VISUALIZING AND PREDICTING HEART DISEASES WITH AN INTERACTIVE DASH BOARD

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Among the different causes of human death, heart disease is one of the most common causes of non-communicable and silent death in the world. It is a challenge to early predict heart disease by using clinical data for better treatment. After evolving machine learning, its importance is incessantly being increased in every field of life. From the last couple of years, Machine learning is also the center of attention of researchers in field medical sciences. Researchers use different tools and techniques of machine learning for the early prediction of diseases. Regarding the different causes of heart disease, analysing what causes heart disease has become mainstream nowadays. After an in-depth understanding of data analysis and machine learning-related knowledge, data analysis and data training are carried out on a dataset. This project proposes a prediction model to predict whether a people have a heart disease or not and visualize it using a dashboard.

1.2 PURPOSE

With the ever increasing population of the world, diseases and their possibilities are also increasing at an alarming rate. As time passes by, diagnosing diseases and providing appropriate treatment at the right time has become quite a challenge. Heart diseases, for one, have been a major cause of death worldwide. Finding an efficient way to predict the chances of a heart failure is indeed a need.

CHAPTER-2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Existing solutions are found to have,

1. Inefficient and inaccurate heart disease prediction system

2. No proper assistance of medical professionals in evaluating a patient's heart disease based on the clinical data of the patient.

2.2 REFERENCES

[1]-https://www.sciencedirect.com/science/article/pii/S1877050920315210

[2]-https://www.mdpi.com/2227-9032/8/2/111

[3]-https://ieeexplore.ieee.org/abstract/document/8550857?casa_token=IAOsNVA-LQ0AAAAA:JW0OSEBNpWx3Yk1GZI82uW7I20SoTluRGfPpwM6WqEgQO62tyIUaG7GQ2ts02tCggDHsmR70aoA

[4]- Real-time machine learning for early detection of heart disease using big data approach | IEEE Conference Publication | IEEE Xplore

[5]- Visualization and Prediction of Heart Diseases Using Data Science Framework | IEEE Conference Publication | IEEE Xplore

[6]- (PDF) IRJET- A LITERATURE SURVEY OF PREDICTING HEART DISEASE | IRJET Journal - Academia.edu

[7]- Heart Disease Prediction using Machine Learning Techniques | SpringerLink

[8]- Heart Disease Prediction Using Various Machine Learning Algorithms |SpringerLink

2.3 PROBLEM STATEMENT DEFINITION

The leading cause of death is heart disease and therefore there needs to be work done to help prevent the heart disease. Day by day the cases of heart diseases are increasing at a rapid rate and it's very important and concerning to predict any such diseases beforehand. This diagnosis is a difficult task i.e. it should be performed precisely and efficiently.

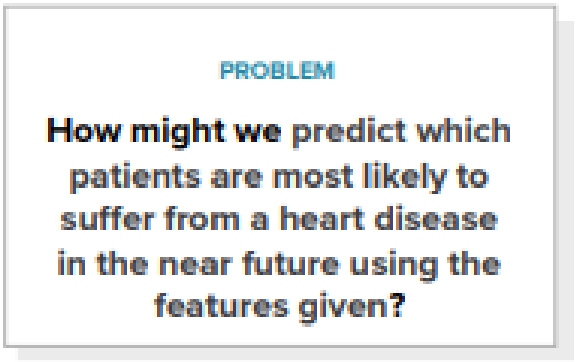
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

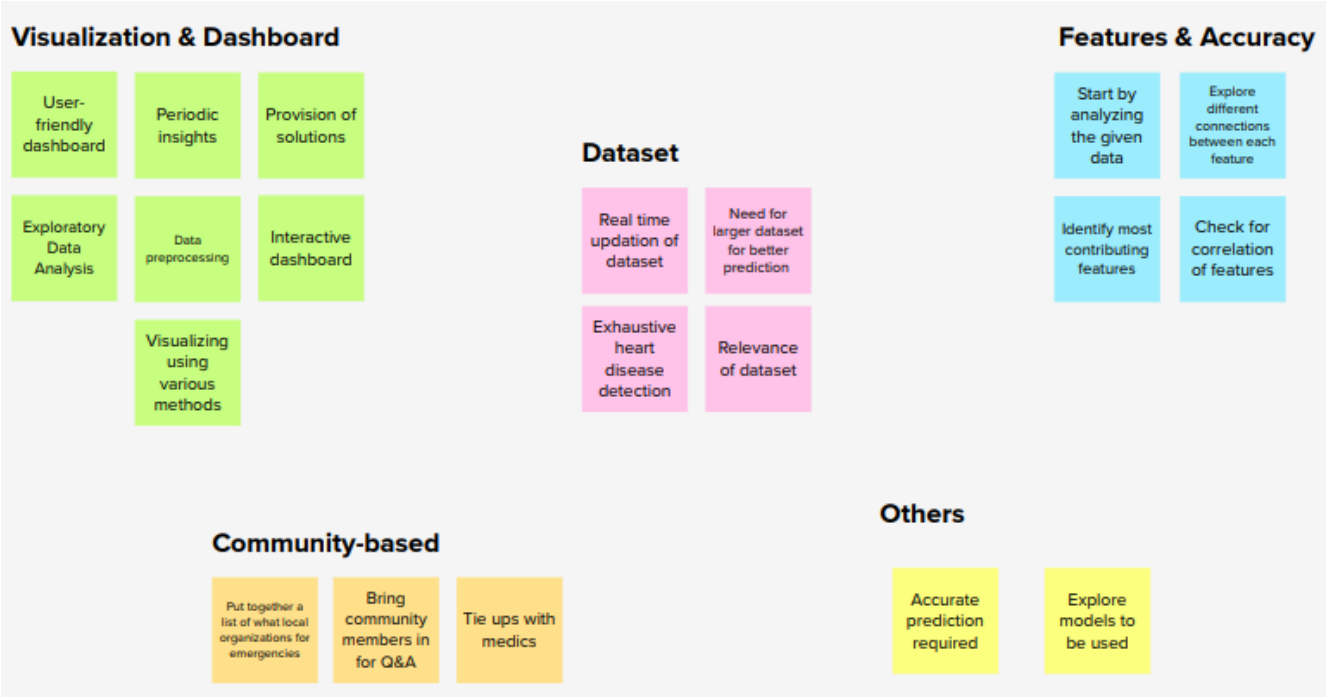
3.2 IDEATION & BRAINSTORMING

Step-1: Team Gathering and Select the Problem Statement



Step-2: Brainstorm, Idea Listing

Step-3: Grouping ideas



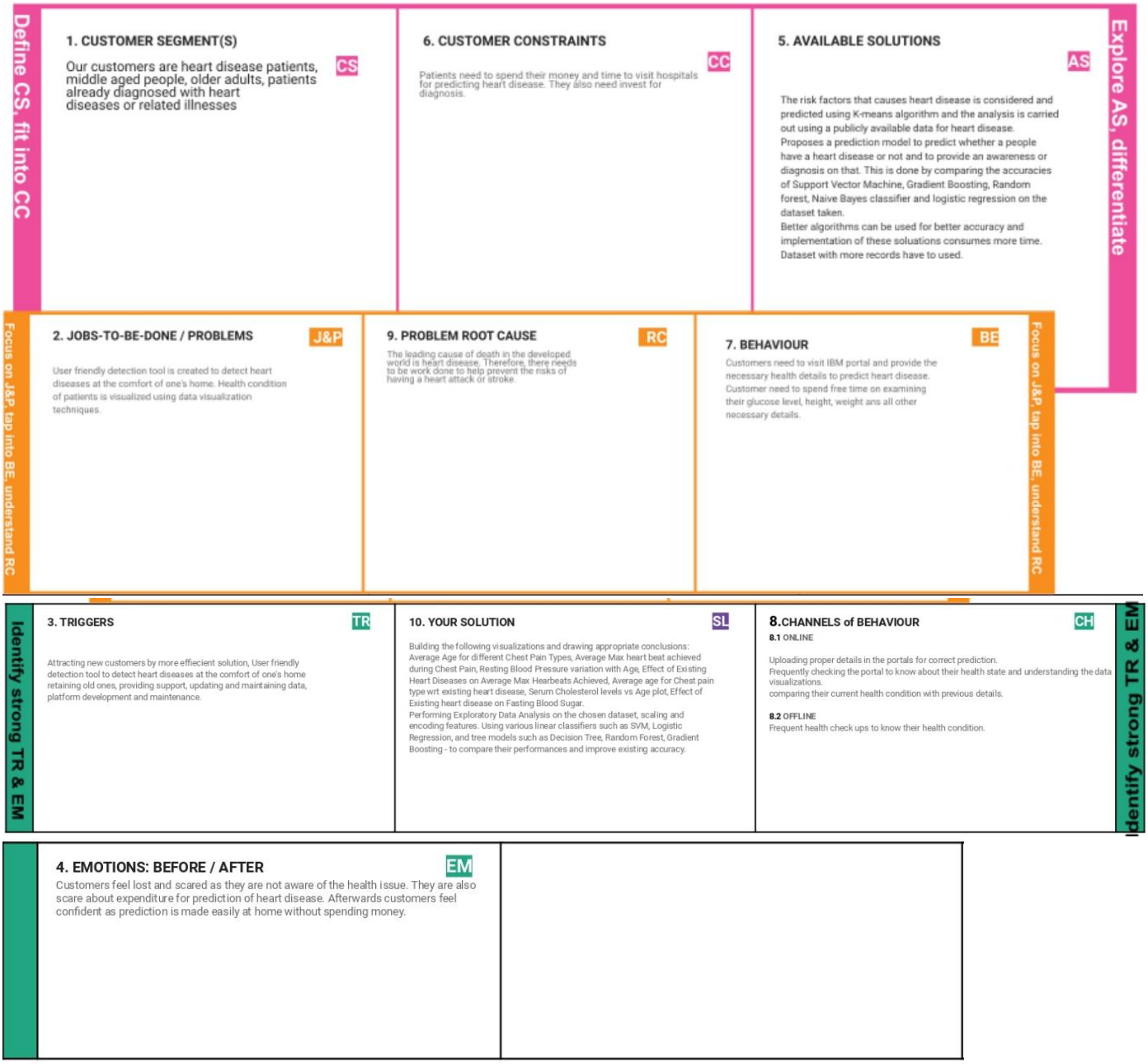
Step-4: Prioritize

3.3 PROPOSED SOLUTION

S.NO.	PARAMETER	DESCRIPTION
1.	Problem Statement (Problem to be solved)	▪ Users create multiple analytical graphs/charts/Visualizations. ▪ Using the Analytical Visualizations, build required Dashboard(s). ▪ Saving and visualizing the final dashboard in the IBM Cognos Analytics.
2.	Idea / Solution description	Building the following visualizations and drawing appropriate conclusions: ▪ Average Age for different Chest Pain Types ▪ Average Max heart beat achieved during Chest Pain ▪ Resting Blood Pressure variation with Age ▪ Effect of Existing Heart Diseases on Average Max Hearbeats Achieved ▪ Average age for Chest pain type wrt existing heart disease ▪ Serum Cholesterol levels vs Age plot ▪ Effect of Existing heart disease on Fasting Blood Sugar
3.	Novelty / Uniqueness	▪ Performing Exploratory Data Analysis on the chosen dataset, scaling and encoding features. ▪ Using various linear classifiers such as SVM, Logistic Regression, and tree models such as Decision Tree, Random Forest, Gradient Boosting - to compare their performances and improve existing accuracy.
4.	Social Impact / Customer Satisfaction	The leading cause of death in the developed world is heart disease. Therefore, there needs to be work done to help prevent the risks of having a heart attack or stroke.

5.	Business Model (Revenue Model)	Key Partners: 3rd Party Applications, foundations, Govt. Agencies Cost Structure: Tech, Investment (fixed) Revenue Streams: public and private contracts Customer segments: Middle aged people, older adults, patients already diagnosed with heart diseases or related illnesses Customer Relationship: Customer support, exclusive channels Customer channels: website, chatbot Value proposition: User friendly detection tool to detect heart diseases at the comfort of one's home Key Activities: Attracting new customers, retaining old ones, providing support, updating and maintaining data, platform development and maintenance Key Resources: Digital platform, loyal customer base, dedicated team
6.	Scalability of the Solution	There are 2 different ways to accomplish scaling for the proposed solution - Horizontal & Vertical scaling. This solution can be vertically scaled with increasing the capacity of existing dataset with no change in the code. In terms of horizontal scalability, the proposed model can be scaled wider to deal with the traffic. This can be done by increasing the GPUs (Graphical Processing Units) & TPUs (Tensor Processing Units) and working together as a single logical unit for faster access.

3.4 PROBLEM SOLUTION FIT



CHAPTER-4

REQUIREMENTS ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Verification	Verification through CAPTCHA
FR-4	User Authentication	Resending the code incase of a forgotten password, Retyping password
FR-5	User Validation	Reconfirming the new password
FR-6	User Submission	Submission through form Submission through Email.

4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The EHDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge, eg, relationships between medical factors related to heart disease and patterns, to be established.
NFR-2	**Security**	When it comes to health factors, we should provide good security services. There should be no errors, lagging , base of data of a patient profile, while working on the software or product.
NFR-3	**Reliability**	Reliability is said to be the measure of stability or consistency of test scores shown in your product. Therefore this product will normally be a good performing one in the field of accuracy.
NFR-4	**Performance**	The performance should be fast relaying. This prediction system should be made available in cloud to ensure better accessibility and setting a milestone

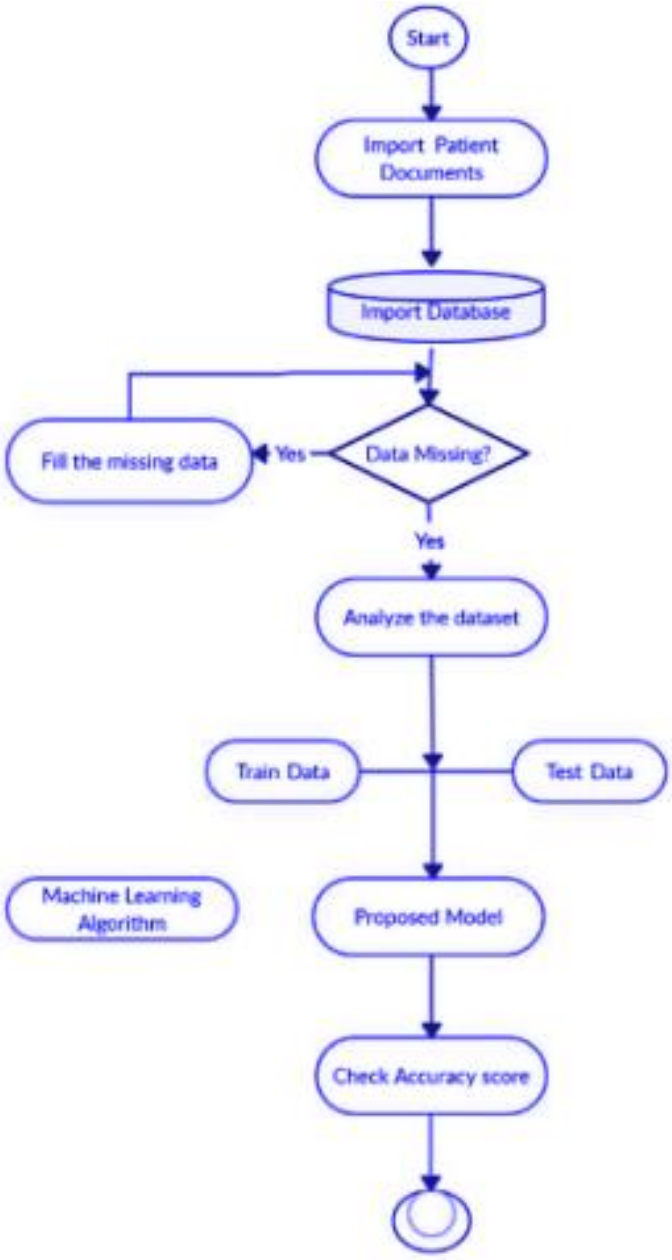
		in providing good quality affordable healthcare.
NFR-5	**Availability**	The Availability of getting used to this software or product design is through by accessing IBM Cognos Analytics and IBM Cloud.
NFR-6	**Scalability**	It is based on the number of users who maintaining the software or a system according to its performance like workflow, increase or decrease in efficiency , response time etc. It scalability can be measured by maintenance, checking in for software updates, fixing errors if occurred in the server. By this a good quality product is got.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

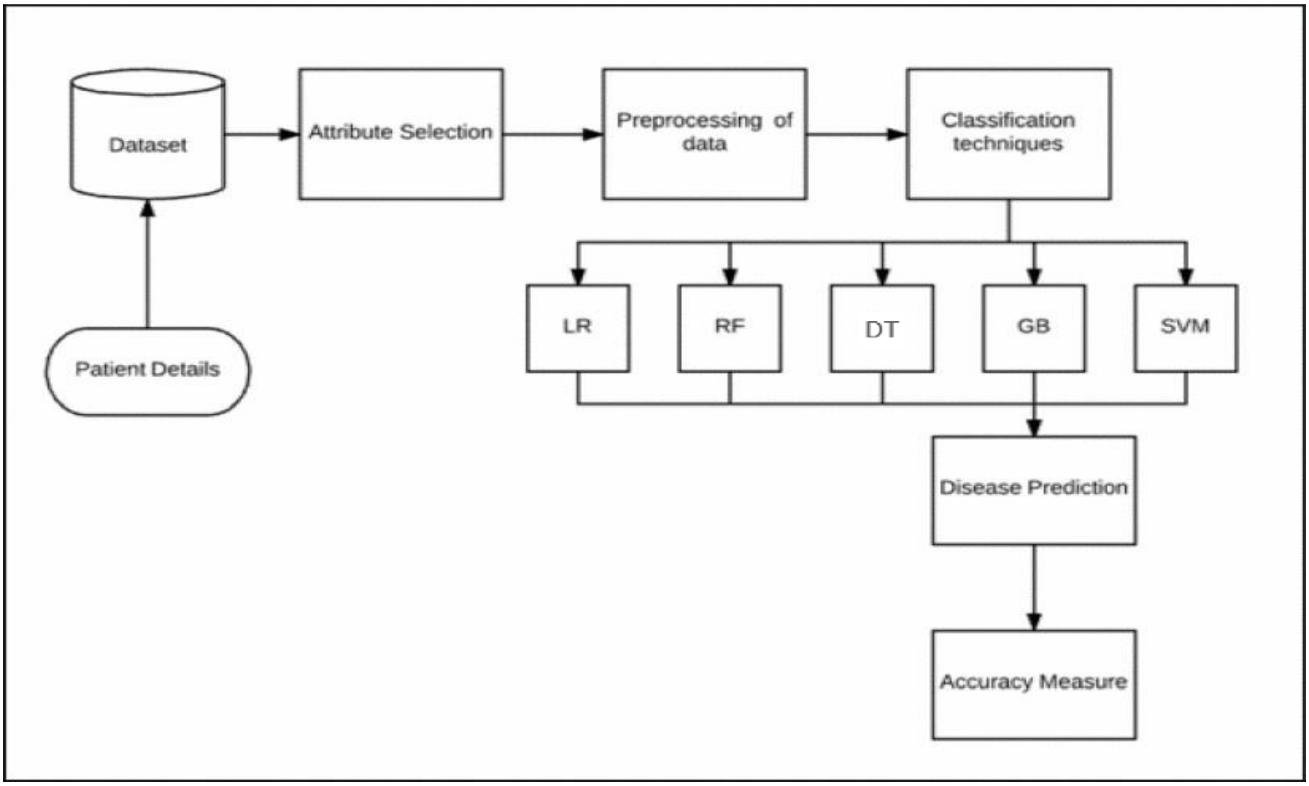


5.2 SOLUTION & TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

SOLUTION ARCHITECTURE DIAGRAM:



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile User)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register & access the dashboard	High	Sprint-1
	Dashboard	USN-6	Profile - View & Update Profile	I can view and update my profile	Medium	Sprint-2
		USN-7	Change Password - user can change their password	I can change my password	Medium	Sprint-1
		USN-8	Home Page - Analyze your health!	I can detect my health condition from wherever I want.	High	Sprint-1
		USN-9	The user will have to fill in the below 13 fields for the system to predict a disease -Age in Year -Gender -Chest Pain Type -Fasting Blood Sugar -Resting Electrographic Results(Restecg) -Exercise Induced Angina(Exang) -The slope of the peak exercise ST segment -CA – Number of major vessels colored by fluoroscopy -Thal	These are the categories available in the application.	High	Sprint-2
			-Trest Blood Pressure -Serum Cholesterol -Maximum heart rate achieved(Thalach) -ST depression induced by exercise(Oldpeak)			
		USN-10	View Doctors - view doctor detail by searching by names or filter by user's choice and location.	Using this functionality, users can find doctors in their locality..	Medium	Sprint-1
Customer (Web user)	System Requirement	USN-11	I. Hardware Requirement i. Laptop or PC: - I5 processor system or higher - 4 GB RAM or higher - 128 GB ROM or higher ii. Android Phone (12.0 or above)	These are the specifications that should be available on the user's PC.	Medium	Sprint-2
		USN-12	II. Software Requirement iii. Laptop or PC - Windows 8 or higher	Install this application that can be used to predict the presence of heart disease.	Medium	Sprint-2
		USN-13	Reference- https://ieeexplore.ieee.org/document/9619208/		Medium	Sprint-1
Customer Care Executive	Dashboard	USN-14	Queries	Users can post your queries in the text box available in that application.	Medium	Sprint-2
		USN-15	Toll Free	Toll free number for customer care services.	Medium	Sprint-2
		USN-16	Ratings	User can rate the application and give their reviews	Medium	Sprint-2
Administrator	Dashboard	USN-17	Verification	Verification through CAPTCHA	Medium	Sprint-2
		USN-18	Validation	Reconfirming the new password.	Medium	Sprint-2
		USN-19	Feedback - send feedback to the Admin.	Users can send their	Medium	Sprint-2

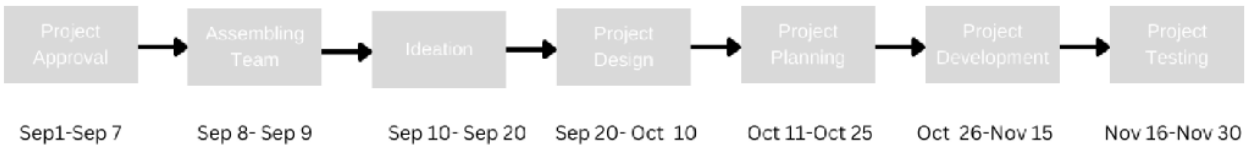
CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-8	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-9	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-10	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-11	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-12	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-13	As a user, I can upload my relevant health data	I can enter the parameters in the correct format and receive confirmation	High	Sprint-1
	Dashboard	USN-14	As a user, I can view my visualizations and prediction	I can receive successfully computed message and results.	High	Sprint-1
Customer Care Executive	Message box	USN-15	As an executive, I'm able to view any complaint or feedback message left by the user in the message box.	I can view the arrival of new feedback	Low	Sprint-2
Administrator	Dashboard	USN-16	As an admin, I'm able to view all historical data, uploaded data of a user and predictions and visualizations.	I can open any data at any time	Low	Sprint-2

6.2 SPRINT DELIVERY SCHEDULE



CHAPTER-7

CODING & SOLUTIONING

7.1 FEATURE 1

Logging in and Registering:

Here a new patient will be able to make an account and upload his details for prediction and an existing registered patient will be able to log in with his already registered details.

7.2 FEATURE 2

Predicting and Visualizing:

In prediction, the form inputs relevant entered details. Fields are provided for the same and a submit button to post the details

In visualization, we are able to view patient progress over a period of time.

CHAPTER-8

TESTING

8.1 TEST CASES

```
# Scaling
from sklearn.preprocessing import RobustScaler

# Train Test Split
from sklearn.model_selection import train_test_split

# Models
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Metrics
from sklearn.metrics import accuracy_score, classification_report, roc_curve

# Cross Validation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

print('Packages imported...')
```

```
Packages imported...
```

4.3.2 Train and test split

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)
print("The shape of X_train is      ", X_train.shape)
print("The shape of X_test is       ",X_test.shape)
print("The shape of y_train is      ",y_train.shape)
print("The shape of y_test is       ",y_test.shape)
```

```
The shape of X_train is       (242, 22)
The shape of X_test is        (61, 22)
The shape of y_train is       (242, 1)
The shape of y_test is        (61, 1)
```

CHAPTER-9

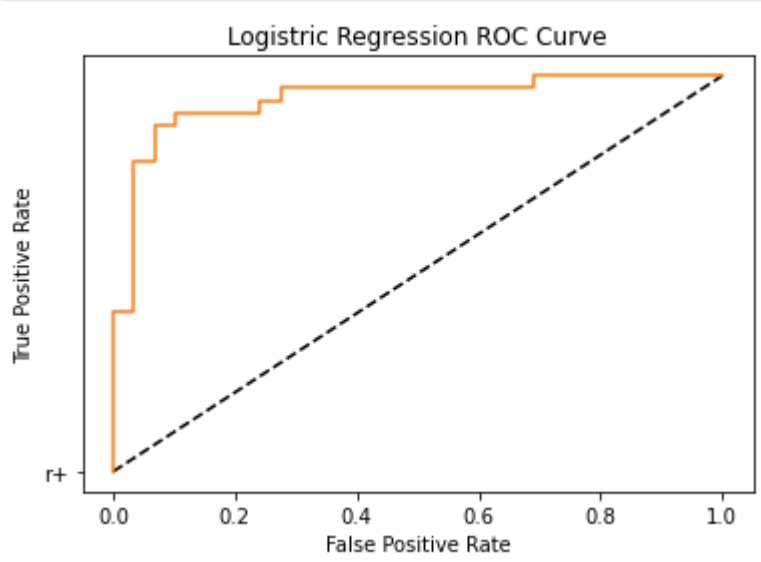
RESULTS

9.1 PERFORMANCE METRICS

```
The test accuracy score of SVM is  0.8688524590163934
```

```
The best params are : {'C': 3, 'gamma': 0.1}
The best score is   : 0.8384353741496599
The test accuracy score of SVM after hyper-parameter tuning is  0.9016393442622951
```

```
The test accuracy score of Logistric Regression is  0.9016393442622951
```




```
The test accuracy score of Decision Tree is  0.7868852459016393
```

```
The test accuracy score of Random Forest is  0.7868852459016393
```

```
The test accuracy score of Gradient Boosting Classifier is  0.8688524590163934
```

CHAPTER-10

ADVANTAGES & DISADVANTAGES

ADVANTAGES :

● Medical professionals can quickly recognize and respond to potential dangers with the right visualization tools.

● Doctors may better define patient populations and allocate resources by displaying health data in real-time.

● Doctors may better forecast their patients' health and make more accurate diagnoses using data visualization software equipped with predictive analytics technologies.

● Clear and simple charts and infographics improve patient awareness and increase participation, making visualization a helpful tool for patient education.

● Visualization tools can significantly improve presentations and reports.

● Good data visualization aids in making the call to impose health regulations.

DISADVANTAGES :

● Due to the risk of malpractice, a visualization intended for use by health care professionals must be so clear and precise as to render misinterpretation virtually impossible.

● Insufficient data or poor structuring of data objects in the database may lead to inaccurate interpretation of visualized data.

CHAPTER-11

CONCLUSION

Diagnosis of cardiac disease is the sternest challenge in the medical profession. It is based on the thorough review by medical experts of the various clinical and medical data of the patient. The methods which are used for comparison of ML models are confusion matrix, heat map.. For the 13 features which were in the dataset, SVM with hyperparameter tuning performed better in the ML approach when data pre-processing is applied.

CHAPTER-12

FUTURE SCOPE

● The current project does for prediction and visualization of heart diseases. It can be extended to add features related to hospital management.

● The dataset size can be increased and then deep learning with various other optimizations can be used and more promising results can be achieved.

● More ways could be found where we could integrate heart-disease-trained ML and DL models with certain multimedia for the ease of patients and doctors.

● If a large dataset is present, the results can increase very much in deep learning and ML as well.

● The prediction of heart diseases by using advanced techniques and algorithms in less time complexity.

CHAPTER-13

APPENDIX

13.1 SOURCE CODE

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

import warnings
warnings.filterwarnings("ignore")


df = pd.read_csv(r"C:\Users\aarth\Downloads\heart.csv")


print("The shape of the dataset is : ", df.shape)


df.head()


dict = {}
for i in list(df.columns):
    dict[i] = df[i].value_counts().shape[0]

pd.DataFrame(dict,index=["unique count"]).transpose()


cat_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']
con_cols = ["age","trtbps","chol","thalachh","oldpeak"]
target_col = ["output"]
print("The categorial cols are : ", cat_cols)
print("The continuous cols are : ", con_cols)
print("The target variable is :  ", target_col)


df[con_cols].describe().transpose()


df.isnull().sum()


fig = plt.figure(figsize=(18,15))
```

```
gs = fig.add_gridspec(3,3)
gs.update(wspace=0.5, hspace=0.25)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[0,2])
ax3 = fig.add_subplot(gs[1,0])
ax4 = fig.add_subplot(gs[1,1])
ax5 = fig.add_subplot(gs[1,2])
ax6 = fig.add_subplot(gs[2,0])
ax7 = fig.add_subplot(gs[2,1])
ax8 = fig.add_subplot(gs[2,2])

background_color = "#ffe6e6"
color_palette = ["#800000","#8000ff","#6aac90","#5833ff","#da8829"]
fig.patch.set_facecolor(background_color)
ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)
ax6.set_facecolor(background_color)
ax7.set_facecolor(background_color)
ax8.set_facecolor(background_color)

# Title of the plot
ax0.spines["bottom"].set_visible(False)
ax0.spines["left"].set_visible(False)
ax0.spines["top"].set_visible(False)
ax0.spines["right"].set_visible(False)
ax0.tick_params(left=False, bottom=False)
ax0.set_xticklabels([])
ax0.set_yticklabels([])


ax0.text(0.5,0.5,
        'Count plot for various\n categorical features\n_____________________',
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=18, fontweight='bold',
        fontfamily='serif',
        color="#000000")

# Sex count
ax1.text(0.3, 220, 'Sex', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax1,data=df,x='sex',palette=color_palette)
```

```
ax1.set_xlabel("")
ax1.set_ylabel("")

# Exng count
ax2.text(0.3, 220, 'Exng', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax2.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax2,data=df,x='exng',palette=color_palette)
ax2.set_xlabel("")
ax2.set_ylabel("")

# Caa count
ax3.text(1.5, 200, 'Caa', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax3,data=df,x='caa',palette=color_palette)
ax3.set_xlabel("")
ax3.set_ylabel("")

# Cp count
ax4.text(1.5, 162, 'Cp', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax4.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax4,data=df,x='cp',palette=color_palette)
ax4.set_xlabel("")
ax4.set_ylabel("")


# Fbs count
ax5.text(0.5, 290, 'Fbs', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax5,data=df,x='fbs',palette=color_palette)
ax5.set_xlabel("")
ax5.set_ylabel("")

# Restecg count
ax6.text(0.75, 165, 'Restecg', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax6.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax6,data=df,x='restecg',palette=color_palette)
ax6.set_xlabel("")
ax6.set_ylabel("")

# Slp count
ax7.text(0.85, 155, 'Slp', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax7.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax7,data=df,x='slp',palette=color_palette)
ax7.set_xlabel("")
ax7.set_ylabel("")
```

```
# Thall count
ax8.text(1.2, 180, 'Thall', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax8.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax8,data=df,x='thall',palette=color_palette)
ax8.set_xlabel("")
ax8.set_ylabel("")


for s in ["top","right","left"]:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)
    ax3.spines[s].set_visible(False)
    ax4.spines[s].set_visible(False)
    ax5.spines[s].set_visible(False)
    ax6.spines[s].set_visible(False)
    ax7.spines[s].set_visible(False)
    ax8.spines[s].set_visible(False)


fig = plt.figure(figsize=(18,16))
gs = fig.add_gridspec(2,3)
gs.update(wspace=0.3, hspace=0.15)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[0,2])
ax3 = fig.add_subplot(gs[1,0])
ax4 = fig.add_subplot(gs[1,1])
ax5 = fig.add_subplot(gs[1,2])

background_color = "#ffe6e6"
color_palette = ["#800000","#8000ff","#6aac90","#5833ff","#da8829"]
fig.patch.set_facecolor(background_color)
ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)

# Title of the plot
ax0.spines["bottom"].set_visible(False)
ax0.spines["left"].set_visible(False)
ax0.spines["top"].set_visible(False)
ax0.spines["right"].set_visible(False)
ax0.tick_params(left=False, bottom=False)
```

```
ax0.set_xticklabels([])
ax0.set_yticklabels([])
ax0.text(0.5,0.5,
        'Boxen plot for various\n continuous features\n_____',
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=18, fontweight='bold',
        fontfamily='serif',
        color="#000000")


# Age
ax1.text(-0.05, 81, 'Age', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax1,y=df['age'],palette=["#800000"],width=0.6)
ax1.set_xlabel("")
ax1.set_ylabel("")

# Trtbps
ax2.text(-0.05, 208, 'Trtbps', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax2.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax2,y=df['trtbps'],palette=["#8000ff"],width=0.6)
ax2.set_xlabel("")
ax2.set_ylabel("")

# Chol
ax3.text(-0.05, 600, 'Chol', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax3,y=df['chol'],palette=["#6aac90"],width=0.6)
ax3.set_xlabel("")
ax3.set_ylabel("")

# Thalachh
ax4.text(-0.09, 210, 'Thalachh', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax4.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax4,y=df['thalachh'],palette=["#5833ff"],width=0.6)
ax4.set_xlabel("")
ax4.set_ylabel("")

# oldpeak
ax5.text(-0.1, 6.6, 'Oldpeak', fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
```

```
sns.boxenplot(ax=ax5,y=df['oldpeak'],palette=["#da8829"],width=0.6)
ax5.set_xlabel("")
ax5.set_ylabel("")


for s in ["top","right","left"]:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)
    ax3.spines[s].set_visible(False)
    ax4.spines[s].set_visible(False)
    ax5.spines[s].set_visible(False)


fig = plt.figure(figsize=(18,7))
gs = fig.add_gridspec(1,2)
gs.update(wspace=0.3, hspace=0.15)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])

background_color = "#ffe6e6"
color_palette = ["#800000","#8000ff","#6aac90","#5833ff","#da8829"]
fig.patch.set_facecolor(background_color)
ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)

# Title of the plot
ax0.text(0.5,0.5,"Count of the target\n_____________",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')

ax0.set_xticklabels([])
ax0.set_yticklabels([])
ax0.tick_params(left=False, bottom=False)


# Target Count
ax1.text(0.35,177,"Output",fontsize=14, fontweight='bold', fontfamily='serif',
color="#000000")
ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax1, data=df, x = 'output',palette = color_palette)
ax1.set_xlabel("")
ax1.set_ylabel("")
ax1.set_xticklabels(["Low chances of attack(0)","High chances of attack(1)"])
```

```
ax0.spines["top"].set_visible(False)
ax0.spines["left"].set_visible(False)
ax0.spines["bottom"].set_visible(False)
ax0.spines["right"].set_visible(False)
ax1.spines["top"].set_visible(False)
ax1.spines["left"].set_visible(False)
ax1.spines["right"].set_visible(False)


fig = plt.figure(figsize=(10,10))
gs = fig.add_gridspec(1,1)
gs.update(wspace=0.3, hspace=0.15)
ax0 = fig.add_subplot(gs[0,0])

color_palette = ["#5833ff","#da8829"]
mask = np.triu(np.ones_like(df_corr))
ax0.text(1.5,-0.1,"Correlation Matrix",fontsize=22, fontweight='bold', fontfamily='serif',
color="#000000")
df_corr = df[con_cols].corr().transpose()
sns.heatmap(df_corr,mask=mask,fmt=".1f",annot=True,cmap='YlGnBu')
plt.show()


fig = plt.figure(figsize=(12,12))
corr_mat = df.corr().stack().reset_index(name="correlation")
g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette="YlGnBu", hue_norm=(-1, 1), edgecolor=".7",
    height=10, sizes=(50, 250), size_norm=(-.2, .8),
)
g.set(xlabel="features on X", ylabel="featurs on Y", aspect="equal")
g.fig.suptitle('Scatterplot heatmap',fontsize=22, fontweight='bold', fontfamily='serif',
color="#000000")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".7")
plt.show()
fig = plt.figure(figsize=(18,18))
gs = fig.add_gridspec(5,2)
gs.update(wspace=0.5, hspace=0.5)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
```

```
ax2 = fig.add_subplot(gs[1,0])
ax3 = fig.add_subplot(gs[1,1])
ax4 = fig.add_subplot(gs[2,0])
ax5 = fig.add_subplot(gs[2,1])
ax6 = fig.add_subplot(gs[3,0])
ax7 = fig.add_subplot(gs[3,1])
ax8 = fig.add_subplot(gs[4,0])
ax9 = fig.add_subplot(gs[4,1])

background_color = "#ffe6e6"
color_palette = ["#800000","#8000ff","#6aac90","#5833ff","#da8829"]
fig.patch.set_facecolor(background_color)
ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)
ax6.set_facecolor(background_color)
ax7.set_facecolor(background_color)
ax8.set_facecolor(background_color)
ax9.set_facecolor(background_color)

# Age title
ax0.text(0.5,0.5,"Distribution of age\naccording to\n target variable\n_____",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax0.spines["bottom"].set_visible(False)
ax0.set_xticklabels([])
ax0.set_yticklabels([])
ax0.tick_params(left=False, bottom=False)


# Chol
ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax5, data=df, x='chol',hue="output",
fill=True,palette=["#8000ff","#da8829"], alpha=.5, linewidth=0)
ax5.set_xlabel("")
ax5.set_ylabel("")

# Thalachh title
ax6.text(0.5,0.5,"Distribution of thalachh\naccording to\n target variable\n_____",
        horizontalalignment = 'center',
```

```
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax6.spines["bottom"].set_visible(False)
ax6.set_xticklabels([])
ax6.set_yticklabels([])
ax6.tick_params(left=False, bottom=False)

# Thalachh
ax7.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax7, data=df, x='thalachh',hue="output",
fill=True,palette=["#8000ff","#da8829"], alpha=.5, linewidth=0)
ax7.set_xlabel("")
ax7.set_ylabel("")

# Oldpeak title
ax8.text(0.5,0.5,"Distribution of oldpeak\naccording to\n target variable\n_____________",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax8.spines["bottom"].set_visible(False)
ax8.set_xticklabels([])
ax8.set_yticklabels([])
ax8.tick_params(left=False, bottom=False)


# Oldpeak
ax9.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax9, data=df, x='oldpeak',hue="output",
fill=True,palette=["#8000ff","#da8829"], alpha=.5, linewidth=0)
ax9.set_xlabel("")
ax9.set_ylabel("")

for i in ["top","left","right"]:
    ax0.spines[i].set_visible(False)
    ax1.spines[i].set_visible(False)
    ax2.spines[i].set_visible(False)
    ax3.spines[i].set_visible(False)
    ax4.spines[i].set_visible(False)
    ax5.spines[i].set_visible(False)
    ax6.spines[i].set_visible(False)
    ax7.spines[i].set_visible(False)
```

```
    ax8.spines[i].set_visible(False)
    ax9.spines[i].set_visible(False)


fig = plt.figure(figsize=(18,20))
gs = fig.add_gridspec(6,2)
gs.update(wspace=0.5, hspace=0.5)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[1,0])
ax3 = fig.add_subplot(gs[1,1])
ax4 = fig.add_subplot(gs[2,0])
ax5 = fig.add_subplot(gs[2,1])
ax6 = fig.add_subplot(gs[3,0])
ax7 = fig.add_subplot(gs[3,1])
ax8 = fig.add_subplot(gs[4,0])
ax9 = fig.add_subplot(gs[4,1])
ax10 = fig.add_subplot(gs[5,0])
ax11 = fig.add_subplot(gs[5,1])

background_color = "#ffe6e6"
color_palette = ["#800000","#8000ff","#6aac90","#5833ff","#da8829"]
fig.patch.set_facecolor(background_color)
ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)
ax6.set_facecolor(background_color)
ax7.set_facecolor(background_color)
ax8.set_facecolor(background_color)
ax9.set_facecolor(background_color)
ax10.set_facecolor(background_color)
ax11.set_facecolor(background_color)


# Cp title
# 0 = Typical Angina, 1 = Atypical Angina, 2 = Non-anginal Pain, 3 = Asymptomatic
ax0.text(0.5,0.5,"Chest pain\ndistribution\n_____________",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax0.spines["bottom"].set_visible(False)
```

```
ax0.set_xticklabels([])
ax0.set_yticklabels([])
ax0.tick_params(left=False, bottom=False)
ax0.text(1,.5,"0 - Typical Angina\n1 - Atypical Angina\n2 - Non-anginal Pain\n3 -
Asymptomatic",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )

# Cp
ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax1, data=df, x='cp',hue="output", fill=True,palette=["#8000ff","#da8829"],
alpha=.5, linewidth=0)
ax1.set_xlabel("")
ax1.set_ylabel("")

# Caa title
ax2.text(0.5,0.5,"Number of\nmajor vessels\n_____________",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax2.text(1,.5,"0 vessels\n1 vessel\n2 vessels\n3 vessels\n4vessels",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )


ax2.spines["bottom"].set_visible(False)
ax2.set_xticklabels([])
ax2.set_yticklabels([])
ax2.tick_params(left=False, bottom=False)

# Caa
ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax3, data=df, x='caa',hue="output",
fill=True,palette=["#8000ff","#da8829"], alpha=.5, linewidth=0)
ax3.set_xlabel("")
ax3.set_ylabel("")

# Sex title
ax4.text(0.5,0.5,"Heart Attack\naccording to\nsex\n______",
        horizontalalignment = 'center',
```

```
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax4.text(1,.5,"0 - Female\n1 - Male",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )
ax4.spines["bottom"].set_visible(False)
ax4.set_xticklabels([])
ax4.set_yticklabels([])
ax4.tick_params(left=False, bottom=False)

# Sex
ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.countplot(ax=ax5,data=df,x='sex',palette=["#8000ff","#da8829"], hue='output')
ax5.set_xlabel("")
ax5.set_ylabel("")


# Thall title
ax6.text(0.5,0.5,"Distribution of thall\naccording to\n target variable\n_____________",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax6.text(1,.5,"Thalium Stress\nTest Result\n0, 1, 2, 3",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )
ax6.spines["bottom"].set_visible(False)
ax6.set_xticklabels([])
ax6.set_yticklabels([])
ax6.tick_params(left=False, bottom=False)

# Thall
ax7.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.kdeplot(ax=ax7, data=df, x='thall',hue="output",
fill=True,palette=["#8000ff","#da8829"], alpha=.5, linewidth=0)
ax7.set_xlabel("")
ax7.set_ylabel("")
```

```
# Thalachh title
ax8.text(0.5,0.5,"Boxen plot of\nthalachh wrt\noutcome\n_______",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax8.text(1,.5,"Maximum heart\nrate achieved",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )


ax8.spines["bottom"].set_visible(False)
ax8.set_xticklabels([])
ax8.set_yticklabels([])
ax8.tick_params(left=False, bottom=False)

# Thalachh
ax9.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax9, data=df,x='output',y='thalachh',palette=["#8000ff","#da8829"])
ax9.set_xlabel("")
ax9.set_ylabel("")



# Exng title
ax10.text(0.5,0.5,"Strip Plot of\nexng vs age\n______",
        horizontalalignment = 'center',
        verticalalignment = 'center',
        fontsize = 18,
        fontweight='bold',
        fontfamily='serif',
        color='#000000')
ax10.text(1,.5,"Exercise induced\nangina\n0 - No\n1 - Yes",
        horizontalalignment = 'center',
         verticalalignment = 'center',
        fontsize = 14
        )
ax10.spines["bottom"].set_visible(False)
ax10.set_xticklabels([])
ax10.set_yticklabels([])
ax10.tick_params(left=False, bottom=False)
```

```
# Exng
ax11.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.stripplot(ax=ax11, data=df,x='exng',y='age',hue='output',palette=["#8000ff","#da8829"])
ax9.set_xlabel("")
ax9.set_ylabel("")


for i in ["top","left","right"]:
    ax0.spines[i].set_visible(False)
    ax1.spines[i].set_visible(False)
    ax2.spines[i].set_visible(False)
    ax3.spines[i].set_visible(False)
    ax4.spines[i].set_visible(False)
    ax5.spines[i].set_visible(False)
    ax6.spines[i].set_visible(False)
    ax7.spines[i].set_visible(False)
    ax8.spines[i].set_visible(False)
    ax9.spines[i].set_visible(False)
    ax10.spines[i].set_visible(False)
    ax11.spines[i].set_visible(False)


sns.pairplot(df,hue='output',palette = ["#8000ff","#da8829"])
plt.show()


# Scaling
from sklearn.preprocessing import RobustScaler

# Train Test Split
from sklearn.model_selection import train_test_split

# Models
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Metrics
from sklearn.metrics import accuracy_score, classification_report, roc_curve

# Cross Validation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

print('Packages imported...')
```

```
# creating a copy of df
df1 = df

# define the columns to be encoded and scaled
cat_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']
con_cols = ["age","trtbps","chol","thalachh","oldpeak"]

# encoding the categorical columns
df1 = pd.get_dummies(df1, columns = cat_cols, drop_first = True)

# defining the features and target
X = df1.drop(['output'],axis=1)
y = df1[['output']]

# instantiating the scaler
scaler = RobustScaler()

# scaling the continuous featuree
X[con_cols] = scaler.fit_transform(X[con_cols])
print("The first 5 rows of X are")
X.head()



X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)
print("The shape of X_train is      ", X_train.shape)
print("The shape of X_test is       ",X_test.shape)
print("The shape of y_train is      ",y_train.shape)
print("The shape of y_test is       ",y_test.shape)


# instantiating the object and fitting
clf = SVC(kernel='linear', C=1, random_state=42).fit(X_train,y_train)

# predicting the values
y_pred = clf.predict(X_test)

# printing the test accuracy
print("The test accuracy score of SVM is ", accuracy_score(y_test, y_pred))


# instantiating the object
svm = SVC()
```

```
# setting a grid - not so extensive
parameters = {"C":np.arange(1,10,1),'gamma':[0.00001,0.00005,
0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5]}

# instantiating the GridSearchCV object
searcher = GridSearchCV(svm, parameters)

# fitting the object
searcher.fit(X_train, y_train)

# the scores
print("The best params are :", searcher.best_params_)
print("The best score is   :", searcher.best_score_)

# predicting the values
y_pred = searcher.predict(X_test)

# printing the test accuracy
print("The test accuracy score of SVM after hyper-parameter tuning is ",
accuracy_score(y_test, y_pred))


# instantiating the object
logreg = LogisticRegression()

# fitting the object
logreg.fit(X_train, y_train)

# calculating the probabilities
y_pred_proba = logreg.predict_proba(X_test)

# finding the predicted valued
y_pred = np.argmax(y_pred_proba,axis=1)

# printing the test accuracy
print("The test accuracy score of Logistic Regression is ", accuracy_score(y_test, y_pred))


# calculating the probabilities
y_pred_prob = logreg.predict_proba(X_test)[:,1]

# instantiating the roc_cruve
fpr,tpr,thresholds=roc_curve(y_test,y_pred_prob)

# plotting the curve
plt.plot([0,1],[0,1],"k--",'r+')
plt.plot(fpr,tpr,label='Logistic Regression')
```

```
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistric Regression ROC Curve")
plt.show()



# instantiating the object
dt = DecisionTreeClassifier(random_state = 42)

# fitting the model
dt.fit(X_train, y_train)

# calculating the predictions
y_pred = dt.predict(X_test)

# printing the test accuracy
print("The test accuracy score of Decision Tree is ", accuracy_score(y_test, y_pred))


# instantiating the object
rf = RandomForestClassifier()

# fitting the model
rf.fit(X_train, y_train)

# calculating the predictions
y_pred = dt.predict(X_test)

# printing the test accuracy
print("The test accuracy score of Random Forest is ", accuracy_score(y_test, y_pred))


# instantiate the classifier
gbt = GradientBoostingClassifier(n_estimators =
300,max_depth=1,subsample=0.8,max_features=0.2,random_state=42)

# fitting the model
gbt.fit(X_train,y_train)

# predicting values
y_pred = gbt.predict(X_test)
print("The test accuracy score of Gradient Boosting Classifier is ", accuracy_score(y_test,
y_pred))
```