INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Applied Data Science

Capstone Project – Health Care

FINAL PRESENTATION FOR SUBMISSION

"PREDICTING THE MORTALITY CAUSED BY HEART FAILURE USING MACHINE LEARNING MODELS"

Prepared by : ARUN RAMAMURTHY
Date : 15 - May 2023

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Organizational Background

Organizational Background:

Ram Health care is a reputed company, established in 1981 in the field of Bio-Medical instrument manufacturing segment. They have several medical instrument including Thermometer, Blood pressure monitoring device, Endoscopes, Ventilator etc.

As their business expansion, they planned to develop a medical application. This Application (App) can be used by various individuals to predict the death caused by Cardio-Vascular diseases.

This prediction is purely based on the several input factors entered by individuals who are interested to use this App.

Some of the highlights of this application is:

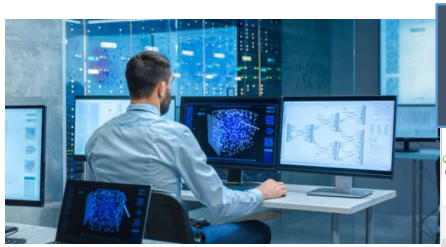
- Individuals can enter input data (which shall be explained in the upcoming slides) such as Blood pressure, Diabetes condition of the individuals, etc.
- The application software will analyse the data using in-built Python software and Machine Learning algorithm for prediction
- The purpose of this App is to predict the chances of death caused due to Cardio-vascular disease based on the individual health related input data.

This Application is named as **"RAM - Cardiac Risk Calculator"**

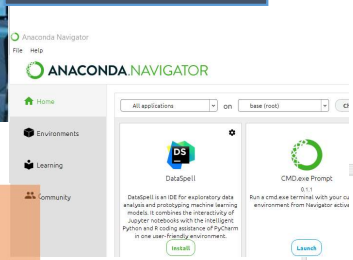
Resource and software requirement for this project

Resource and software:

In order to develop this Application software, they handed over project to In-house development team and the team listed out the requirement specification as mentioned below.



Software engineer with knowledge of Python programming and ML algorithms



Anaconda Navigator software which is available for free download from Google

age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	t
75	0	582	0	20	1	265000	1.9	130	1	0	
55	0	7861	0	38	0	263358	1.1	136	1	0	
65	0	146	0	20	0	162000	1.3	129	1	1	
50	1	111	0	20	0	210000	1.9	137	1	0	
65	1	160	1	20	0	327000	2.7	116	0	0	
90	1	47	0	40	1	204000	2.1	132	1	1	
75	1	246	0	15	0	127000	1.2	137	1	0	
60	1	315	1	60	0	454000	1.1	131	1	1	
65	0	157	0	65	0	263358					
80	1	123	0	35	1	3880					

Valid dataset which defines which are the inputs to consider for predicting mortality caused by Cardio-vascular disease

Jupyter notebook, which is embedded within Anaconda navigator

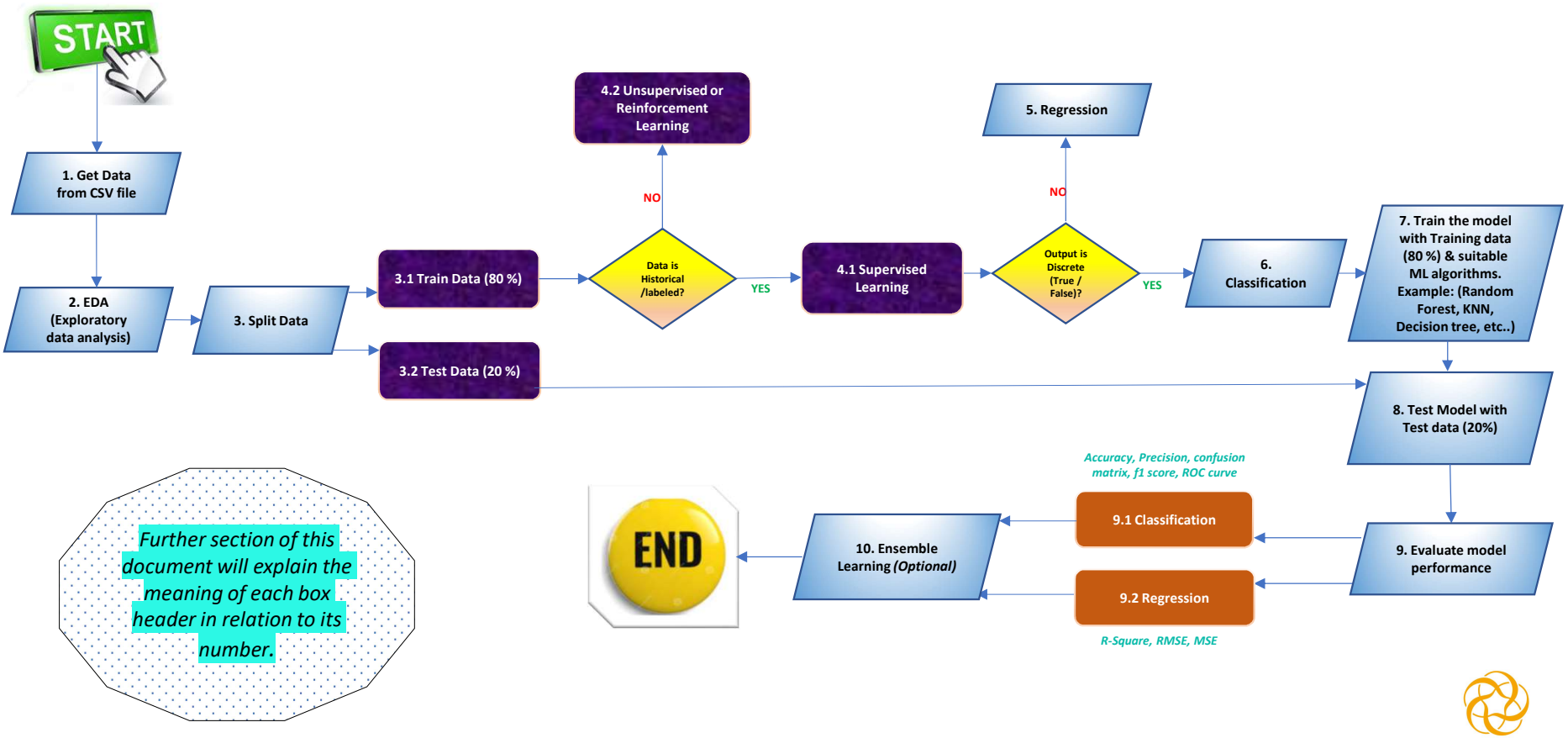
Python programming environment –which is available in Jupyter notebook

Further slides will take you through Dataset and software details used in this project to execute the project

"RAM - Cardiac Risk Calculator"

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Steps need to follow to prepare a model in order to detect mortality caused by cardiovascular disease



Further section of this document will explain the meaning of each box header in relation to its number.

1. Get Data from CSV

In this project, as a first step we need to get the given data from Excel into python project for further analyzation.

As a common practice, this is done by importing panda's library.

Function used for this is
pd.read_csv("./Data/….filename…. .csv")

Additional information about dataset:

- *As shown in previous slide, we have several input parameters.*
- *Output prediction is "Death Event" as mentioned in the last column*
- *Purpose of this machine learning algorithm is to predict the "Death event" based on all given input parameters*

age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
75	0	582	0	20	1	265000	1.9	130	1	0	4	1
55	0	7861	0	38	0	263358	1.1	136	1	0	6	1
65	0	146	0	20	0	162000	1.3	129	1	1	7	1
50	1	111	0	20	0	210000	1.9	137	1	0	7	1
65	1	160	1	20	0	327000	2.7	116	0	0	8	1
90	1	47	0	40	1	204000	2.1	132	1	1	8	1
75	1	246	0	15	0	127000	1.2	137	1	0	10	1
60	1	315	1	60	0	454000	1.1	131	1	1	10	1
65	0	157	0	65	0	263358	1.5	138	0	0	10	1
80	1	123	0	35	1	388000	9.4	133	1	1	10	1
75	1	81	0	38	1	368000	4	131	1	1	10	1
62	0	231	0	25	1	253000	0.9	140	1	1	10	1
45	1	981	0	30	0	136000	1.1	137	1	0	11	1
50	1	168	0	38	1	276000	1.1	137	1	0	11	1
49	1	80	0	30	1	427000	1	138	0	0	12	0
82	1	379	0	50	0	47000	1.3	136	1	0	13	1
87	1	149	0	38	0	262000	0.9	140	1	0	14	1
45	0	582	0	14	0	166000	0.8	127	1	0	14	1
70	1	125	0	25	1	237000	1	140	0	0	15	1
48	1	582	1	55	0	87000	1.9	121	0	0	15	1
65	1	52	0	25	1	276000	1.3	137	0	0	16	0
65	1	128	1	30	1	297000	1.6	136	0	0	20	1
68	1	220	0	35	1	289000	0.9	140	1	1	20	1
53	0	63	1	60	0	368000	0.8	135	1	0	22	0

1.1 – Get data from CSV file + Import all required libraries

In this project, already data is provided in the CSV format. We need to get this data in to our Python environment so that, we can start working with this.

Following methods are used to complete this task.

- Import necessary libraries in Python code. (Panda's library is the one which help us to get the Data form CSV file)
- Here, data frame name is **cvd**
- Read the Header of top 5 lines to get a quick look into data and its contents.
- The Python code to do this job is enclosed in this slide.
- Remaining libraries are imported for other purpose, which shall be explained in next sections

In [1]:
```
print ("ಓಂ ಗಣೇಶಾಯ ನಮಃ")
```

```
ಓಂ ಗಣೇಶಾಯ ನಮಃ
```

1. Import all necessary libraries and read the CSV file(Dataset)

In [2]:
```
import warnings
warnings.filterwarnings('ignore')

#data wrangling & processing
import pandas as pd
import numpy as np


#data plotting & visualization
import seaborn as sns
import matplotlib # for Visualisations
import matplotlib.pyplot as plt
%matplotlib inline


from scipy import stats
```

In [3]:
```
cvd = pd.read_csv("Data/CVD.csv")
```

In [4]:
```
cvd.head(5)
```

Out[4]:

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

2. EDA (Exploratory Data Analysis)

EDA involves various process related to data.

1. Collect the data to get application usage

2. Make sure, we have all relevant data and no null values

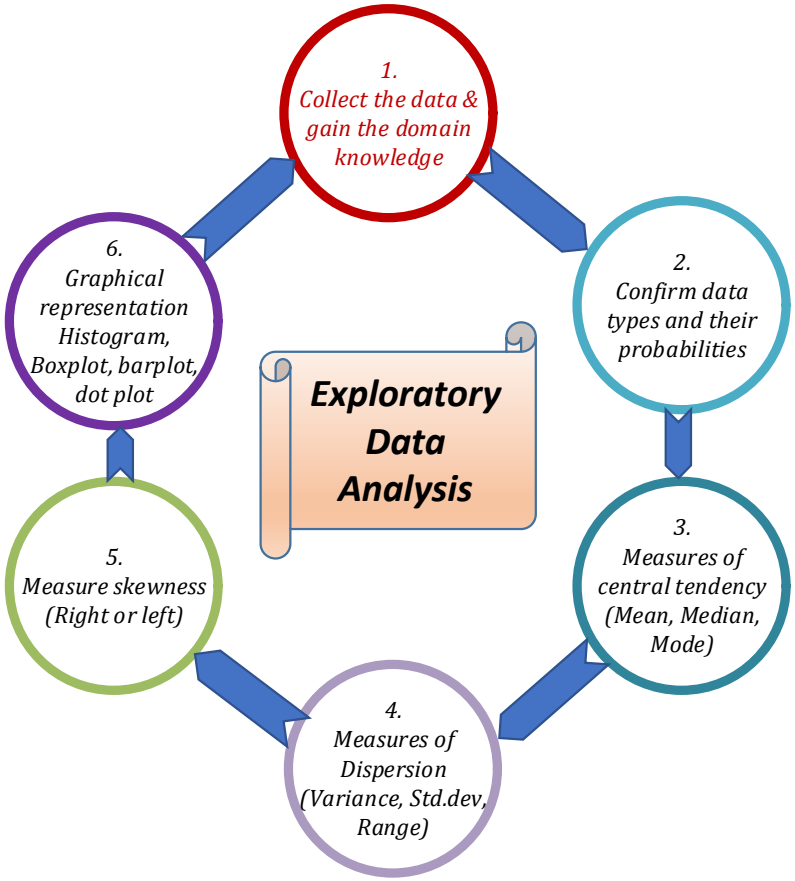
3. Measure the mean/median of data set

4. Measure the data variance and deviation

5. Check for right or left skewness of data

6. Plot various charts of data like Histogram, boxplot, bar plot, etc.

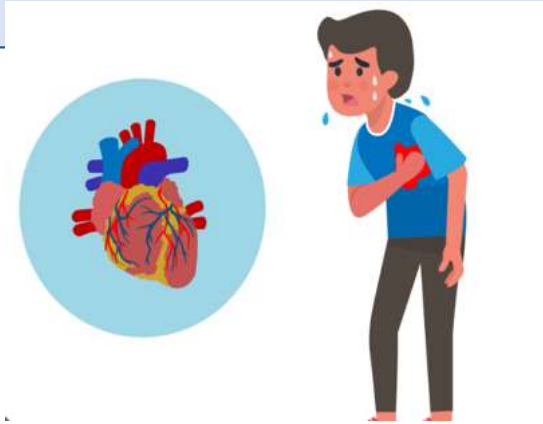
To conclude this is vary basic step and the output data from this process helps us to predict better results



- - As per WHO statistics, it has been estimated that **27%** of the deaths in India are caused by **Heart disease**.

https://www.who.int/india/health-topics/cardiovascular-diseases

- **One-Third** of the global deaths are due to Heart disease.
- This situation is same in other developed countries (US/ Europe) as well.
- Around **17.9 million** people die due to Cardiovascular disease (CVD) every year worldwide.



2. EDA (Exploratory Data Analysis)

As a initial step, first let us understand the given dataset and meaning of variables

Sl.No	Feature as per given dataset	Rename within project for clarity	Details / Meaning	Data type
1	Age	--	Patients Age in Years	Numeric
2	anaemia	--	Anaemia is a medical condition characterized by a lack of red blood cells or hemoglobin (Anaemia present = 1, No Anaemia = 0)	**Boolean**
3	creatinine_phosphokinase	CPK	Enzime indicating muscle damage and important for diagnosing heart attack. Normal range (10 - 120 units per Ltr)	Numeric
4	diabetes	--	Metabolic disorder characterized by high blood sugar levels due to insulin dysfunction. (Diabetic = 1, Non Diabetic = 0)	**Boolean**
5	ejection_fraction	EF	Ejection factor is a measure of how well the heart is pumping blood out of its chambers. (Normal range 50% to 70%)	Numeric
6	high_blood_pressure	High BP	It is a condition where the force of blood against the walls of the arteries is consistently too high. A person, with consistenltly above 130/80 mmHg is considered as high BP.	**Boolean**
7	platelets	--	Platelets are small blood cells that help with clotting. The normal platelet count in a healthy person is typically between 150,000 to 450,000 platelets per microliter of blood.	Numeric
8	serum_creatinine	Scr	Measure of the kidneys' ability to filter waste products from the blood and is used to diagnose and monitor kidney-related conditions (Normal range 0.5 mgg/dl o 1.3 mg/Dl)	Numeric
9	serum_sodium	S_Sodium	it is an electrolyte that helps regulate fluid balance and blood pressure in the body. Normal range (135 to 145 mEq / L)	Numeric
10	sex	--	Gender of Patient (Male = 1, Female = 0)	**Boolean**
11	smoking	--	To indicate smoking habbit (Smoker = 1 : Non-smoker = 0)	**Boolean**
12	time	--	Not related variable. We can remove this data for our ML model	Numeric
13	DEATH_EVENT	target	It is the target variable which we have to predict 1 means patient is sufeering from Heart disease & 0 means patient is normal	**Boolean**

(EDA) 2.2 – Rename column names + remove column which is not necessary

Rename the dataset column names

In [5]:
```
##For better understanding, we shall rename the data set contents to meaningful names. This will help us to visualize graphs
##and furhter processing of the variable names where-ever needed ##

cvd = cvd.rename(columns = {"creatinine_phosphokinase" : "CPK", "ejection_fraction" : "EF", "high_blood_pressure" : "High BP",
                            "serum_creatinine": "Scr","serum_sodium" : "S_Sodium", "DEATH_EVENT": "target" })
```

In [6]:
```
cvd.head(3) # Check for the revised or modified variable names in the dataset
```

Out[6]:

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	time	target
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

As a next step, in the given dataset, TIME column is not giving us any information. Hence, we can delete this column

In [7]:
```
#Removing the time column from dataset using drop command#

cvd = cvd.drop(columns = ['time'])
```

As a next step, in the given dataset, CPK value is given in 3 digits. Where as standard CPK value is 10 to 120 micrograms per liter(mcg/L). Hence, we divide this by fixed value 10 and store the data back to dataframe. This will make sure, we are using standard unit of CPK in our project

In [8]:
```
cvd['CPK'] = cvd['CPK']/10
print(cvd)
```

```
      age  anaemia    CPK  diabetes  EF  High BP  platelets  Scr  S_Sodium  \
0    75.0        0   58.2         0  20        1  265000.00  1.9       130
1    55.0        0  786.1         0  38        0  263358.03  1.1       136
2    65.0        0   14.6         0  20        0  162000.00  1.3       129
3    50.0        1   11.1         0  20        0  210000.00  1.9       137
4    65.0        1   16.0         1  20        0  327000.00  2.7       116
..    ...      ...    ...       ...  ..      ...        ...  ...       ...
294  62.0        0    6.1         1  38        1  155000.00  1.1       143
295  55.0        0  182.0         0  38        0  270000.00  1.2       139
296  45.0        0  206.0         1  60        0  742000.00  0.8       138
297  45.0        0  241.3         0  38        0  140000.00  1.4       140
298  50.0        0   19.6         0  45        0  395000.00  1.6       136

     sex  smoking  target
0      1        0       1
1      1        0       1
2      1        1       1
3      1        0       1
4      0        0       1
..   ...      ...     ...
294    1        1       0
295    0        0       0
296    0        0       0
297    1        1       0
298    1        1       0

[299 rows x 12 columns]
```

(EDA) 2.3 – Rename dataset + check header & tail of dataset + check null values + Check datatypes + convert data type if needed

For better understanding, we rename contents of dataset to meaning ful names instead of 1 & 0 for variable SEX

In [9]:
```
#Use Lambda python function and assign if column has value=1 (Male) or else Value=0 (Female)

cvd['sex'] = cvd.sex.apply(lambda x:'male' if x==1 else 'female')

cvd.head(3)          #Verify the previous steps by checking column "sex" & verify the removal of column "time"
```

Out[9]:

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
0	75.0	0	58.2	0	20	1	265000.00	1.9	130	male	0	1
1	55.0	0	786.1	0	38	0	263358.03	1.1	136	male	0	1
2	65.0	0	14.6	0	20	0	162000.00	1.3	129	male	1	1

In [10]:
```
cvd.tail(3)
```

Out[10]:

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
296	45.0	0	206.0	1	60	0	742000.0	0.8	138	female	0	0
297	45.0	0	241.3	0	38	0	140000.0	1.4	140	male	1	0
298	50.0	0	19.6	0	45	0	395000.0	1.6	136	male	1	0

Find shape of our dataset (Rows & Columns) + find the data types + Check for Null values / Float values and resolve where ever needed

In [11]:
```
cvd.shape
```

Out[11]:
```
(299, 12)
```

In [12]:
```
print ("Number of Rows", cvd.shape[0])
print ("Number of Columns", cvd.shape[1])
```

```
Number of Rows 299
Number of Columns 12
```

Check Null values in the dataset + Get overall statistics about dataset

In [13]:
```
cvd.isnull().sum()
```

Out[13]:
```
age          0
anaemia      0
CPK          0
diabetes     0
EF           0
High BP      0
platelets    0
Scr          0
S_Sodium     0
sex          0
smoking      0
target       0
dtype: int64
```

In [14]:
```
cvd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        299 non-null    float64
 1   anaemia    299 non-null    int64
 2   CPK        299 non-null    float64
 3   diabetes   299 non-null    int64
 4   EF         299 non-null    int64
 5   High BP    299 non-null    int64
 6   platelets  299 non-null    float64
 7   Scr        299 non-null    float64
 8   S_Sodium   299 non-null    int64
 9   sex        299 non-null    object
 10  smoking    299 non-null    int64
 11  target     299 non-null    int64
dtypes: float64(4), int64(7), object(1)
memory usage: 28.2+ KB
```

This shows, there are 2 Float values at several column which is not really required

In [15]:
```
##Convert floating point values to Integer for variables "age" & "platelets" for further clarity & store it in same DataFrame
##Then verify the datatypes, so that Age & Platelets datatypes are converted to integer

cvd = cvd.astype({"age":"int", "platelets": "int"})
print(cvd.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        299 non-null    int32
 1   anaemia    299 non-null    int64
 2   CPK        299 non-null    float64
 3   diabetes   299 non-null    int64
 4   EF         299 non-null    int64
 5   High BP    299 non-null    int64
 6   platelets  299 non-null    int32
 7   Scr        299 non-null    float64
 8   S_Sodium   299 non-null    int64
 9   sex        299 non-null    object
 10  smoking    299 non-null    int64
 11  target     299 non-null    int64
dtypes: float64(2), int32(2), int64(7), object(1)
memory usage: 25.8+ KB
None
```

(EDA) 2.4 –check dataset tendency + Check for co-relation + Heatmap

Measure the central tendency of the data set (Mean, Min, Max, 25%, 50%.... etc)

In [16]:
```
cvd.describe()
```

Out[16]:

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	smoking	
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.00000	299.
mean	60.829431	0.431438	58.183946	0.418060	38.083612	0.351171	263358.026756	1.39388	136.625418	0.32107	0.
std	11.894997	0.496107	97.028788	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477	0.46767	0.
min	40.000000	0.000000	2.300000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000	0.00000	0.
25%	51.000000	0.000000	11.650000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000	0.00000	0.
50%	60.000000	0.000000	25.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000	0.00000	0.
75%	70.000000	1.000000	58.200000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000	1.00000	1.
max	95.000000	1.000000	786.100000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000	1.00000	1.

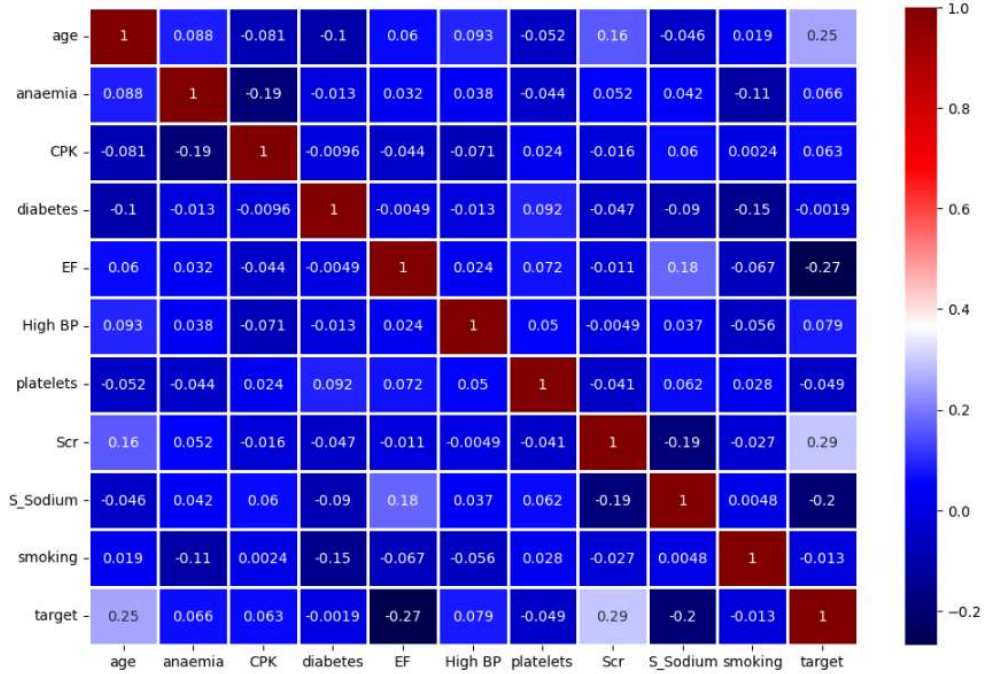
Find the co-relation between the data & draw seaborn heatmap

In [17]:
```
cvd.corr()
```

Out[17]:

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	smoking	target
age	1.000000	0.087574	-0.081406	-0.101454	0.060195	0.093170	-0.052475	0.159237	-0.045912	0.018926	0.253581
anaemia	0.087574	1.000000	-0.190741	-0.012729	0.031557	0.038182	-0.043786	0.052174	0.041882	-0.107290	0.066270
CPK	-0.081406	-0.190741	1.000000	-0.009639	-0.044080	-0.070590	0.024463	-0.016408	0.059550	0.002421	0.062728
diabetes	-0.101454	-0.012729	-0.009639	1.000000	-0.004850	-0.012732	0.092193	-0.046975	-0.089551	-0.147173	-0.001943
EF	0.060195	0.031557	-0.044080	-0.004850	1.000000	0.024445	0.072177	-0.011302	0.175902	-0.067315	-0.268603
High BP	0.093170	0.038182	-0.070590	-0.012732	0.024445	1.000000	0.049963	-0.004935	0.037109	-0.055711	0.079351
platelets	-0.052475	-0.043786	0.024463	0.092193	0.072177	0.049963	1.000000	-0.041198	0.062125	0.028234	-0.049139
Scr	0.159237	0.052174	-0.016408	-0.046975	-0.011302	-0.004935	-0.041198	1.000000	-0.189095	-0.027414	0.294278
S_Sodium	-0.045912	0.041882	0.059550	-0.089551	0.175902	0.037109	0.062125	-0.189095	1.000000	0.004813	-0.195204
smoking	0.018926	-0.107290	0.002421	-0.147173	-0.067315	-0.055711	0.028234	-0.027414	0.004813	1.000000	-0.012623
target	0.253581	0.066270	0.062728	-0.001943	-0.268603	0.079351	-0.049139	0.294278	-0.195204	-0.012623	1.000000

In [18]:
```
##Heatmap to find the co-relation
plt.figure(figsize = (12,8))
sns.heatmap(cvd.corr(), linewidth = 1, annot = True, cmap = "seismic")
plt.show()
```



Conclusion from Heatmap (co-relation) statistics:

In our dataset, "**Age**" & **"SCR"** are positively co-related with target variable(as they are showing very light colour)

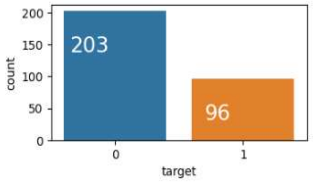
"EF" & "S_Sodium" are negatively co-related with target variable (as they are showing very dark colour)

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.5 – Plotting related to target variable + Histograms

According to dataset, how many people have Heart disease(target = 1 & how many people are Normal(target = 0) , show it in graph

In [19]:
```
plt.figure(figsize = (4,2))

ax=sns.countplot(x=cvd['target'], data=cvd);

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.2, p.get_height()), ha='center', va='top', color='white', size=18)
plt.show()

cvd['target'].value_counts().to_frame()
```



Out[19]:

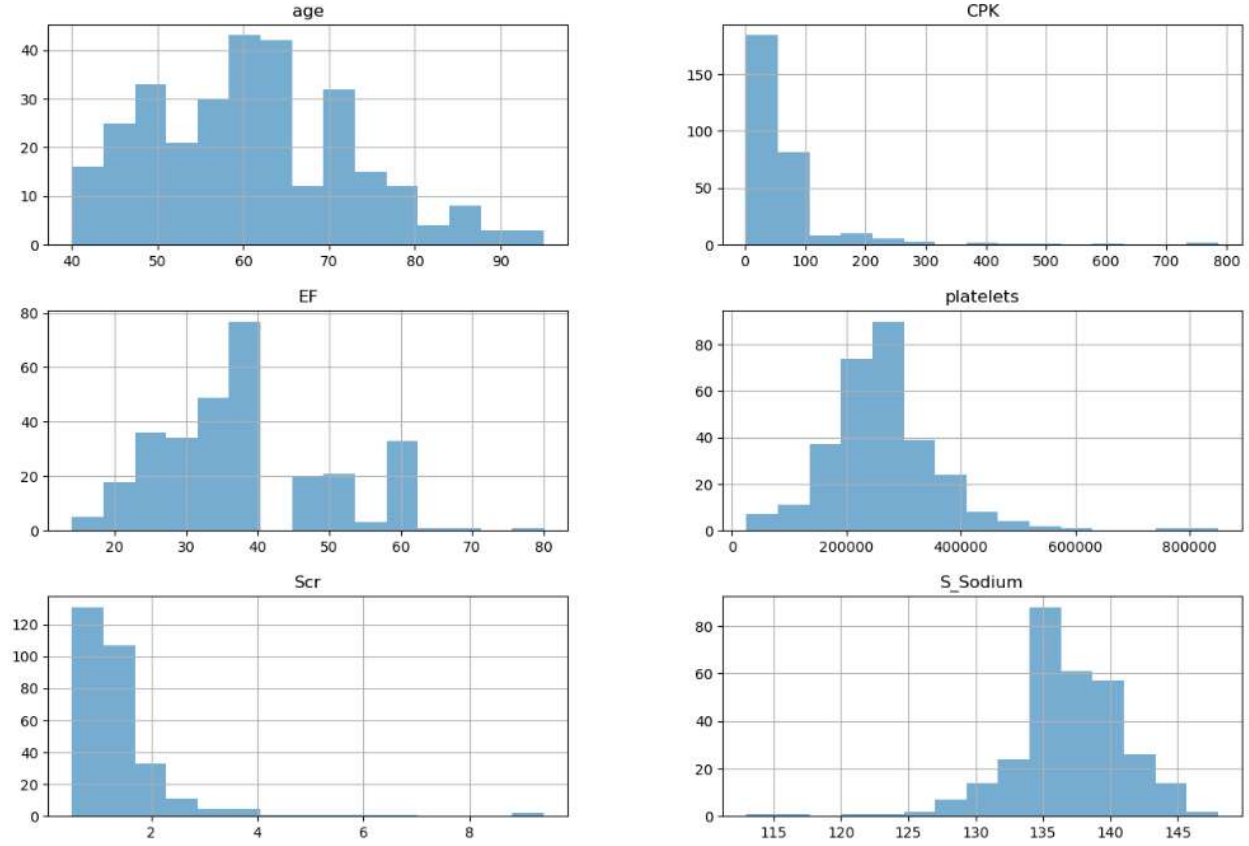
	target
0	203
1	96

- The ratio of the people who are with Heart disease are more comparitively to the people who have less chance of getting it (203 = Heart attack / 9 risk)

- Heart disease -> Blue
- Normal -> Orange
- Conclusion: In this dataset, the ratio of the people who already got heart attack is more comparatively to the ople who have less chance of getting

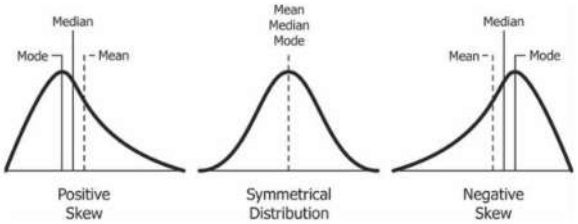
As a first step, we can visualize the Histogram to understand our data. about each parameter and mejority of values of dataset. Example,
- this in this dataset, 55 years to 65 years patients are more
- with in this dataset, S_Sodium value of most of the people is between the range 132 to 142

```
cvd[['age', 'CPK', 'EF', 'platelets', 'Scr', 'S_Sodium']]. hist(bins=15, figsize=(15,10), alpha=0.6, grid=True)
plt.show()
```

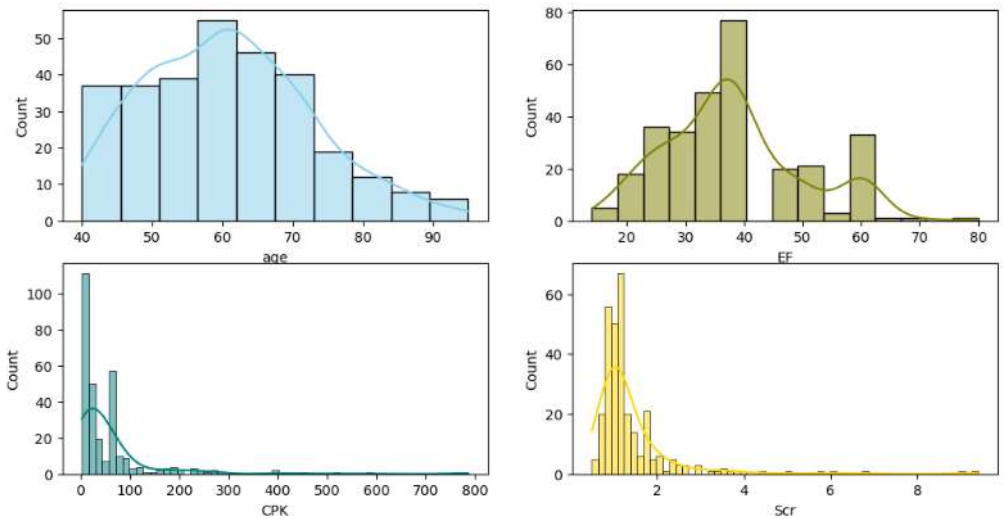


(EDA – Plots & Graphs) 2.6 – Check skewness of data using seaborn library "histplot" & "swarmplot"

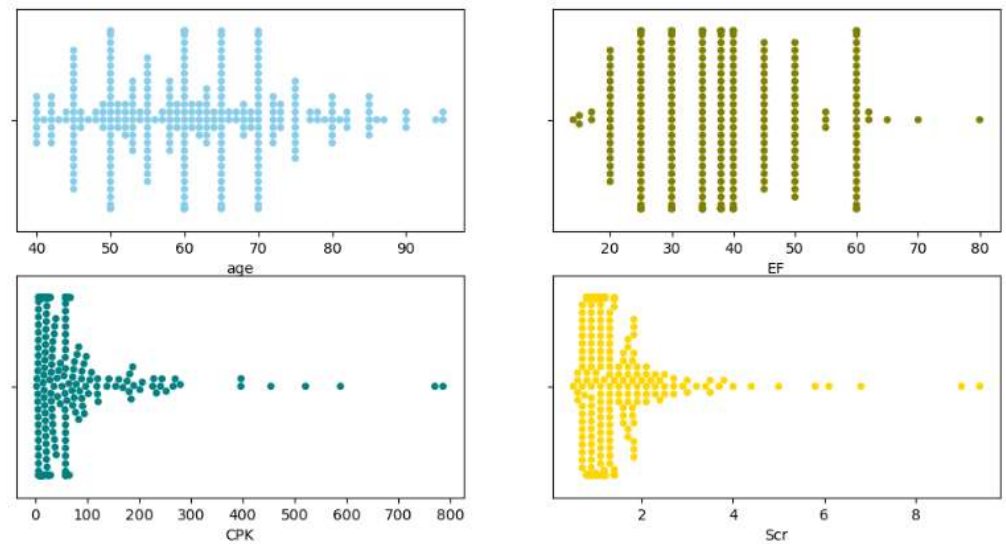
As a next step, we can see the subplots and also we can see the data skewness - (Positive skew, Negetive skew or Symmetrical distribution of data)
- To achive this, we are going to histplot from seaborn library
- Along with this, we also see data distribution and have a look into where most of the data value fits, and see outliers on each numerical data




```
#data shown using Histplot seaborn functionality. Data shown here are Age, EF, CPK, SCR
fig, axs = plt.subplots(2, 2, figsize=(12,6))
sns.histplot(data=cvd, x='age', kde=True, color="skyblue", ax=axs[0,0])
sns.histplot(data=cvd, x='EF', kde=True, color="olive", ax=axs[0,1])
sns.histplot(data=cvd, x='CPK', kde=True, color="teal", ax=axs[1,0])
sns.histplot(data=cvd, x='Scr', kde=True, color="gold", ax=axs[1,1])
plt.show()
```




```
#data shown using Histplot swarmplot functionality. Data shown here are Age, EF, CPK, SCR
fig, axs = plt.subplots(2, 2, figsize=(12,6))
sns.swarmplot(data=cvd, x='age', color="skyblue", ax=axs[0,0])
sns.swarmplot(data=cvd, x='EF', color="olive", ax=axs[0,1])
sns.swarmplot(data=cvd, x='CPK', color="teal", ax=axs[1,0])
sns.swarmplot(data=cvd, x='Scr', color="gold", ax=axs[1,1])
plt.show()
```



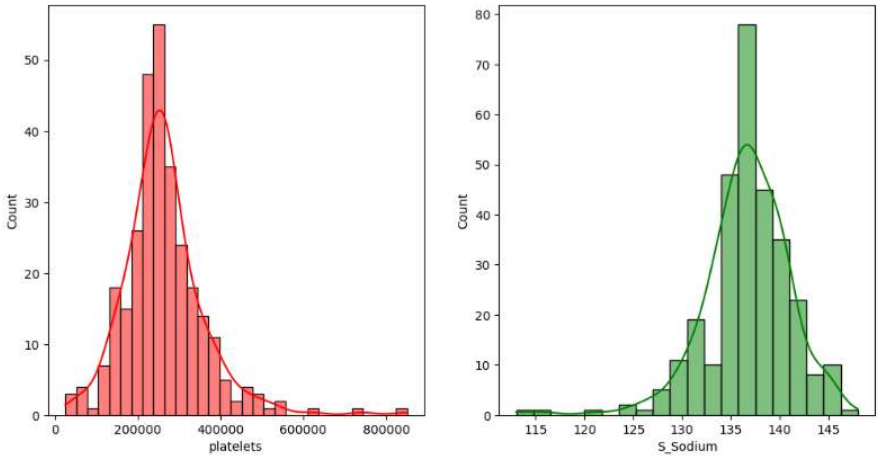
By looking into above graphs, we can understand
- Age: we have more people between age group 50 to 70 in our dataset (Symmetrical skew)
- EF: 25 to 40 is the value of EF most of the patients in our dataset (Symmetrical skew)
- CPK: most of the patients CPK level in our dtaset is between 5.0 to 100.0 (Normal value 10 -120) & (left skew)
- SCR: most of the patients SCR level is 0 - 1.5. (left skew)

Other important thing, outliers are present all the 4 variables

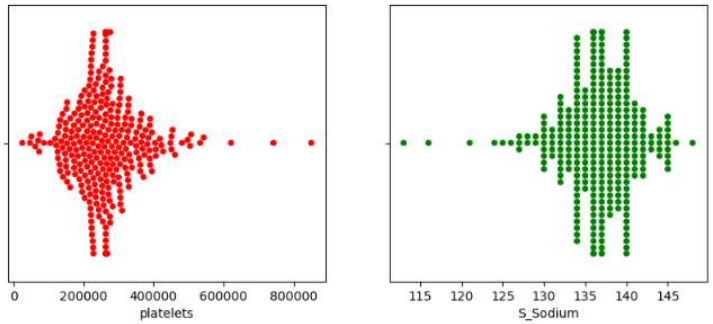
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.7 – Check skewness of data using seaborn library “histplot” & “swarmplot” + Create separate data frames related to target

```
sns.histplot(data=cvd, x='S_Sodium', kde=True, color="green", ax=axs[1])

plt.show()
```




```
#data shown using seaborn swarmplot functionality. Data shown here are platelets & S_Sodium
fig, axs = plt.subplots(1, 2, figsize=(10, 4))
sns.swarmplot(data=cvd, x='platelets',color="red", ax=axs[0])
sns.swarmplot(data=cvd, x='S_Sodium',color="green", ax=axs[1])
plt.show()
```



By looking into above graphs, we can understand

- Platelets: Most of the data is between 140000 to 400000 (left skew)
- S_Sodium: Most of the data is between 130 to 140 (Right skew)

Other important thing, outliers are present in both variables

Creating separate data frames for normal & Heart disease patients to establish co-relation between different parameters

```
attr_1 = cvd[cvd['target'] == 1]   #attr_1 is a new variable, which will be true when Target = 1

attr_0 = cvd[cvd['target'] == 0]   #attr_0 is a new variable, which will be true when Target = 0
```

```
attr_1.head(3) # This will list out the dataset when target value is 1 (Patient who has Heart disease)
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
0	75	0	58.2	0	20	1	265000	1.9	130	male	0	1
1	55	0	786.1	0	38	0	263358	1.1	136	male	0	1
2	65	0	14.6	0	20	0	162000	1.3	129	male	1	1

```
attr_0.head(3) # This will list out the dataset when target value is 0 (Patient who are normal)
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
14	49	1	8.0	0	30	1	427000	1.0	138	female	0	0
20	65	1	5.2	0	25	1	276000	1.3	137	female	0	0
23	53	0	6.3	1	60	0	368000	0.8	135	male	0	0

We can comapre age with few input parameters to check relation

In below plots, it shows two things

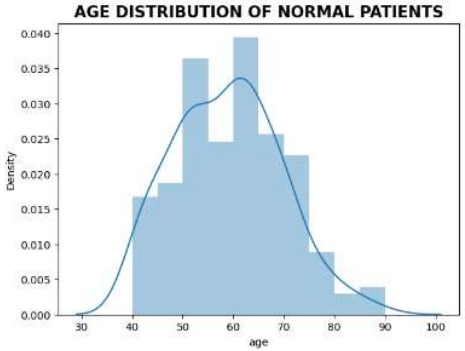
-histogram of patients "age" with heart disease and without heart disease (normal)

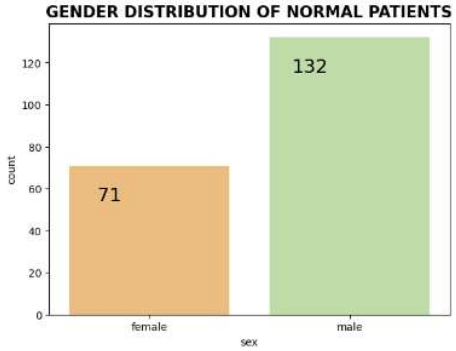
-second one shows Gender distribution graph of Male and Female with heart disease and normal

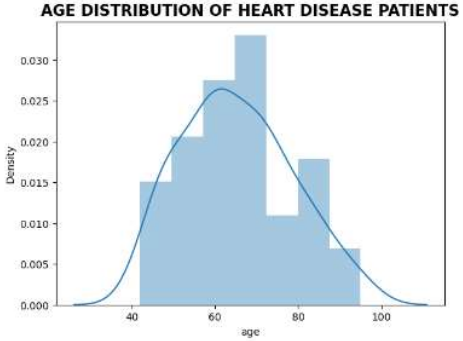
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

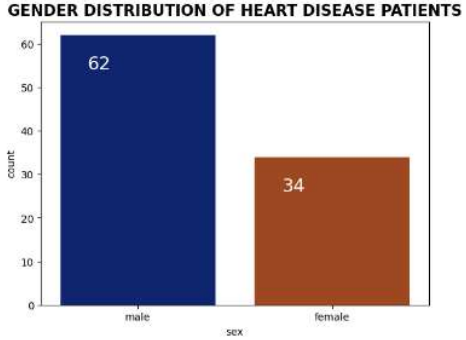
(EDA – Plots & Graphs) 2.8 – plotting all the input variables related to target (Heart disease)

```
#Plotting heart disease (CVD) patients
fig = plt.figure(figsize=(15,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.distplot(attr_1['age'])
plt.title('AGE DISTRIBUTION OF HEART DISEASE PATIENTS', fontsize = 15, weight='bold')

#This below code is to show the numbers on the graph
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attr_1['sex'], palette='dark')
plt.title('GENDER DISTRIBUTION OF HEART DISEASE PATIENTS', fontsize = 15, weight='bold')

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='white', size=18)

plt.show()
```







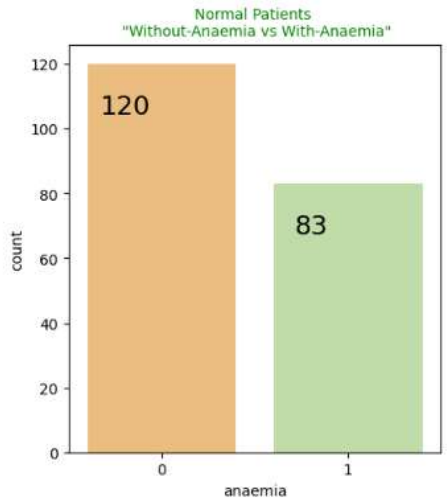


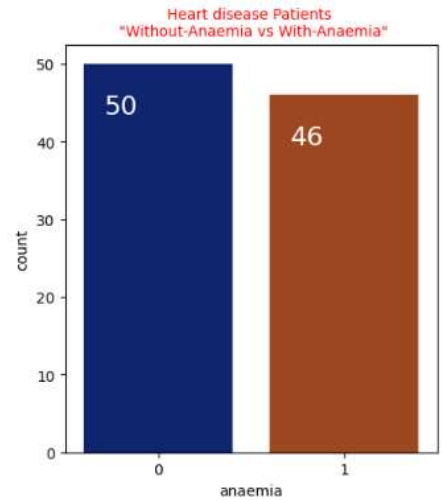
As per the dataset, these plots shows 2 things

- Most of the normal patients(Without Heart disease) are Male compare to females
- Male have more heart disease couunts compare to female

In below plots, it shows relation between "Heart disease" and "anaemia"

```
#plotting normal patients
fig = plt.figure(figsize=(10,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.countplot(attr_0['anaemia'], palette = 'Spectral')
plt.title('Normal Patients \n "Without-Anaemia vs With-Anaemia"', fontsize=10, color="green")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='black', size=18)

#plotting Heart disease patients
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attr_1['anaemia'], palette = 'dark')
plt.title('Heart disease Patients \n "Without-Anaemia vs With-Anaemia"', fontsize=10, color="red")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='white', size=18)
plt.show()
```

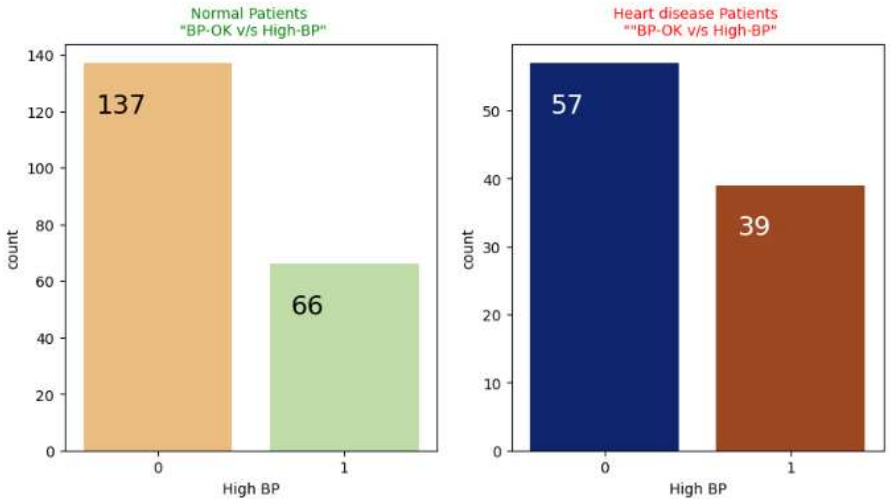




(EDA – Plots & Graphs) 2.9 – plotting all the input variables related to target (Heart disease)

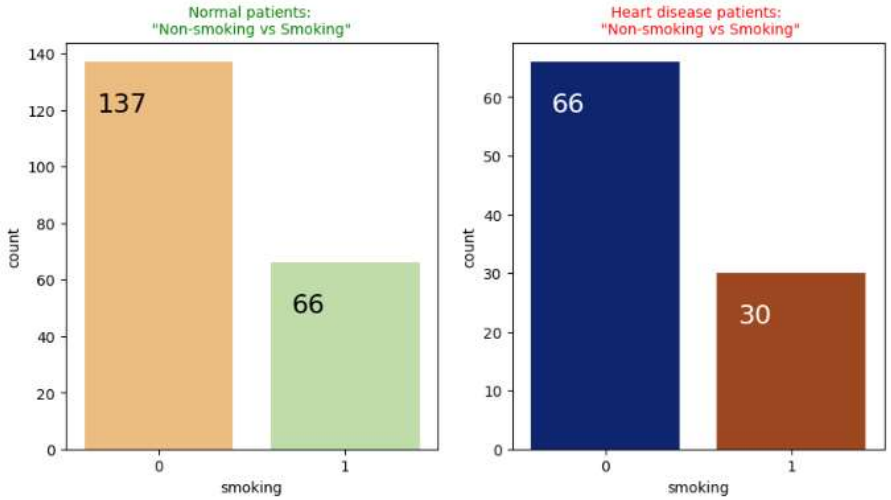
In below plots, it shows relation between "Heart disease" and "High blood pressure"

```
#plotting normal patients
fig = plt.figure(figsize=(10,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.countplot(attr_0['High BP'], palette = 'Spectral')
plt.title('Normal Patients \n "BP-OK v/s High-BP"', fontsize=10, color="green")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='black', size=18)

#plotting Heart disease patients
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attr_1['High BP'], palette = 'dark')
plt.title('Heart disease Patients \n ""BP-OK v/s High-BP"', fontsize=10, color="red")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='white', size=18)
plt.show()
```



In below plots, it shows relation between "Heart disease" and "smoking habits" of patients

```
#plotting normal patients
fig = plt.figure(figsize=(10,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.countplot(attr_0['smoking'], palette = 'Spectral')
plt.title('Normal patients: \n "Non-smoking vs Smoking" ', fontsize=10, color="green")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='black', size=18)

#plotting Heart disease patients
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attr_1['smoking'], palette = 'dark')
plt.title('Heart disease patients: \n "Non-smoking vs Smoking"', fontsize=10, color="red")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='white', size=18)
plt.show()
```

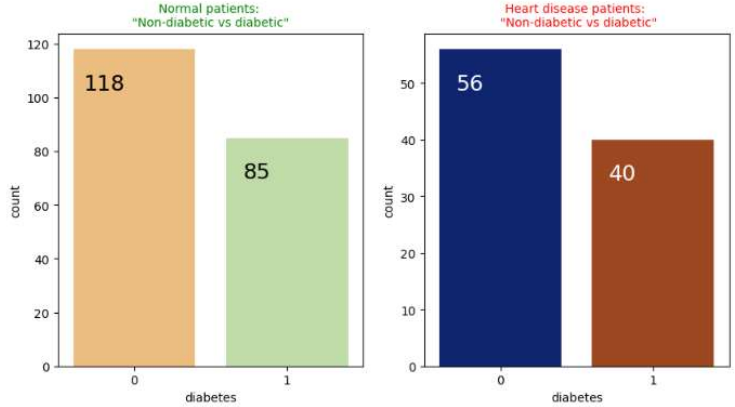


INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.10 – plotting all the input variables related to target (Heart disease)

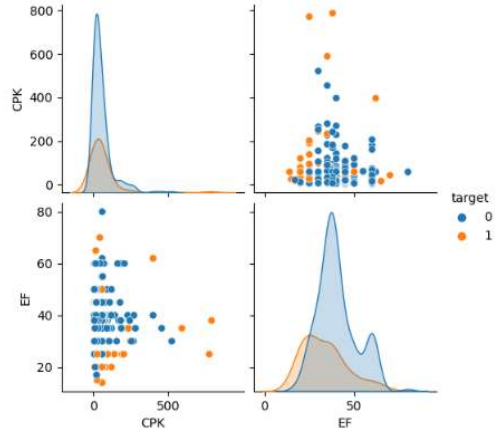
In below plots, it shows relation between "Heart disease" and "diabeties"

```
#plotting normal patients
fig = plt.figure(figsize=(10,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.countplot(attr_0['diabetes'], palette = 'Spectral')
plt.title('Normal patients: \n "Non-diabetic vs diabetic" ', fontsize=10, color="green")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='black', size=18

#plotting Heart disease patients
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attr_1['diabetes'], palette = 'dark')
plt.title('Heart disease patients: \n "Non-diabetic vs diabetic"', fontsize=10, color="red")

for a in ax1.patches:
    ax1.annotate(f'\n{a.get_height()}', (a.get_x()+0.2, a.get_height()), ha='center', va='top', color='white', size=18
plt.show()
```

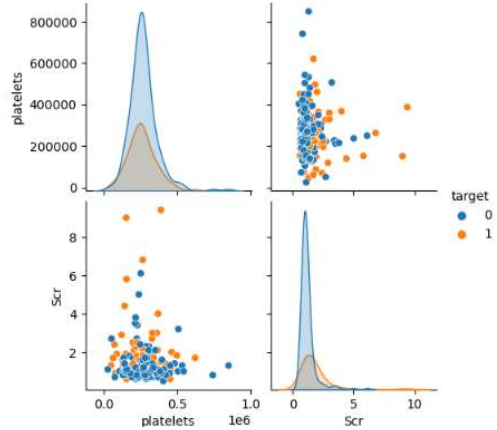


We can use pairplot to see the relation between target and few other variables

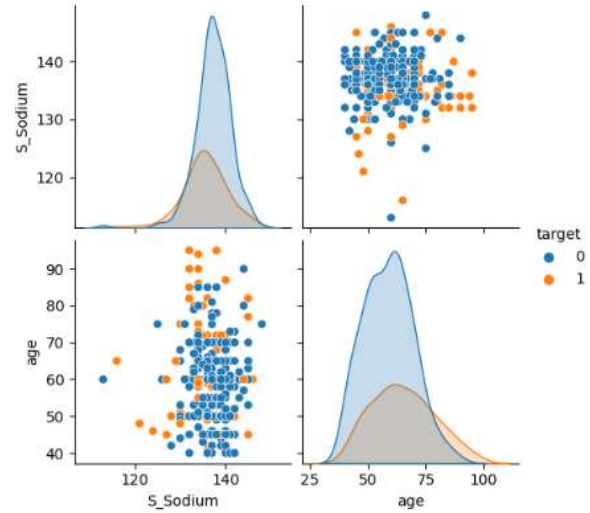
```
#This below pairplots show us the relation of CPK and EF with target
sns.pairplot(cvd, hue='target', vars = ['CPK', 'EF' ])
plt.show()
```




```
#This below pairplots show us the relation of platelets and Scr with target
sns.pairplot(cvd, hue='target', vars = ['platelets', 'Scr'])
plt.show()
```




```
#This below pairplots show us the relation of S_Sodium and age with target
sns.pairplot(cvd, hue='target', vars = ['S_Sodium', 'age'])
plt.show()
```



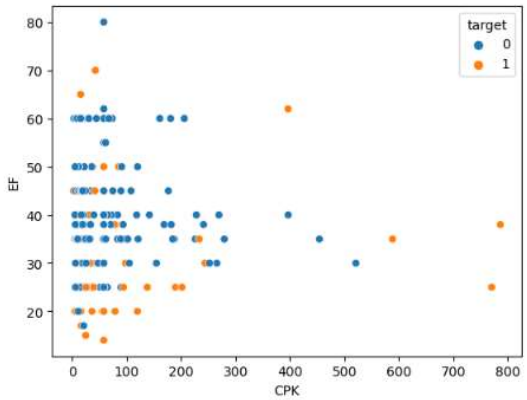
All the above plots and graphs show us some relation between variables and also it mentions

(EDA – Plots & Graphs) 2.11 – plotting all the input variables related to target (Heart disease) – Scatter plot from seaborn library + Outliers Z-Score

Scatter plots are mentioned below and this clearly inform us about outliers present in different variables

```
sns.scatterplot(x = 'CPK', y = 'EF', hue = 'target', data = cvd )
```

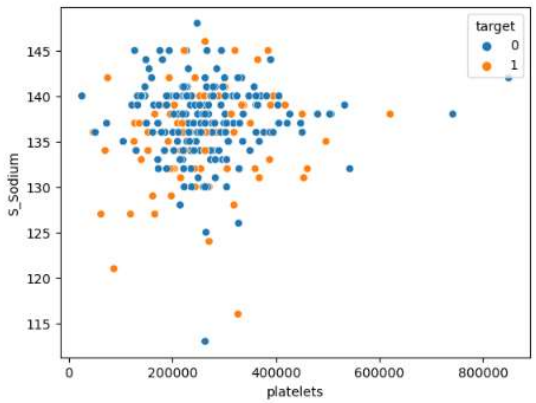
```
<AxesSubplot:xlabel='CPK', ylabel='EF'>
```



Scatter plots are mentioned below and this clearly inform us about outliers present in different variables

```
sns.scatterplot(x = 'platelets', y = 'S_Sodium', hue = 'target', data = cvd )
```

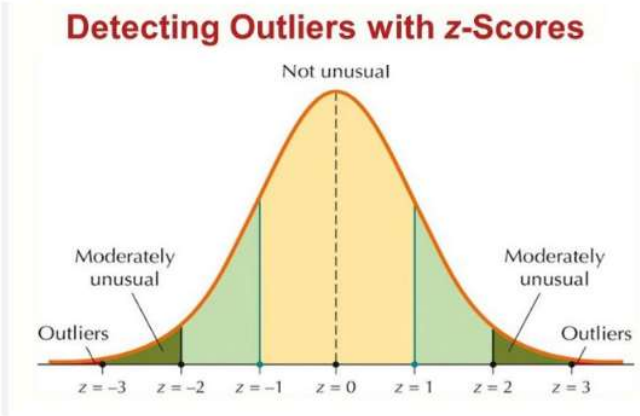
```
<AxesSubplot:xlabel='platelets', ylabel='S_Sodium'>
```



In above mentioned scatterplots, we can see some relation between variables and also outliers are clearly visible

#Outliner Detection and Removal

An Outlier is an extremely large or extremely small datavalue relative to the rest of the datset. It may represent a data entry error, or it may be genuine data.



$$Z\ Score = \frac{(\text{Observation} - \text{Mean})}{\text{Standard Deviation}}$$

$$Z\ Score = \frac{X - \mu}{\sigma}$$

```
#Create a new dataframe for all numerical (Non-boolean) variables from our dataset
cvd_numeric = cvd[['age', 'CPK', 'EF', 'platelets', 'Scr', 'S_Sodium']]
```

```
cvd_numeric.head(5)
```

	age	CPK	EF	platelets	Scr	S_Sodium
0	75	58.2	20	265000	1.9	130
1	55	786.1	38	263358	1.1	136
2	65	14.6	20	162000	1.3	129
3	50	11.1	20	210000	1.9	137
4	65	16.0	20	327000	2.7	116

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.12 –Finding Outliers & using Z-Score

By using Z-Score method, we are going to remove outliers in the data set. Below mentioned code will help us achieve this thing

```
# calculating the z-score of numeric columns in the dataset
# Note : Make sure to import Scipy as we are using this library to find z-score

z = np.abs(stats.zscore(cvd_numeric))
print(z)
```

```
          age       CPK        EF     platelets       Scr  S_Sodium
0    1.193302  0.000166  1.530560  1.681651e-02  0.490057  1.504036
1    0.490896  7.514640  0.007077  2.740240e-07  0.284552  0.141976
2    0.351203  0.449939  1.530560  1.038073e+00  0.090900  1.731046
3    0.911945  0.486071  1.530560  5.464741e-01  0.490057  0.085034
4    0.351203  0.435486  1.530560  6.517986e-01  1.264666  4.682176
..        ...       ...       ...           ...       ...       ...
294  0.098573  0.537688  0.007077  1.109765e+00  0.284552  1.447094
295  0.490896  1.278215  0.007077  6.802474e-02  0.187726  0.539054
296  1.332995  1.525979  1.854958  4.902082e+00  0.575031  0.312044
297  1.332995  1.890398  0.007077  1.263389e+00  0.005926  0.766064
298  0.911945  0.398321  0.585389  1.348231e+00  0.199578  0.141976

[299 rows x 6 columns]
```

```
#Defining threshold for filtering outliers
threshold = 3
print(np.where(z > 3))
```

```
(array([  1,   4,   9,  19,  28,  52,  52,  60,  64,  72, 103, 105, 109,
       131, 134, 171, 199, 217, 228, 296], dtype=int64), array([1, 5, 4, 5, 4, 1, 4, 1, 2, 1, 1, 3, 3, 4, 1, 1, 5, 4, 4, 3],
      dtype=int64))
```

///

In the above result, please note these things - First array give us information us the Row number - Second array give us information about column number - Two example to give clarity on this. - when we observe the first one of each array, it is mentioned as 1 & 1. **When we look into data set as mentioned below, 01 row & 01 column indicates CPK = 786.1 (which is unusual) - Normal value is 10 to 120 (mcg/L)**

- when we observe the 4th one of each array, it is mentioned as 19 & 5. **When we look into data set as mentioned below, 19 row & 5 column indicates S.Sodium = 121 (which is unusual) - Normal value is 135 to 145 (mcg/L)**

- when we observe the 6th one of each array, it is mentioned as 52 & 1. **When we look into data set as mentioned below, 52 row & 1 column indicates CPK = 396.4 (which is unusual) - Normal value is 10 to 120 (mcg/L)**

- In total, we have 20 Z score parameters or outliers in our given data set.

///

```
cvd_numeric.head(53)
```

	age	CPK	EF	platelets	Scr	S_Sodium
0	75	58.2	20	265000	1.90	130
1	55	786.1	38	263358	1.10	136
2	65	14.6	20	162000	1.30	129
3	50	11.1	20	210000	1.90	137
4	65	16.0	20	327000	2.70	116
5	90	4.7	40	204000	2.10	132
6	75	24.6	15	127000	1.20	137
7	60	31.5	60	454000	1.10	131
8	65	15.7	65	263358	1.50	138
9	80	12.3	35	388000	9.40	133
10	75	8.1	38	368000	4.00	131
11	62	23.1	25	253000	0.90	140
12	45	98.1	30	136000	1.10	137
13	50	16.8	38	276000	1.10	137
14	49	8.0	30	427000	1.00	138
15	82	37.9	50	47000	1.30	136
16	87	14.9	38	262000	0.90	140
17	45	58.2	14	166000	0.80	127
18	70	12.5	25	237000	1.00	140
19	48	58.2	55	87000	1.90	121
20	65	5.2	25	276000	1.30	137
21	65	12.8	30	297000	1.60	136
22	68	22.0	35	289000	0.90	140
23	53	6.3	60	368000	0.80	135
24	75	58.2	30	263358	1.83	134
25	80	14.8	38	149000	1.90	144
26	95	11.2	40	196000	1.00	138
27	70	12.2	45	284000	1.30	136
28	58	6.0	38	153000	5.80	134
29	82	7.0	30	200000	1.20	132
30	94	58.2	38	263358	1.83	134
31	85	2.3	45	360000	3.00	132
32	50	24.9	35	319000	1.00	128
33	50	15.9	30	302000	1.20	138
34	65	9.4	50	188000	1.00	140
35	69	58.2	35	228000	3.50	134
36	90	6.0	50	226000	1.00	134
37	82	85.5	50	321000	1.00	145
38	60	265.6	30	305000	2.30	137
39	60	23.5	38	329000	3.00	142
40	70	58.2	20	263358	1.83	134
41	50	12.4	30	153000	1.20	136
42	70	57.1	45	185000	1.20	139

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.13 –Finding Outliers & using Z-Score

```
#Filtering outliers & retain only data points which are below threshold
#Before removing outliers, this is the shape of data we have (299 rows & 6 columns)
cvd_numeric.shape
```

```
(299, 6)
```

new dataframe "cvd1" shall be created to store the new dataframe after removing outliers from the original data set

```
cvd1 = cvd[(z < 3).all(axis=1)]
```

```
cvd1.shape
```

```
(280, 12)
```

new dataframe "cvd1" contains only 280 rows after removing the outliers

```
#verify the cvd_numeric dataframe
cvd_numeric.head(3)
```

	age	CPK	EF	platelets	Scr	S_Sodium
0	75	58.2	20	265000	1.9	130
1	55	786.1	38	263358	1.1	136
2	65	14.6	20	162000	1.3	129

```
#verify the cvd_1 dataframe, in which outliers are removed. If you see below, after sl.no 3, 4 is not
#removed due to outlier present in row 4
cvd1.head(5)
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
0	75	0	58.2	0	20	1	265000	1.9	130	male	0	1
2	65	0	14.6	0	20	0	162000	1.3	129	male	1	1
3	50	1	11.1	0	20	0	210000	1.9	137	male	0	1
5	90	1	4.7	0	40	1	204000	2.1	132	male	1	1
6	75	1	24.6	0	15	0	127000	1.2	137	male	0	1

```
cvd1.shape
```

```
(280, 12)
```

Conclusion: - Now, we have removed 20 outliers which were in our record
- Earlier our data set had (299 rows, 6 columns)
- After removing outliers we have (280 rows, 6 columns)

```
cvd1.head(55)
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
0	75	0	58.2	0	20	1	265000	1.90	130	male	0	1
2	65	0	14.6	0	20	0	162000	1.30	129	male	1	1
3	50	1	11.1	0	20	0	210000	1.90	137	male	0	1
5	90	1	4.7	0	40	1	204000	2.10	132	male	1	1
6	75	1	24.6	0	15	0	127000	1.20	137	male	0	1
7	60	1	31.5	1	60	0	454000	1.10	131	male	1	1
8	65	0	15.7	0	65	0	263358	1.50	138	female	0	1
10	75	1	8.1	0	38	1	368000	4.00	131	male	1	1
11	62	0	23.1	0	25	1	253000	0.90	140	male	1	1
12	45	1	98.1	0	30	0	136000	1.10	137	male	0	1
13	50	1	16.8	0	38	1	276000	1.10	137	male	0	1
14	49	1	8.0	0	30	1	427000	1.00	138	female	0	0
15	82	1	37.9	0	50	0	47000	1.30	136	male	0	1
16	87	1	14.9	0	38	0	262000	0.90	140	male	0	1
17	45	0	58.2	0	14	0	166000	0.80	127	male	0	1
18	70	1	12.5	0	25	1	237000	1.00	140	female	0	1
20	65	1	5.2	0	25	1	276000	1.30	137	female	0	0
21	65	1	12.8	1	30	1	297000	1.60	136	female	0	1
22	68	1	22.0	0	35	1	289000	0.90	140	male	1	1
23	53	0	6.3	1	60	0	368000	0.80	135	male	0	0
24	75	0	58.2	1	30	1	263358	1.83	134	female	0	1
25	80	0	14.8	1	38	0	149000	1.90	144	male	1	1
26	95	1	11.2	0	40	1	196000	1.00	138	female	0	1
27	70	0	12.2	1	45	1	284000	1.30	136	male	1	1
29	82	0	7.0	1	30	0	200000	1.20	132	male	1	1
30	94	0	58.2	1	38	1	263358	1.83	134	male	0	1
31	85	0	2.3	0	45	0	360000	3.00	132	male	0	1
32	50	1	24.9	1	35	1	319000	1.00	128	female	0	1
33	50	1	15.9	1	30	0	302000	1.20	138	female	0	0
34	65	0	9.4	1	50	1	188000	1.00	140	male	0	1
35	69	0	58.2	1	35	0	228000	3.50	134	male	0	1
36	90	1	6.0	1	50	0	226000	1.00	134	male	0	1
37	82	1	85.5	1	50	1	321000	1.00	145	female	0	1
38	60	0	265.6	1	30	0	305000	2.30	137	male	0	0
39	60	0	23.5	1	38	0	329000	3.00	142	female	0	1
40	70	0	58.2	0	20	1	263358	1.83	134	male	1	1
41	50	0	12.4	1	30	1	153000	1.20	136	female	1	1
42	70	0	57.1	1	45	1	185000	1.20	139	male	1	1
43	72	0	12.7	1	50	1	218000	1.00	134	male	0	0

The blue marked shows removed rows. Example, row 4 & row 9 are removed as highlighted

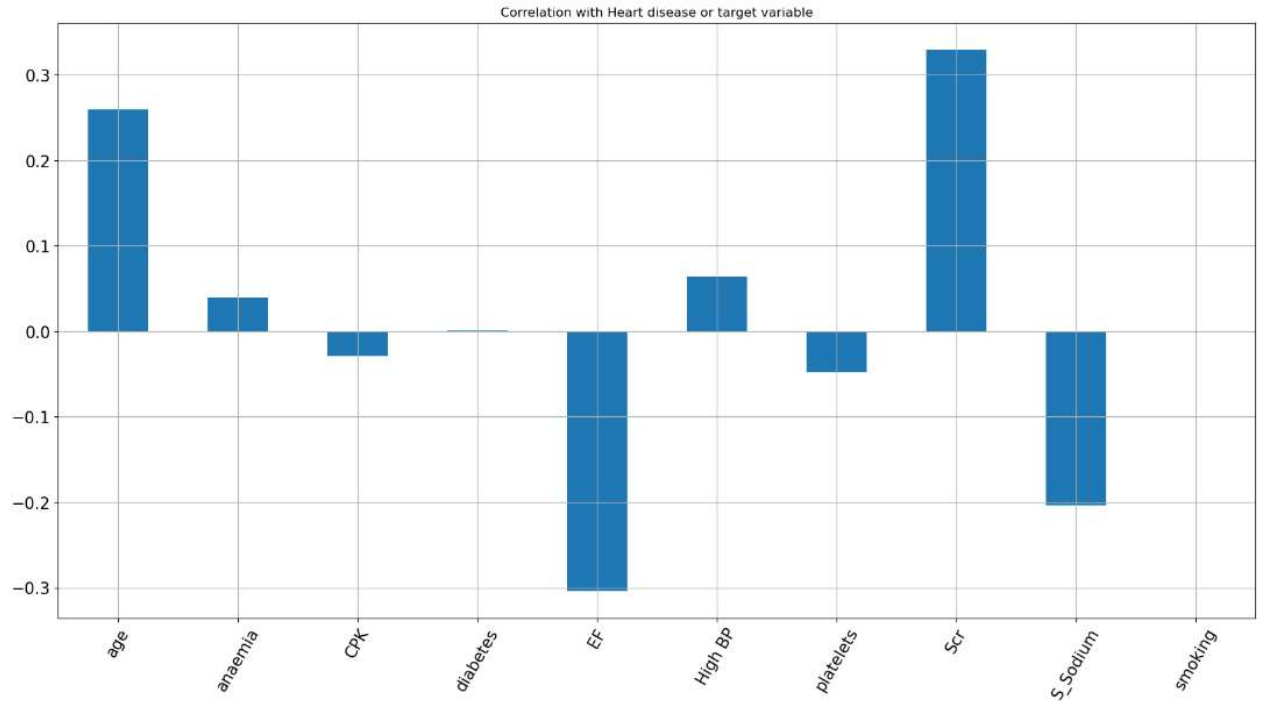
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.14 – Checking co relation & heatmap after removing outliers

```
#Segregating datsset into features . Means X and target variables i.e., y
X = cvd1.drop(['target'], axis=1)
y = cvd1['target']
```

We can again check the co-relation of all the variables against target value by using below plot. Note that, this new plot is after removing the outliers

```
X.corrwith(y).plot.bar(
            figsize = (20,10), title = "Correlation with Heart disease or target variable", fontsize = 15, rot = 60, grid = True)
plt.show()
```

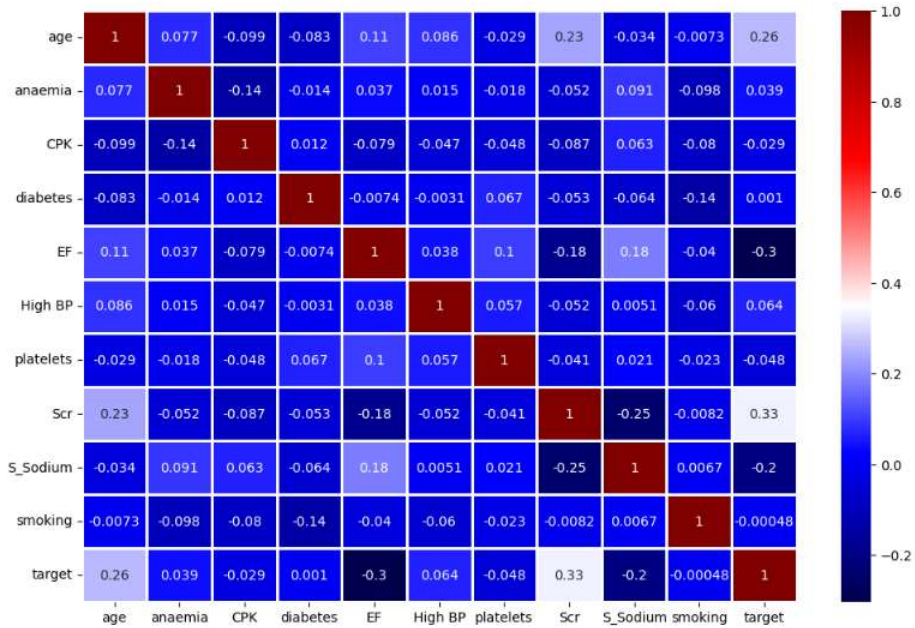


By looking into above plot, it is clear that EF & S_Sodium are negetively co-related with target variable. Whereas, AGE & SCR is positively corelated with target variable. Here target variable is Heart disease

(EDA – Plots & Graphs) 2.15 – Checking co relation & heatmap after removing outliers

```
##Again co-relation can be seen using heatmap to cross verify
plt.figure(figsize = (12,8))
sns.heatmap(cvd1.corr(), linewidth = 1, annot = True, cmap = "seismic")
```

```
<AxesSubplot:>
```




```
cvd1.corr()
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	smoking	target
age	1.000000	0.076898	-0.099022	-0.083030	0.106512	0.085913	-0.028550	0.231842	-0.033671	-0.007287	0.259125
anaemia	0.076898	1.000000	-0.144906	-0.014291	0.037141	0.014892	-0.017951	-0.051654	0.090741	-0.098411	0.039480
CPK	-0.099022	-0.144906	1.000000	0.011658	-0.078583	-0.046820	-0.047819	-0.087296	0.063430	-0.079764	-0.029289
diabetes	-0.083030	-0.014291	0.011658	1.000000	-0.007449	-0.003070	0.067170	-0.053498	-0.063822	-0.136820	0.001009
EF	0.106512	0.037141	-0.078583	-0.007449	1.000000	0.038059	0.101141	-0.177942	0.182741	-0.040007	-0.303506
High BP	0.085913	0.014892	-0.046820	-0.003070	0.038059	1.000000	0.057272	-0.051845	0.005079	-0.059963	0.064143
platelets	-0.028550	-0.017951	-0.047819	0.067170	0.101141	0.057272	1.000000	-0.041153	0.021204	-0.023075	-0.048259
Scr	0.231842	-0.051654	-0.087296	-0.053498	-0.177942	-0.051845	-0.041153	1.000000	-0.245860	-0.008193	0.329460
S_Sodium	-0.033671	0.090741	0.063430	-0.063822	0.182741	0.005079	0.021204	-0.245860	1.000000	0.006677	-0.203471
smoking	-0.007287	-0.098411	-0.079764	-0.136820	-0.040007	-0.059963	-0.023075	-0.008193	0.006677	1.000000	-0.000476
target	0.259125	0.039480	-0.029289	0.001009	-0.303506	0.064143	-0.048259	0.329460	-0.203471	-0.000476	1.000000

Conclusion from Heatmap (co-relation) statistics: (As mentioned earlier)

In our dataset, "**Age**" & **"SCR"** are positively co-related with target variable(as they are showing very light colour)

"EF" & "S_Sodium" are negatively co-related with target variable (as they are showing very dark colour)

The above plot is also showing the same thing

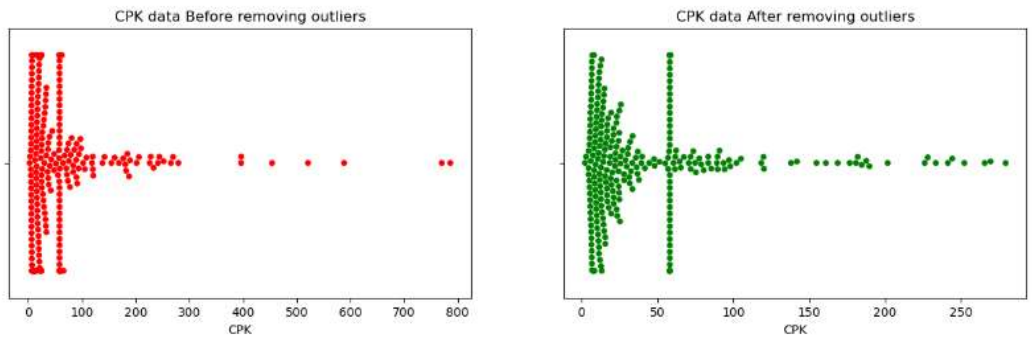
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

(EDA – Plots & Graphs) 2.16 – Check tendency of data after removing outliers + compare few input variables before vs after removing outliers

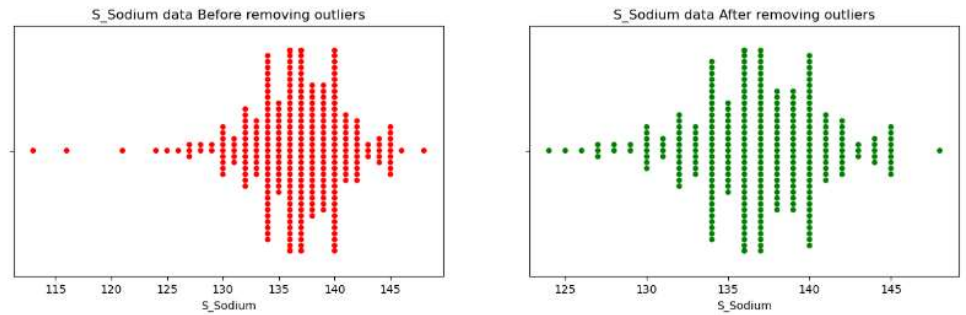
```
cvd1.describe()
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	smoking	target
count	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000
mean	60.967857	0.435714	45.672500	0.417857	37.835714	0.360714	258820.064286	1.275393	136.825000	0.314286	0.307143
std	11.815595	0.496738	53.710757	0.494089	11.470461	0.481068	85605.349796	0.615743	3.924203	0.465062	0.462135
min	40.000000	0.000000	2.300000	0.000000	14.000000	0.000000	25100.000000	0.500000	124.000000	0.000000	0.000000
25%	51.000000	0.000000	11.450000	0.000000	30.000000	0.000000	210750.000000	0.900000	134.000000	0.000000	0.000000
50%	60.000000	0.000000	23.950000	0.000000	38.000000	0.000000	261000.000000	1.100000	137.000000	0.000000	0.000000
75%	70.000000	1.000000	58.200000	1.000000	45.000000	1.000000	302000.000000	1.400000	140.000000	1.000000	1.000000
max	95.000000	1.000000	279.400000	1.000000	65.000000	1.000000	543000.000000	4.400000	148.000000	1.000000	1.000000

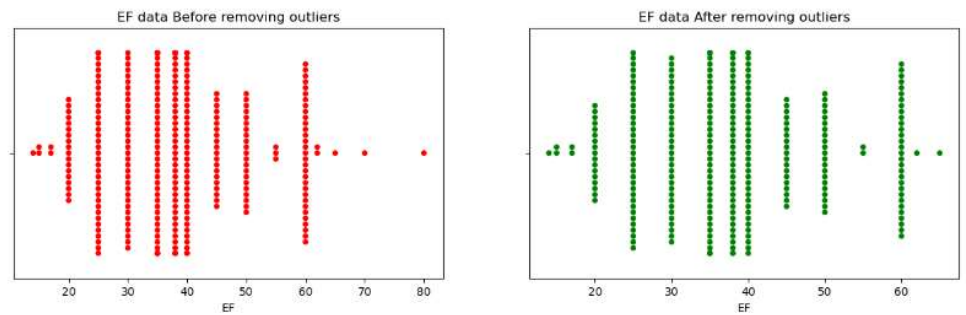
```
fig, axs = plt.subplots(1, 2, figsize=(15, 4))
sns.swarmplot(data=cvd, x='CPK',color="red", ax=axs[0]).set(title='CPK data Before removing outliers')
sns.swarmplot(data=cvd1, x='CPK',color="green", ax=axs[1]).set(title='CPK data After removing outliers')
plt.show()
```




```
fig, axs = plt.subplots(1, 2, figsize=(15, 4))
sns.swarmplot(data=cvd, x='S_Sodium',color="red", ax=axs[0]).set(title='S_Sodium data Before removing outliers')
sns.swarmplot(data=cvd1, x='S_Sodium',color="green", ax=axs[1]).set(title='S_Sodium data After removing outliers')
plt.show()
```




```
fig, axs = plt.subplots(1, 2, figsize=(15, 4))
sns.swarmplot(data=cvd, x='EF',color="red", ax=axs[0]).set(title='EF data Before removing outliers')
sns.swarmplot(data=cvd1, x='EF',color="green", ax=axs[1]).set(title='EF data After removing outliers')
plt.show()
```



Conclusion: - After removing outliers, it is clearly visible that the data quality is good

- CPK: Earlier CPK 0 - 800. After, outlier removal, it is having 0 to 260
- S_Sodium: Earlier S_Sodium graph values are 110 - 150. After, outlier removal, it is 125 - 145
- EF: Earlier EF graph was having from 20 to 80. After, outlier removal, it is having 20 to 60

With all above steps, we completed Exploratory Data Analysis process.

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

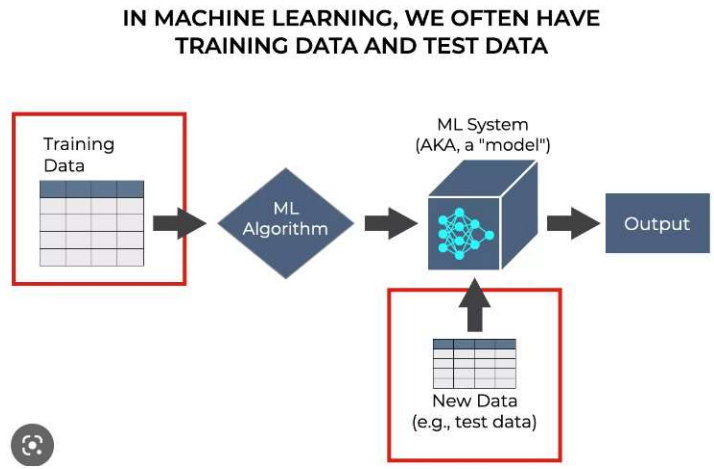
3. Split Data

3.1 - Training data & 3.2 - Test data

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

<u>The procedure involves:</u>

- *Divide the data set into two subsets.*
- *First subset is used to fit the model and is referred to as the training dataset.*
- *Second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values.*
- *The second dataset is referred to as the test dataset.*
- *The objective is to estimate the performance of the machine learning model on test data.*
- *In this model, we are going to split data on:*
  Train data : 80 %, Test data: 20 %
- *We are going to use **SKLearn** library train_test_split to perform this task. (Example code is shown in this slide)*



SKlearn Train split example

```
]: from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
   print ('Train set:', X_train.shape,  y_train.shape)
   print ('Test set:', X_test.shape,  y_test.shape)

   Train set: (800, 11) (800, 1)
   Test set: (200, 11) (200, 1)

]: type(y_test)

]: pandas.core.frame.DataFrame
```

3. Train Test Split the data

```
#for test and train split we need to import a library from sklearn

from sklearn.model_selection import train_test_split
```

Splitting data into Features and target

```
cvd1.head(2)
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking	target
0	75	0	58.2	0	20	1	265000	1.9	130	1	0	1
2	65	0	14.6	0	20	0	162000	1.3	129	1	1	1

In below step, we are going to segregate the dataset into input parameters & target dataset
This means, X contains only input variable parameters, where as Y should contain only target variable

```
X = cvd1.drop(columns = 'target', axis = 1)
Y = cvd1['target']
```

```
X.head(3) # This is dataset without target variable in this
```

	age	anaemia	CPK	diabetes	EF	High BP	platelets	Scr	S_Sodium	sex	smoking
0	75	0	58.2	0	20	1	265000	1.9	130	1	0
2	65	0	14.6	0	20	0	162000	1.3	129	1	1
3	50	1	11.1	0	20	0	210000	1.9	137	1	0

```
Y.head(2) # This is dataset only with target variable in this
```

```
0    1
2    1
Name: target, dtype: int64
```

Splitting data into Training data & testing data

- below is the function to split data into Test & train
- test_size = 0.2 indicates, we are dividing the data as 80% (training) and 20% (testing)
- stratify = Y --- means, it will distribute equal data (1, 0), in the data sets

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify=Y, random_state = 2)
```

```
print (" ************ Training data SET *************** ")
print (X.shape, X_train.shape, X_test.shape)
print (Y.shape, Y_train.shape, Y_test.shape)
```

```
 ************ Training data SET ***************
(280, 11) (224, 11) (56, 11)
(280,) (224,) (56,)
```

In the above division - we can visualize below mentioned things.
- Training dataset has 244 rows & 11 columns
- Testing dataset has 56 rows & 11 columns

```
#Check the distribution of target variable in training data and test data after train-test-split

print ("Distribution of target variable in training set")
print (Y_train.value_counts())

print ("Distribution of target variable in test set")
print (Y_test.value_counts())
```

```
Distribution of target variable in training set
0    155
1     69
Name: target, dtype: int64
Distribution of target variable in test set
0    39
1    17
Name: target, dtype: int64
```

In the above results - we successfully divided the Test set & training set of data
- In training data set, we have 155 patients without heart disease & 69 patients who have heart disease
- In test data set, we have 39 patients without heart disease & 17 patients who have heart disease

4.1 & 4.2 Supervised learning model and unsupervised learning model

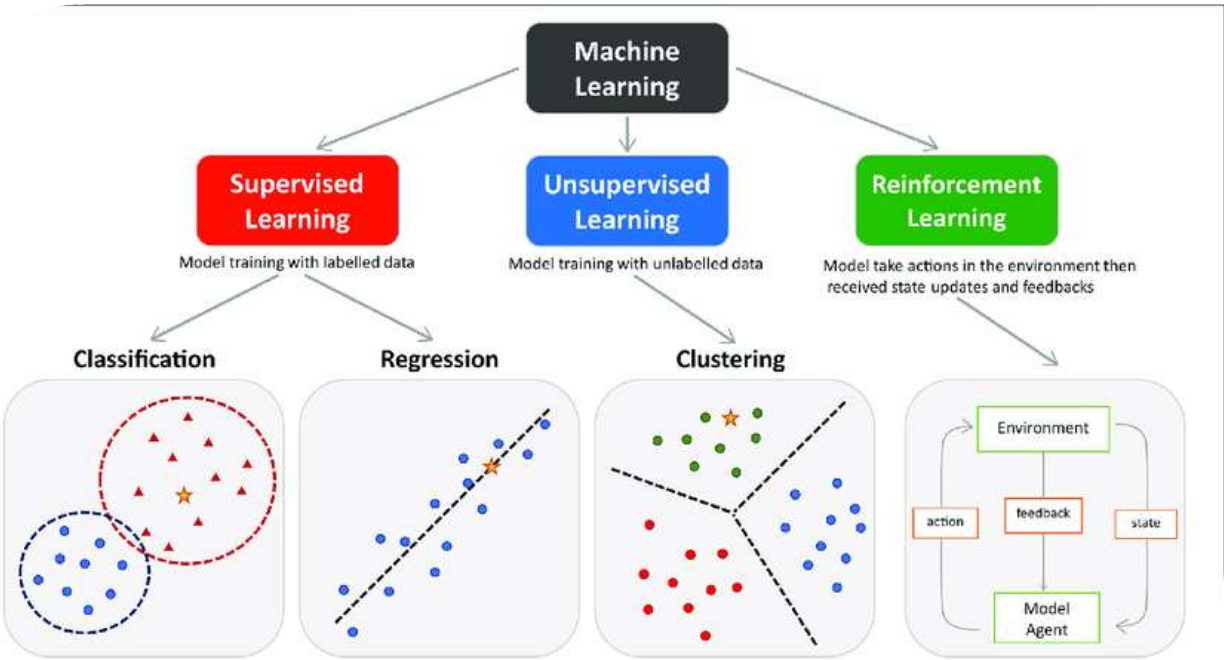
Machine learning tasks in are classified into 3 types

1) Supervised learning
2) Unsupervised learning
3) Reinforcement learning

- *In our data set, we can classify this as Supervised learning because, we are going to use Historical / labelled data to produce the output.*
- *Unsupervised learning refers to non-availability of "Historical or labelled data to produce the output" - Not applicable to this project*
- *Reinforcement learning refers to "output depends on the current input and next input depends on the previous output". In other words, machine learns from rewards or feedback – Not applicable to this project*

Supervised learning is divided into 2 methods. Next slides explain further about this.

- ***Classification***
- ***Regression***



5. Regression

Regression is another division of supervised learning machine learning model. Few details about Regression model are given below for information only:

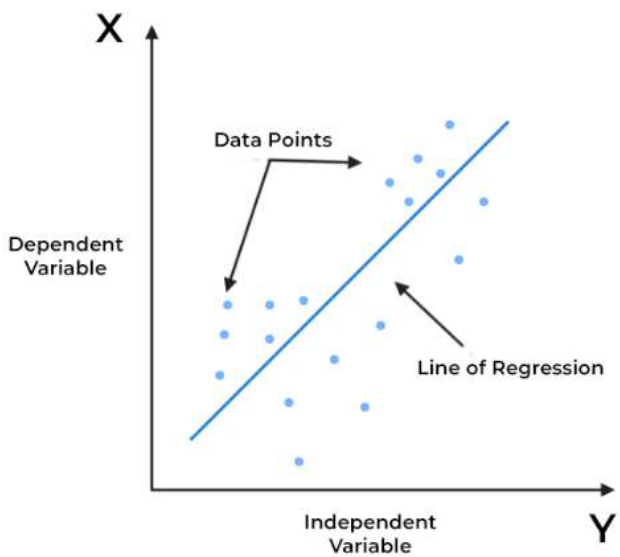
** Regression finds co-relations between dependent and independent variables. Therefore, regression algorithms help predict continuous variables such as House prices, market trends, weather patterns, oil and gas prices etc.*

** The Regression algorithm's task is finding the mapping function so we can map the input variable of "x" to the continuous output variable of "y"*

** One of the important feature of regression model is the output is continuous.*

** To conclude, in this project or ML model as our output is discrete (Heart disease present) we are not going to use Regression model. This information is given here only as a reference.*

Example chart of Regression model



6. Classification

There are 4 different types of classification tasks in machine learning.

- Binary Classification
- Multi-Class Classification
- Multi-Label Classification
- Imbalanced Classificcation
- *With reference to this ML model and with the given data set, as mentioned in earlier slides the outcome of the model is either of one condition*

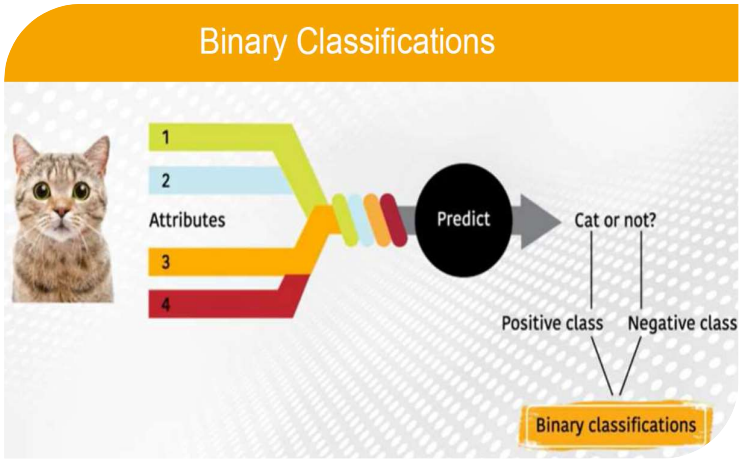
Heart disease present *or* ***No heart disease***

- *Hence this model classification is treated as Binary classification.*

Meaning of Binary classification:

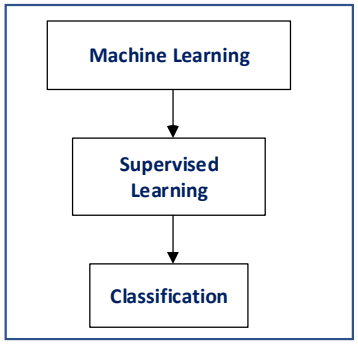
"In machine learning, binary classification is a supervised learning algorithm that categorizes new observations into one of two classes. Where in, in this case the output is discrete"

Example of Binary classification

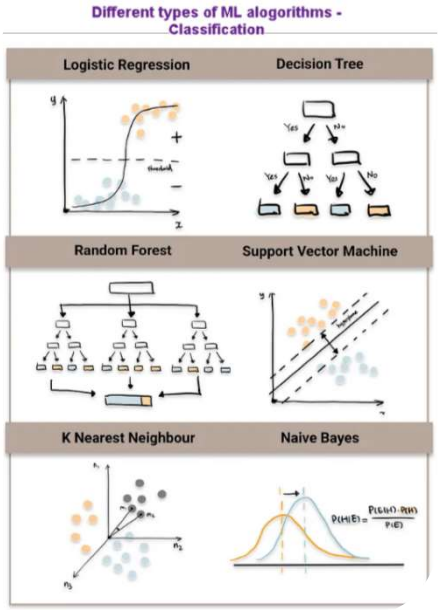


7. Trained Model

As Explained in Earlier section, with the given data set, we can conclude on following things and based on the classification, we can select the most suitable ML model we are going to use in this project with given data set.



- *With in the Classification, there are many machine learning techniques which we can use to predict the output "Heart disease".*
- *Figure enclosed in this slide show some of the ML models which can be used in our project to predict. Among these, any 2 models shall be used to predict the output.*
- *In case we use Decision Tree classifier, example python code is shown in the figure in this slide*



. Decision Tree Classifier Hyperparameter tuning

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

params = {
    "criterion":("gini", "entropy"),
    "splitter":("best", "random"),
    "max_depth":(list(range(1, 20))),
    "min_samples_split":[2, 3, 4],
    "min_samples_leaf":list(range(1, 20)),
}


tree_clf = DecisionTreeClassifier(random_state=42)
tree_cv = GridSearchCV(tree_clf, params, scoring="accuracy", n_jobs=-1, verbose=1, cv=3)
tree_cv.fit(X_train, y_train)
best_params = tree_cv.best_params_
print(f"Best paramters: {best_params})")

tree_clf = DecisionTreeClassifier(**best_params)
tree_clf.fit(X_train, y_train)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

7.1 – Train the model with suitable machine learning algorithm. Here we are training our data with 7 various ML algorithm models.

1- Logistic Regression

```
# import the Logistic regression model
from sklearn.linear_model import LogisticRegression

# import accuracy score & other things
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
#defining Logistic regression
lr = LogisticRegression()

#In this step model is trained with training data on Logistic regression ML algorithm
lr.fit(X_train, Y_train)
```

```
LogisticRegression()
```

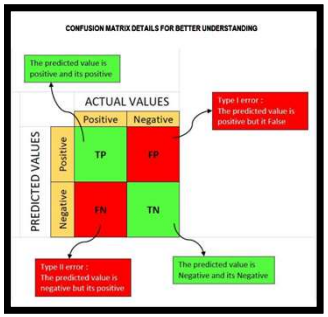
```
y_pred1 = lr.predict(X_test)
accuracy_score(Y_test, y_pred1)
```

```
0.8035714285714286
```

Logistic Regression - model evaluation - accuracy, classification, confusion matrix

```
#In our case, Y_test is the testing dataset & y_pred1 contains X_test dataset with model predictions.
#Hence, we need to check accuracy scores related to Y_test dataset vs X_test dataset with model predict

print(f"Accuracy Score: {accuracy_score(Y_test, y_pred1) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred1)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred1)}\n")
```

```
Accuracy Score: 80.36%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.80      0.95      0.87        39
           1       0.80      0.47      0.59        17

    accuracy                           0.80        56
   macro avg       0.80      0.71      0.73        56
weighted avg       0.80      0.80      0.79        56

_____________________________________________
_____________________________________________
Confusion Matrix:
 [[37  2]
 [ 9  8]]
```



7 Machine learning classifiers used in this project (ML algorithms) are listed below:

1. Logistic Regression
2. Decision tree
3. Random Forest
4. SVM (Support Vector Model)
5. KNN (K-nearest neighbors)
6. Naive Bayes
7. Gradient Boosting

Logistic Regression classifier– Facts

- ✓ Logistic Regression is a supervised machine learning algorithm used for binary classification tasks.
- ✓ It models the relationship between the features and the probability of the data point belonging to a particular class.
- ✓ It uses the logistic function (also known as the sigmoid function) to transform the linear regression output into a probability value between 0 and 1.
- ✓ The model is trained using a maximum likelihood estimation to find the optimal coefficients that maximize the likelihood of the observed data.
- ✓ Logistic Regression is interpretable, computationally efficient, and can handle both numerical and categorical features, but assumes a linear relationship between features and the log-odds of the target variable.

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

7.2 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

2- Decision tree

```
# import the Logistic regression model
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()


dt.fit(X_train, Y_train) #In this step model is trained with training data on Decisiontree M
```

```
DecisionTreeClassifier()
```

```
y_pred2 = dt.predict(X_test)
accuracy_score(Y_test, y_pred2)
```

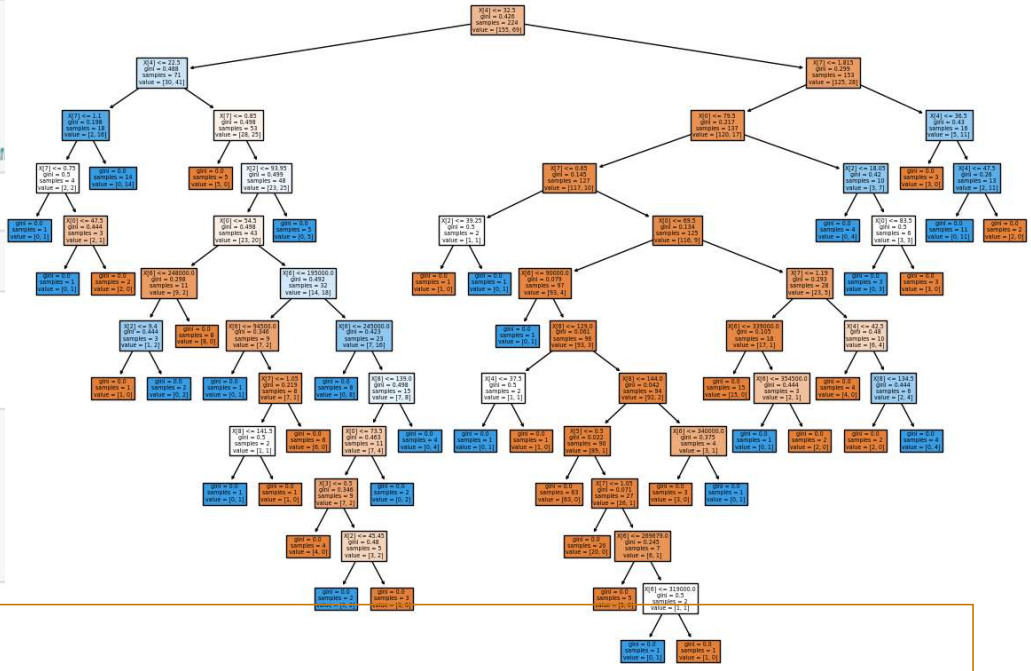
```
0.7142857142857143
```

Decision tree - model evaluation - accuracy, classification, confusion matrix

```
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred2) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred2)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred2)}\n")
```

```
Accuracy Score: 71.43%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.81      0.77      0.79        39
           1       0.53      0.59      0.56        17

    accuracy                           0.71        56
   macro avg       0.67      0.68      0.67        56
weighted avg       0.72      0.71      0.72        56

_____________________________________________
_____________________________________________
Confusion Matrix:
 [[30  9]
 [ 7 10]]
```

Decision Tree – pictorial representation



Decision tree Classifier – Facts

- ✓ Decision Tree is a supervised machine learning algorithm used for classification and regression tasks
- ✓ It builds a tree-like model of decisions and their possible consequences based on feature values
- ✓ The tree structure consists of internal nodes representing features, branches representing decisions, and leaf nodes representing the predicted class or value.
- ✓ Decision Tree splits data based on feature thresholds to create homogeneous subsets at each node
- ✓ It is interpretable, can handle both categorical and numerical data, and is prone to overfitting on complex datasets.

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

7.3 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

3- Random Forest classifier

```
# import the Logistic regression model
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()


rf.fit(X_train, Y_train)  #In this step model is trained with training data on Random Forest ML algorithm
```

```
RandomForestClassifier()
```

```
y_pred3 = rf.predict(X_test)
accuracy_score(Y_test, y_pred3)
```

```
0.7857142857142857
```

Random Forest - model evaluation - accuracy, classification, confusion matrix

```
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred3) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred3)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred3)}\n")
```

```
Accuracy Score: 78.57%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.80      0.92      0.86        39
           1       0.73      0.47      0.57        17

    accuracy                           0.79        56
   macro avg       0.76      0.70      0.71        56
weighted avg       0.78      0.79      0.77        56

_____________________________________________
_____________________________________________
Confusion Matrix:
 [[36  3]
 [ 9  8]]
```

Random forest – Facts

- ✓ Ensemble machine learning model
- ✓ Constructed using multiple decision trees
- ✓ Each tree is trained on a random subset of data
- ✓ Combines predictions of individual trees for final classification
- ✓ Handles high-dimensional data, avoids overfitting, and provides feature importance rankings

7.4 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

4- SVM (Support Vector Model)

```
from sklearn import svm

svm = svm.SVC()

svm.fit(X_train, Y_train) #In this step model is trained with training data on SVM ML algorithm
```

```
SVC()
```

```
y_pred4 = svm.predict(X_test)
accuracy_score(Y_test, y_pred4)
```

```
0.6964285714285714
```

SVM- model evaluation - accuracy, classification, confusion matrix

```
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred4) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred4)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred4)}\n")
```

```
Accuracy Score: 69.64%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82        39
           1       0.00      0.00      0.00        17

    accuracy                           0.70        56
   macro avg       0.35      0.50      0.41        56
weighted avg       0.49      0.70      0.57        56

_____________________________________________
_____________________________________________
Confusion Matrix:
 [[39  0]
 [17  0]]
```

SVM classifier – Facts

- ✓ Support Vector Machine (SVM) is a supervised machine learning algorithm for classification and regression
- ✓ It creates a hyperplane that separates data points into different classes based on their features
- ✓ SVM aims to maximize the margin, i.e., the distance between the hyperplane and the nearest data points of different classes
- ✓ It can handle linear and non-linear classification problems by using different kernel functions
- ✓ SVM is effective for high-dimensional data and can handle both binary and multi-class classification tasks.

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

7.5 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

5- KNN (K Nearest Neighbour)

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X_train, Y_train) #In this step model is trained with training data on KNN Neighbor
```

```
KNeighborsClassifier()
```

```
y_pred5 = knn.predict(X_test)
accuracy_score(Y_test, y_pred5)
```

```
0.5535714285714286
```

KNN model evaluation - accuracy, classification, confusion matrix

```
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred5) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred5)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred5)}\n")
```

```
Accuracy Score: 55.36%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.68      0.69      0.68        39
           1       0.25      0.24      0.24        17

    accuracy                           0.55        56
   macro avg       0.46      0.46      0.46        56
weighted avg       0.55      0.55      0.55        56

_____________________________________________
_____________________________________________
Confusion Matrix:
 [[27 12]
 [13  4]]
```

KNN classifier – Facts

- ✓ K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification
- ✓ It classifies a new data point by considering the majority class of its K nearest neighbors in the training dataset.
- ✓ The value of K, the number of neighbors to consider, is a hyperparameter that needs to be specified.
- ✓ KNN uses a distance metric (e.g., Euclidean distance) to determine the similarity between data points.
- ✓ It is a non-parametric algorithm and can handle both numerical and categorical data but can be computationally expensive for large datasets.

7.6 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

6- Naive Bayes

```
from sklearn.naive_bayes import GaussianNB


nb = GaussianNB()

nb.fit(X_train, Y_train) #In this step model is trained with training data on Gaissoam NB algorithm
```

```
GaussianNB()
```

```
y_pred6 = nb.predict(X_test)
accuracy_score(Y_test, y_pred6)
```

```
0.75
```

Naive Bayes evaluation - accuracy, classification, confusion matrix

```
print(("____________________________________________"))
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred6) * 100:.2f}%")
print(("____________________________________________"))
print("____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred6)}")
print("____________________________________________")
print("____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred6)}\n")
```

```
____________________________________________
Accuracy Score: 75.00%
____________________________________________
____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.75      0.97      0.84        39
           1       0.80      0.24      0.36        17

    accuracy                           0.75        56
   macro avg       0.77      0.60      0.60        56
weighted avg       0.76      0.75      0.70        56


____________________________________________
____________________________________________
Confusion Matrix:
 [[38  1]
 [13  4]]
```

Naive Bayes classifier – Facts

- ✓ Naive Bayes is a supervised machine learning algorithm based on Bayes' theorem and assumes independence between features.
- ✓ It is commonly used for classification tasks and calculates the probability of a data point belonging to each class.
- ✓ Naive Bayes computes the likelihood of each feature given the class and multiplies them to obtain the posterior probability.
- ✓ It assumes that features are conditionally independent, simplifying the computation.
- ✓ Naive Bayes is efficient, especially with high-dimensional data, but can be limited by the independence assumption and may not capture complex relationships.

7.7 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

7 - Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

gbc.fit(X_train, Y_train)  #In this step model is trained with training data on gradient boosting ML alogirithm
```

```
GradientBoostingClassifier()
```

```
y_pred7 = gbc.predict(X_test)
accuracy_score(Y_test, y_pred7)
```

```
0.7321428571428571
```

Gradient Boosting classifier : evaluation- accuracy, classification, confusion matrix

```
print(("_____________________________________________"))
print(f"Accuracy Score: {accuracy_score(Y_test, y_pred7) * 100:.2f}%")
print(("_____________________________________________"))
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{classification_report(Y_test, y_pred7)}")
print("_____________________________________________")
print("_____________________________________________")
print(f"Confusion Matrix: \n {confusion_matrix(Y_test, y_pred7)}\n")
```

```
_____________________________________________
Accuracy Score: 73.21%
_____________________________________________
_____________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.79      0.85      0.81        39
           1       0.57      0.47      0.52        17

    accuracy                           0.73        56
   macro avg       0.68      0.66      0.67        56
weighted avg       0.72      0.73      0.72        56


_____________________________________________
_____________________________________________
Confusion Matrix:
 [[33  6]
 [ 9  8]]
```

Gradient Boosting– Facts

- ✓ Gradient Boosting Classifier is a machine learning algorithm that combines multiple weak learners to create a strong predictive model.
- ✓ It is an ensemble method that builds the model in a sequential manner, where each subsequent model corrects the mistakes made by the previous models.
- ✓ Gradient Boosting works by optimizing a loss function using gradient descent to minimize the errors between predicted and actual values.
- ✓ It assigns weights to the weak learners based on their performance, allowing the model to focus on challenging instances.
- ✓ Gradient Boosting is versatile, handles various types of data, and often provides high predictive accuracy, but can be computationally expensive and prone to overfitting if not properly tuned.

7.8 – Train the model with suitable machine learning algorithms (Model implementation + Accuracy scores + Classification report + Confusion matrix)

Compare the accuracy of all the models

```
final_model = pd.DataFrame({'Models':['LR', 'DT', 'RF', 'SVM', 'KNN', 'NB', 'GBC'],
                            'ACC in %': [accuracy_score(Y_test, y_pred1)*100,
                                         accuracy_score(Y_test, y_pred2)*100,
                                         accuracy_score(Y_test, y_pred3)*100,
                                         accuracy_score(Y_test, y_pred4)*100,
                                         accuracy_score(Y_test, y_pred5)*100,
                                         accuracy_score(Y_test, y_pred6)*100,
                                         accuracy_score(Y_test, y_pred7)*100]})

#In above code, we are multiplying by 100 to make sure, we can see the accuracy in percentage
```

```
final_model1 = final_model.round(2) #This is to limit deviding result to 2 decimal points
final_model1
```

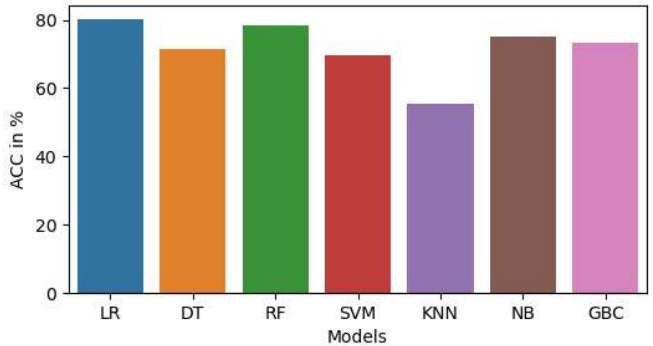
	Models	ACC in %
0	LR	80.36
1	DT	71.43
2	RF	78.57
3	SVM	69.64
4	KNN	55.36
5	NB	75.00
6	GBC	73.21

This **Barplot** clearly shows:

- Linear regression ML algorithm is giving us highest accuracy **(80.36 %)**

- 2nd highest accuracy is Random Forest ML algorithm **(78.57 %)**

```
plt.figure(figsize=(6,3))
sns.barplot(final_model1['Models'], final_model1['ACC in %'])
plt.show()
```



8. Test Model

Once the model is trained, we need to test the model with Test data. As explained in Sectin.3, we still have “Test Data set” and this will be used to test our model.

Until now, model is trained on “Training Data set”

- *In this step, the selected model will be tested on Training data and Test data set as well.*
- *Picture mentioned in this slide shows an example python code from **scikit-learn library**. This shows an example python code of model being tested on Training data and Test data set.*

Similar technique shall be used in this project with given data set.

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

This step has been completed in previous slides (7.1 to 7.6)

- *First, we train the model with training data*
- *Then, we test the model with testing data*
- *Further testing of the model with random data shall be executed in upcoming slides 8-1 to 8-3*

9. Evaluate Model Performance

Evaluation metrics quantify the performance of a machine learning model. It involves training a model and then comparing the predictions to expected values.

Once the model is trained, we need to test the model with Test data. As explained in Sectin.3, we still have “Test Data set” and this will be used to test our model.

Until now, model is trained on “Training Data set”

- *For classification models, the evaluation method includes Classification report which includes*

- *Precision*
- *Recall*
- *F1-score*
- *Support*
- *Confusion matrix*
- *Accuracy score*
- *ROC Curve*

One such example where, the model is tested and evaluated with above said evaluation methods are mentioned in this slide for reference purpose.

Similar technique shall be used in this project with given data set to predict the “Death event”

```
rf_cv.fit(X_train, y_train)
best_params = rf_cv.best_params_
print(f"Best parameters: {best_params}")

rf_clf = RandomForestClassifier(**best_params)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 3 folds for each of 768 candidates, totalling 2304 fits
Best parameters: {'bootstrap': False, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 1500}
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision   1.0    1.0       1.0        1.0           1.0
recall      1.0    1.0       1.0        1.0           1.0
f1-score    1.0    1.0       1.0        1.0           1.0
support   853.0  176.0       1.0     1029.0        1029.0
_______________________________________________
Confusion Matrix:
 [[853   0]
 [  0 176]]

Test Result:
================================================
Accuracy Score: 85.94%
_______________________________________________
CLASSIFICATION REPORT:
                   0          1  accuracy   macro avg  weighted avg
precision   0.871495   0.461538   0.85941    0.666517      0.814789
recall      0.981579   0.098361   0.85941    0.539970      0.859410
f1-score    0.923267   0.162162   0.85941    0.542715      0.817990
support   380.000000  61.000000   0.85941  441.000000    441.000000
_______________________________________________
Confusion Matrix:
 [[373   7]
 [ 55   6]]
```

This step has been completed in previous slides (7.1 to 7.6)

- *After testing the various models, we are evaluating the model performance*
- *In slide 7.8, we are also comparing which model performance is better. (highest accuracy)*

8.1 - Test the model: Test the model with random input data & check the result

Train the data on entire dataset on Logistic regression model predicted highest accuracy for given dataset("80.36 % accuracy")

```
# import the Logistic regression model as this is the one with highest accuracy score
from sklearn.linear_model import LogisticRegression

lr_full = LogisticRegression()
lr_full.fit(X,Y)    #X & Y are having full data without training and testing division
```

```
LogisticRegression()
```

```
#As a testing, here we can enter the data as we need to test the model
new_data = pd.DataFrame({
            'age': 50,
            'anaemia': 1,
            'cpk': 200,
            'diabetes': 0,
            'EF': 20,
            'High BP': 1,
            'platelets': 300000,
            'Scr': 1.5,
            'S_Sodium': 190,
            'Sex': 1,
            'smoking': 1,
}, index=[0])
```

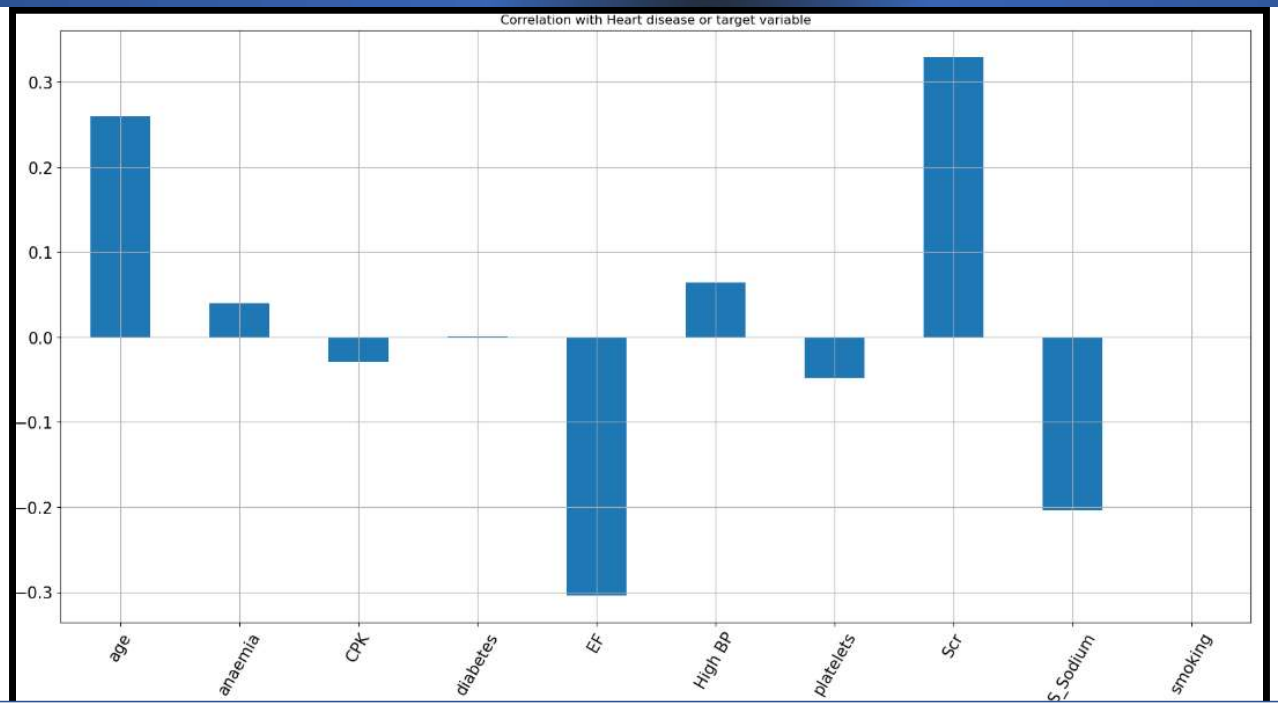
```
new_data #Above entered data is represented in row/colum
```

	age	anaemia	cpk	diabetes	EF	High BP	platelets	Scr	S_Sodium	Sex	smoking
0	50	1	200	0	20	1	300000	1.5	190	1	1

```
#In case, the given input data (new_data) is giving the result of "1", this means patient is having having disease. Otherwise,
#with the given data, model will predict there is no heart disease

LR_Model_result = lr_full.predict(new_data)
if LR_Model_result[0] == 0:
    print("No Heart disease")
else:
    print("Patient has heart disease")
```

```
No Heart disease
```

8.2 - Test the model: Test the model with random input data & check the result



In the above co-relation chart, we can clearly see the Positive & negative co-related factors with reference to target variable

As we can see, Positively co-related input factors are

- "Age"
- "SCR"

Negatively co-related input factors are

- "EF"
- "S_Sodium"

In next slide:

- *In first example, we modify (**Increase**) the AGE/SCR and check the result as these are **positively** co-related with "target" variable*
- *In Second example, we modify(**Decrease**) the EF/S_Sodium and check the result s these are **negetively** co-related with "target" variable*

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

8-3 Test the model: Example-1 & Example-2 with random variables in the input data (Check result in both the cases highlighted in RED)

Example:1 - Age is increased to 80 & SCR is increased to 3.0

```
#As a testing, here we can enter the data as we need to test the model
new_data = pd.DataFrame({
            'age': 80,
            'anaemia': 1,
            'cpk': 200,
            'diabetes': 0,
            'EF': 20,
            'High BP': 1,
            'platelets': 300000,
            'Scr': 3.0,
            'S_Sodium': 190,
            'Sex': 1,
            'smoking': 1,
}, index=[0])
```

```
new_data #Above entered data is represented in row/colum
```

	age	anaemia	cpk	diabetes	EF	High BP	platelets	Scr	S_Sodium	Sex	smoking
0	80	1	200	0	20	1	300000	3.0	190	1	1

```
#In case, the given input data (new_data) is giving the result of "1", thi
#with the given data, model will predict there is no heart disease

LR_Model_result = lr_full.predict(new_data)
if LR_Model_result[0] == 0:
    print("No Heart disease")
else:
    print("Patient has heart disease")
```

```
Patient has heart disease
```

RESULT

Example:2 - EF is decreased to 20 & S_Sodium is decreased to 60

```
#As a testing, here we can enter the data as we need to test the model
new_data = pd.DataFrame({
            'age': 50,
            'anaemia': 1,
            'cpk': 200,
            'diabetes': 0,
            'EF': 20,
            'High BP': 1,
            'platelets': 300000,
            'Scr': 0.5,
            'S_Sodium': 60,
            'Sex': 1,
            'smoking': 1,
}, index=[0])
```

```
new_data #Above entered data is represented in row/colum
```

	age	anaemia	cpk	diabetes	EF	High BP	platelets	Scr	S_Sodium	Sex	smoking
0	50	1	200	0	20	1	300000	0.5	60	1	1

```
#In case, the given input data (new_data) is giving the result of "1", this m
#with the given data, model will predict there is no heart disease

LR_Model_result = lr_full.predict(new_data)
if LR_Model_result[0] == 0:
    print("No Heart disease")
else:
    print("Patient has heart disease")
```

```
Patient has heart disease
```

RESULT

10. Ensemble Learning (Optional)

Ensemble Learning is a general meta-approach to Machine Learning that seeks better predictive performance by combining the predictions from multiple models.

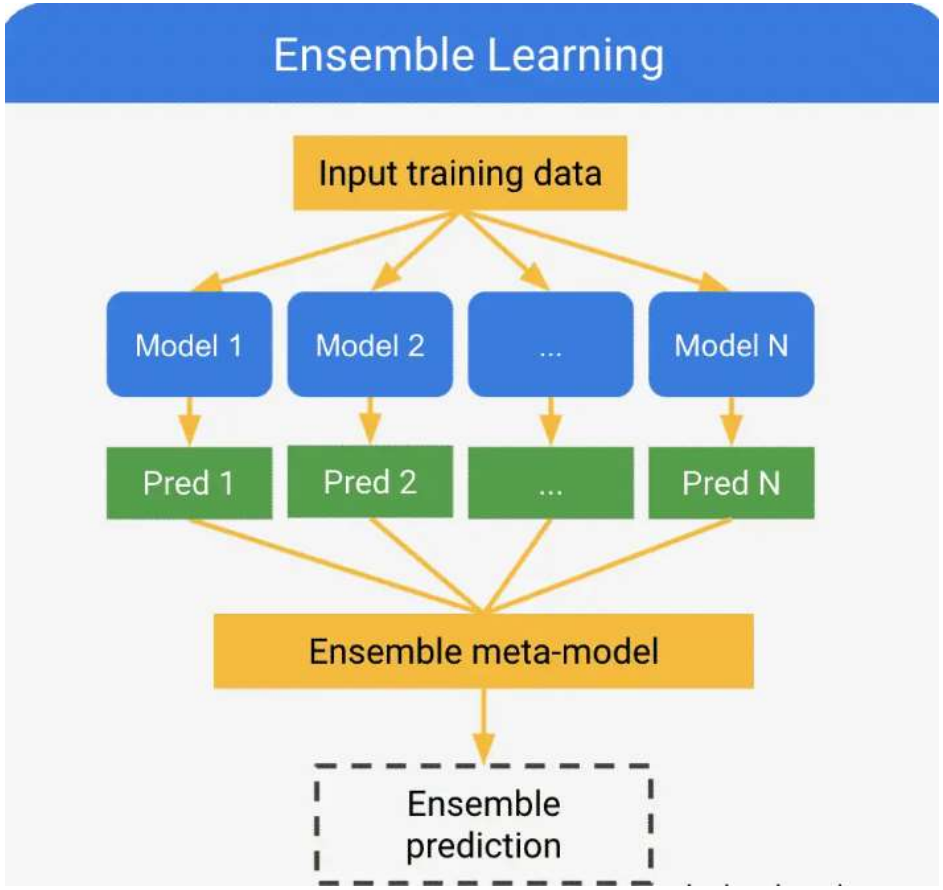
The predictions made by different models are taken as separate votes. Subsequently, the prediction made by most models is treated as the ultimate prediction.

To give an example, we might use Random forest and KNN ML algorithms for given data set. In the end, we are taking predictions of both of these models and do the Ensemble learning.

Types of Ensemble Modeling Techniques are:

- **Bagging**
- **Boosting**
- **Stacking**
- **Blending**

With in this healthcare project, this shall be done only if needed. Here it is mentioned to give brief introduction about this feature of Machine learning algorithm.



This step is not executed for the given data set / this project

Conclusion – Summary of the Project - "RAM - Cardiac Risk Calculator"

- We are at the last section of this presentation and here we are going to summarize the outcome of this project.
- "**RAM - Cardiac Risk Calculator**" has been successfully developed with help of all the available resources as mentioned in the beginning of the presentation. (Software engineer, Python software and using all suitable Machine learning algorithm models)
- We concluded which ML model is suitable for this data set and to achieve better accuracy with the given data set. Hence, given data is trained on the suitable or appropriate Machine learning algorithm.
- At this point, the software development is completed for this project.
- Now, Software development team will handover the developed software to Graphics team to design an attractive GUI which is essential for the App. This GUI will contain company logo and several field to enter the input data.
- This Application - GUI (Mobile application or software) should allow patient to enter all the necessary parameters as explained in the beginning of this presentation.
- All these inputs shall be taken into the software, which was prepared as explained in all the previous slides, and we can predict the chances of death due to Cardio-Vascular disease.
- Based on this, patient can take remedial action to improve his health. For example, doing regular exercise to improve the health, consuming healthy food etc.
- In this way, the company (**Ram Health care**) can release this App "**RAM - Cardiac Risk Calculator**" in the market and patients can use this application to know the chances of getting death(mortality) due to heart attack by entering few essential parameters related to their health.
- This will increase the market cap for the company. Increasing brand visibility and recognition by appearing in app stores and search results.
- Encouraging customer engagement and interaction with the company, which can lead to increased customer loyalty and advocacy.
- Last but not the least, company can collect all the data of the users and customer behavior and preferences, which can be used to inform future marketing strategies.

"A healthy heart is a fountain of gladness, making everything fresh and new." - Lydia M. Child

"Take care of your body. It's the only place you have to live." - Jim Rohn

"Your body hears everything your mind says. Stay positive." - Unknown

"शरीरमाद्यं खलु धर्मसाधनम्" - कालिदास , कुमारसम्भवम्

- "The body is the foremost means of achieving righteousness." – Kalidasa from Kumarasabhavam

WHAT WE THINK,

WE BECOME.

Buddha

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD