in sklearn includes

- ✓ Support vector machines (SVM)
- ✓ Nearest neighbors
- ✓ Random forest

Model selection algorithms

- ✓ Grid search
- ✓ Cross validation
- ✓ metrics

5.2.6 Jupyter notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text

Jupyter notebook are most widely used for machine Learning , Data analysis works, it helps us to document our code which coding itself

To install this use the following command
>pip isstall jupyter
To run the server for jupyter
>jupyer notebook

5.2.7 Kivy

kivy is an open source multi-platform GUI development library for Python and can run on iOS, Android, Windows, OS X, and GNU/Linux. It helps develop applications that make use of innovative, multi-touch UI

The fundamental idea behind kivy is to enable the developer to build and app once tand use it across all devices , making the code reusable and deplorable, allowing for quick and easy interaction design and rapid prototyping.

This is an easy to use frame work

- ➢ Extensive input support for input devices such as mouse, keyboard
- ➢ A graphic library using only openGL
- ➢ A wide range of widgets built with mutli-touch support
- ➢ An intermediate language Kv language , used to design cutsom widgets easily

To install kivy package use this command
>pip install kivy

5.2.8 pyinstaller

It freezes (packages) python applications into stand-alone executable under windows,GNU/LINUX,Mac Os X, …

The main goal of PyInstaller is to be **compatible with 3rd-party packages out-of-the-box**. This means that, with PyInstaller, all the required tricks to make external packages work are already **integrated within PyInstaller itself** so that there is no user intervention required. You'll never be required to look for tricks in wikis and apply custom modification to your files or your setup scripts. As an example, **libraries like PyQt, Django or matplotlib are fully supported**, without having to handle plugins or external data files manually.

To install this library, use the following command in CMD

>pip install pyinstaller

I am going to use this module to generated and exe file out of python code

5.3 InnoSetupCompiler

Inno Setup is a *free* installer for Windows programs by Jordan Russell and Martijn Laan. First introduced in 1997, Inno Setup today rivals and even surpasses many commercial installers in feature set and stability.

Key features:

- Support for every Windows release since 2006, including: Windows 10, Windows 10 on ARM, Windows Server 2019, Windows Server 2016, Windows 8.1, Windows 8, Windows Server 2012, Windows 7, Windows Server 2008 R2, Windows Server 2008, and Windows Vista. (No service packs are required.)

- Extensive support for installation of **64-bit applications** on the 64-bit editions of Windows. The x64, ARM64 and Itanium architectures are all supported.

- Extensive support for both administrative and non administrative installations.

- Supports creation of a **single EXE** to install your program for easy online distribution. Disk spanning is also supported.

- Standard Windows wizard interface.

- **Customize setup types**, e.g. Full, Minimal, Custom.

- Complete **uninstall** capabilities.

- Installation of files:
  Includes integrated support for "deflate", bzip2, and **7-Zip LZMA/LZMA2 file compression**. The installer has the ability to compare file version info, replace in-use files, use shared file counting, register DLL/OCX's and type libraries, and install fonts.
- Creation of shortcuts anywhere, including in the Start Menu and on the desktop.

- Creation of registry and .INI entries.

- Running other programs before, during or after install.

- Support for **multilingual** installs, including right-to-left language support.

- Support for pass-worded and encrypted installs.

- Support for **digitally signed** installs and uninstalls, including dual signing (SHA1 & SHA256).

Use below link to download and install the Inno setup compiler (ctrl+click)

https://jrsoftware.org/download.php/is.exe

6. Code Documentation

For a programmer reliable documentation is always a must. The presence of documentation helps keep track of all aspects of an application and it improves on the quality of a software product. Its main focuses are development, maintenance and knowledge transfer to other developers.

Successful documentation will make information easily accessible, provide a limited number of user entry points, help new users learn quickly, simplify the product and help cut support costs. Documentation is usually focused on the following components that make up an application: server environments, business rules, databases/files, troubleshooting, application installation and code deployment.

The code documentation is the backbone of every application. Code documentation can be split in multiple parts. The first one, the most helpful for programmers are the comment blocks. These will be found through every file explaining classes, methods, parameters, possible errors. Then comes the specific file documentations. These are usually generated through a third party script which will parse a file and, based on the comment blocks, will create an explicit PDF. Afterwards there should be information regarding the code repository, where the file updates are found, and where they need to be moved. In addition, there should be step-by-step instructions on how to create an application package or a build to be deployed.

In Python to document a code by default we have a module called pydoc

6.1 Pydoc

This is in build available in python

Pydoc is the standard documentation module for the programming language Python. Pydoc is used to extract documentation from the source code itself. ... More comprehensive documentation is generated from external restructured-text documents using the Sphinx documentation system.

Out of our comments and import information in our it generates its documentation in text format if need we can get the information in HTML also

6.2 How to document code

It is very easy by using pydoc just use (“““ ””” or ‘‘‘ ’’’) use either of these two with in the function or class after defining it and write its importance point like logic, about data, interconnections etc …

Eg :-

```
def add(a,b):
    ‘‘‘ This function is used to add two numbers
        And returns its sum
    ’’’
    return a+b
```

This part comes under documentation

6.3 Local Host (Document view CMD)

To view the documentation of the code use command prompt and follow these steps

>pydoc

‘This command will help use to know extra details about this module and some arguments that we can pass and how these arguments effect the output’

```
pydoc - the Python documentation tool

pydoc-script <name> ...
    Show text documentation on something.    <name> may be the name of a
    Python keyword, topic, function, module, or package, or a dotted
    reference to a class or function within a module or module in a
    package.    If <name> contains a '\', it is used as the path to a
    Python source file to document. If name is 'keywords', 'topics',
    or 'modules', a listing of these things is displayed.

pydoc-script -k <keyword>
    Search for a keyword in the synopsis lines of all available modules.

pydoc-script -n <hostname>
    Start an HTTP server with the given hostname (default: localhost).

pydoc-script -p <port>
    Start an HTTP server on the given port on the local machine.    Port
    number 0 can be used to get an arbitrary unused port.

pydoc-script -b
    Start an HTTP server on an arbitrary unused port and open a Web browser
    to interactively browse documentation.    This option can be used in
    combination with -n and/or -p.

pydoc-script -w <name> ...
    Write out the HTML documentation for a module to a file in the current
    directory.    If <name> contains a '\', it is treated as a filename; if
    it names
```

1. Directly view the doc in command prompt
2. In browser (HTML)
3. Save the HTML file in local directory

6.3.1 Directly view in Command Prompt

dir>pydoc <class or function name>

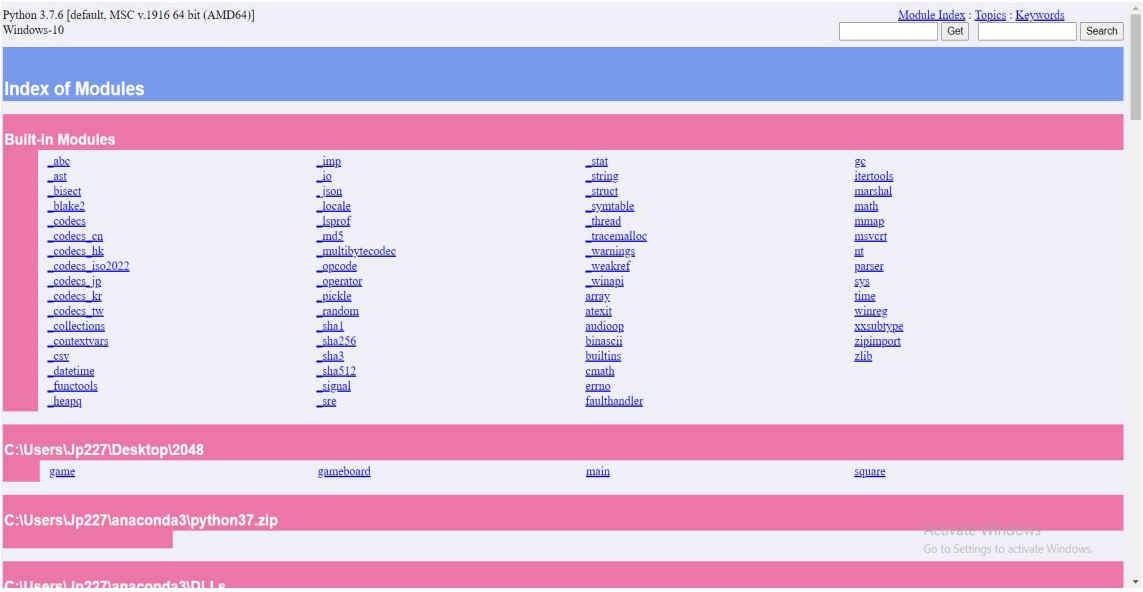
Eg:-

```
C:\Users\Jp227\Desktop\2048>pydoc random.shuffle
Help on method shuffle in random:

random.shuffle = shuffle(x, random=None) method of random.Random instance
    Shuffle list x in place, and return None.

    Optional argument random is a 0-argument function returning a
    random float in [0.0, 1.0); if it is the default None, the
    standard random.random will be used.
```

6.3.2 In browser (HTML)

dir>pydoc -b or dir>python -m pydoc - b

Browser



6.3.3 Save the HTML File in Local Directory

dir > pydoc -w <Name of the python file>

It will create a html file with same python file name and exaltation in .html

7. Jupyter notebook (Machine Learning)

Heart Failure

In[1]:

```
get_ipython().run_cell_magic('javascript', '',
'IPython.OutputArea.auto_scroll_threshold = 9999')
```

importing important packages/modules

In[2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In[3]:

```
DataF = pd.read_csv("heart1.csv")
```

viewing first 5 rows of data

In[4]:

```
DataF.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1

information about data available

In[5]:

```
DataF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

Description of Data

In[6]:

```
DataF.describe()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.000000	299.00000	299.000000	299.00000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418	0.648829	0.32107	130.260870	0.32107
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477	0.478136	0.46767	77.614208	0.46767
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000	0.000000	0.00000	4.000000	0.00000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000	0.000000	0.00000	73.000000	0.00000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000	1.000000	0.00000	115.000000	0.00000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000	1.000000	1.00000	203.000000	1.00000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000	1.000000	1.00000	285.000000	1.00000

Checking for null values

In[7]:

```
print('{:38}{:7}'.format("Features","Count of Nan"))
print()

for col in DataF.columns:
    print('{:35} : {:7}'.format(col,DataF[col].isna().sum()))
```

```
Features                            Count of Nan

age                                 :       0
anaemia                             :       0
creatinine_phosphokinase            :       0
diabetes                            :       0
ejection_fraction                   :       0
high_blood_pressure                 :       0
platelets                           :       0
serum_creatinine                    :       0
serum_sodium                        :       0
sex                                 :       0
smoking                             :       0
time                                :       0
DEATH_EVENT                         :       0
```

hist plot

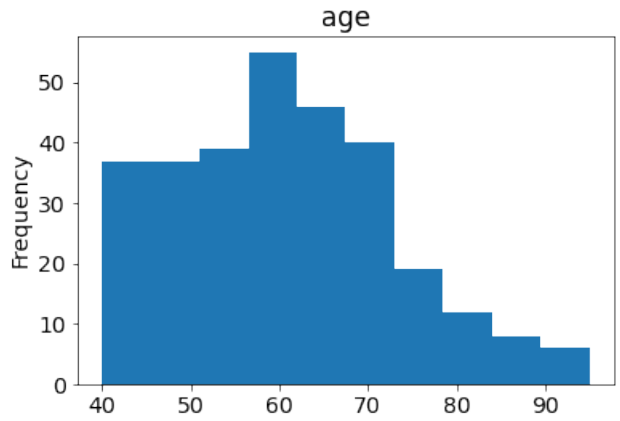
In[8]:

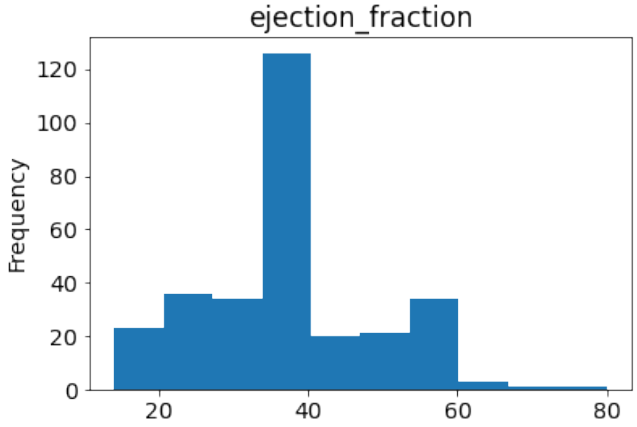
```
def hist_plot(feature):
    DataF[feature].plot(kind="hist")
    plt.title(f"{feature}")
    plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+feature+"_hist"+'.png', transparent=True)
    plt.show()
```

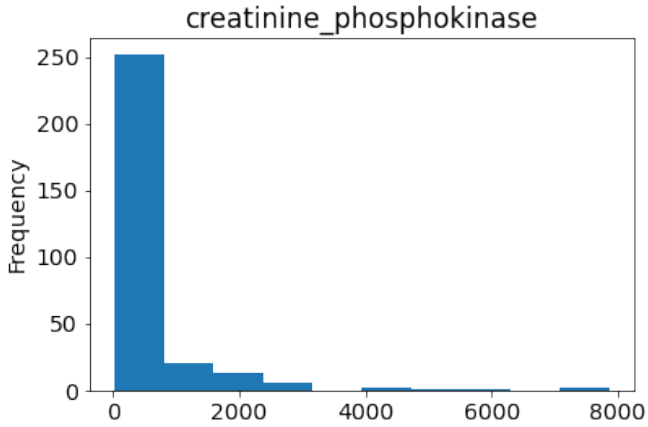
using above function to generate graphs

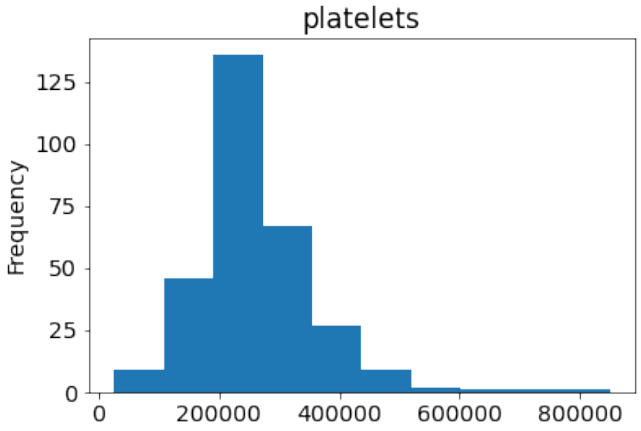
In[9]:

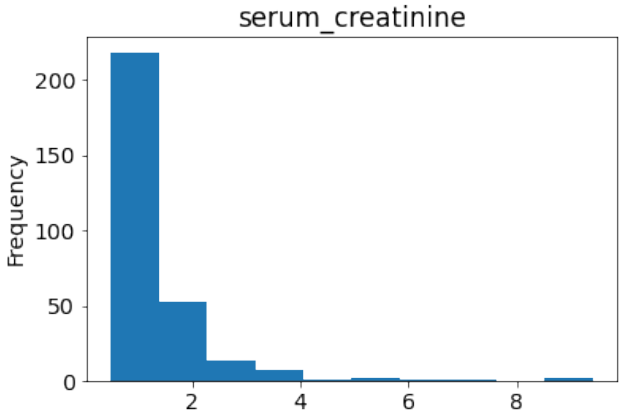
```
features =
["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_creatinine","serum_sodium","time"]
plt.rcParams.update({'font.size': 14})
for feature in features:
    hist_plot(feature)
```

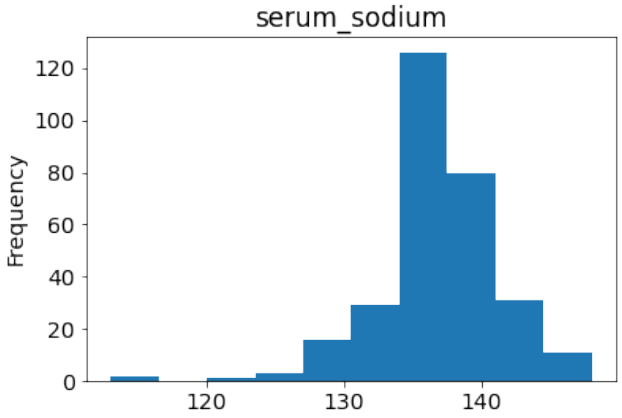


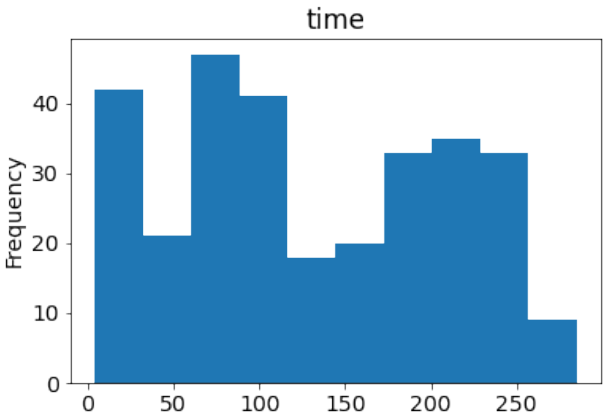





platelets
Frequency
125
100
75
50
25
0
0
200000
400000
600000
800000


serum_creatinine
Frequency
200
150
100
50
0
2
4
6
8


serum_sodium
Frequency
120
100
80
60
40
20
0
120
130
140


time
Frequency
40
30
20
10
0
0
50
100
150
200
250

function for Categorical values bar plit

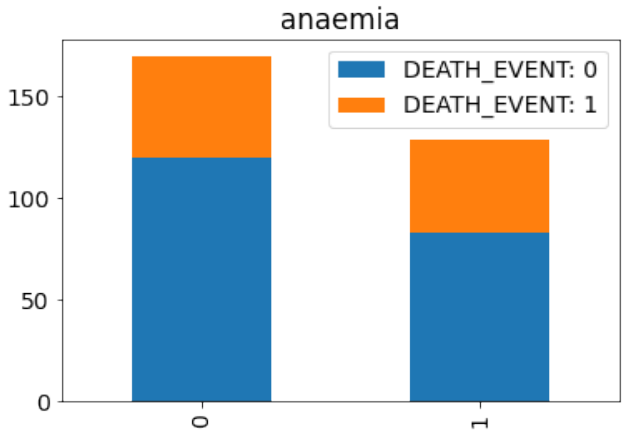
In[10]:

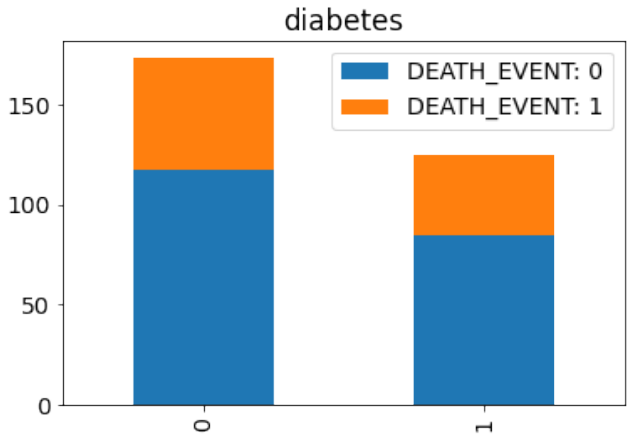
```
def barPlot(cat):
        DistValues = DataF[[cat,"DEATH_EVENT"]].value_counts()
        List =pd.DataFrame({"DEATH_EVENT:
0":[DistValues[(0,0)],DistValues[(1,0)]],"DEATH_EVENT:
1":[DistValues[(0,1)],DistValues[(1,1)]]})
        List.plot(kind='bar', stacked=True,title = cat)
        plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+cat+"_bar"+'.png', transparent=True)
        plt.show()
```

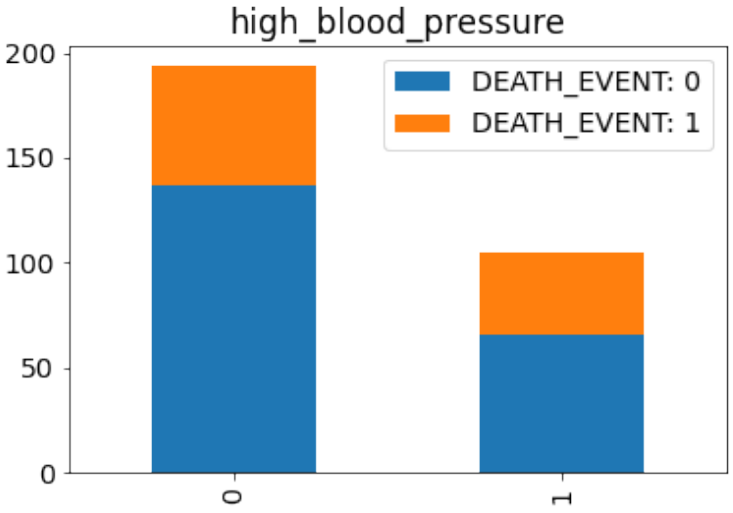
Bar plots for Catagorical data

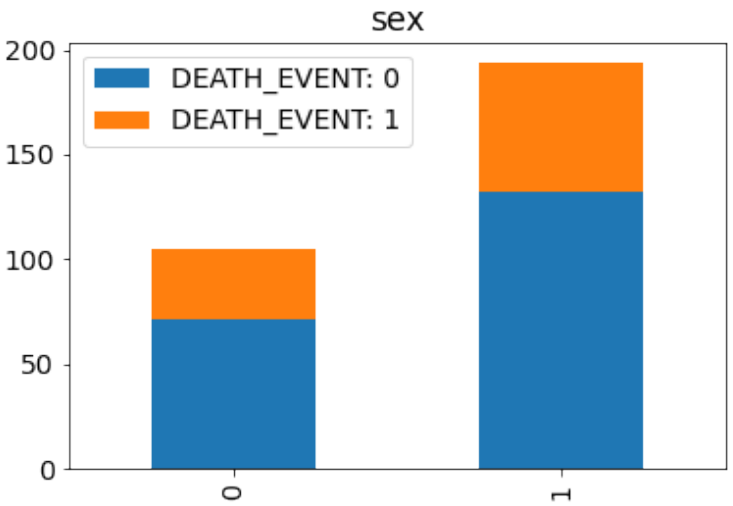
In[11]:

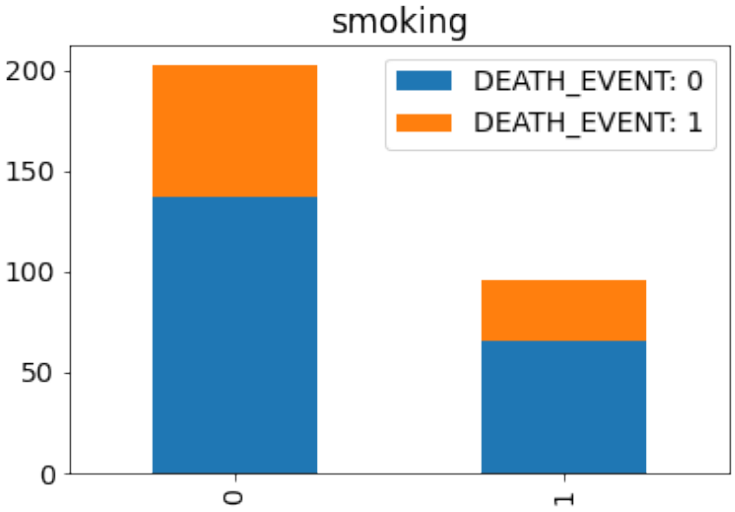
```
cat = ["anaemia","diabetes","high_blood_pressure","sex","smoking"]
plt.rcParams.update({'font.size': 14})
for i in cat:
        barPlot(i)
```






high_blood_pressure
DEATH_EVENT: 0
DEATH_EVENT: 1
200
150
100
50
0
0
1

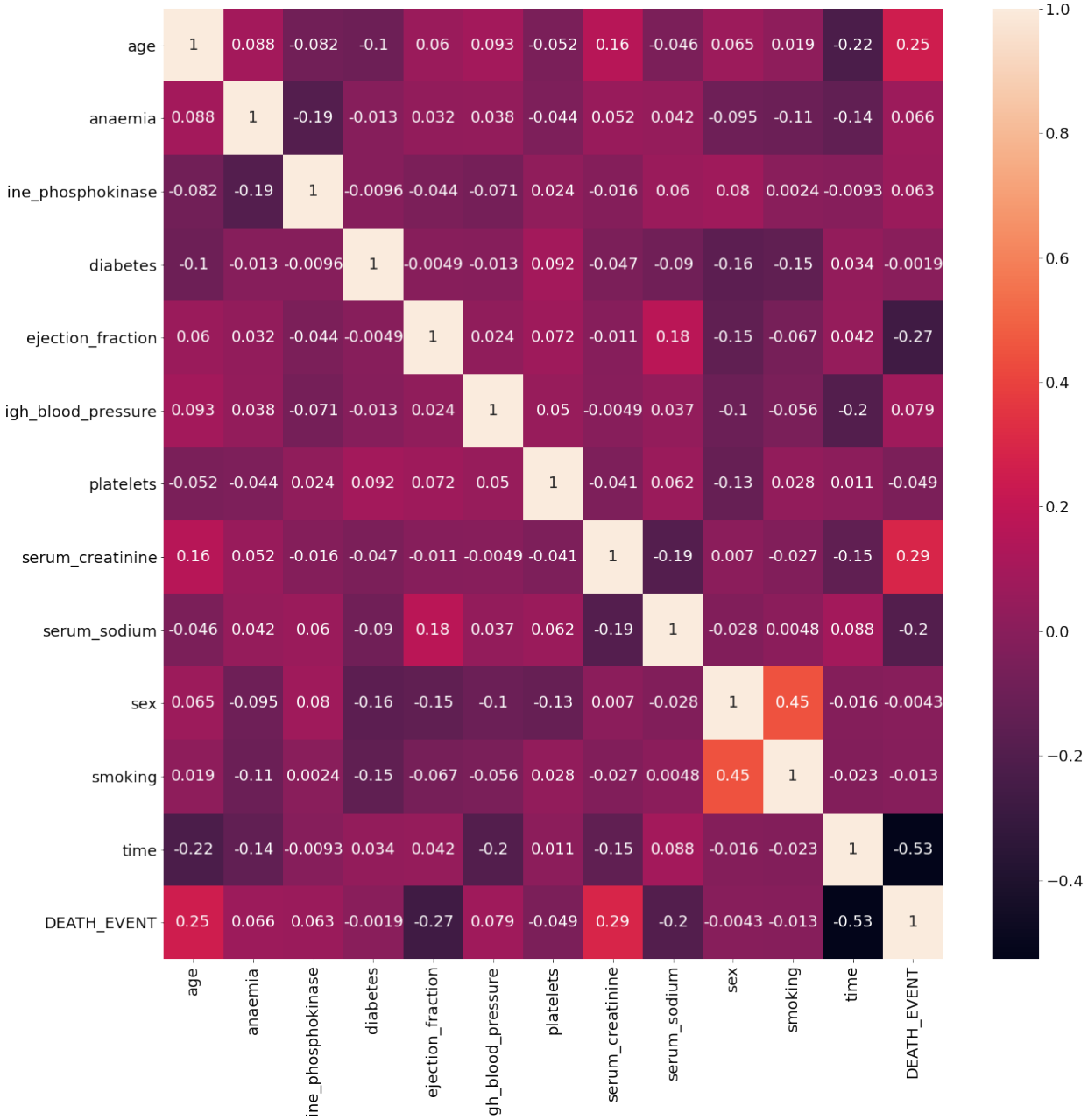

sex
DEATH_EVENT: 0
DEATH_EVENT: 1
200
150
100
50
0
0
1


smoking
DEATH_EVENT: 0
DEATH_EVENT: 1
200
150
100
50
0
0
1

heatmap for correlation between features

In[12]:

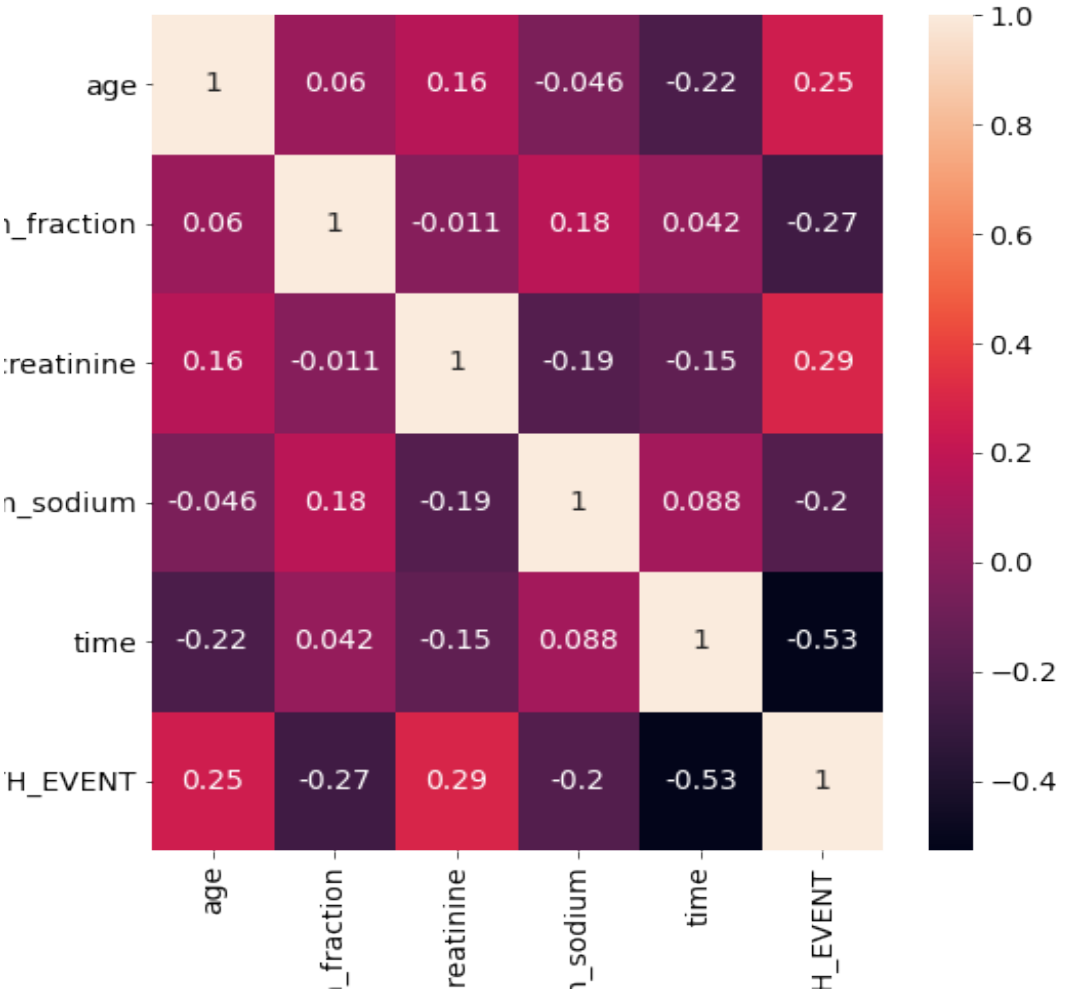
```
plt.figure(figsize=(20,20))
plt.rcParams.update({'font.size': 18})
sns.heatmap(DataF.corr(), annot = True)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"corr"+'.png', transparent=True)
plt.show()
```



heatmap for features whose correlation in >0.1

In[13]:

```
matrix = DataF.corr()
plt.rcParams.update({'font.size': 14})
base = 0.1

features = np.abs(matrix["DEATH_EVENT"]) > base
corr_features = list(matrix.columns[features])

plt.figure(figsize=(8,8))
sns.heatmap(DataF[corr_features].corr(), annot = True)

plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"corr_imp"+'.png', transparent=True)
plt.show()
```



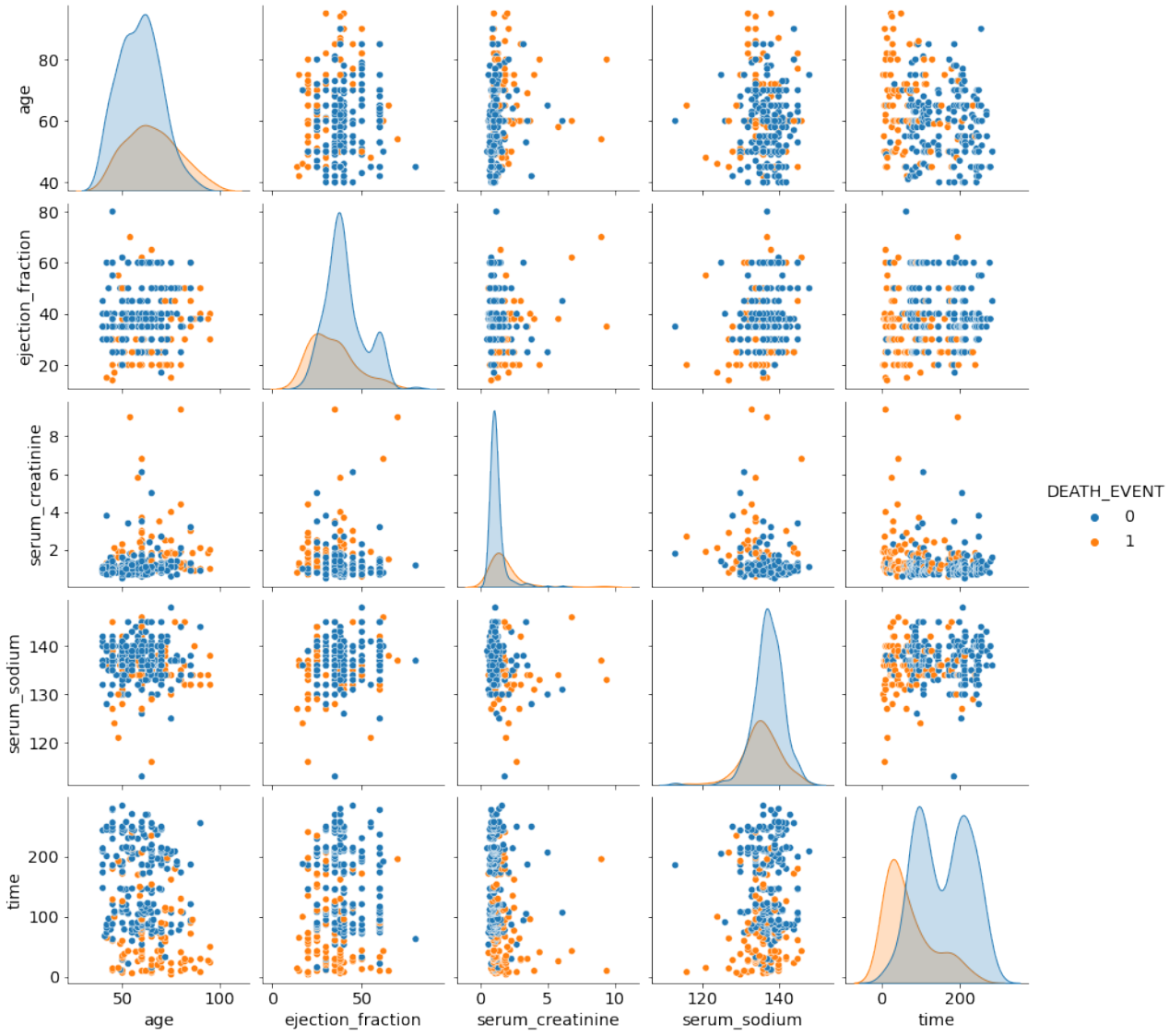
In[14]:

```
importent_features = ["age", "ejection_fraction", "serum_creatinine",
"serum_sodium","time"]
```

Pair Plot with hue = Death_event to compare two features

In[15]:

```
sns.pairplot(DataF[corr_features],hue = "DEATH_EVENT")
plt.rcParams.update({'font.size': 14})
plt.figure(figsize = (30,30))
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"pairplot"+'.png', transparent=True)
plt.show()
```



Data preprocessing

Function to return number of outliers

In[16]:

```
def outliers(x):
    q1 = np.percentile(x,25)
    q3 = np.percentile(x,75)
    iqr = q3-q1
    floor = q1 - 1.5*iqr
    ceiling = q3 + 1.5*iqr
    leng =len(x[(x<floor)|(x>ceiling)])
    return leng
```

Using above function to find number of outliers

In[17]:

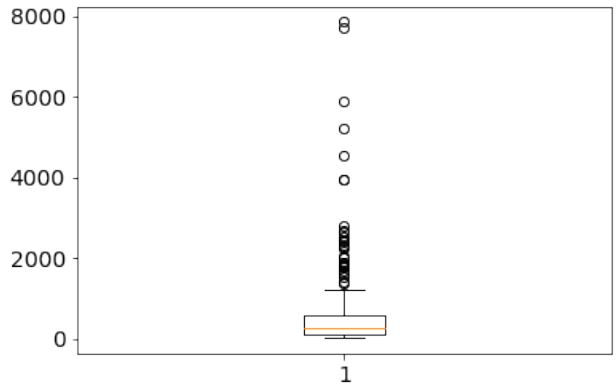
```
for col in DataF.columns:
    print('{:35} : {:5}'.format(col,outliers(DataF[col])))
```

```
age                                 :     0
anaemia                             :     0
creatinine_phosphokinase            :    29
diabetes                            :     0
ejection_fraction                   :     2
high_blood_pressure                 :     0
platelets                           :    21
serum_creatinine                    :    29
serum_sodium                        :     4
sex                                 :     0
smoking                             :     0
time                                :     0
DEATH_EVENT                         :     0
```

max 7 outliers indices in Creatinine Phosphokinase

In[18]:

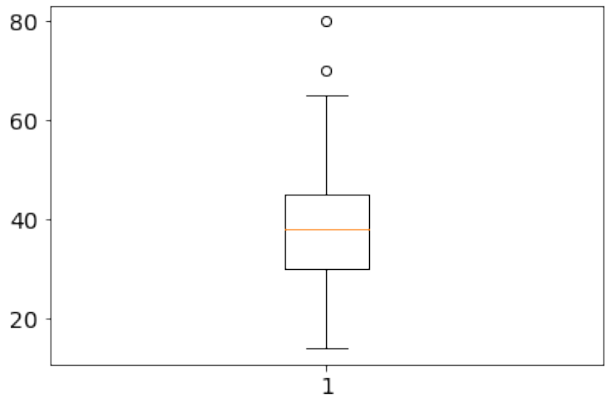
```
plt.boxplot([DataF["creatinine_phosphokinase"]])
indices_cp = DataF.sort_values(by = "creatinine_phosphokinase").tail(7)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"boxplot_cp"+'.png', transparent=True)
```



max 2 outliers indices in Ejection Fraction

In[19]:

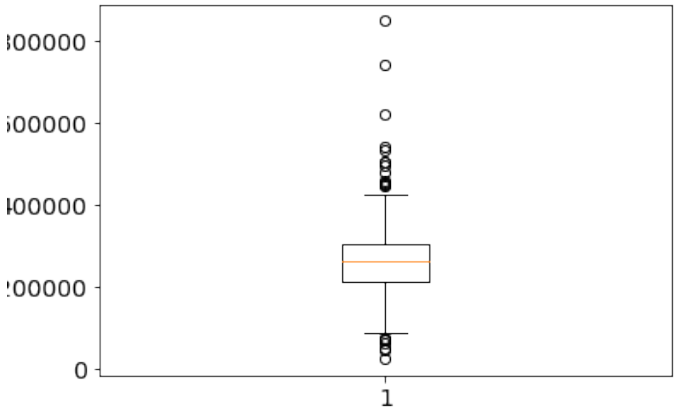
```
plt.boxplot([DataF["ejection_fraction"]])
indices_ef = DataF.sort_values(by = "ejection_fraction").tail(2)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"boxplot_ej"+'.png', transparent=True)
```



max 3 outliers indices in Platelets

In[20]:

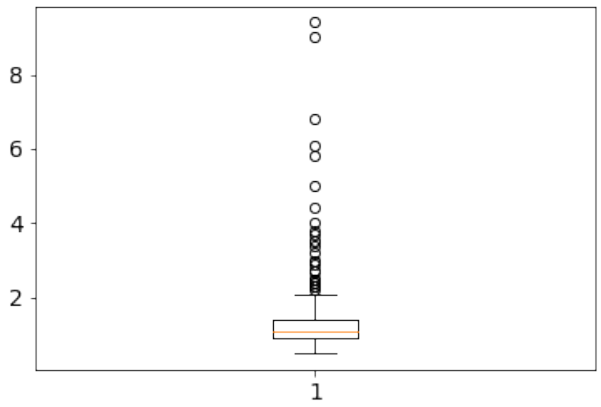
```
plt.boxplot([DataF["platelets"]])
indices_p = DataF.sort_values(by = "platelets").tail(3)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"boxplot_p"+'.png', transparent=True)
```



max 8 outliers indices in serum creatinine

In[21]:

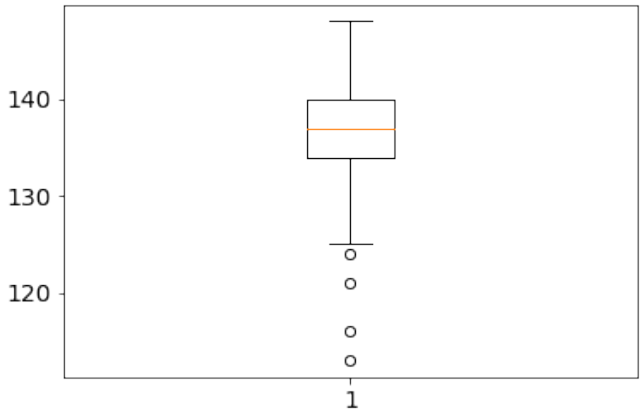
```
plt.boxplot([DataF["serum_creatinine"]])
indices_sc = DataF.sort_values(by = "serum_creatinine").tail(8)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"boxplot_sc"+'.png', transparent=True)
```



min 4 outliers indices in serum sodium

In[22]:

```
plt.boxplot([DataF["serum_sodium"]])
indices_ss = DataF.sort_values(by = "serum_sodium").head(4)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"boxplot_ss"+'.png', transparent=True)
```



Total no of outliers and there indices

In[23]:

```
index = []
for i in
[indices_ef.index,indices_ss.index,indices_sc.index,indices_cp.index,indices_p.index]:
    index.extend(list(i))
l1 = sorted(set(index))
print(len(l1))
print(l1)
```

```
22
[1, 4, 9, 10, 19, 28, 48, 52, 60, 64, 72, 103, 105, 109, 126, 131, 1
34, 171, 199, 217, 228, 296]
```

Function to find indices of outliers

In[24]:

```
def detected_outliers_index(x):
    q1 = np.percentile(x,25)
    q3 = np.percentile(x,75)
    iqr = q3-q1
    floor = q1 - 1.5*iqr
    ceiling = q3 + 1.5*iqr
    outliers = x[(x<floor)|(x>ceiling)]
    return list(outliers.index)
```

Records to find more than 2 outliers

In[25]:

```
m_index = []
for i in DataF.columns:
    m_index.extend(detected_outliers_index(DataF[i]))
```

```
l2 = []
for i in m_index:
        if m_index.count(i)>1 and i not in l2:
                l2.append(i)
print(sorted(l2))
l2 = sorted(l2)

indices = sorted(set(l1+l2))

print((DataF.iloc[indices])["DEATH_EVENT"].value_counts())

DataF.iloc[indices]
```

```
[4, 38, 52, 117, 163, 167, 200, 217, 281, 296]
1     15
0     13
Name: DEATH_EVENT, dtype: int64
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
1	55.0	0	7861	0	38	0	263358.03	1.10	136	1	0	6	1
4	65.0	1	160	1	20	0	327000.00	2.70	116	0	0	8	1
9	80.0	1	123	0	35	1	388000.00	9.40	133	1	1	10	1
10	75.0	1	81	0	38	1	368000.00	4.00	131	1	1	10	1
19	48.0	1	582	1	55	0	87000.00	1.90	121	0	0	15	1
28	58.0	1	60	0	38	0	153000.00	5.80	134	1	0	26	1
38	60.0	0	2656	1	30	0	305000.00	2.30	137	1	0	30	0
48	80.0	1	553	0	20	1	140000.00	4.40	133	1	0	41	1
52	60.0	0	3964	1	62	0	263358.03	6.80	146	0	0	43	1
60	45.0	0	7702	1	25	1	390000.00	1.00	139	1	0	60	1
64	45.0	0	582	0	80	0	263358.03	1.18	137	0	0	63	0
72	85.0	0	5882	0	35	0	243000.00	1.00	132	1	1	72	1
103	42.0	0	5209	0	30	0	226000.00	1.00	140	1	1	87	0
105	72.0	1	328	0	30	1	621000.00	1.70	138	0	1	88	1
109	45.0	0	292	1	35	0	850000.00	1.30	142	1	1	88	0
117	85.0	1	102	0	60	0	507000.00	3.20	138	0	0	94	0
126	46.0	0	168	1	17	1	271000.00	2.10	124	0	0	100	1
131	60.0	1	1082	1	45	0	250000.00	6.10	131	1	0	107	0
134	81.0	0	4540	0	35	0	231000.00	1.18	137	1	1	107	0
163	50.0	1	2334	1	35	0	75000.00	0.90	142	0	0	126	1
167	59.0	0	66	1	20	0	70000.00	2.40	134	1	0	135	1
171	52.0	0	3966	0	40	0	325000.00	0.90	140	1	1	146	0
199	60.0	0	1211	1	35	0	263358.03	1.80	113	1	1	186	0
200	63.0	1	1767	0	45	0	73000.00	0.70	137	1	0	186	0
217	54.0	1	427	0	70	1	151000.00	9.00	137	0	0	196	1
228	65.0	0	56	0	25	0	237000.00	5.00	130	0	0	207	0
281	70.0	0	582	0	40	0	51000.00	2.70	136	1	1	250	0
296	45.0	0	2060	1	60	0	742000.00	0.80	138	0	0	278	0

Droping recordes based on index as above

In[26]:

```
new_data = DataF.copy()
new_data.drop(DataF.index[indices],inplace = True)
new_data.reset_index(drop=True,inplace = True)
new_data.head()
```

Description of modified data set

In[27]:

```
new_data.describe()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
count	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000	271.000000
mean	60.827801	0.431734	441.232472	0.416974	37.966790	0.361624	260552.098266	1.233985	136.896679	0.653137	0.317343	133.516605	0.298893
std	11.765012	0.496234	508.506740	0.493971	11.388217	0.481360	81801.526054	0.540673	3.870643	0.476852	0.466303	77.188506	0.458620
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.500000	125.000000	0.000000	0.000000	4.000000	0.000000
25%	51.000000	0.000000	115.000000	0.000000	30.000000	0.000000	214000.000000	0.900000	134.000000	0.000000	0.000000	74.500000	0.000000
50%	60.000000	0.000000	235.000000	0.000000	38.000000	0.000000	262000.000000	1.100000	137.000000	1.000000	0.000000	120.000000	0.000000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	302000.000000	1.300000	140.000000	1.000000	1.000000	206.500000	1.000000
max	95.000000	1.000000	2794.000000	1.000000	65.000000	1.000000	543000.000000	3.800000	148.000000	1.000000	1.000000	285.000000	1.000000

information of modified data set

In[28]:

```
new_data.info()
```

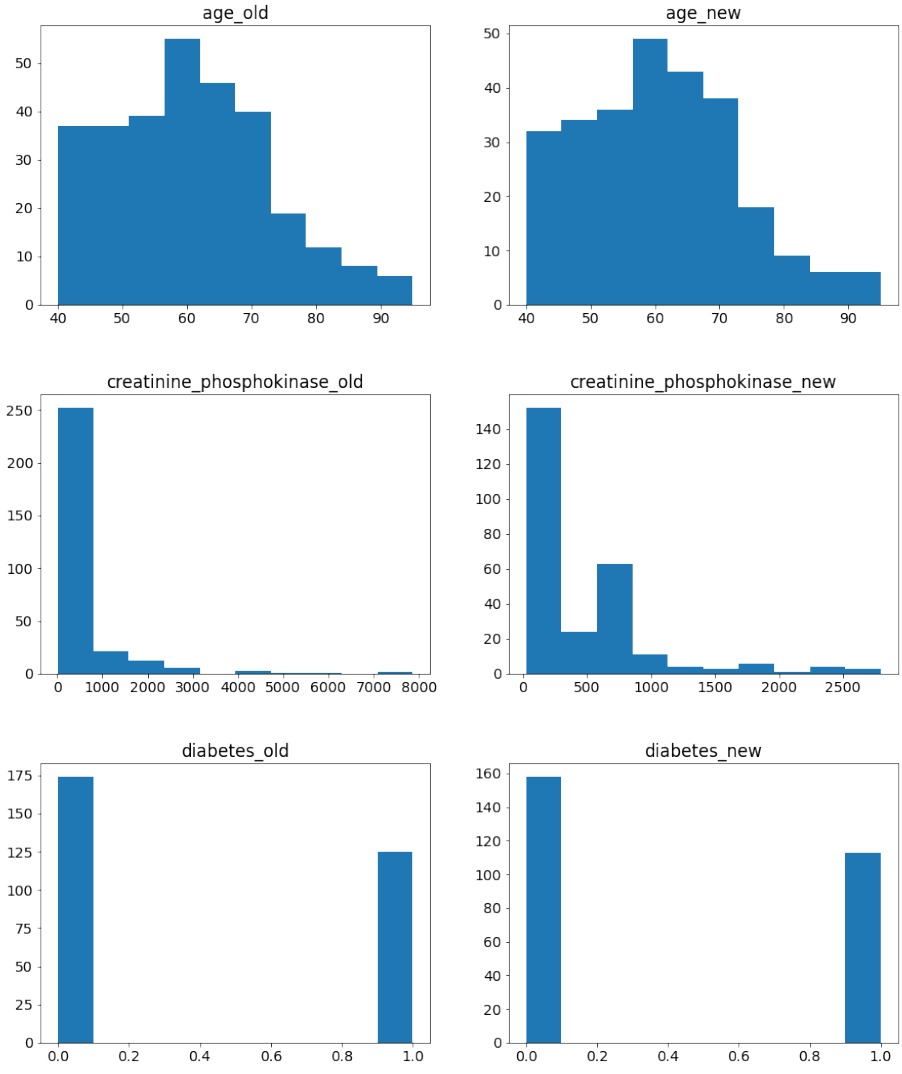
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271 entries, 0 to 270
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age                       271 non-null    float64
 1   anaemia                   271 non-null    int64  
 2   creatinine_phosphokinase  271 non-null    int64  
 3   diabetes                  271 non-null    int64  
 4   ejection_fraction         271 non-null    int64  
 5   high_blood_pressure       271 non-null    int64  
 6   platelets                 271 non-null    float64
 7   serum_creatinine          271 non-null    float64
 8   serum_sodium              271 non-null    int64  
 9   sex                       271 non-null    int64  
 10  smoking                   271 non-null    int64  
 11  time                      271 non-null    int64  
 12  DEATH_EVENT               271 non-null    int64  
```

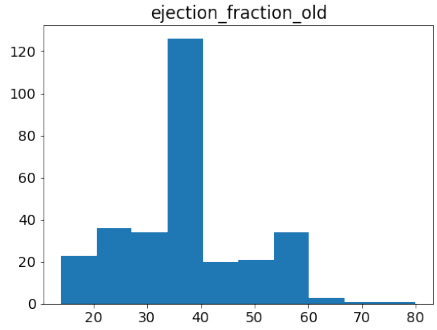
```
dtypes: float64(3), int64(10)
memory usage: 27.6 KB
```

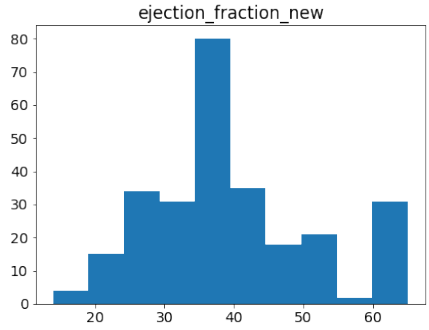
Comparing Old data with new

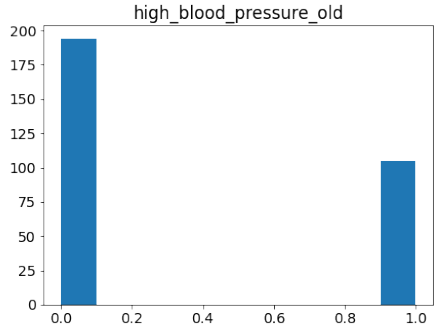
In[29]:

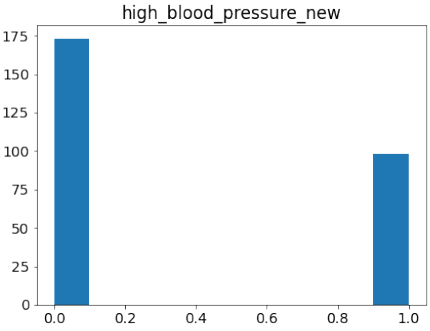
```
for col in DataF.columns[:-1]:
    plt.figure(figsize = (15,5))
    plt.subplot(1,2,1)
    plt.hist(DataF[col])
    plt.title(col+'_old')
    plt.subplot(1,2,2)
    plt.hist(new_data[col])
    plt.title(col+'_new')
    plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+col+'comp.png', transparent=True)
    plt.show()
```

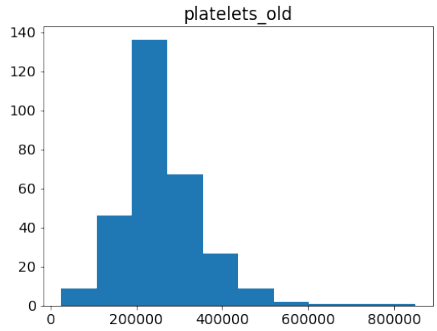


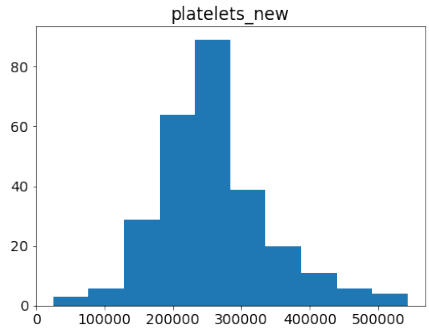

ejection_fraction_old
0
20
40
60
80
100
120
20
30
40
50
60
70
80

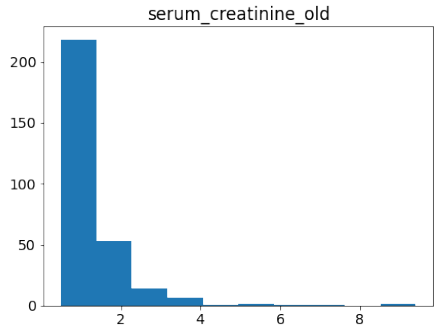

ejection_fraction_new
0
10
20
30
40
50
60
70
80
20
30
40
50
60

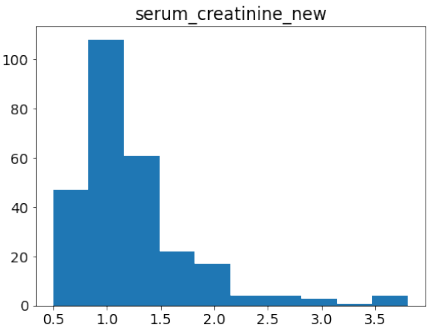

high_blood_pressure_old
0
25
50
75
100
125
150
175
200
0.0
0.2
0.4
0.6
0.8
1.0

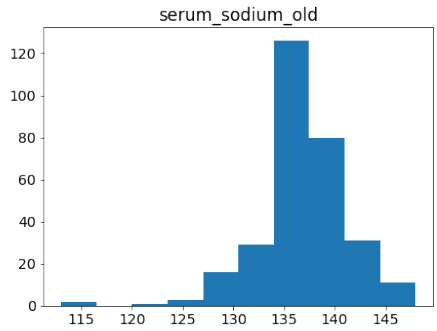

high_blood_pressure_new
0
25
50
75
100
125
150
175
0.0
0.2
0.4
0.6
0.8
1.0

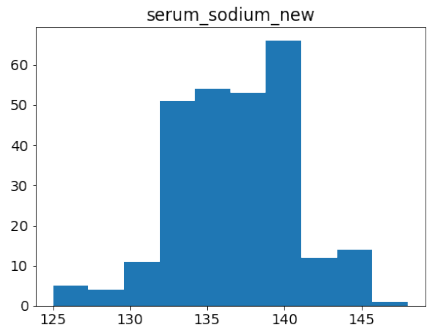

platelets_old
0
20
40
60
80
100
120
140
0
200000
400000
600000
800000

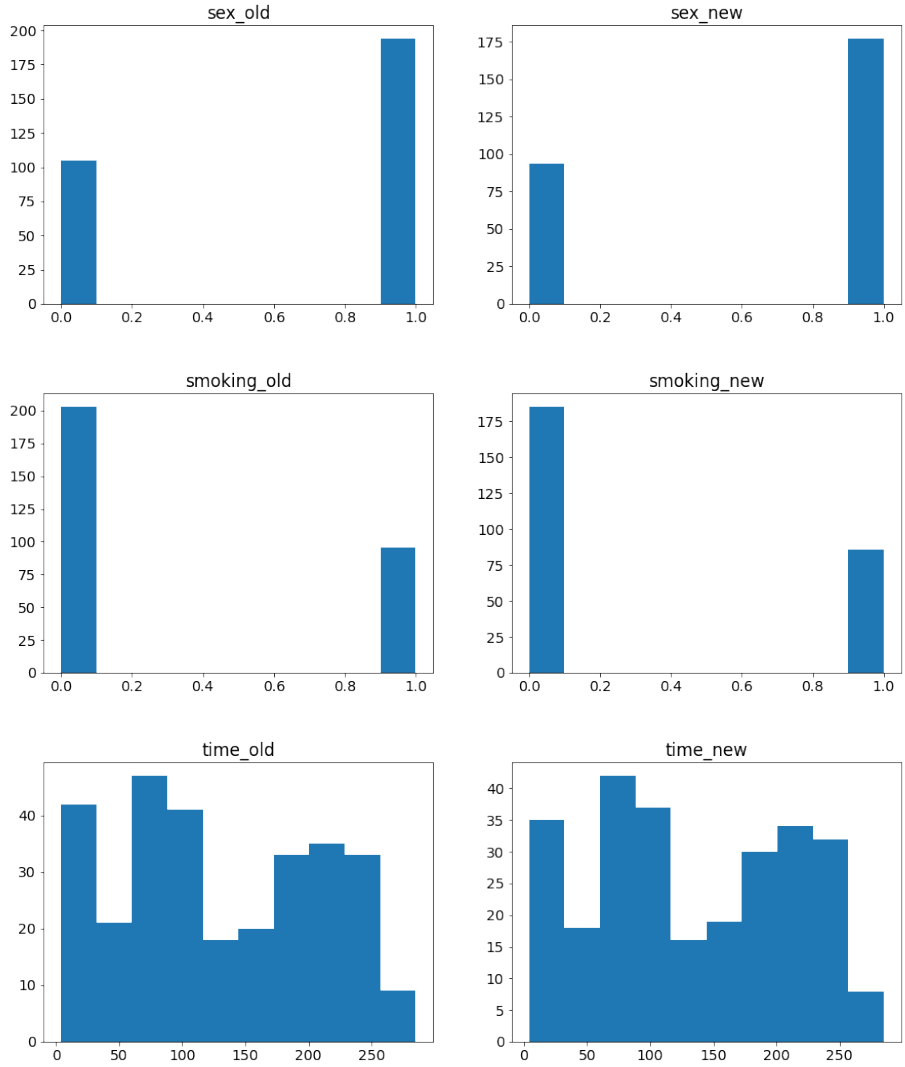

platelets_new
0
20
40
60
80
0
100000
200000
300000
400000
500000


serum_creatinine_old
0
50
100
150
200
2
4
6
8


serum_creatinine_new
0
20
40
60
80
100
0.5
1.0
1.5
2.0
2.5
3.0
3.5


serum_sodium_old
0
20
40
60
80
100
120
115
120
125
130
135
140
145


serum_sodium_new
0
10
20
30
40
50
60
125
130
135
140
145



Spliting data into two parts Test and Train

Dividing data into X,Y where X is input features Y is predictable feature

In[30]:

```
new_data =DataF.copy()
y = new_data['DEATH_EVENT']
x = new_data.drop(['DEATH_EVENT'],axis = 1)
```

Counting no of 0,1 in the targeted output

In[31]:

```
y.value_counts()
```

```
0    190
1     81
Name: DEATH_EVENT, dtype: int64
```

We need to smote the data

In[32]:

```
from imblearn.over_sampling import SMOTE
s = SMOTE(random_state=50)
x_s,y_s = s.fit_resample(x,y)
```

In[33]:

```
y_s.value_counts()
```

```
0    190
1    190
Name: DEATH_EVENT, dtype: int64
```

importing function and spliting data data in Test,Train

In[34]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_s,y_s,test_size = 0.20,random_state =
42)
```

In[35]:

```
print("X_train   shape",x_train.shape)
print("Y_train   shape",y_train.shape)
print("X_test    shape",x_test.shape)
print("X_test    shape",y_test.shape)
```

```
X_train  shape (304, 12)
Y_train  shape (304,)
X_test   shape (76, 12)
X_test   shape (76,)
```

Models Generation

DecisionTree Model

importing DecisionTreeClassifier class from sklearn

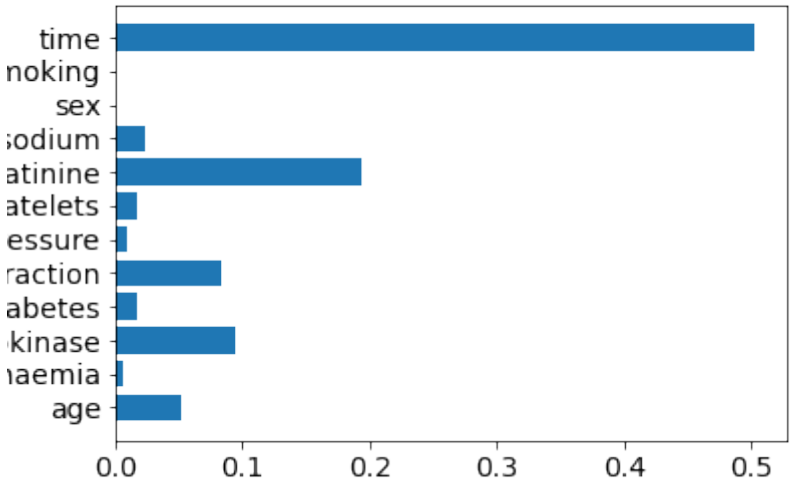
In[36]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
Models = []
```

Finding Beast Features sotes the model

In[37]:

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)

plt.barh(x_train.columns, decision_tree.feature_importances_)

plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"decinsion_tree_imp"+'_m.png', transparent=True)

plt.show()
```



Training Model

In[38]:

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV
accuracy = 0
dt_Model = None
for i in range(100):
    decision_tree = DecisionTreeClassifier()
    decision_tree.fit(x_train,y_train)
    y_pred = decision_tree.predict(x_test)
```

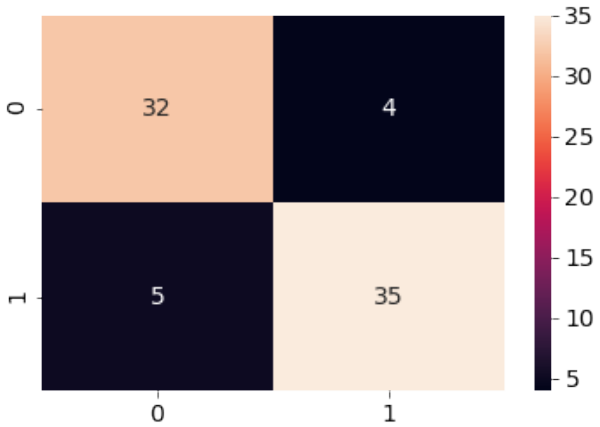
```
    c_matrix = confusion_matrix(y_pred, y_test)
    accuracy_dt = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
    if accuracy_dt>accuracy:
        print(accuracy_dt)
        accuracy = accuracy_dt
        dt_Model = decision_tree
```

Testing Model

In[39]:

```
y_pred = dt_Model.predict(x_test)
c_matrix = confusion_matrix(y_pred, y_test)
accuracy_dt = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))

comp = y_pred == y_test
comp_dt = comp[comp==False]

print(comp_dt.index)

sns.heatmap(c_matrix,annot=True)

result = {"Decision Tree":[accuracy_dt,]}
print(accuracy_dt)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"decision_tree_acc"+'.png', transparent=True)

Models.append((dt_Model,comp_dt.index))
```

0.881578947368421



Random Forest

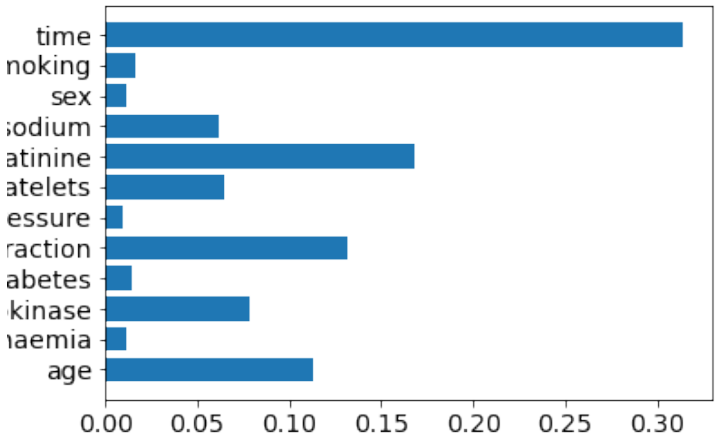
Importing RandomForestClassifier from skleran

In[40]:

```
from sklearn.ensemble import RandomForestClassifier
```

In[41]:

```
RandomForest = RandomForestClassifier()
RandomForest.fit(x_train, y_train)

plt.barh(x_train.columns, RandomForest.feature_importances_)

plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"Randome_Forest_imp"+'.png', transparent=True)

plt.show()
```



Selecting features

In[42]:

```
features_rf =
['time','serum_creatinine','ejection_fraction','creatinine_phosphokinase','age','diabetes','serum_creatinine']
x_train_rf = x_train[features_rf]
x_test_rf = x_test[features_rf]
```

Training Model

In[43]:

```
accuracy = 0
for i in range(100):
    RandomForest = RandomForestClassifier(n_estimators = 100, max_depth = 15,
```

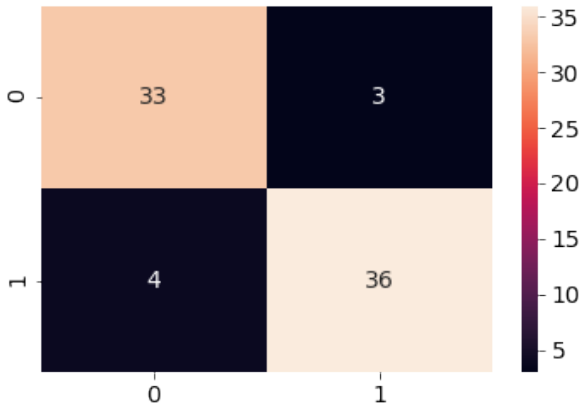
```
max_leaf_nodes = 40)
        RandomForest.fit(x_train,y_train)
        y_pred = RandomForest.predict(x_test)
        c_matrix = confusion_matrix(y_pred, y_test)
        accuracy_rf = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
        if accuracy_rf>accuracy:
                accuracy = accuracy_rf
                rf_Model=RandomForest
                print(accuracy)
```

Testing Model

In[44]:

```
y_pred = rf_Model.predict(x_test)

c_matrix = confusion_matrix(y_pred, y_test)
accuracy_rf = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))

sns.heatmap(c_matrix,annot=True)


print(accuracy_rf)
result["Randome Forest"]=[accuracy_rf,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for
Doc\\jupyter\\'+"RandomForest_acc"+'.png', transparent=True)
comp = y_pred == y_test
comp_rf = comp[comp==False]
comp_rf.index

Models.append((rf_Model,comp_rf.index))
```

0.9078947368421053



Logistic Regression

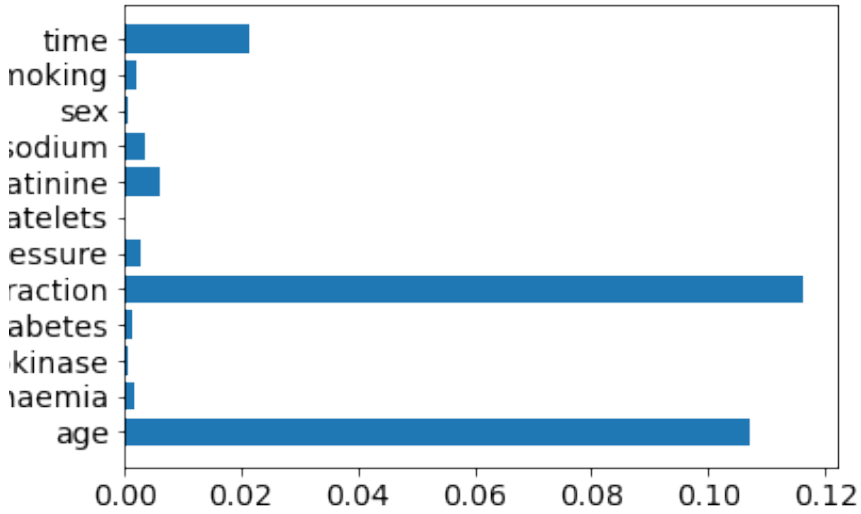
importing LogisticRegression class from sklearn

In[45]:

```
from sklearn.linear_model import LogisticRegression
```

In[46]:

```
LogisticRegression_ = LogisticRegression()
LogisticRegression_.fit(x_train, y_train)
List = [abs(i) for i in LogisticRegression_.coef_[0]]
plt.barh(x_train.columns,List)
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"logistic_regression_imp"+'.png', transparent=True)

plt.show()
```



In[47]:

```
features_lr = ['time', 'ejection_fraction', 'age','serum_creatinine']
x_train_lr = x_train[features_lr]
x_test_lr = x_test[features_lr]
```

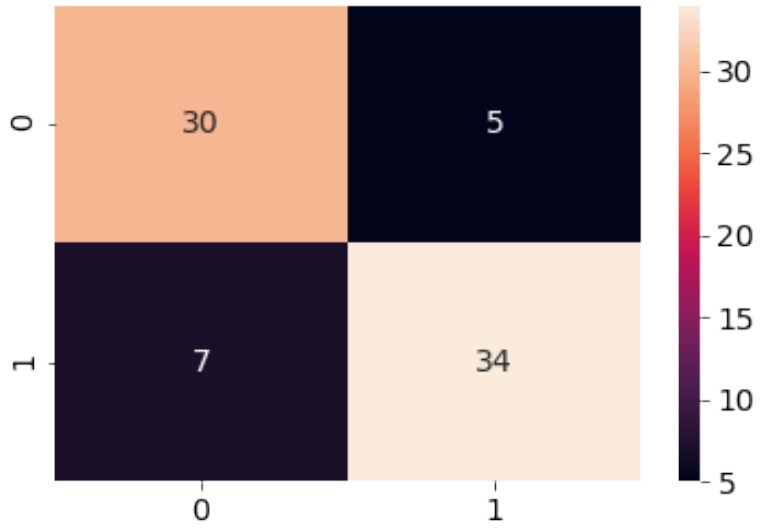
In[48]:

```
LogisticRegression_ = LogisticRegression()
LogisticRegression_.fit(x_train_lr, y_train)
```

In[49]:

```
y_pred = LogisticRegression_.predict(x_test_lr)

c_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(c_matrix,annot=True)

comp = y_pred == y_test
comp_lr = comp[comp==False]
comp_lr.index

accuracy_lr = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
print(accuracy_lr)
result["Logistic Regression"]=[accuracy_lr,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"LogisticRegression_acc"+'.png', transparent=True)

Models.append((LogisticRegression,comp_lr.index))
```

0.8421052631578947



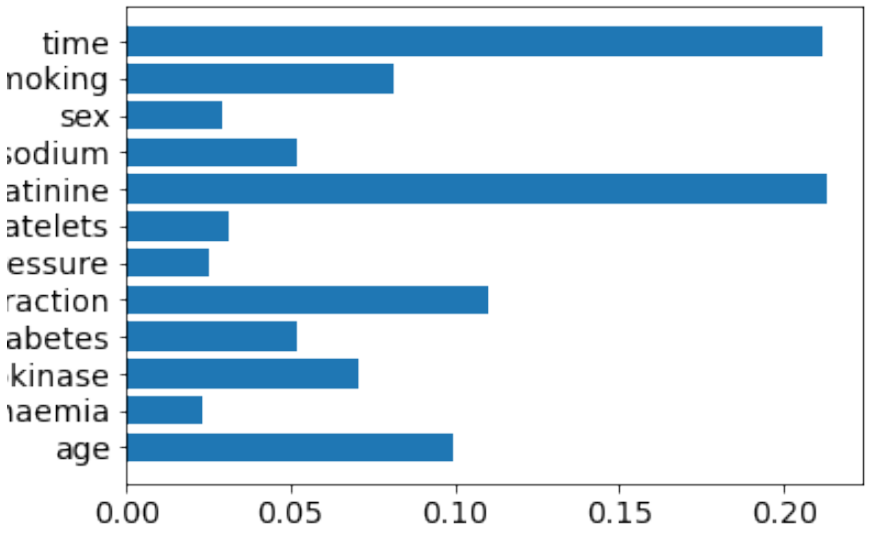
XG Boost model

In[50]:

```
from xgboost import XGBClassifier
```

In[51]:

```
XGB = XGBClassifier(max_dept = 5,n_estimator= 50)
XGB.fit(x_train, y_train)

plt.barh(x_train.columns,XGB.feature_importances_)

plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"XGB_imp"+'.png', transparent=True)

plt.show()
```



In[52]:

```
features_xgb = ['time','serum_creatinine','ejection_fraction','creatinine_phosphokinase','age','diabetes','serum_creatinine']
x_train_xgb  = x_train[features_xgb]
x_test_xgb = x_test[features_xgb]
```

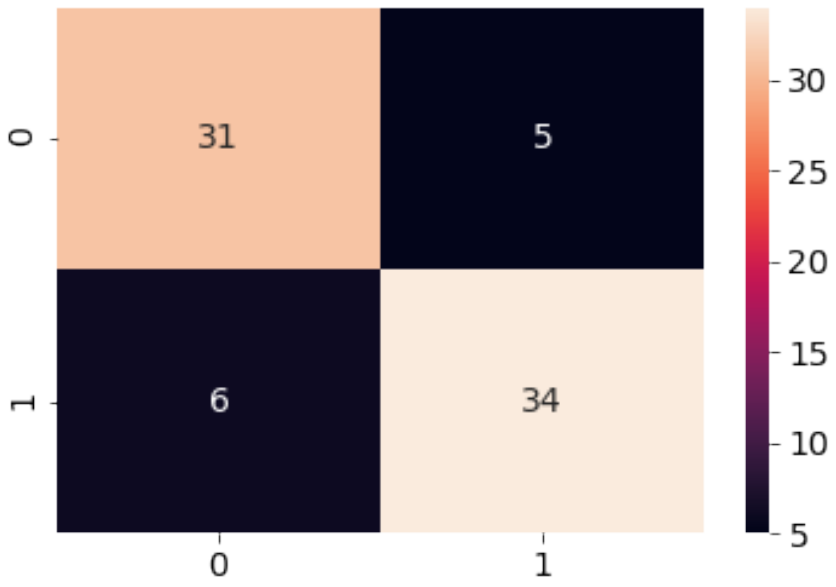
In[53]:

```
XGB = XGBClassifier(n_estimator = 50)
XGB.fit(x_train, y_train)
```

In[54]:

```
y_pred = XGB.predict(x_test)
c_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(c_matrix,annot=True)

comp = y_pred == y_test
comp_xb = comp[comp==False]
print(comp_xb.index)

accuracy_xg = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
print(accuracy_xg)
result["XGBClassigier"]=[accuracy_xg,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"XGB_acc"+'.png', transparent=True)

Models.append((XGB,comp_xb.index))
```

0.8552631578947368



Navie Bayes

In[55]:

```
from sklearn.naive_bayes import GaussianNB
```

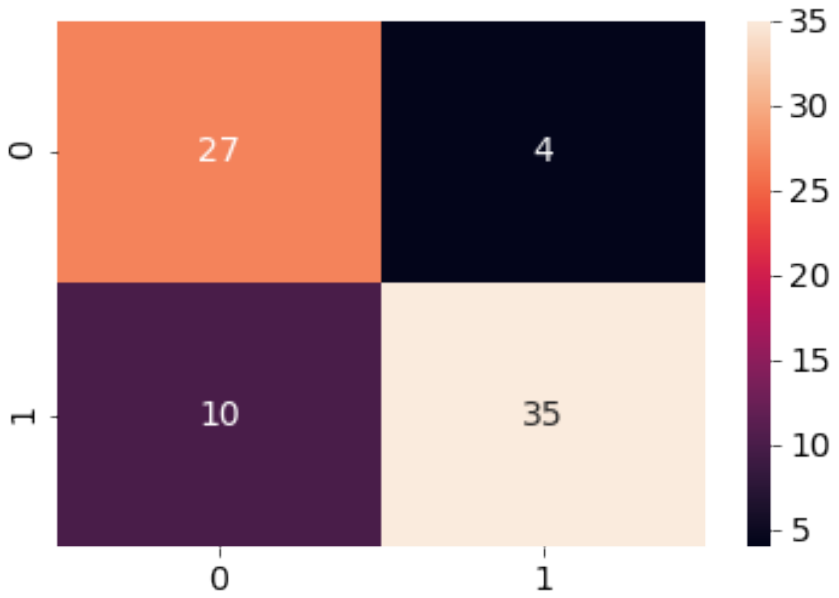
In[56]:

```
NaiveBaies = GaussianNB()
NaiveBaies.fit(x_train,y_train)
```

In[57]:

```
y_pred = NaiveBaies.predict(x_test)
c_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(c_matrix,annot=True)

comp = y_pred == y_test
comp_nb = comp[comp==False]
print(comp_nb.index)

accuracy_nb = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
print(accuracy_nb)
result["NaiveBaies"]=[accuracy_nb,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"naive_Baies_acc"+'.png', transparent=True)

Models.append((NaiveBaies,comp_nb.index))
```

```
0.8157894736842105
```



K nearst neighbour Classificatoin

In[58]:

```
from sklearn.neighbors import KNeighborsClassifier
```

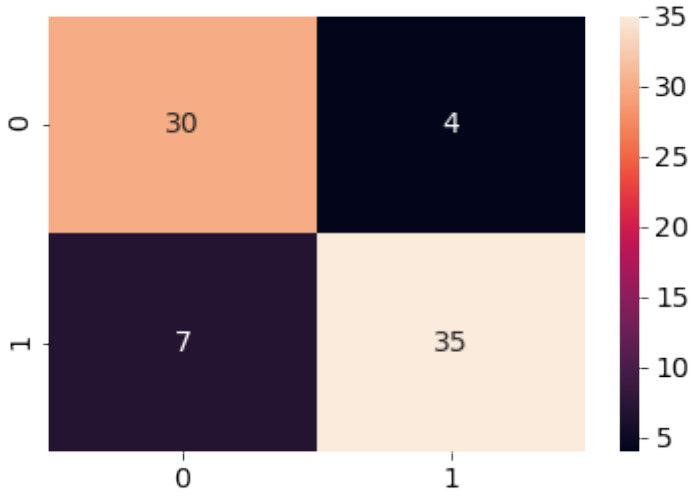
In[59]:

```
knn = KNeighborsClassifier()
knn.fit(x_train[importent_features],y_train)
```

In[60]:

```
y_pred = knn.predict(x_test[importent_features])
c_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(c_matrix,annot=True)

comp = y_pred == y_test
comp_kn = comp[comp == False]
print(comp_kn.index)

accuracy_kn = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
print(accuracy_kn)
result["KNeighborsClassigier"]=[accuracy_kn,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"Knn_acc"+'.png', transparent=True)

Models.append((knn,comp_kn.index))
```

0.8552631578947368



Selection vector Classification

In[61]:

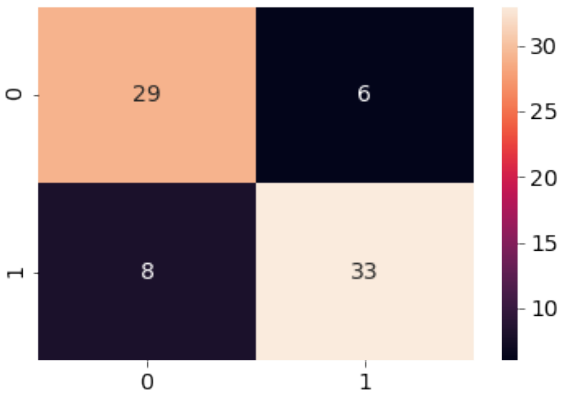
```
from sklearn.svm import SVC
```

In[62]:

```
model_svm = SVC(kernel="linear")
model_svm.fit(x_train, y_train)
importance = model_svm.coef_[0]

plt.bar([x for x in range(len(importance))], importance)
```

In[63]:

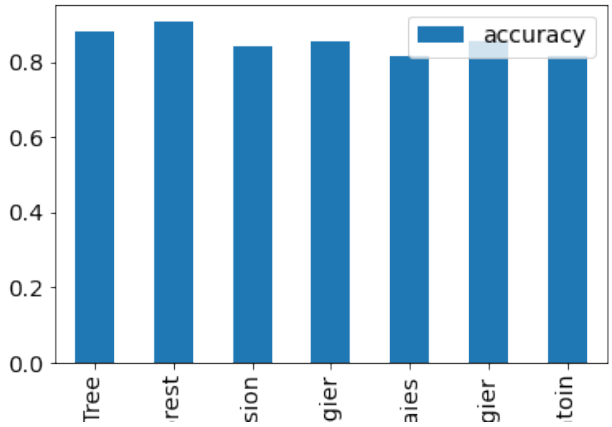
```
y_pred = model_svm.predict(x_test)
c_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(c_matrix,annot=True)

comp = y_pred == y_test
comp_svm = comp[comp==False]
print(comp_svm.index)

accuracy_svm = (c_matrix[0,0]+c_matrix[1,1])/(sum(sum(c_matrix)))
print(accuracy_svm)
result["Selection Vector Classificatoin"]=[accuracy_svm,]
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"svm_acc"+'.png', transparent=True)
Models.append((model_svm,comp_svm.index))
```



Selecting Best Model

In[64]:

```
resultdf = pd.DataFrame(result,index=["accuracy"]).transpose()
resultdf.plot(kind="bar")
plt.savefig('C:\\Users\\Jp227\\Desktop\\Project Files\\Images for Doc\\jupyter\\'+"results"+'.png', transparent=True)
```



In[65]:

```
resultdf
```

	accuracy
Decision Tree	0.881579
Randome Forest	0.907895
Logistic Regression	0.789474
XGBClassigier	0.855263
NaiveBaies	0.815789
KNeighborsClassigier	0.855263
Selection Vector Classificatoin	0.815789

From above values we can choose Random Forest
Storing model in .pkl format

In[66]:

```
import joblib
joblib.dump(Models[1][0],"HeartFailPredmodel.pkl")
```

8. Graphical user interface

In our user interface we have 3 Screens

- ✓Welcome Screen
- ✓Form Screen
- ✓File upload Screen

Each screen its own importance in the interface

Welcome Screen

- It contains 2 buttons
  - Form
  - Fileupload

To support the screen movements

Form Screen

It is a form Window which takes input from user and produces predicted output

FileUpload Screen

It is used to load a file and predicted the targeted output display it in the window in scroll-able form

Code :

Widgets.py (some modifications to widgets for form window)

```
from kivy.uix.popup import Popup
from kivy.uix.label import Label
from kivy.uix.checkbox import CheckBox
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.lang import Builder

Builder.load_file("widgets.kv")


class MyPopup(Popup):
    """
    inherited all the properties of Popup class
    by calling open function of this class object a window gets popuped
    """
    pass
```

```
class MyLabel(Label):
    """
    inherited all the properties of Label
    additional features are added to make the label
    > bit transparent
    > rounded at left end
    """
    pass


class MyBoxLayout(BoxLayout):
    """
    inherited all the properties of BoxLayout
    additional features are added to make the boxlayout
    > bit transparent
    > rounded at right end
    """
    pass


class MyTextInput(TextInput):
    """
    inherited all the properties of TextInput
    additional features are added to
    > bit Transparent
    > text color to white
    > courser color to white
    > text alignment to center
    """
    pass


class RadioButton(CheckBox):
    """
    inherited all the properties of CheckBox
    > By grouping checkbox object they acts as RadioButton
    additional features are added
    > changing color to white
    """
    pass


class RadioLabel(Label):
    """
    inherited all the properties of Label
    mainly created to use them along with RadioButtons
    additional features are
    >Bit Transparent
    >White Text
    """
    pass
```

Widgets.kv

```
<MyPopup>
    auto_dismiss : True
    size_hint : 0.6,0.3
    pos_hint : {"center_x":.5,"center_y":.5}
    background_color : (0,0,0,0)
    canvas.before:
        Color:
            rgba: 0,0,1,0.8
        RoundedRectangle:
            size : self.size
            pos : self.pos
            radius : [0,40,0,40]
    Label :
        id : pop_output
        text : ""

<MyLabel>
    size_hint : .80,.75
    halign : "left"
    valign : "middle"
    font_size : 18
    color : 1,1,1,1
    canvas.before:
        Color:
            rgba : (.4,.5,.6,.7)
        RoundedRectangle:
            size : self.size
            pos : self.pos
            radius : [50,0,0,50]

<MyTextInput>
    halign : "center"
    valign : "middle"
    size_hint : 1.5,.75
    multiline : False
    background_color: 0,0,0,0
    cursor_color: 1,1,1,1
    foreground_color : 1,1,1,1
    canvas.after:
        Color:
            rgba : .1,.2,.3,.5
        RoundedRectangle:
            size : self.size
            pos : self.pos
            radius : [0,50,50,0]
```

```
<MyBoxLayout>
    canvas.before:
        Color:
            rgba : (.1,.2,.3,.5)
        RoundedRectangle:
            size : self.size
            pos : self.pos
            radius : [0,50,50,0]
    size_hint : 1.5,.75

<RadioButton>
    color: 1,1,1,1

<RadioLabel>
    color: 1,1,1,1
```

ModelLoading.py

```
import joblib


class ModelLoading:
    def __init__(self):
        """
        model is loading and creating a model object
        """
        self.model = joblib.load("MachineLearning\\Model.pkl")


    def singleRecord(self, dataf):
        """
        used to predict the output for single record
        :param dataf: input for model
        :return: predicted output
        """
        return self.model.predict(dataf)[0]


    def multipleRecords(self, dataf):
        """
        used to predict the output for more than one record
        :param dataf: input for model ( more than one record)
        :return: predicted output series is returned (list)
        """
        return self.model.predict(dataf)
```

formwindow.py

```
from widgets import MyPopup
from kivy.uix.widget import Widget
import pandas as pd
from modelLoading import ModelLoading
from kivy.lang import Builder

Builder.load_file("formwindow.kv")

class FormWindow(Widget):
    """
    inherited all the properties of Widgets
    contains some class variables for storing input
    > age, creatinine_phosphokinase, ejection_fraction, platelets,
      serum_creatinine, serum_sodium, time, anaemia, diabetes,
      high_blood_pressure, gender, smoking

    it contains multiple function to handle the form
    > radiobutton_anaemia : to handle radio button of anaemia
    > radiobutton_diabetes : to handle radio button of diabetes
    > radiobutton_high_blood_pressure : to handle radio button of high_blood_pressur
    > radiobutton_gender : to handle radio button of gender
    > radiobutton_smoking : to handle radio button of smoking
    > submit : to collect all details and popup result
    > reset : to clear all the inputs

    """
    age = None
    creatinine_phosphokinase = None
    ejection_fraction = None
    platelets = None
    serum_creatinine = None
    serum_sodium = None
    time = None
    anaemia = None
    diabetes = None
    high_blood_pressure = None
    gender = None
    smoking = None

    def radiobutton_anaemia(self, instance, value, text):
        """
        it will store the data to anaemia if the radio button is off None is assigned to
anaemia
        :param instance: radio button object
        :param value: true/false :: on/off
        :param text: to assign value to anaemia variable
        :return: None
        """
        if value:
            FormWindow.anaemia = text
        else:
            FormWindow.anaemia = None
```

```
    def radiobutton_diabetes(self, instance, value, text):
        """
        it will store the data to diabetes if the radio button is off None is assigned to
diabetes
        :param instance: radio button object
        :param value: true/false :: on/off
        :param text: to assign value to diabetes variable
        :return: None
        """
        if value:
            FormWindow.diabetes = text
        else:
            FormWindow.diabetes = None

    def radiobutton_high_blood_pressure(self, instance, value, text):
        """
        it will store the data to high blood pressure if the radio button is off None is
assigned to high blood pressure
        :param instance: radio button object
        :param value: true/false :: on/off
        :param text: to assign value to high_blood_pressure variable
        :return: None
        """
        if value:
            FormWindow.high_blood_pressure = text
        else:
            FormWindow.high_blood_pressure = None

    def radiobutton_gender(self, instance, value, text):
        """
        it will store the data to gender if the radio button is off None is assigned to gender
        :param instance: radio button object
        :param value: true/false :: on/off
        :param text: to assign value to gender variable
        :return: None
        """
        if value:
            FormWindow.gender = text
        else:
            FormWindow.gender = None

    def radiobutton_smoking(self, instance, value, text):
        """
        it will store the data to smoking if the radio button is off None is assigned to
smoking
        :param instance: radio button object
        :param value: true/false :: on/off
        :param text: to assign value to smoking variable
        :return: None
        """
        if value:
            FormWindow.smoking = text
        else:
            FormWindow.smoking = None
```

```
    def submit(self):
        """
        in this call we collect all the data from the form
        creates a list of it
        generate a data frame to give it as input to the model
        creating model object
        predicting output
        MyPopup class object is created
        and uses it to popup the ouput
        if any errors are occurred in middle it will generate a popup
        :return: None
        """
        FormWindow.age = self.ids.age_input.text.strip()
        FormWindow.creatinine_phosphokinase =
self.ids.creatinine_phosphokinase_input.text.strip()
        FormWindow.ejection_fraction = self.ids.ejection_fraction_input.text.strip()
        FormWindow.platelets = self.ids.platelets_input.text.strip()
        FormWindow.serum_creatinine = self.ids.serum_creatinine_input.text.strip()
        FormWindow.serum_sodium = self.ids.serum_sodium_input.text.strip()
        FormWindow.time = self.ids.time_input.text.strip()
        columns = ['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
                   'ejection_fraction', 'high_blood_pressure', 'platelets',
                   'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time', ]
        Heading = ["age", "creatinine_phosphokinase", "ejection_fraction",
"serum_creatinine", "time"]

        List = [[FormWindow.age, FormWindow.anaemia,
                 FormWindow.creatinine_phosphokinase, FormWindow.diabetes,
                 FormWindow.ejection_fraction, FormWindow.high_blood_pressure,
                 FormWindow.platelets, FormWindow.serum_creatinine,
                 FormWindow.serum_sodium, FormWindow.gender,
                 FormWindow.smoking, FormWindow.time]]

        df = pd.DataFrame(List, index=[1], columns=columns)
        pop = MyPopup()

        try:
            #age
            if not df["age"][1].isdigit():
                if FormWindow.age == '':
                    raise ValueError("Fill age")
                else:
                    raise ValueError("Age must be an integer in range 1 to 120")
            if not 0 < int(df["age"][1]) < 120:
                raise ValueError("Age must be in range 1 to 120")

            # anaemia
            if FormWindow.anaemia is None:
                raise ValueError("fill Aneaemia")

            # ceatinine_phosphokinase
            if not df["creatinine_phosphokinase"][1].isdigit():
                if FormWindow.creatinine_phosphokinase == '':
                    raise ValueError("Fill Creatinine Phosphokinase")
```

```
                else:
                    raise ValueError("Creatinine Phosphokinase must be an integer
in range 20 to 7860")
            if not 20 <= int(df["creatinine_phosphokinase"][1]) < 8000:
                raise ValueError("Creatinine Phosphokinase must in range(20,8000)")

            # diabetes
            if FormWindow.diabetes is None:
                raise ValueError("fill Diabetes")

            # ejection_fraction
            if not df["ejection_fraction"][1].isdigit():
                if FormWindow.ejection_fraction == '':
                    raise ValueError("Fill Ejection Fraction")
                else:
                    raise ValueError("Ejection Fraction must be in range 10 to 100")
            if not 14 <= int(df["ejection_fraction"][1]) <= 80:
                raise ValueError("Ejection Fraction in range(14,80)")

            # hight_blood_pressure
            if FormWindow.high_blood_pressure is None:
                raise ValueError("Fill High Blood Pressure")

            # platelets
            if FormWindow.platelets == '':
                raise ValueError("Fill Platelets")
            if FormWindow.platelets.count('.') > 1:
                raise ValueError("Platelets more than one decimal points")
            elif not FormWindow.platelets.replace('.', '').isdigit():
                raise ValueError("Pleas enter valid input for Platelets")
            if not 25000 <= float(df["platelets"][1]) < 850000:
                raise ValueError("Platelets value must be in range(25000,850000)")

            # serum_creatinine
            if FormWindow.serum_creatinine == '':
                raise ValueError("Fill serum Creatinine")
            if FormWindow.serum_creatinine.count('.') > 1:
                raise ValueError("Serum Creatinine more than one decimal points")
            elif not FormWindow.serum_creatinine.replace('.', '').isdigit():
                raise ValueError("Pleas enter valid input for Serum Creatinine")
            if not 0.5 <= float(df["serum_creatinine"][1]) <= 9.5:
                raise ValueError("Serum Creatinine must be in range(.5 to 9.5)")

            # serum_sodium
            if FormWindow.serum_sodium == '':
                raise ValueError("Fill Serum Sodium")
            if FormWindow.serum_sodium.count('.') > 1:
                raise ValueError("Serum Sodium more than one decimal points")
            elif not FormWindow.serum_sodium.replace('.', '').isdigit():
                raise ValueError("Pleas enter valid input for Serum Sodium")
            if 100 < int(df["serum_sodium"][1]) <= 150:
                raise ValueError("Serum Sodium must be in range(100 to 150)")

            # gender
            if FormWindow.gender is None:
```

```
                    raise ValueError("choose gender male/female ")

                # smoking
                if FormWindow.smoking is None:
                    raise ValueError("Fill for smoking")

                model = ModelLoading()
                prediction = model.singleRecord(df[Heading])
                pop.title = "Prediction is Done"
                if prediction == 1:
                    pop.ids.pop_output.text = "There is high chance to survive for this
patient for long time"
                elif prediction == 0:
                    pop.ids.pop_output.text = "There is a chance to survive with good
medical care "

            except ValueError as ve:
                pop.title = " Value Error "
                pop.ids.pop_output.text = str(ve)

            except:
                pop.title = "Error"
                pop.ids.pop_output.text = " Something went wrong in prediction try again"

            finally:
                pop.open()

        def reset(self):
            """
            clears all values
            by replacing text with ""
            and replacing readiobuttons output to None, state to Normal
            :return:None
            """
            self.ids.age_input.text = ''
            self.ids.creatinine_phosphokinase_input.text = ''
            self.ids.ejection_fraction_input.text = ''
            self.ids.platelets_input.text = ''
            self.ids.serum_creatinine_input.text = ''
            self.ids.serum_sodium_input.text = ''
            self.ids.time_input.text = ''
            self.ids.anaemia_yes.state = "normal"
            self.ids.anaemia_no.state = "normal"
            self.ids.diabetes_yes.state = "normal"
            self.ids.diabetes_no.state = "normal"
            self.ids.high_blood_pressure_yes.state = "normal"
            self.ids.high_blood_pressure_no.state = "normal"
            self.ids.gender_female.state = "normal"
            self.ids.gender_male.state = "normal"
            self.ids.smoking_yes.state = "normal"
            self.ids.smoking_no.state = "normal"
            FormWindow.age = None
            FormWindow.creatinine_phosphokinase = None
            FormWindow.ejection_fraction = None
            FormWindow.platelets = None
```

```
        FormWindow.serum_creatinine = None
        FormWindow.serum_sodium = None
        FormWindow.time = None
        FormWindow.anaemia = None
        FormWindow.diabetes = None
        FormWindow.high_blood_pressure = None
        FormWindow.gender = None
        FormWindow.smoking = None
```

Formwindow.kv

```
#:import all widgets
<FormWindow>:
    canvas.before:
        Rectangle:
            size : self.size
            pos : self.pos
            source : "Heading.jpg"

    BoxLayout:
        orientation : "vertical"
        cols : 2
        size : root.width,root.height
        padding:10
        canvas.before:
            Color:
                rgba: 0,0,0,0.7
            Rectangle:
                size : self.size
                pos : self.pos
#age
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Age"
            MyTextInput :
                id : age_input
                size_hint : 1.5,.75
                multiline : False

#anaemia
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Anaemia"

            MyBoxLayout:
                RadioLabel :
                    text : "Yes"
                RadioButton:
                    id : anaemia_yes
                    group : "anaemia"
                    on_active : root.radiobutton_anaemia(self,self.active,'1')
```

```
                    RadioLabel :
                        text : "No"
                    RadioButton:
                        id : anaemia_no
                        group : "anaemia"
                        on_active :    root.radiobutton_anaemia(self,self.active,'0')

#creatinine_phosphokinase
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Creatinine Phosphokinase"
            MyTextInput :
                id:creatinine_phosphokinase_input

#diabetes
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Diabetes"
            MyBoxLayout:
                RadioLabel :
                    text : "Yes"
                RadioButton:
                    id : diabetes_yes
                    group : "diabetes"
                    on_active :    root.radiobutton_diabetes(self,self.active,'1')
                RadioLabel :
                    text : "No"
                RadioButton:
                    id : diabetes_no
                    group : "diabetes"
                    on_active : root.radiobutton_diabetes(self,self.active,'0')

#ejection_fraction
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Ejection Fraction"
            MyTextInput :
                id:ejection_fraction_input

#high_blood_pressure
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "High Blood Pressure"
            MyBoxLayout:
                size_hint : 1.5,.75
                RadioLabel :
                    text : "Yes"
                RadioButton:
                    id : high_blood_pressure_yes
                    group : "high_blood_pressure"
                    on_active : root.radiobutton_high_blood_pressure(self,self.active,'1')
```

```
                    RadioLabel :
                        text : "No"
                    RadioButton:
                        id : high_blood_pressure_no
                        group : "high_blood_pressure"
                        on_active : root.radiobutton_high_blood_pressure(self,self.active,'0')

#platelets
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Platelets"
            MyTextInput :
                id:platelets_input

#serum_creatinine
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Serum Creatinine"
            MyTextInput :
                id:serum_creatinine_input

#serum_sodium
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Serum Sodium"
            MyTextInput :
                id:serum_sodium_input
#gender
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Gender"
            MyBoxLayout:
                RadioLabel :
                    text : "Female"
                RadioButton:
                    id : gender_female
                    group : "gender"
                    on_active : root.radiobutton_gender(self,self.active,'0')
                RadioLabel :
                    text : "Male"
                RadioButton:
                    id : gender_male
                    group : "gender"
                    on_active : root.radiobutton_gender(self,self.active,'1')
#smoking
        BoxLayout:
            orientaion:"horizontal"
            MyLabel:
                text : "Smoking"
            MyBoxLayout:
```

```
                    RadioLabel :
                        text : "Yes"
                    RadioButton:
                        id : smoking_yes
                        group : "smoking"
                        on_active : root.radiobutton_smoking(self,self.active,'1')
                    RadioLabel :
                        text : "No"
                    RadioButton:
                        id : smoking_no
                        group : "smoking"
                        on_active : root.radiobutton_smoking(self,self.active,'0')
#Time
            BoxLayout:
                orientaion:"horizontal"
                MyLabel:
                    text : "Time"
                MyTextInput :
                    id:time_input
#Buttons
            BoxLayout:
                orientaion:"horizontal"
                spacing : 2
                Button:
                    text : "RESET"
                    background_color : (0,0,0,0)
                    size_hint : .9,.9
                    color : (.1,.1,.7,1)
                    canvas.before:
                        Color:
                            rgba : (.4,.5,.6,.9)
                        RoundedRectangle:
                            size : self.size
                            pos : self.pos
                            radius : [50,0,0,50]
                    on_press : root.reset()

                Button:
                    text : "SUBMIT"
                    background_color : (0,0,0,0)
                    size_hint : .9,.9
                    color : (.7,.1,.1,1)
                    canvas.before:
                        Color:
                            rgba : (.4,.5,.6,1)
                        Rectangle:
                            size : self.size
                            pos : self.pos
                    on_press : root.submit()
                Button:
                    text : "Back"
                    background_color : (0,0,0,0)
                    size_hint : .9,.9
                    color : (.1,.1,.7,1)
                    canvas.before:
```

```
                    Color:
                        rgba : (.4,.5,.6,.9)
                    RoundedRectangle:
                        size : self.size
                        pos : self.pos
                        radius : [0,50,50,0]
                on_press : app.root.current = "WelcomeScreen"
```

Fileuploadwindow.py

```
from kivy.uix.widget import Widget
from kivy.uix.label import Label
import easygui
from kivy.lang import Builder
import pandas as pd
from modelLoading import ModelLoading
from widgets import MyPopup
from kivy.uix.gridlayout import GridLayout
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.progressbar import ProgressBar
from kivy.clock import Clock
import threading
import time


Clock.max_iteration = 100


Builder.load_file("fileuploadwindow.kv")


class RecordLabel(Label):
    """
    inherited all the properties of the Label class
    > bit transparent
    > white font
    """
    pass


class IndexLabel(Label):
    """
    used to place index of the records and heading of the records
    > blue
    > white font
    """
    pass

class IGridLayout(GridLayout):
    """
    To arrange record values in order
    """
```

```
        pass

class IndexBoxLayout(BoxLayout):
        """
        to arrange index of the records
        """
        pass


class HeadingBoxLayout(BoxLayout):
        """
        to arrange headings of the columns in order
        """
        pass


class FileUploadWindow(Widget):
        """
        We have progress, popup as two class variables with None
        they are used while loading data into the Window
        """
        progress = None
        popup = None

        def choose_file(self):
                """
                1. it will as user to choose file (.csv)
                3. generating a popup along with progress bar to show at what extent data loaded
                4 . thread to load data t1 for calling update_layout function which takes file as
parameter
                :return: None
                """
                file = easygui.fileopenbox(default="*.csv")
                self.ids.heading.clear_widgets()
                self.ids.index.clear_widgets()
                self.ids.scroll.clear_widgets()
                if file is not None:
                        self.ids.location.text = file
                        self.file = file.replace('\\', '\\\\')
                        FileUploadWindow.popup = MyPopup()
                        FileUploadWindow.progress = ProgressBar(max=100)
                        FileUploadWindow.popup.content = FileUploadWindow.progress
                        FileUploadWindow.popup.auto_dismiss = False
                        FileUploadWindow.popup.title = "Loading Data ....."
                        FileUploadWindow.popup.open()
                        t1 = threading.Thread(target=self.update_layout, args=[file], name='t2')
                        t1.start()


        def update_layout(self, file):
                """
                dataf is created using .csv file
                updating progress bar
                creating model object
                predicting values
```

```
        loading the data into 3 layouts
        1. index layout
        2. heading layout
        3. records layout
        adding widgets to scroll view
        :param file: name,location of file
        :return: None
        """
        dataf = pd.read_csv(file)
        FileUploadWindow.popup.title = "Loading Data ...."
        FileUploadWindow.progress.value = 10.11
        time.sleep(0.03)
        real = dataf.pop("DEATH_EVENT")
        FileUploadWindow.popup.title = "Creating Layout ...."
        iBox = IGridLayout()
        hBox = HeadingBoxLayout()
        inBox = IndexBoxLayout()
        hBox.add_widget(RecordLabel(text="DEATH_EVENT(p)"))

        for col in list(dataf.columns):
            hBox.add_widget(RecordLabel(text=col))
        FileUploadWindow.progress.value = 22.61
        FileUploadWindow.popup.title = "Loading Model ...."
        time.sleep(0.02)
        model = ModelLoading()
        FileUploadWindow.progress.value = 36.53
        FileUploadWindow.popup.title = "Predicting    ...."
        time.sleep(0.05)
        predict = model.multipleRecords(dataf[["age",
                                               "creatinine_phosphokinase",
                                               "ejection_fraction",
                                               "serum_creatinine",
                                               "time"]])
        FileUploadWindow.progress.value = 50.53
        FileUploadWindow.popup.title = "Displaying Data    ...."
        time.sleep(0.03)
        for i in range(dataf.shape[0]):
            inBox.add_widget(IndexLabel(text=str(i + 1)))
            iBox.add_widget(RecordLabel(text=str(predict[i])))
            for j in dataf.columns:
                iBox.add_widget(RecordLabel(text=str(dataf[j][i])))
        FileUploadWindow.progress.value = 80.93
        time.sleep(0.3)
        self.ids.scroll.add_widget(iBox)
        self.ids.heading.add_widget(hBox)
        self.ids.index.add_widget(inBox)
        FileUploadWindow.popup.dismiss()
        FileUploadWindow.popup = None
        FileUploadWindow.progress = None
```

Fileuploadwindow.kv

```
<RecordLabel>:
    color : 1,1,1,1
    size_hint : (None,None)
    height : 30
    width : 130
    bold : True

<IndexLabel>:
    color : 1,1,1,1
    size_hint : (None,None)
    height : 30
    width :  40
    bold : True

<ProgressPopup>:
    auto_dismiss : True
    size_hint : 0.6,0.3
    pos_hint : {"center_x":.5,"center_y":.5}
    background_color : (0,0,0,0)
    canvas.before:
        Color:
            rgba: 0,0,1,0.8
        RoundedRectangle:
            size : self.size
            pos : self.pos
            radius : [0,40,0,40]

<IGridLayout>:
    id : grid
    cols : 13
    size_hint : (None,None)
    height : self.minimum_height
    width : self.minimum_width

<IndexBoxLayout>:
    canvas.before:
        Color :
            rgba : (0,0,1,0.8)
        Rectangle:
            size : self.size
            pos : self.pos
    orientation : "vertical"
    size_hint : (None,None)
    height : self.minimum_height
    width : self.minimum_width

<HeadingBoxLayout>:
    canvas.before:
        Color :
            rgba : (0,0,1,0.8)
        Rectangle:
```

```
                size : self.size
                pos : self.pos
        orientation : "horizontal"
        id:heading
        size_hint : (None,None)
        height : self.minimum_height
        width : self.minimum_width


<FileUploadWindow>:
        canvas.before:
                Rectangle:
                        size : self.size
                        pos : self.pos
                        source : "Heading.jpg"
        FloatLayout:
                size : root.width,root.height
                canvas.before:
                        Color :
                                rgba : (0,0,0,.7)
                        Rectangle:
                                size : self.size
                                pos : self.pos
                TextInput:
                        id : location
                        font_size : 24
                        text : ""
                        size_hint : .7,.07
                        pos_hint:{"center_x":.5,"center_y":.07}
                        multiline : False
                        background_color: 0,0,0,.5
                        cursor_color: 1,1,1,1
                        foreground_color : 1,1,1,1
                Button:
                        text : "Choose File"
                        size_hint : .15,.07
                        pos_hint :{"x":0,"center_y":.07}
                        background_color :0,.5,.9,1
                        on_release : root.choose_file()

                Button :
                        text : "Back"
                        size_hint :.15,.07
                        pos_hint :{"x":.85,"center_y":.07}
                        background_color :0,.5,.9,1
                        on_release : app.root.current = "WelcomeScreen"

                FloatLayout:
                        id : layout
                        size_hint : (1,.90)
                        pos_hint :{"x":0,"y":.1}
                        orientation : "vertical"
                        ScrollView:
                                id : heading
                                scroll : scroll
```

```
                size_hint : (.95,.05)
                pos_hint : {'x':.05,'y':0.95}
                do_scroll_y : False
                do_scroll_x : False
                scroll_x : self.scroll.scroll_x

            ScrollView:
                id : index
                scroll : scroll
                size_hint : (.05,.95)
                pos_hint : {'x':0,'y':0}
                do_scroll_y : False
                do_scroll_x : False
                scroll_y : self.scroll.scroll_y


            ScrollView:
                size_hint : (.95,.95)
                pos_hint :    {'x':.05,'y':0}
                id : scroll
                scroll_type : ["bars"]
                bar_margin : 5
                bar_width : 10
```

Windowmanager.py

```
from kivy.uix.screenmanager import ScreenManager
from kivy.lang import Builder

Builder.load_string(
    """
#:import FormWindow formwindow
#:import FileUploadWindow fileuploadwindow
#:import WelcomeWindow welcomewindow
#:import Screen kivy.uix.screenmanager

<WindowManager>
    WelcomeScreen:
    FormScreen:
    FileUploadScreen:

<WelcomeScreen@Screen>:
    name : "WelcomeScreen"
    WelcomeWindow:

<FormScreen@Screen>:
    name:"FormScreen"
    FormWindow:

<FileUploadScreen@Screen>:
    name : "FileUploadScreen"
    FileUploadWindow:
```

```
    """
)


class WindowManager(ScreenManager):
    """
    inherited all the properties of ScreenManager class
    created 3 screens in this class
    1.WelcomeScreen
        > it contains WelcomeWindow object
    2.FormScreen
        > it contains FormWindow object
    3.FileUploadScreen
        > it contains FileUploadWindow object
    This class is going to manage all screens of our applications
    """
    pass
```

Main.py

```
from kivy.uix.screenmanager import FadeTransition
from windowmanager import WindowManager
from kivy.app import App


class HearFailure(App):
    """
    inherited all the properties of App class
    in this we will create WindowManager object
    sent that object to run
    """
    def build(self):
        """
        :return:WindowManager object
        """
        return WindowManager(transition=FadeTransition(duration=.5))

if __name__ == '__main__':
    """
      main namespace to star execution
      1. crating HearFailure class object
      2. calling its run function
    """
    HearFailure().run()
```

9.Packing Executable file

From python code executable file can be generating by using pyinstaller package
which is already discussed in page 19

To generate an exe file follow the below steps

Step 1:- keep all the required files and the python files in one directory

Step 2:- Make sure ,for which python file we need to generate exe file

Step 3:- Open command Prompt

Step 4:- Check whether pyinstaller in installed or not by using bellow command

```
> pyinstaller
```

If some output except error is printed on screen then pyinstaller is installed

Step 5:- If not installed then use the below command to install

```
>pip install pyinstaller
```

Step 6:- Go to the directory where your python file is located

Step 7:- use the following command to generate Executable file

```
>pyinstaller main.py
```

Step 8:- “main.spec” file will be created modify that with required arguments as below

Main.spec

```
# -*- mode: python ; coding: utf-8 -*-
from kivy_deps import sdl2, glew

block_cipher = None
```

```
a = Analysis(['ModelGUI.py'],
                     pathex=['C:\\Users\\Jp227\\Desktop\\Project\\Kivy\\Kivy_Folder'],
                     binaries=[],
                     datas=[],
                     hiddenimports=[],
                     hookspath=[],
                     runtime_hooks=[],
                     excludes=[],
                     win_no_prefer_redirects=False,
                     win_private_assemblies=False,
                     cipher=block_cipher,
                     noarchive=False)
pyz = PYZ(a.pure, a.zipped_data,
                     cipher=block_cipher)

a.datas +=
[('ModelGUI.kv','C:\\Users\\Jp227\\Desktop\\Project\\Kivy\\Kivy_Folder\\ModelGUI.
kv','DATA'),]



exe = EXE(pyz,
                a.scripts,
                [],
                exclude_binaries=True,
                name='ModelGUI',
                debug=False,
                bootloader_ignore_signals=False,
                strip=False,
                upx=True,
                console=False)

coll = COLLECT(exe,
                       Tree('C:\\Users\\Jp227\\Desktop\\Project\\Kivy\\Kivy_Folder\\'),
                       a.binaries,
                       a.zipfiles,
                       a.datas,
                       *[Tree(p) for p in (sdl2.dep_bins + glew.dep_bins)],
                       strip=False,
                       upx=True,
                       upx_exclude=[],
```

```
                    name='ModelGUI')
```

Step 9 :- execute the file command again as below

```
>pyinstaller main.spec
```

Step 10:- Go to directory <dist> which is newly created after executing the above command there u can see the exe file and some additional files required to run executable file

Note : To run the exe file it required some resource those are same that we used while coding like images, text files, etc…

10. Setup File generation

Inno Setup compiler which is going to bind the resource and the executable file at one location and help to install and uninstall option to the user to install our application in there PC and uninstall it by executing uninstall

Makes it easy for user and gives a grate look as an windows application

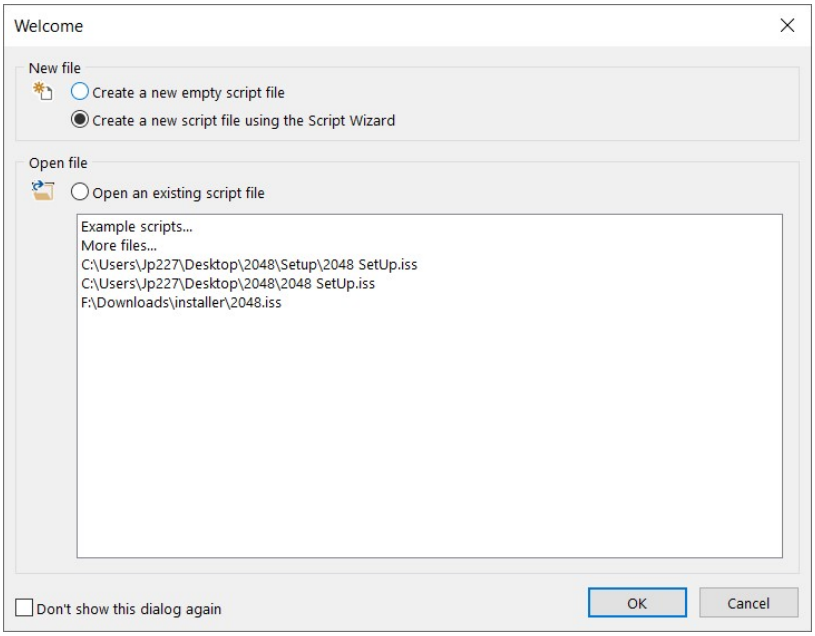
Inno Setup compiler can auto generate its own code

Follow the below steps :

Step 1:- Go to page 8 in this Document and use the link to install the Compiler

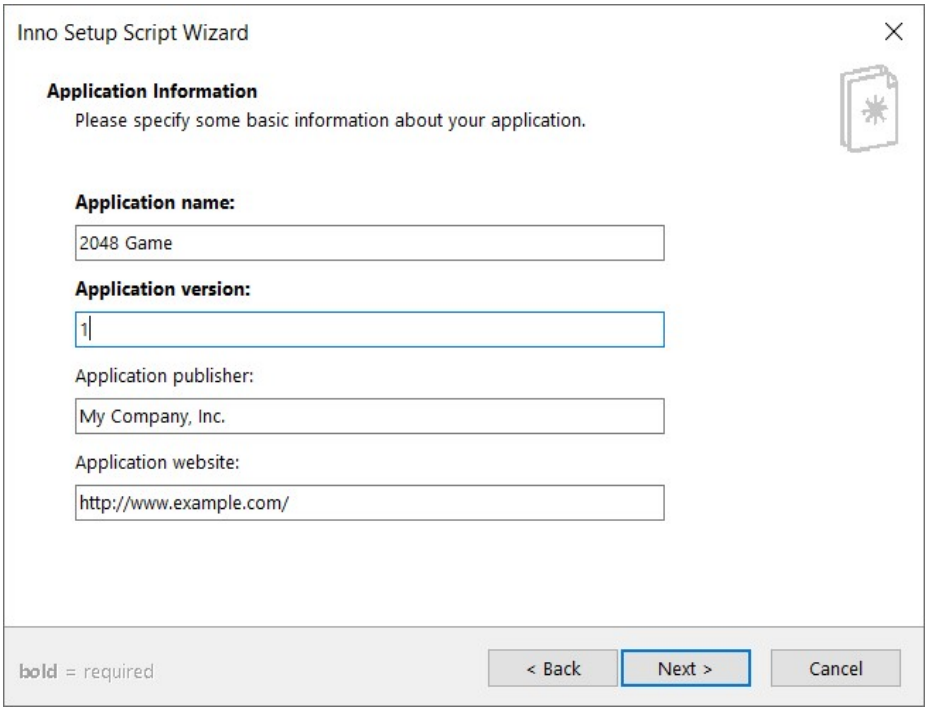
Step 2:- Open it

Step 3:- Choose Option < create new empty Script using Script Wizard> and OK



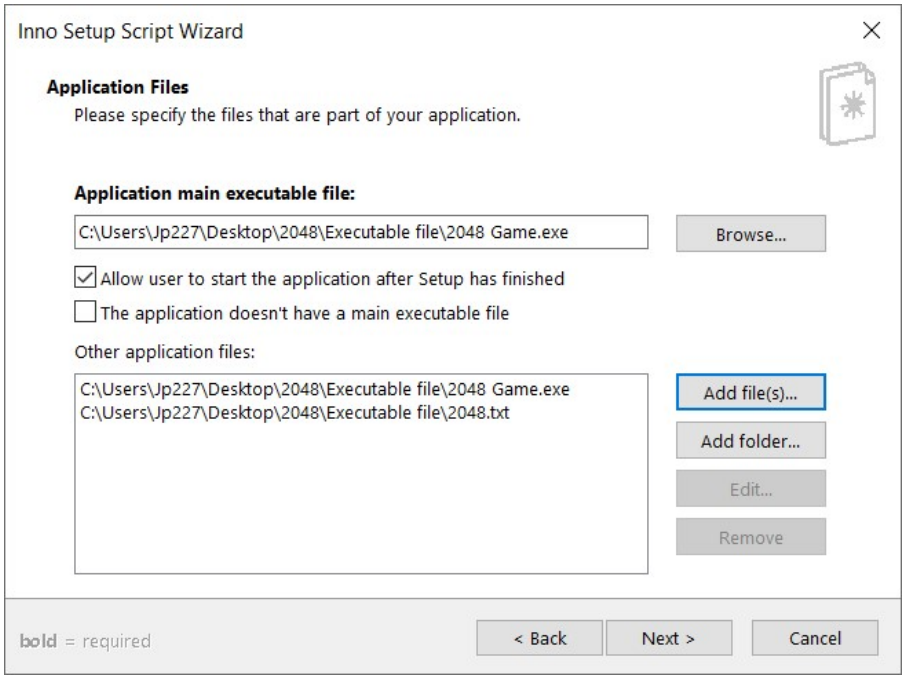
Step 4:- Just click <Next> (No changes at all)

Step 5:- Fill the all the required fields in the form and <click on Next>



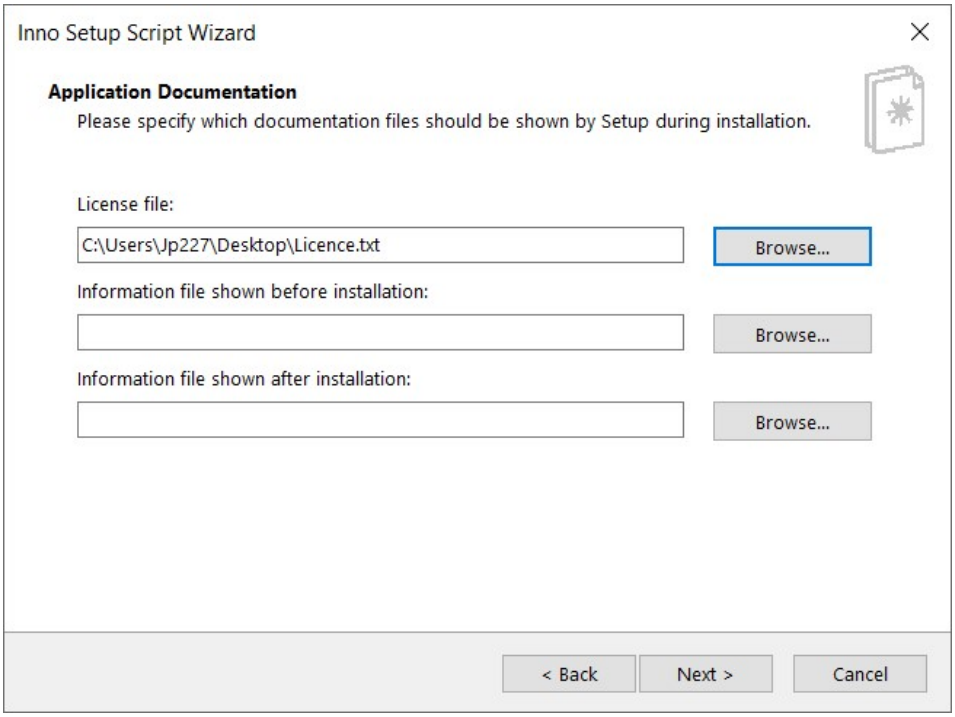
Step 6 :- Click on Next again

Step 7 :- Add Main file/ other required files (paths) as shown in below image (Next)



Step 8:- (Next)

Step 9: Add the Licence file(write something regarding your details in text file and use it as your Licence)
<Next>



Step 10:- complete all nexts till end <Next> and ok (save the file and run it)

HeartFailure.iss

```
; Script generated by the Inno Setup Script Wizard.
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!

#define MyAppName "Heart Failure"
#define MyAppVersion "1.5"
#define MyAppPublisher "CSE B Group 1"
#define MyAppURL "http://www.example.com/"
#define MyAppExeName "ModelGUI.exe"

[Setup]
; NOTE: The value of AppId uniquely identifies this application. Do not use the same AppId value in installers for other applications.
; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)
AppId={{43A7A9CC-3818-426A-A402-BBF3F3405412}
AppName={#MyAppName}
AppVersion={#MyAppVersion}
;AppVerName={#MyAppName} {#MyAppVersion}
AppPublisher={#MyAppPublisher}
AppPublisherURL={#MyAppURL}
AppSupportURL={#MyAppURL}
AppUpdatesURL={#MyAppURL}
DefaultDirName={autopf}\{#MyAppName}
DisableProgramGroupPage=yes
LicenseFile=C:\Users\Jp227\Desktop\Licence.txt
; Uncomment the following line to run in non administrative install mode (install for current user only.)
;PrivilegesRequired=lowest
OutputDir=C:\Users\Jp227\Desktop\Project
OutputBaseFilename=Heart_Failure
Compression=lzma
SolidCompression=yes
```

```
WizardStyle=modern

[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"

[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]
Source: "C:\Users\Jp227\Desktop\Project\Kivy\Kivy_Folder\dist\ModelGUI\ModelGUI.exe"; DestDir: "{app}"; Flags: ignoreversion
Source: "C:\Users\Jp227\Desktop\Project\Kivy\Kivy_Folder\dist\ModelGUI\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs ; Permissions: users-modify
Source: "C:\Users\Jp227\Desktop\HeartFailure.ico"; DestDir: "{app}"; Flags: ignoreversion
; NOTE: Don't use "Flags: ignoreversion" on any shared system files

[Icons]
Name: "{autoprograms}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"
Name: "{autodesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; Tasks: desktopicon
Name: "{autoprograms}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; IconFilename : "{app}\HeartFailure.ico"
Name: "{autodesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; IconFilename : "{app}\HeartFailure.ico"
[Run]
Filename: "{app}\{#MyAppExeName}"; Description: "{cm:LaunchProgram,{#StringChange(MyAppName, '&', '&&')}}"; Flags: nowait postinstall skipifsilent
```

Run this file

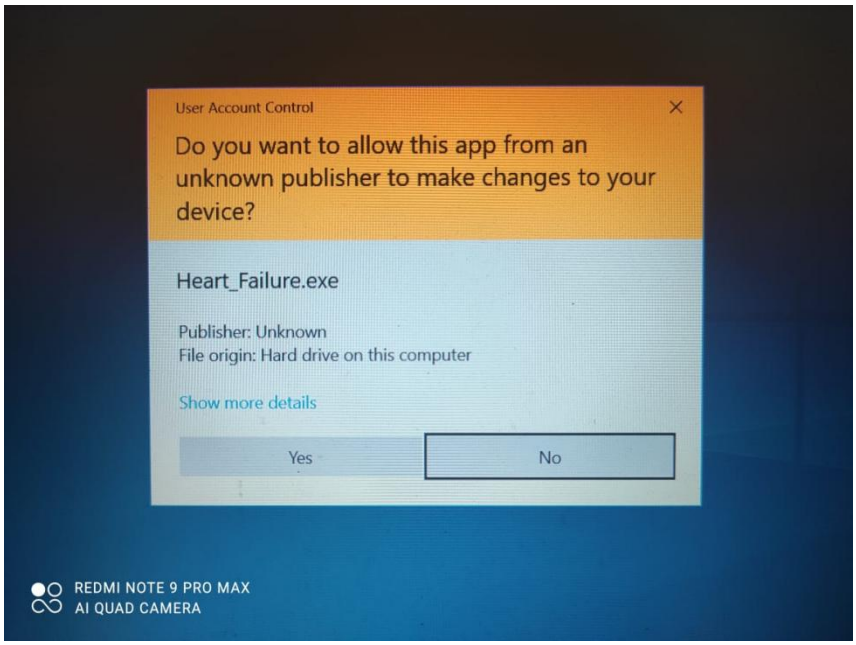
It will take some time

setup file size will be in between 30 to 50 MB size

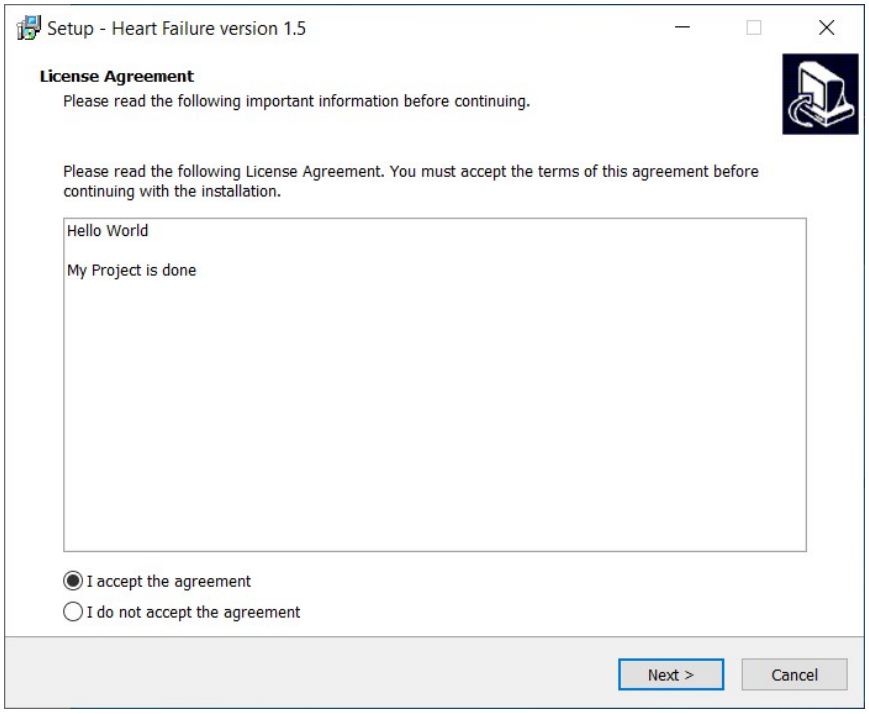
11. User Manual

11.1 Installation

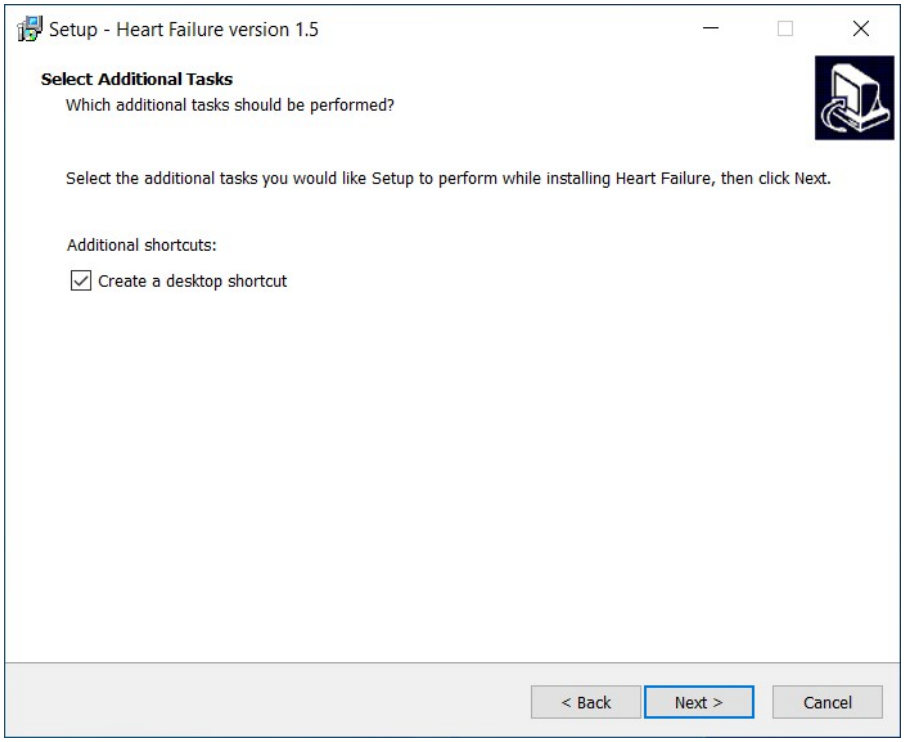
i. Double tap on Setup File and Allow app to install it in your system



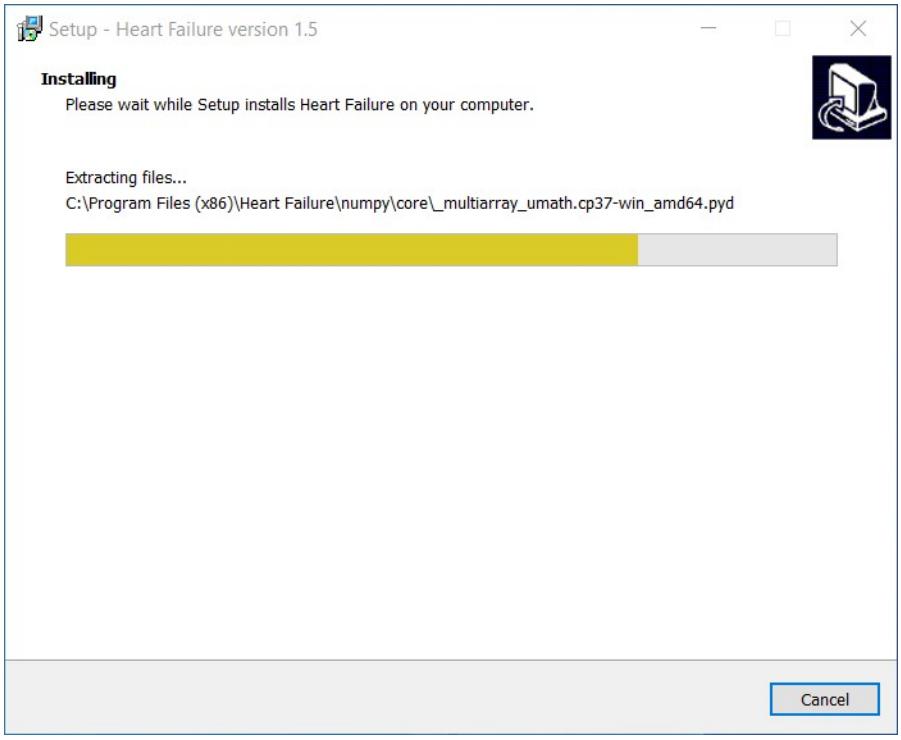
ii. Accept the agreements and press next



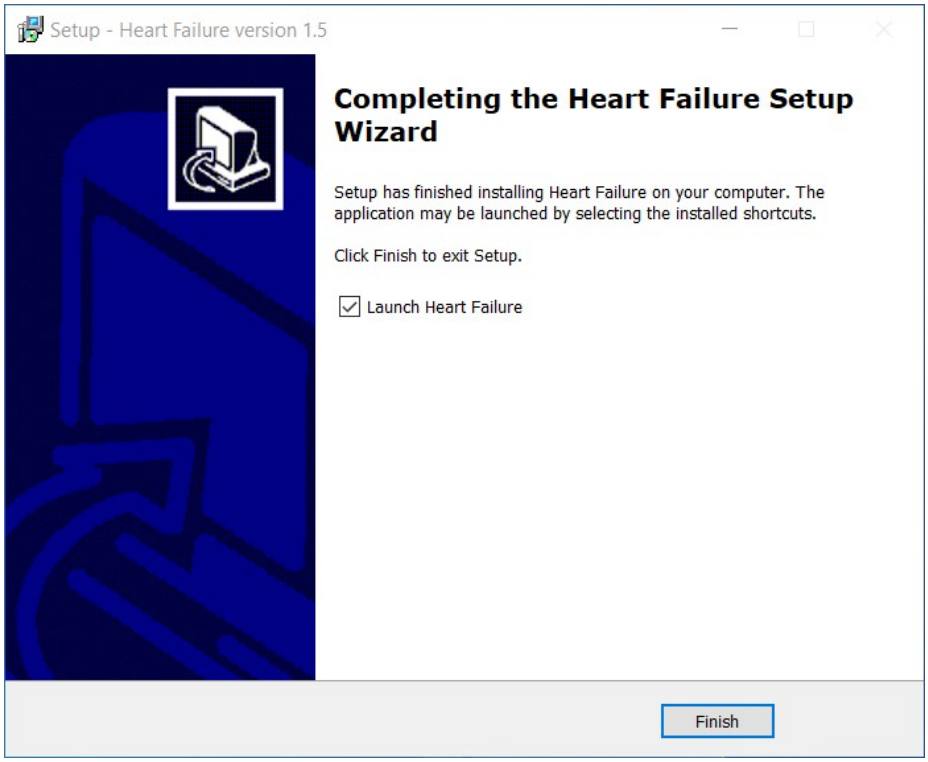
iii. Check the create a desktop shortcut (check box)



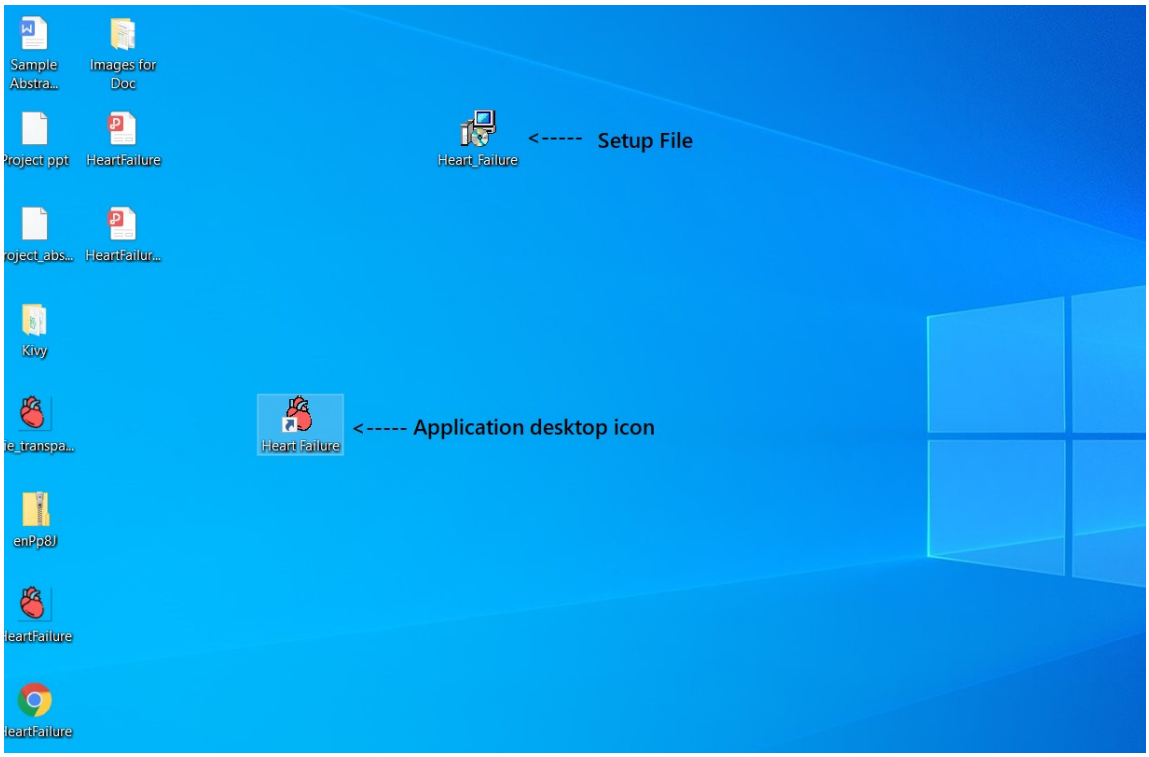
iv. Click on Install



v. Press on finish button

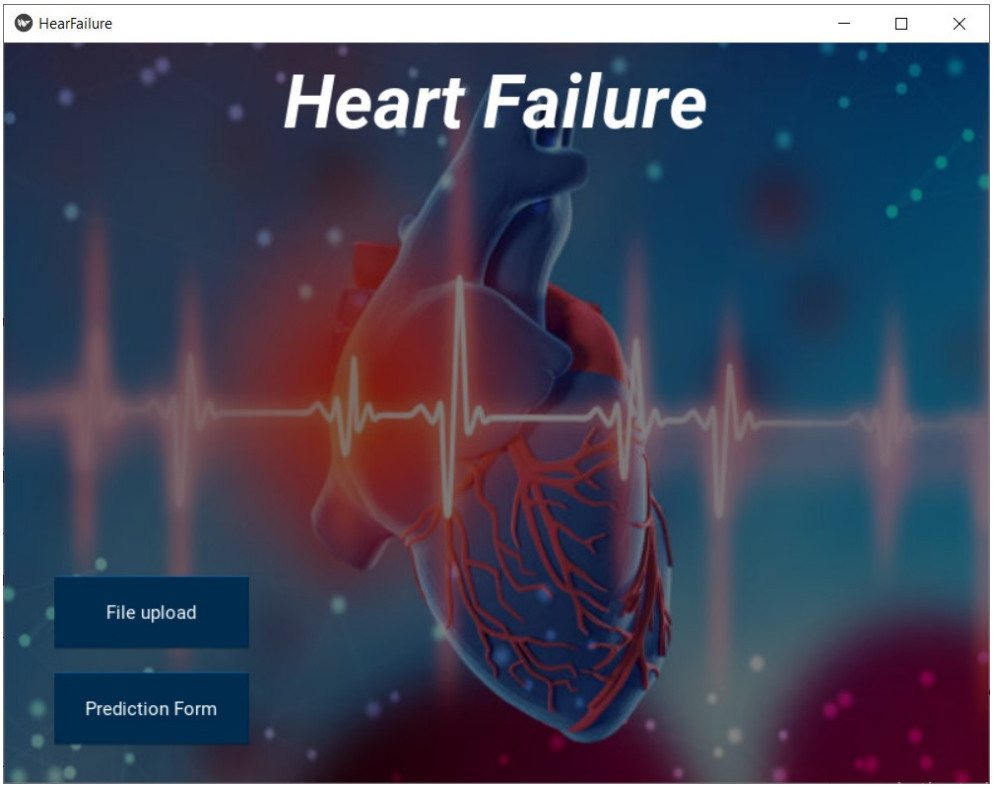


vi. We can view application short cut on desktop

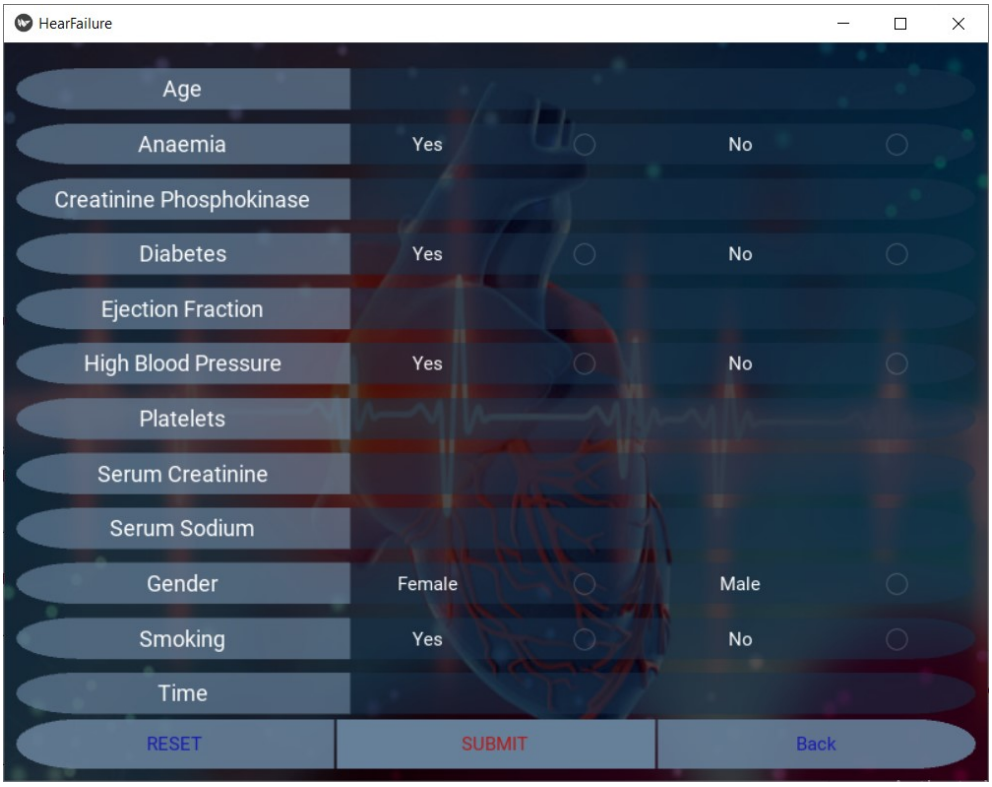


11.2 Application usage

i. Welcome Window

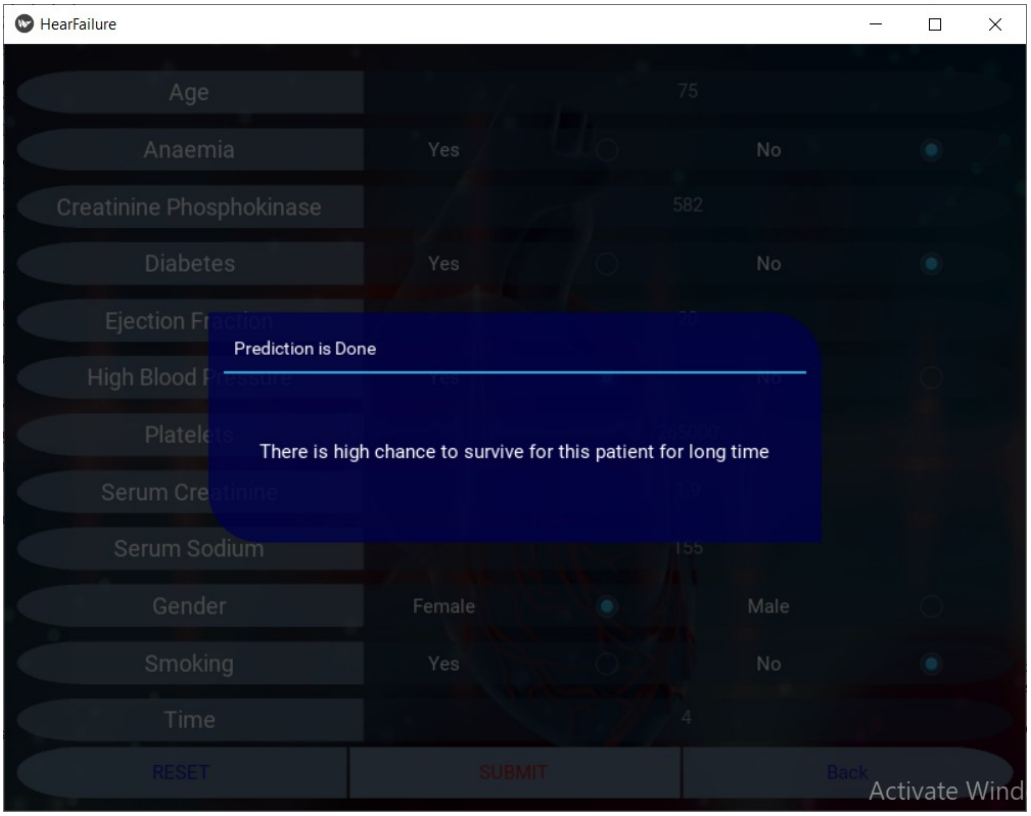


ii. Clicking on prediction Form button



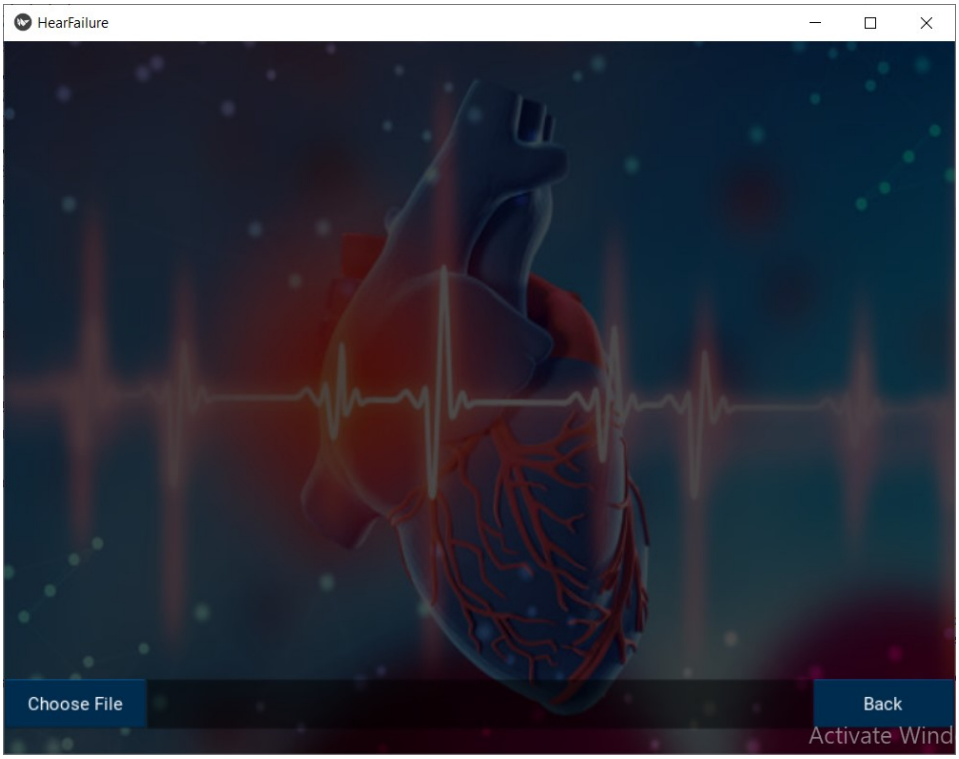
iii. Filling form

iv. Click on Submit button (pop up will be generated on prediction)

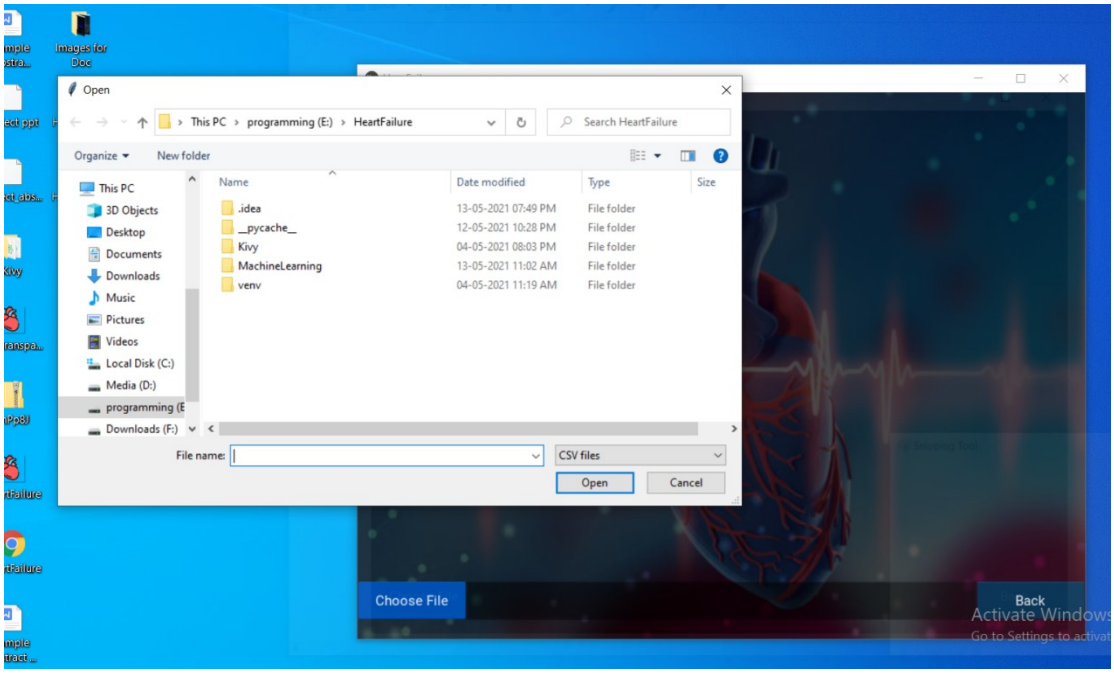


By pressing on any where inside window other than popup will close popup

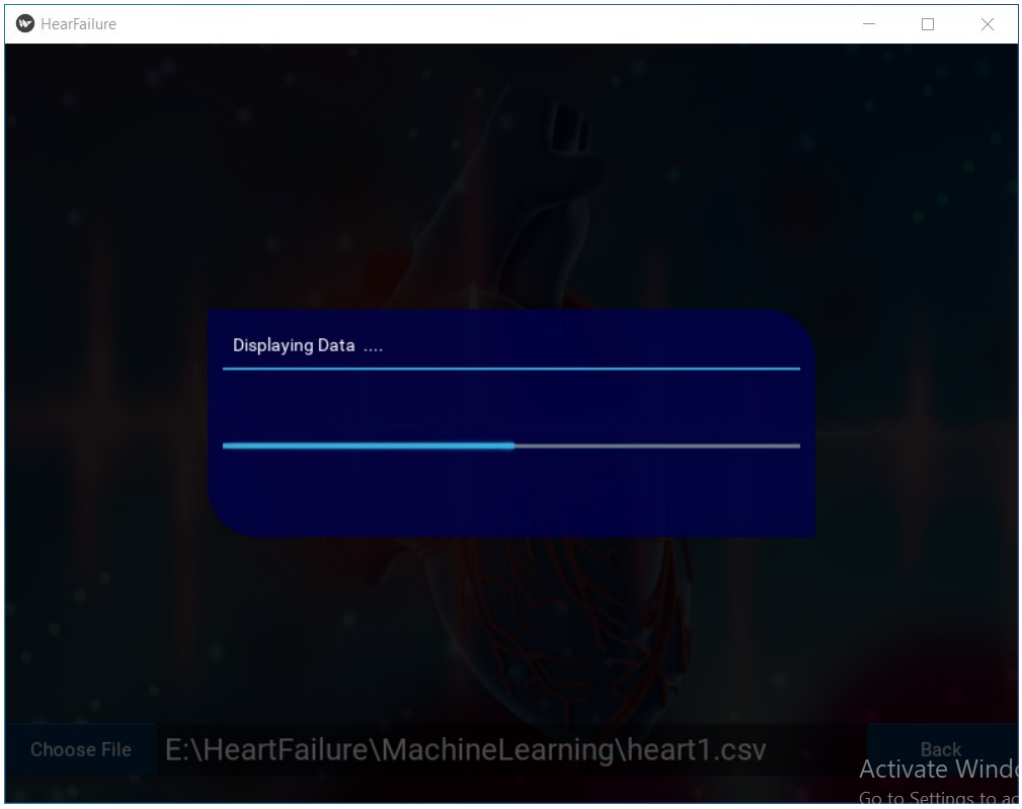
v. Pressing on File



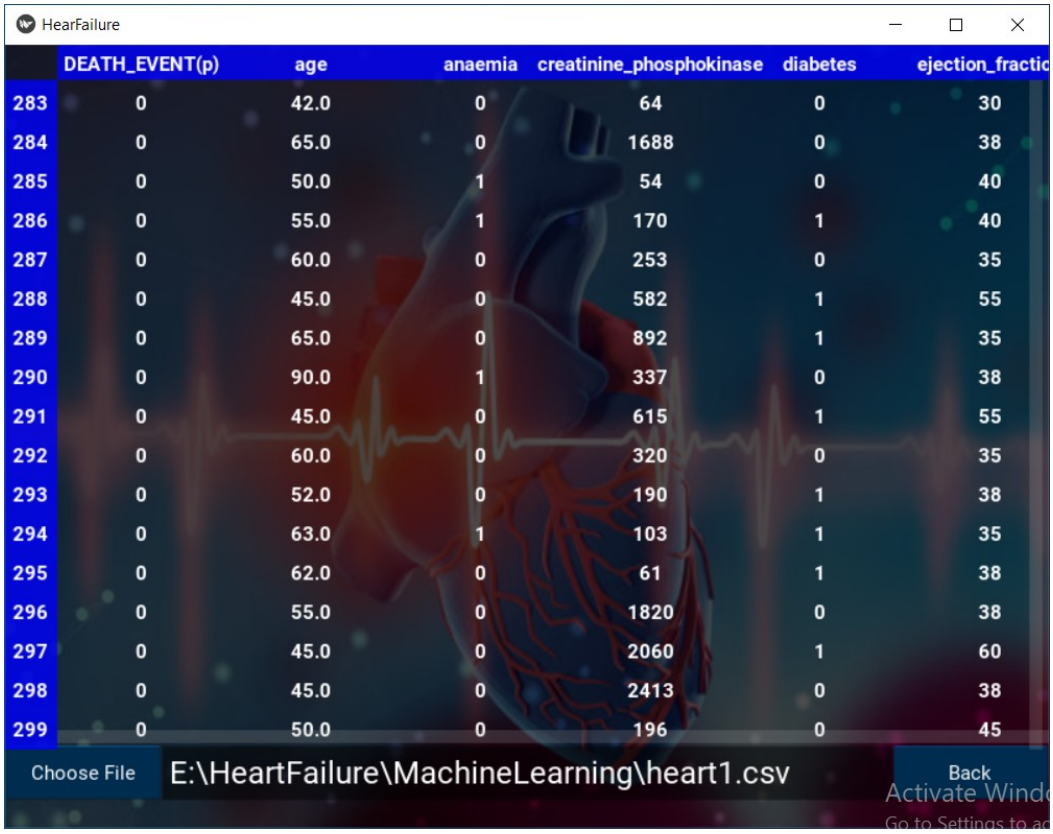
vi. Pressing on choose file button



vii. Selecting file and data loading popup will be displayed

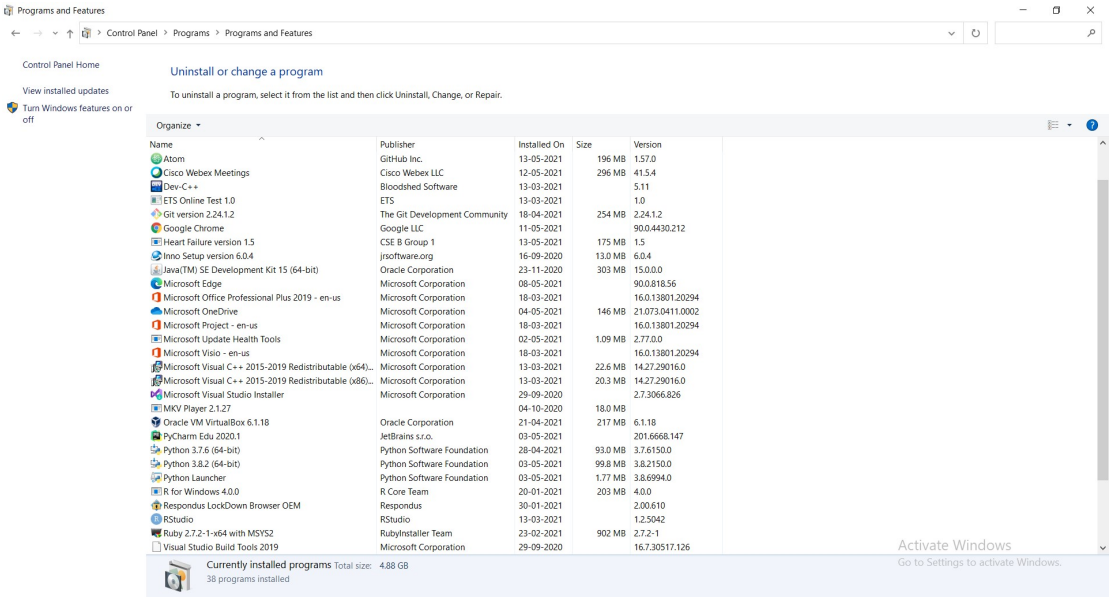


viii. Data displayed

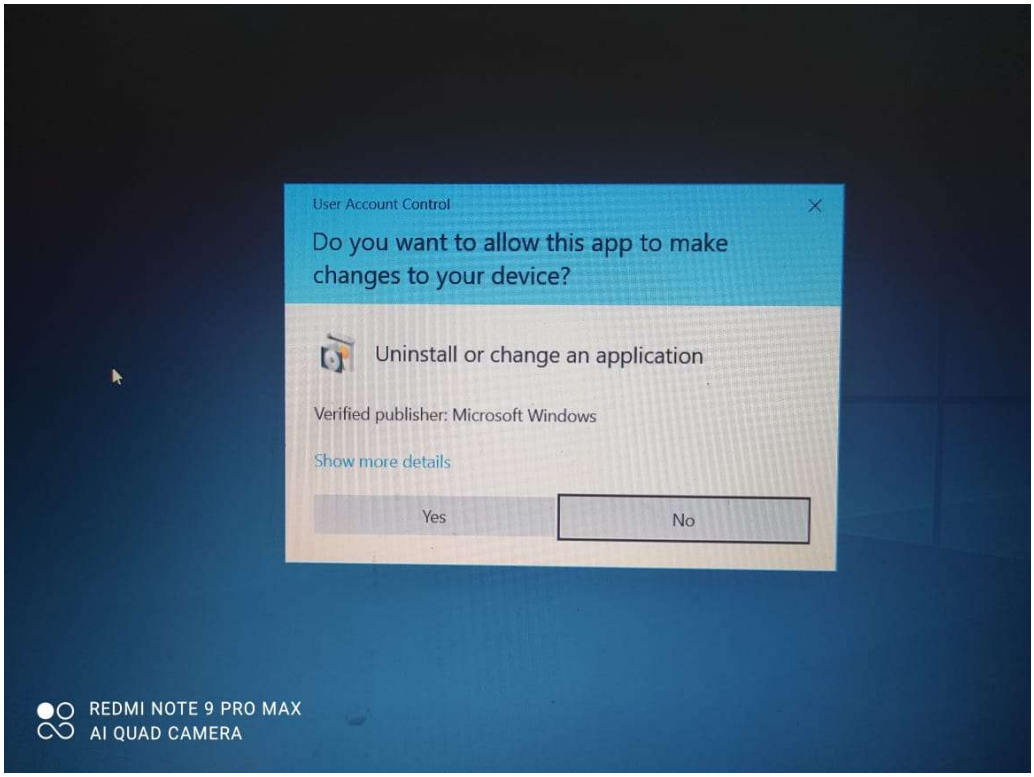


11.3 Uninstalling

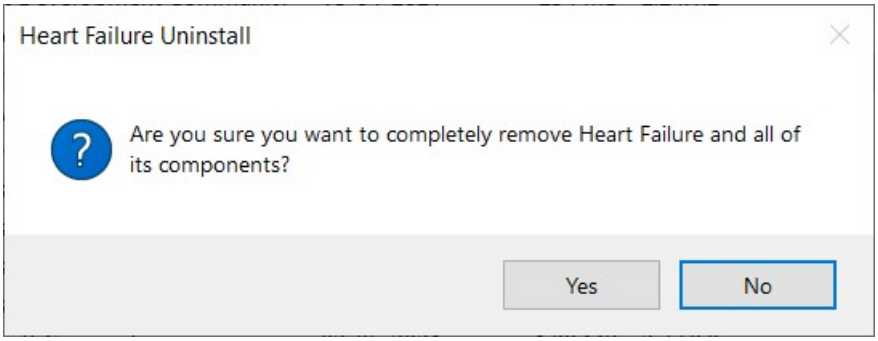
i. Go to control panel



ii. Uninstall



iii. Verification for uninstall HearFailure



iv. Successfully uninstalled

12. Bibliography

World Health Organization, World Heart Day. https://www.who.int/cardiovascular_diseases/world-heart day/en/. Accessed 7 May 2019.

The Guardian. UK heart disease fatalities on the rise for the first time in 50 years. https://www.theguardian.com/society/2019/may/13/heart circulatory-disease-fatalities-on-rise-in-UK. Accessed 25 Oct 2019.

National Heart Lung and Blood Institute (NHLBI). Heart failure. https://www.nhlbi.nih.gov/health-topics/heart-failure. Accessed 20 June 2019.

Data source (2015 years data) https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

For kivy and other application based information

Kivy.org , stackoverflow.com

Youtube playlist on kivy

https://www.youtube.com/watch?v=dLgquj0c5_U&list=PLCC34OHNcOtpz7PJQ7Tv7hqFBP_xDDjqg

Data collected from

https://www.kaggle.com/andrewmvd/hear-failure-clinical-data

Machine Learning free course from (Grayatom)

https://greyatom.com