Fundamentals of Data Mining – IT3051

Heart Disease Prediction & Visualization

Group Members:

Name	Registration Number
Hiroshan I	IT21034572
Fernando M.T.S	IT21064654
Shifan M.R.M	IT21002274
Vithanage C.V	IT21038396

Table of Contents

Background

Dataset name - UCI Heart Disease Data:

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date.

In data mining, classification is one of the most important ways to label certain information so that it makes it easier for end-users to come to a decision. We are using this dataset to build a classification model that can predict whether a person has heart disease or not, based on the given attributes of that patient. Furthermore, an exploratory data analysis will be performed to diagnose and find out various insights from this dataset which could help in understanding different areas of the problem.

Identified Problem:

The identified problem is the need for an efficient and accurate heart disease prediction system. Early detection and prediction of heart disease is crucial for timely intervention and patient care. The problem includes:

- The lack of a reliable predictive tool for identifying individuals at high risk of heart disease.
- The potential for misdiagnosis or late diagnosis of heart disease.
- The need to improve patient outcomes and reduce healthcare costs through early intervention.

Target Groups:

The primary beneficiaries of this project are healthcare providers and patients. Healthcare providers will use the system for early heart disease detection, while patients will benefit from improved health outcomes and reduced medical costs.

Dataset Link - https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data

Original Dataset Description

Column Name	Description	Data Type
id	Unique id for each patient	Int
age	Age of the patient in years	Int
dataset	Location of data collection	String
sex	Gender (Male/Female)	String
cp	Chest pain type (typical angina, atypical angina, non-anginal, asymptomatic)	String
trestbps	Resting blood pressure (in mm Hg on admission to the hospital)	Float
chol	Serum cholesterol measure in mg/dl	Float
fbs	If fasting blood sugar > 120 mg/dl	Boolean
restecg	Resting electrocardiographic results (normal, stt abnormality, lv hypertrophy)	String
thalach	Maximum heart rate achieved	Float
exang	Exercise-induced angina (True/ False)	String
oldpeak	ST depression induced by exercise relative to rest	Float
slope	The slope of the peak exercise ST segment	String
ca	Number of major vessels (0-3) colored by fluoroscopy	Float
thal	(normal; fixed defect; reversible defect)	String
num	The predicted attribute (Level of severity of the heart disease(0-4))	Int

Methodology

Supervised machine learning approach was used to build and optimize the predictive model. Feature engineering was involved in selecting and transforming variables to improve model performance.

Tools and Technologies

Programming Languages:

Python

Machine Learning Libraries:

Scikit Learn

Tensorflow

Development Environment:

Google Colab

VSCode

Data Visualization:

Matplotlib

Seaborn

Data Analysis Tools:

Pandas

Numpy

Web Application Development:

Streamlit

Version Control:

Git

Scope of Work

1. **Data Collection and Identification:**
   Gathering and initial cleaning of the original dataset.

2. **Exploratory Data Analysis:**
   Exploring the characteristics and impact of features to gain insights from the original dataset.

3. **Data Preprocessing:**
   Treating null values, identifying, and engineering relevant features to enhance prediction accuracy.

4. **Model Development and Evaluation:**
   Building suitable machine learning models and fine-tuning them using cross-validation and evaluation metrics.

5. **Web Application Implementation:**
   Creating a user-friendly interface for healthcare practitioners to access and utilize the predictive model.

1. Data Collection and Identification

1.1 Read Dataset

The total number of rows and columns and the first five records were read and displayed to identify the features and shape of the original dataset.

```
# Read dataset
data = pd.read_csv('input/heart_disease_uci.csv')
display(data.head().style.hide(axis='index'))
print(f"Dataset shape : {data.shape[0]} rows x {data.shape[1]} columns")
```

id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
1	63	Male	Cleveland	typical angina	145.000000	233.000000	True	lv hypertrophy	150.000000	False	2.300000	downsloping	0.000000	fixed defect	0
2	67	Male	Cleveland	asymptomatic	160.000000	286.000000	False	lv hypertrophy	108.000000	True	1.500000	flat	3.000000	normal	2
3	67	Male	Cleveland	asymptomatic	120.000000	229.000000	False	lv hypertrophy	129.000000	True	2.600000	flat	2.000000	reversable defect	1
4	37	Male	Cleveland	non-anginal	130.000000	250.000000	False	normal	187.000000	False	3.500000	downsloping	0.000000	normal	0
5	41	Female	Cleveland	atypical angina	130.000000	204.000000	False	lv hypertrophy	172.000000	False	1.400000	upsloping	0.000000	normal	0

```
Dataset shape : 920 rows x 16 columns
```

1.2 Basic Data Clean-up

The following initial data cleaning tasks were done to make data more consistent and easier to work with in the steps to follow.

- Rename target column to “class.”
- Drop rows if target is null.
- Drop duplicate rows.
- Replace attributes that has space with “_”

```
def basic_clean_data(data):
    data.rename(mapper={'num':'class'}, axis=1, inplace=True) # rename target column to "class"
    data.dropna(subset=['class'], axis=0, inplace=True) # drop rows if target is null
    data.drop_duplicates(keep='first', inplace=True) # drop duplicate rows
    #rename attributes that has space with _
    data["restecg"].replace({"lv hypertrophy": "lv_hypertrophy","st-t abnormality": "stt_abnormality" }, inplace=True)
    data['thal'].replace({'fixed defect':'fixed_defect' , 'reversable defect': 'reversable_defect' }, inplace =True)
    data['cp'].replace({'typical angina':'typical_angina', 'atypical angina': 'atypical_angina' }, inplace =True)
    return data
data = basic_clean_data(data)
display(data.head().style.hide(axis='index'))
print(f"Dataset shape : {data.shape[0]} rows x {data.shape[1]} columns")
```

id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	class
1	63	Male	Cleveland	typical_angina	145.000000	233.000000	True	lv_hypertrophy	150.000000	False	2.300000	downsloping	0.000000	fixed_defect	0
2	67	Male	Cleveland	asymptomatic	160.000000	286.000000	False	lv_hypertrophy	108.000000	True	1.500000	flat	3.000000	normal	2
3	67	Male	Cleveland	asymptomatic	120.000000	229.000000	False	lv_hypertrophy	129.000000	True	2.600000	flat	2.000000	reversable_defect	1
4	37	Male	Cleveland	non-anginal	130.000000	250.000000	False	normal	187.000000	False	3.500000	downsloping	0.000000	normal	0
5	41	Female	Cleveland	atypical_angina	130.000000	204.000000	False	lv_hypertrophy	172.000000	False	1.400000	upsloping	0.000000	normal	0

```
Dataset shape : 920 rows x 16 columns
```

2. Exploratory Data Analysis

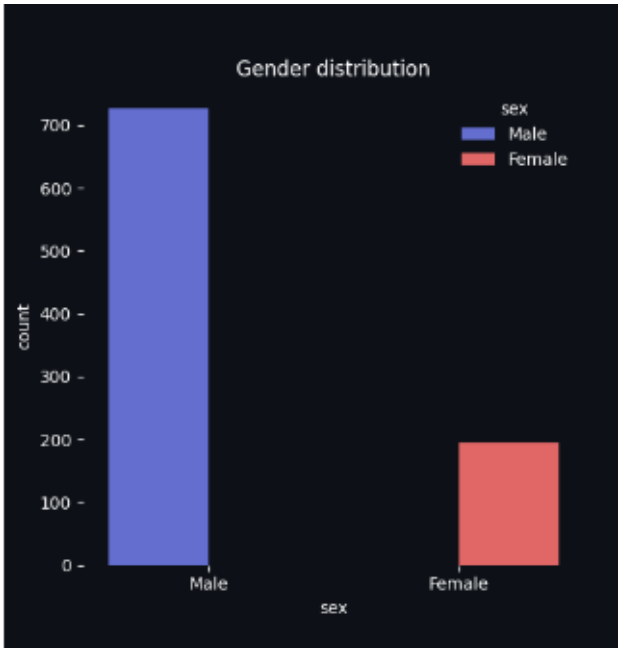
Under exploratory data analysis (EDA), univariate and bivariate data analysis were done to gain insights from the original dataset. Furthermore, the amount of missing data was analyzed feature wise to understand the approach for preprocessing step.

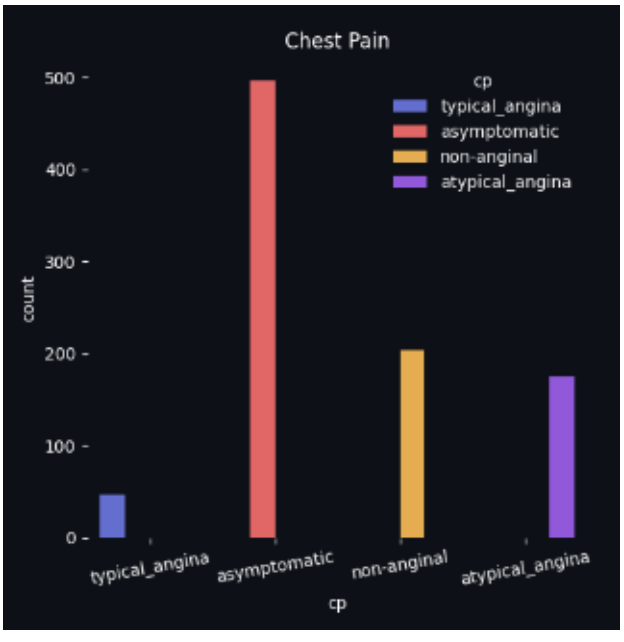
2.1 Univariate Analysis

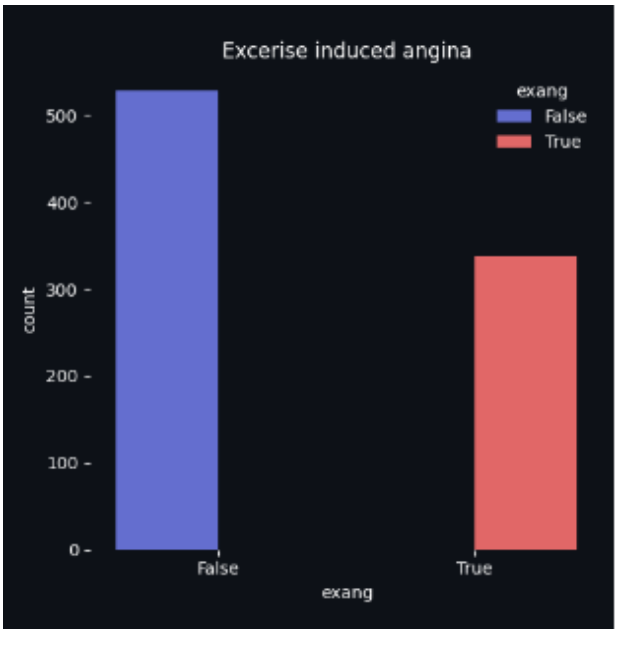
The following techniques were used to visualize the distribution of data using the Seaborn and Matplotlib libraries.

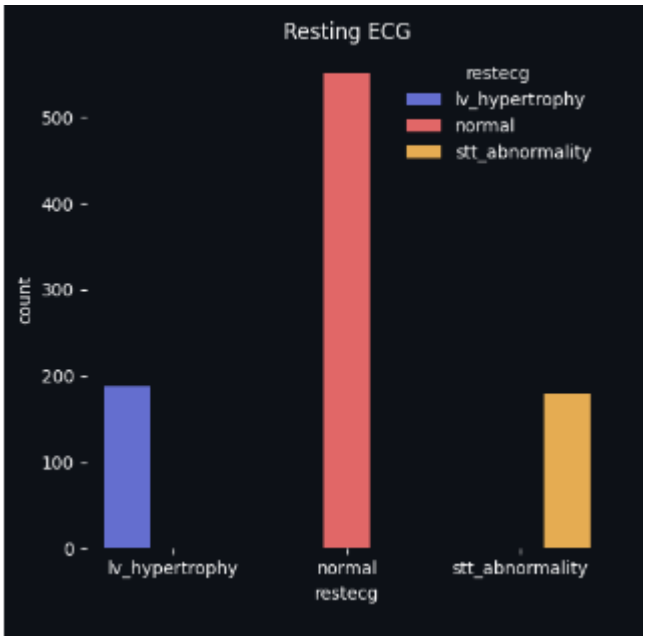
- Bar plots to visualize the distribution of categorical features.
- Box plots to visualize the distribution of numerical features.
- Pie chart to visualize the distribution of target feature.

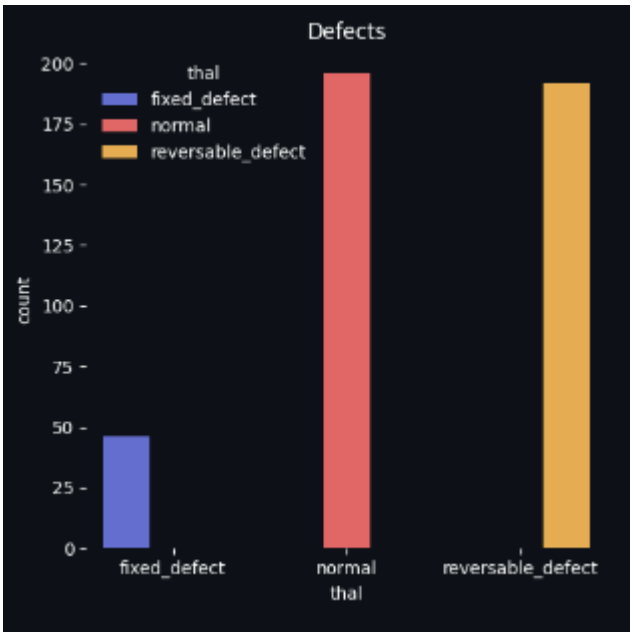
2.1.1 Bar plots

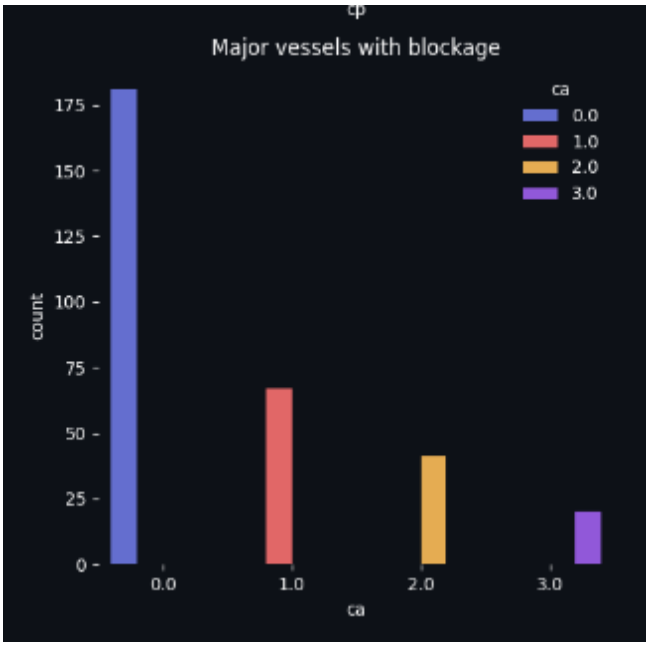


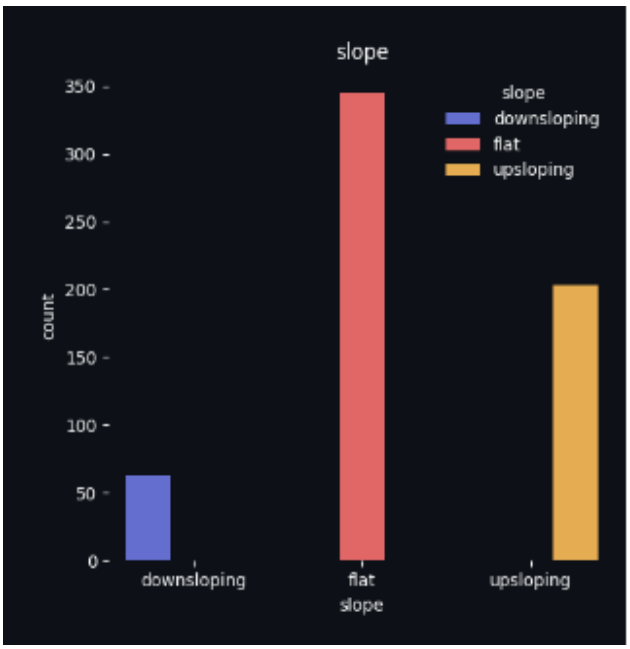




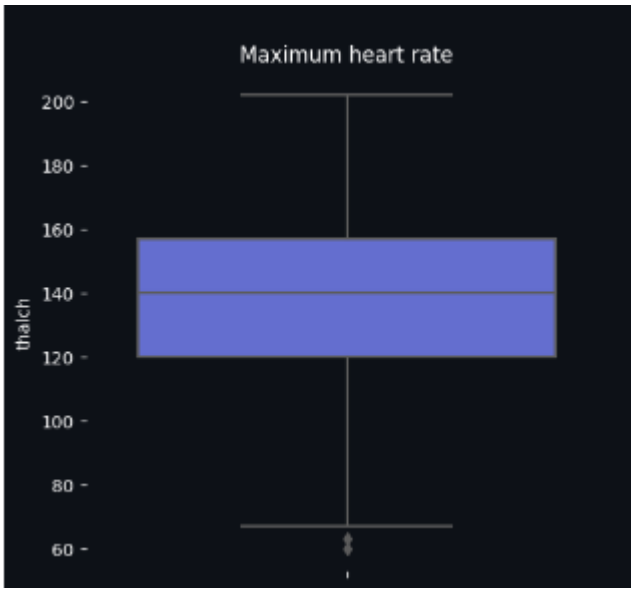

Resting ECG
restecg
lv_hypertrophy
normal
stt_abnormality
count
500
400
300
200
100
0
lv_hypertrophy
normal
stt_abnormality
restecg

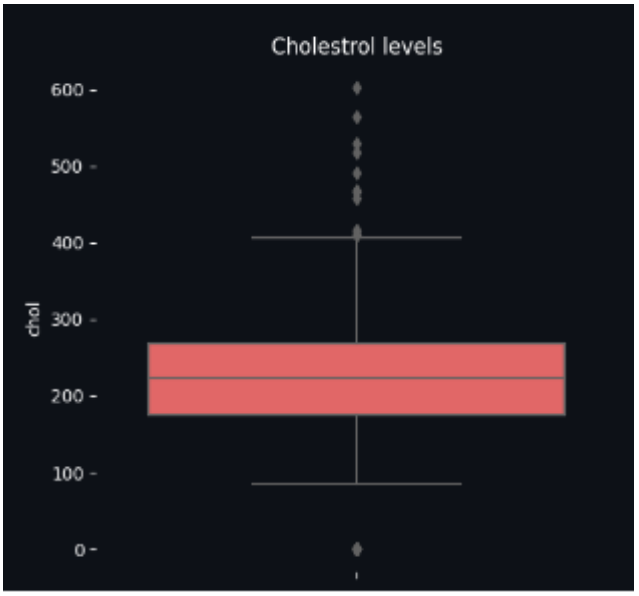

Defects
thal
fixed_defect
normal
reversable_defect
count
200
175
150
125
100
75
50
25
0
fixed_defect
normal
reversable_defect
thal

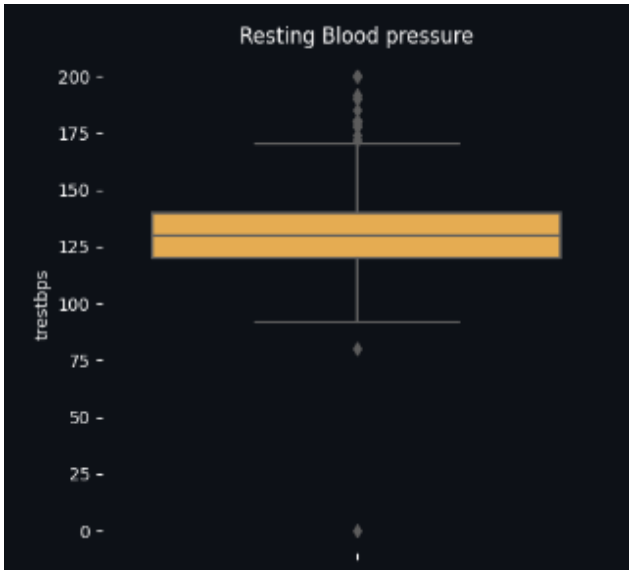

cp
Major vessels with blockage
ca
0.0
1.0
2.0
3.0
count
175
150
125
100
75
50
25
0
0.0
1.0
2.0
3.0
ca

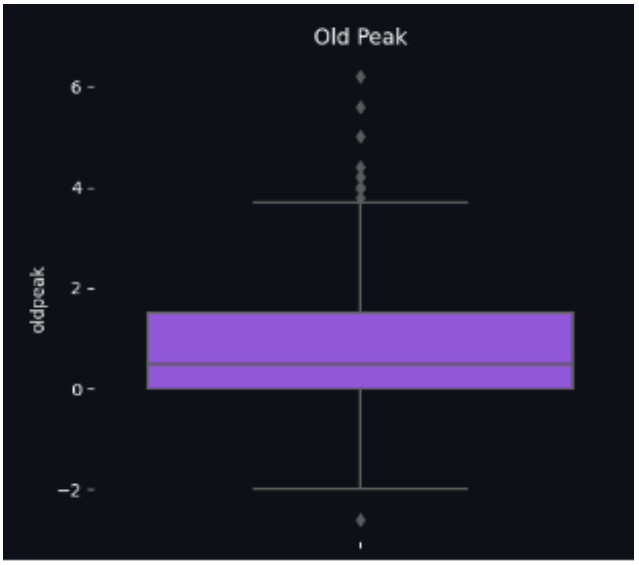

slope
slope
downsloping
flat
upsloping
count
350
300
250
200
150
100
50
0
downsloping
flat
upsloping
slope

2.1.2 Box plots

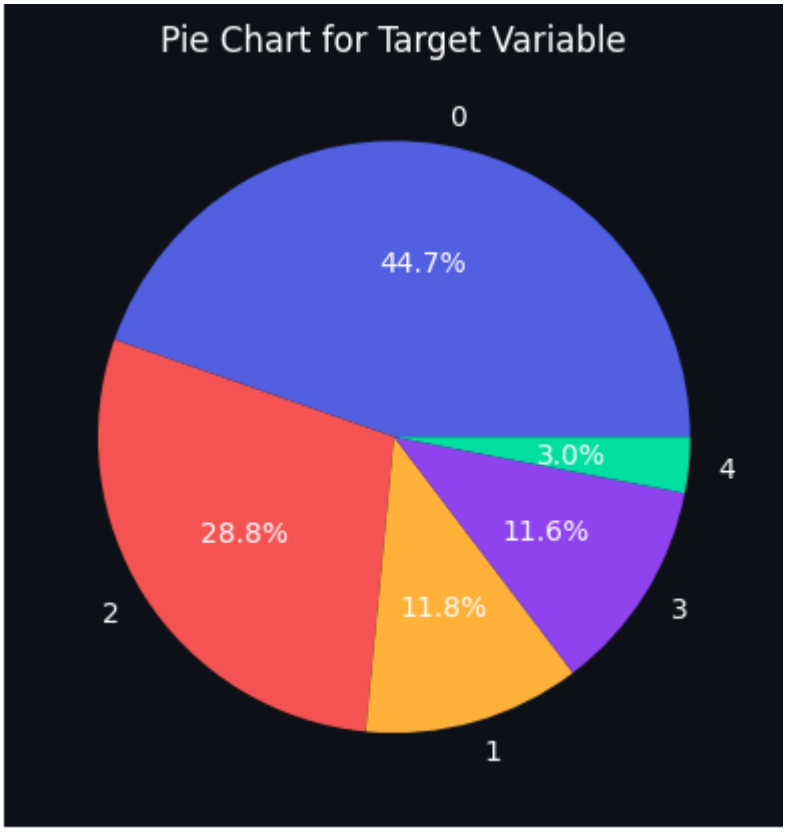








2.1.3 Pie Chart



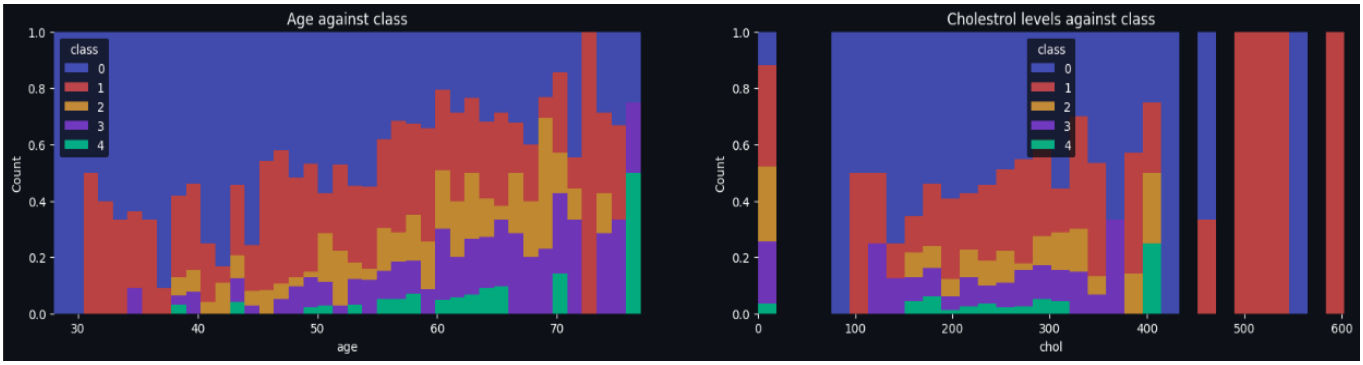
Target feature values range from 0 to 4,0 means no heart disease and 4 means high chances of heart disease. According to the Pie chart, in this dataset, more than half of the people are being diagnosed with some level of heart disease.

2.2 Bivariate Analysis

The following techniques were used to explore how one variable is related to or influenced by another variable in the original dataset.

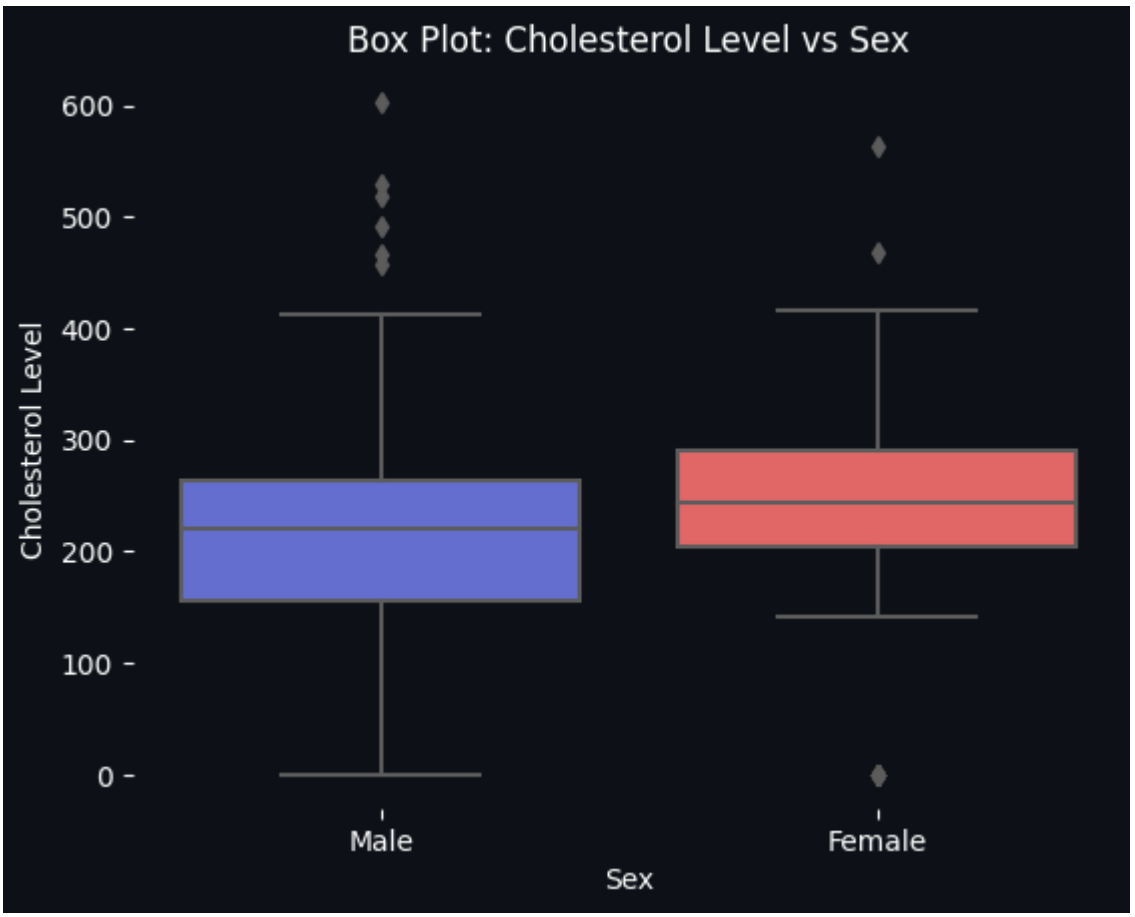
- Histograms visualize the distribution of age and cholesterol level against the target variable.
- Box plots to visualize the distribution of cholesterol level against sex variable.
- Scatter plots to visualize the distribution of maximum heart rate and cholesterol level against the age.

2.2.1 Histograms



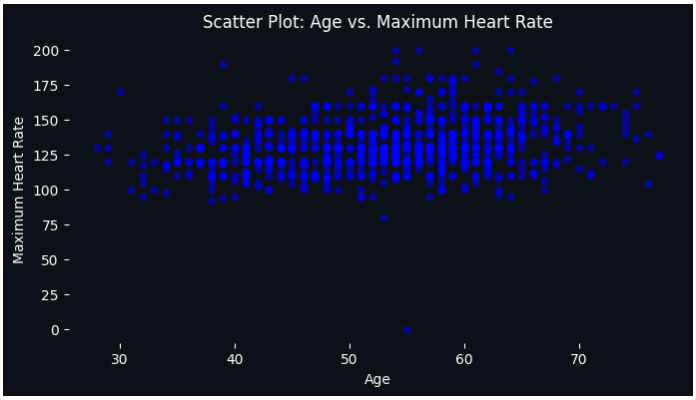
According to the above two histograms, majority of people who are diagnosed with heart disease, are above 50 years of age and have cholesterol level above 300.

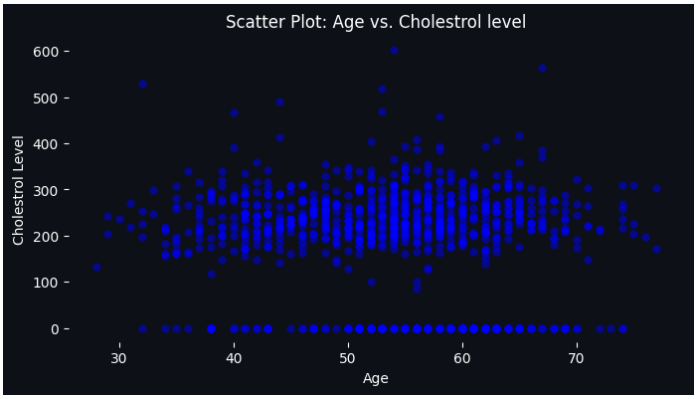
2.2.2 Box Plots



Above figure shows that on average, female have higher level of cholesterol reading as compared to male.

2.2.3 Scatter Plots





Highly dense region of Maximum heart rate and Cholesterol Level is seen for age group 40 to 60 years.

3. Data Preprocessing

The following tasks were done to preprocess the data before building models.

1. Handling missing data
2. Splitting the dataset into training and testing data
3. Data transformation
4. Feature engineering

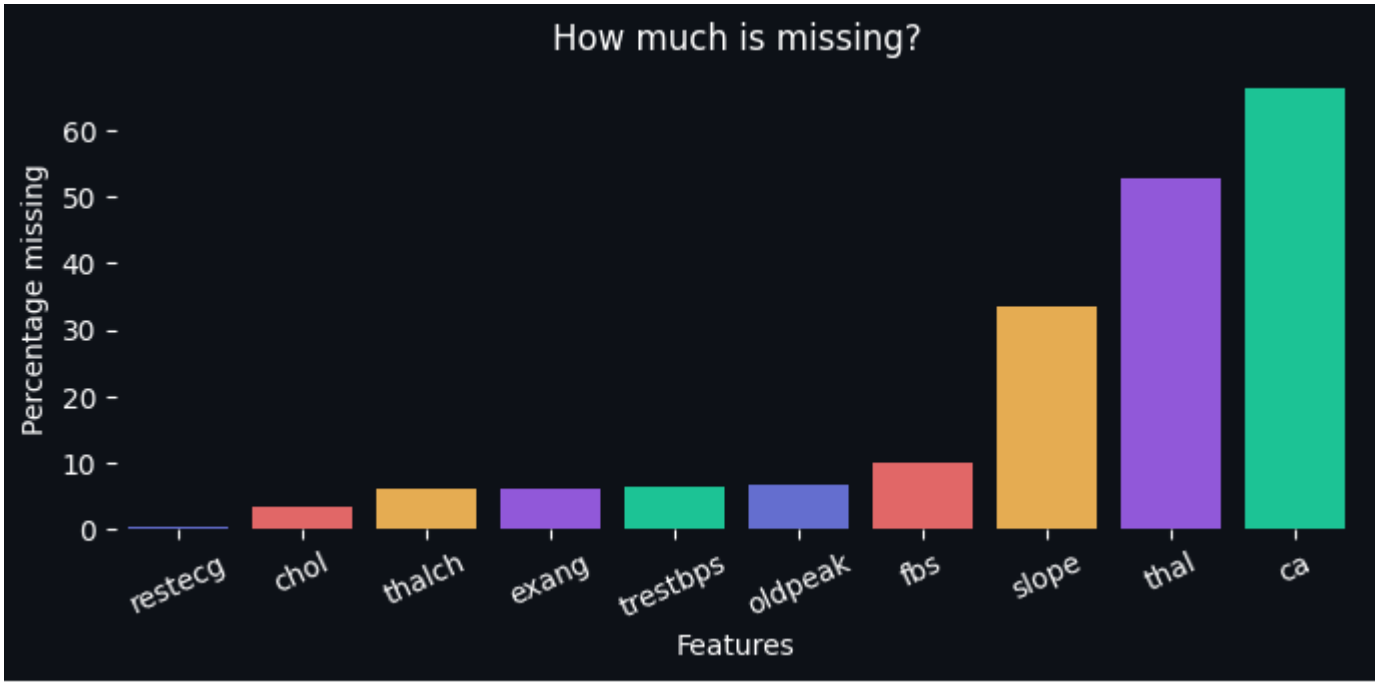
3.1 Handling missing data

```
[14] # Checking for null values

     data.isnull().sum()
```

```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
class         0
dtype: int64
```

The original dataset has several null values. In the following step the distribution of the null values has been analyzed.

```
[ ]  # Missing column names
     missing_val_cols = data.columns[data.isnull().any()]
     # No. of missing values for each column
     missing_vals = data[missing_val_cols].isnull().sum().sort_values()

     plt.figure(figsize=(8,3))
     sns.barplot(x=missing_vals.index, y=round(missing_vals/data.shape[0] * 100, 1), palette=MY_PALETTE)
     plt.xticks(rotation=25)
     plt.title("How much is missing?")
     plt.xlabel("Features")
     plt.ylabel("Percentage missing")
     plt.show()
```



- age column does not have missing values.
- More than half of ca, slope, thal values are missing.
- Other features are missing less than 10% of the values.

Records with null values were taken separately to help identify the impact of each feature on the original dataset.

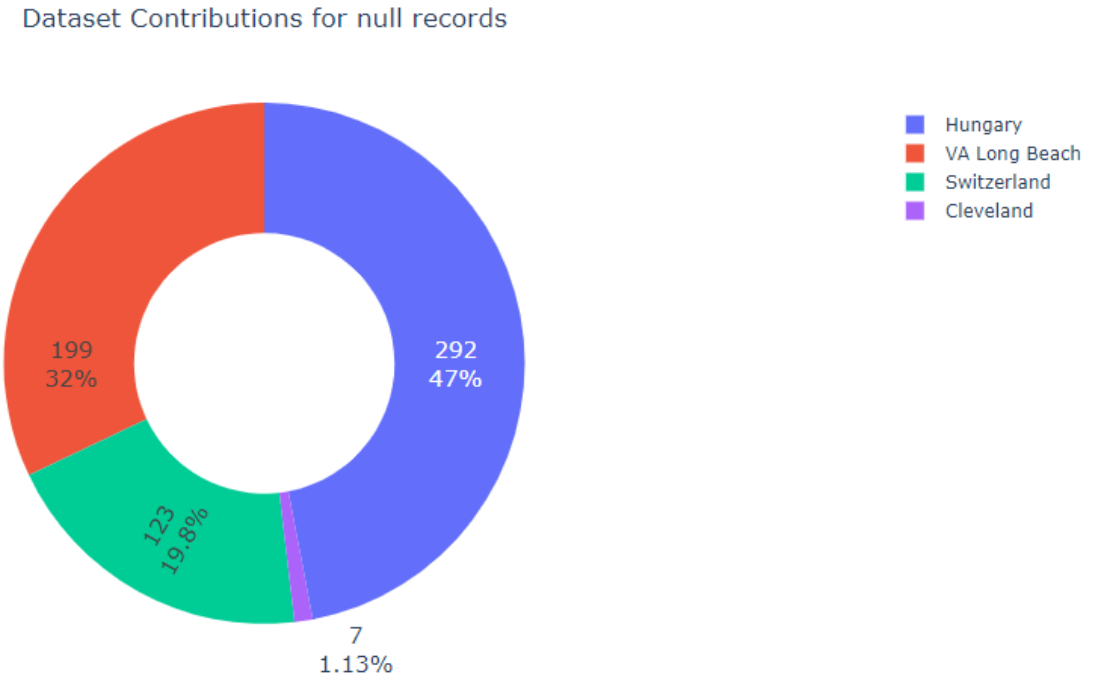
```
#Records with null values
records_with_null = data[data.isnull().any(axis=1)]
records_with_null
```

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	class
87	88	53	Female	Cleveland	non-anginal	128.0	216.0	False	lv_hypertrophy	115.0	False	0.0	upsloping	0.0	NaN	0
166	167	52	Male	Cleveland	non-anginal	138.0	223.0	False	normal	169.0	False	0.0	upsloping	NaN	normal	0
192	193	43	Male	Cleveland	asymptomatic	132.0	247.0	True	lv_hypertrophy	143.0	True	0.1	flat	NaN	reversable_defect	1
266	267	52	Male	Cleveland	asymptomatic	128.0	204.0	True	normal	156.0	True	1.0	flat	0.0	NaN	2
287	288	58	Male	Cleveland	atypical_angina	125.0	220.0	False	normal	144.0	False	0.4	flat	NaN	reversable_defect	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
915	916	54	Female	VA Long Beach	asymptomatic	127.0	333.0	True	stt_abnormality	154.0	False	0.0	NaN	NaN	NaN	1
916	917	62	Male	VA Long Beach	typical_angina	NaN	139.0	False	stt_abnormality	NaN	NaN	NaN	NaN	NaN	NaN	0
917	918	55	Male	VA Long Beach	asymptomatic	122.0	223.0	True	stt_abnormality	100.0	False	0.0	NaN	NaN	fixed_defect	2
918	919	58	Male	VA Long Beach	asymptomatic	NaN	385.0	True	lv_hypertrophy	NaN	NaN	NaN	NaN	NaN	NaN	0
919	920	62	Male	VA Long Beach	atypical_angina	120.0	254.0	False	lv_hypertrophy	93.0	True	0.0	NaN	NaN	NaN	1

621 rows × 16 columns

To identify from which dataset(location) that contains data with the least number of null values, a doughnut chart was plotted.

```
#Dataset(location) Contributions for null records
df=records_with_null['dataset'].value_counts().reset_index().rename(columns={'index':'dataset','dataset':'count'})
fig = go.Figure([go.Pie(labels=df['dataset'],values=df['count'], hole = 0.5)])
fig.update_traces(hoverinfo='label+percent', textinfo='value+percent', textfont_size=15,insidetextorientation='radial')
fig.update_layout(title="Dataset Contributions for null records",title_x=0.5)
fig.show()
```



- According to the doughnut chart, Cleveland dataset has the least impact on missing records.
- Therefore, only data from Cleveland dataset were selected as the dataset for the tasks to follow.

```
X = data[data['dataset'] == "Cleveland"]
X.drop(['dataset'], axis=1, inplace=True)
display(X.head().style.hide(axis='index'))
print(f"Dataset shape : {X.shape[0]} rows x {X.shape[1]} columns")
```

id	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	class
1	63	Male	typical_angina	145.000000	233.000000	True	lv_hypertrophy	150.000000	False	2.300000	downsloping	0.000000	fixed_defect	0
2	67	Male	asymptomatic	160.000000	286.000000	False	lv_hypertrophy	108.000000	True	1.500000	flat	3.000000	normal	2
3	67	Male	asymptomatic	120.000000	229.000000	False	lv_hypertrophy	129.000000	True	2.600000	flat	2.000000	reversable_defect	1
4	37	Male	non-anginal	130.000000	250.000000	False	normal	187.000000	False	3.500000	downsloping	0.000000	normal	0
5	41	Female	atypical_angina	130.000000	204.000000	False	lv_hypertrophy	172.000000	False	1.400000	upsloping	0.000000	normal	0

```
Dataset shape : 304 rows x 15 columns
```

3.2 Data Splitting

The dataset was split into training and testing, this was done early before manipulation to prevent data leakage. Training to testing data ratio was taken as 80:20.

```
[ ]  # Constants
     RANDOM_STATE=42
     NUM_COLS = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']
     CAT_COLS = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
```

```
[ ]  # Separate data into target and features
     y = X['class']
     X = X.drop('class', axis=1)
     X_train, X_valid, y_train, y_valid = train_test_split(X, y,
                                                           train_size=0.8, test_size=0.2,
                                                           random_state=RANDOM_STATE)

     print(tabulate(zip(['Training set', 'Testing set'],[X_train.shape, X_valid.shape]),
                    headers=['Data', '(Rows, Columns)']))
```

```
Data          (Rows, Columns)
------------  -----------------
Training set  (243, 14)
Testing set   (61, 14)
```

Training dataset:

```
#Training data
X_train
```

	id	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal
269	270	42	Male	non-anginal	130.0	180.0	False	normal	150.0	False	0.0	upsloping	0.0	normal
211	212	38	Male	typical_angina	120.0	231.0	False	normal	182.0	True	3.8	flat	0.0	reversable_defect
197	198	45	Female	asymptomatic	138.0	236.0	False	lv_hypertrophy	152.0	True	0.2	flat	0.0	normal
75	76	65	Female	non-anginal	160.0	360.0	False	lv_hypertrophy	151.0	False	0.8	upsloping	0.0	normal
177	178	56	Male	asymptomatic	132.0	184.0	False	lv_hypertrophy	105.0	True	2.1	flat	1.0	fixed_defect
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
188	189	54	Male	atypical_angina	192.0	283.0	False	lv_hypertrophy	195.0	False	0.0	upsloping	1.0	reversable_defect
71	72	67	Male	asymptomatic	125.0	254.0	True	normal	163.0	False	0.2	flat	2.0	reversable_defect
106	107	59	Male	asymptomatic	140.0	177.0	False	normal	162.0	True	0.0	upsloping	1.0	reversable_defect
270	271	61	Male	asymptomatic	140.0	207.0	False	lv_hypertrophy	138.0	True	1.9	upsloping	1.0	reversable_defect
102	103	57	Female	asymptomatic	128.0	303.0	False	lv_hypertrophy	159.0	False	0.0	upsloping	1.0	normal

243 rows × 14 columns

Testing dataset:

```
#Testing data
X_valid
```

	id	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal
180	181	48	Male	asymptomatic	124.0	274.0	False	lv_hypertrophy	166.0	False	0.5	flat	0.0	reversable_defect
154	155	64	Male	asymptomatic	120.0	246.0	False	lv_hypertrophy	96.0	True	2.2	downsloping	1.0	normal
111	112	56	Male	asymptomatic	125.0	249.0	True	lv_hypertrophy	144.0	True	1.2	flat	1.0	normal
247	248	47	Male	asymptomatic	110.0	275.0	False	lv_hypertrophy	118.0	True	1.0	flat	1.0	normal
60	61	51	Female	asymptomatic	130.0	305.0	False	normal	142.0	True	1.2	flat	0.0	reversable_defect
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
218	219	64	Female	asymptomatic	130.0	303.0	False	normal	122.0	False	2.0	flat	2.0	normal
104	105	49	Male	non-anginal	120.0	188.0	False	normal	139.0	False	2.0	flat	3.0	reversable_defect
301	302	57	Female	atypical_angina	130.0	236.0	False	lv_hypertrophy	174.0	False	0.0	flat	1.0	normal
194	195	68	Female	non-anginal	120.0	211.0	False	lv_hypertrophy	115.0	False	1.5	flat	0.0	normal
185	186	63	Female	atypical_angina	140.0	195.0	False	normal	179.0	False	0.0	upsloping	2.0	normal

61 rows × 14 columns

3.3 Data transformation

Categorical and numerical features identified in the previous steps were segregated from the dataset initially to perform data transformation.

```
[40] # Constants
     NUM_COLS = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']
     CAT_COLS = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
```

Then, the following tasks were done to treat null values and enhance the quality of numerical and categorical data.

- Filling missing numerical values with median.

```
imputer = SimpleImputer(strategy="median")
X_train_num = pd.DataFrame(imputer.fit_transform(X_train[NUM_COLS]))
X_valid_num = pd.DataFrame(imputer.transform(X_valid[NUM_COLS]))
X_train_num.head(2)
```

	age	trestbps	chol	thalch	oldpeak
0	42.0	130.0	180.0	150.0	0.0
1	38.0	120.0	231.0	182.0	3.8

- Standardizing numerical values.

```
[ ] num_scaler = RobustScaler()
    num_scaler.fit(X_train_num)
    X_train_num = pd.DataFrame(num_scaler.transform(X_train_num))
    X_valid_num = pd.DataFrame(num_scaler.transform(X_valid_num))
    X_valid_num.columns, X_train_num.columns = NUM_COLS, NUM_COLS # restore column names
    X_train_num.head(2)
```

	age	trestbps	chol	thalch	oldpeak
0	-0.928571	0.00000	-0.984127	-0.057971	-0.500
1	-1.214286	-0.47619	-0.174603	0.869565	1.875

- Filling missing categorical values with mode.

```
imputer = SimpleImputer(strategy="most_frequent")
X_train_cat = pd.DataFrame(imputer.fit_transform(X_train[CAT_COLS]))
X_valid_cat = pd.DataFrame(imputer.transform(X_valid[CAT_COLS]))
X_train_cat.columns, X_valid_cat.columns = CAT_COLS, CAT_COLS # restore column names
X_train_cat.head(2)
```

	sex	cp	fbs	restecg	exang	slope	ca	thal
0	Male	non-anginal	False	normal	False	upsloping	0.0	normal
1	Male	typical_angina	False	normal	True	flat	0.0	reversable_defect

3.4 Feature Engineering

The following tasks were done in the feature engineering step.

1. Numerical and categorical data segregation.
2. Encoding categorical data.
3. Combining numerical and encoded categorical data.
4. Changing the target class to binary.

3.4.1 Numerical and categorical data segregation

Categorical features identified in the previous steps were segregated from the dataset initially to do feature engineering.

- Categorical data in the training dataset after preprocessing:

```
#Categorical data in training dataset after preprocessing
display(X_train_cat.head().style.hide(axis='index'))
print("Number of records with null values: ",X_train_cat.isnull().any(axis=1).sum())
```

sex	cp	fbs	restecg	exang	slope	ca	thal
Male	non-anginal	False	normal	False	upsloping	0.000000	normal
Male	typical_angina	False	normal	True	flat	0.000000	reversable_defect
Female	asymptomatic	False	lv_hypertrophy	True	flat	0.000000	normal
Female	non-anginal	False	lv_hypertrophy	False	upsloping	0.000000	normal
Male	asymptomatic	False	lv_hypertrophy	True	flat	1.000000	fixed_defect

```
Number of records with null values:  0
```

- Categorical data in the testing dataset after preprocessing:

```
[39] #Categorical data in testing dataset after preprocessing
     display(X_valid_cat.head().style.hide(axis='index'))
     print("Number of records with null values: ",X_valid_cat.isnull().any(axis=1).sum())
```

sex	cp	fbs	restecg	exang	slope	ca	thal
Male	asymptomatic	False	lv_hypertrophy	False	flat	0.000000	reversable_defect
Male	asymptomatic	False	lv_hypertrophy	True	downsloping	1.000000	normal
Male	asymptomatic	True	lv_hypertrophy	True	flat	1.000000	normal
Male	asymptomatic	False	lv_hypertrophy	True	flat	1.000000	normal
Female	asymptomatic	False	normal	True	flat	0.000000	reversable_defect

```
Number of records with null values:  0
```

3.4.2 Encoding Categorical Data

There were both nominal and ordinal types of categorical variables.

1) Nominal Type
   i. Sex
   ii. Fasting Blood Sugar(fbs)
   iii. Exercise Induced Angina (exang)

2) Ordinal type
   i. Chest Pain Type (cp)
   ii. Resting Electrocardiographic Results (restecg)

For nominal features, one-hot encoding was used and for ordinal features, ordinal encoding was used.

3.4.2.1 One hot encoding

```
[43] #One hot encoding for nominal features
     oh_encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
     X_train_cat_ohenc = pd.DataFrame(oh_encoder.fit_transform(X_train_cat))
     X_valid_cat_ohenc = pd.DataFrame(oh_encoder.transform(X_valid_cat))
     X_train_cat_ohenc.head()
```

	0	1	2	3	4	5	6	7	8	9	...	13	14	15	16	17	18	19	20	21	22
0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
2	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

5 rows × 23 columns

3.4.2.2 Ordinal encoding

```
#Ordinal encoding for ordinal features
or_encoder = OrdinalEncoder()
X_train_cat_orenc = pd.DataFrame(or_encoder.fit_transform(X_train_cat))
X_valid_cat_orenc = pd.DataFrame(or_encoder.transform(X_valid_cat))
X_train_cat_orenc.head()
```

	0	1	2	3	4	5	6	7
0	1.0	2.0	0.0	1.0	0.0	2.0	0.0	1.0
1	1.0	3.0	0.0	1.0	1.0	1.0	0.0	2.0
2	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0
3	0.0	2.0	0.0	0.0	0.0	2.0	0.0	1.0
4	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0

Then, numerical, and categorical features were concatenated.

3.4.3 Numerical and Categorical data concatenation

```
[ ] # Concatenate column-wise
    X_train_final = pd.concat([X_train_num, X_train_cat_ohenc], axis=1)
    X_valid_final = pd.concat([X_valid_num, X_valid_cat_ohenc], axis=1)
    # Change column name type to string
    X_train_final.columns, X_valid_final.columns = X_train_final.columns.astype(str),  X_valid_final.columns.astype(str)
    print(f"Training set shape : {X_train_final.shape}")
    print(f"Testing set shape : {X_valid_final.shape}")
```

```
Training set shape : (243, 28)
Testing set shape : (61, 28)
```

After combining numerical and categorical data, the original vs. preprocessed dataset looked like this:

```
[ ] print(f"Original ➡ {X_train.shape}")
    display(X_train.head(2))
    print(f"Preprocessed ➡ {X_train_final.shape}")
    display(X_train_final.head(2))
```

Original ➡ (243, 14)

	id	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal
269	270	42	Male	non-anginal	130.0	180.0	False	normal	150.0	False	0.0	upsloping	0.0	normal
211	212	38	Male	typical angina	120.0	231.0	False	normal	182.0	True	3.8	flat	0.0	reversable defect

Preprocessed ➡ (243, 28)

	age	trestbps	chol	thalch	oldpeak	0	1	2	3	4	...	13	14	15	16	17	18	19	20	21	22
0	-0.928571	0.00000	-0.984127	-0.057971	-0.500	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
1	-1.214286	-0.47619	-0.174603	0.869565	1.875	0.0	1.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0

2 rows × 28 columns

3.4.4 Changing target class to binary

The original dataset contains the target as 0, 1, 2, 3, 4. But for simply identifying the presence of disease, the target data in the class column were changed into 1/0 to do binary classification.

```
y_train_binary = pd.Series([int(y != 0) for y in y_train], name="Heart Disease")
y_valid_binary = pd.Series([int(y != 0) for y in y_valid], name="Heart Disease")
y_train_binary
```

```
0      0
1      1
2      0
3      0
4      1
      ..
238    1
239    1
240    1
241    1
242    0
Name: Heart Disease, Length: 243, dtype: int64
```

4. Model Development and Evaluation

The following steps were taken to develop, train and evaluate models and finally identify which models have the highest accuracy.

- Building and training the below classification models.

    1) Random Forest Classifier
    2) Logistic Regression
    3) Artificial Neaural Network (ANN)
    4) Support Vector Classifier (SVC)

- Evaluating each model on both the training and validation datasets using the output of 'classification_report' function from the scikit-learn library.

- The classification report provides a detailed summary of the model's performance, which includes the following evaluation metrics.

    - Precision: Measures the accuracy of positive predictions made by the model. It is calculated as:
      Precision = True Positives / (True Positives + False Positives)
        - True Positives (TP): The number of correctly predicted positive instances.
        - False Positives (FP): The number of negative instances incorrectly predicted as positive.

    - Recall: Measures the model's ability to identify all relevant positive instances. It is calculated as:
      Recall = True Positives / (True Positives + False Negatives)
        - True Negatives (TN): The number of correctly predicted negative instances.
        - False Negatives (FN): The number of positive instances incorrectly predicted as negative.

    - F1 Score: Harmonic mean of precision and recall, providing a balance between these two metrics. It is calculated as:
      Recall = 2.Precision.Recall / (Precision + Recall)

    - Support: Represents the number of instances in the true class. It's a helpful context metric for the other three metrics, providing information about the relative size of each class in the dataset. A class with a higher support may have more influence on overall performance metrics.

- Taking insights from above evaluation metrics was helpful in understanding how well the model is classifying instances in the validation dataset and its ability to correctly predict the presence of heart disease.

- Finally, to identify which models have the highest accuracy all the models were compared.

4.1 Random Forest Classifier

- Training the Random Forest Classifier:

```
[58] #Training the Random Forest Classifier
     rfc_model = RandomForestClassifier(n_estimators=100, random_state=RANDOM_STATE)
     rfc_model.fit(X_train_final, y_train_binary);
```

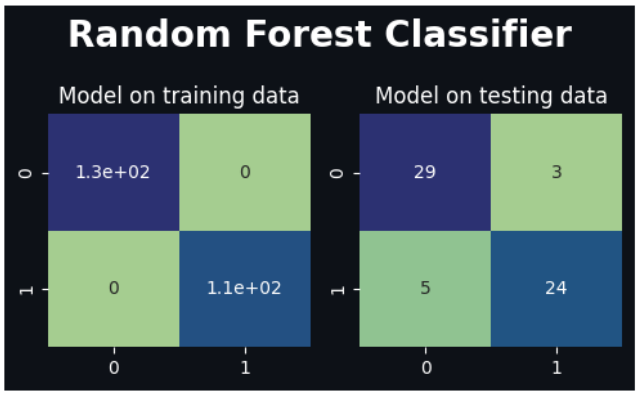
- Evaluating the Random Forest Classifier (RFC) model on both the training and validation datasets:

```
#Evaluating the Random Forest Classifier (RFC) model on both the training and validation datasets

#For the training data
rfc_train_preds = rfc_model.predict(X_train_final)

#For the testing data
rfc_test_preds = rfc_model.predict(X_valid_final)

#Print classification report for the model's predictions on the validation dataset
print(classification_report(y_valid_binary, rfc_test_preds))
```

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88        32
           1       0.89      0.83      0.86        29

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61
```

- Getting the confusion matrix for both training and validation data:

```
fig, axes = plt.subplots(1, 2, figsize=(5,3))
fig.suptitle("Random Forest Classifier")
#Confusion matrix for the training data
axes[0].set_title("Model on training data")
sns.heatmap(confusion_matrix(y_train_binary, rfc_train_preds), annot=True, cmap='crest', cbar=False, ax=axes[0])
#Confusion matrix for the testing data
axes[1].set_title("Model on testing data")
sns.heatmap(confusion_matrix(y_valid_binary, rfc_test_preds), annot=True, cmap='crest', cbar=False, ax=axes[1])
plt.tight_layout()
plt.show()
```



4.2 Logistic Regression

- Training the Logistic Regression model:

```
[62] #Training the Logistic Regression Model
     log_model = LogisticRegression(random_state=RANDOM_STATE)
     log_model.fit(X_train_final, y_train_binary);
```

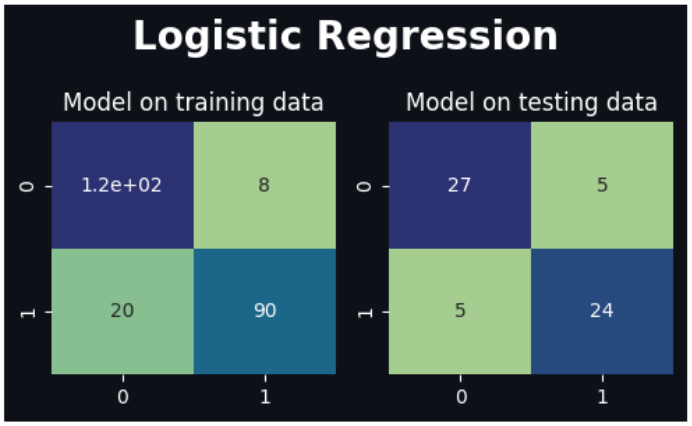
- Evaluating the Logistic Regression model on both the training and validation datasets:

```
[63] #Evaluating the Logistic Regression model on both the training and validation datasets

     #For the training data
     lr_train_preds = log_model.predict(X_train_final)

     #For the testing data
     lr_test_preds = log_model.predict(X_valid_final)

     #Print classification report for the model's predictions on the validation dataset
     print(classification_report(y_valid_binary, lr_test_preds))
```

```
              precision    recall  f1-score   support

           0       0.84      0.84      0.84        32
           1       0.83      0.83      0.83        29

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

- Getting the confusion matrix for both training and validation data:

```
fig, axes = plt.subplots(1, 2, figsize=(5,3))
fig.suptitle("Logistic Regression")
#Confusion matrix for the training data
axes[0].set_title("Model on training data")
sns.heatmap(confusion_matrix(y_train_binary, lr_train_preds), annot=True, cmap='crest', cbar=False, ax=axes[0])
#Confusion matrix for the testing data
axes[1].set_title("Model on testing data")
sns.heatmap(confusion_matrix(y_valid_binary, lr_test_preds), annot=True, cmap='crest', cbar=False, ax=axes[1])
plt.tight_layout()
plt.show()
```



4.3 Artificial Neural Network (ANN)

- Training the ANN model:

```
[57] #Training the ANN Model
     ann_model = keras.Sequential([
         keras.layers.Dense(28, input_shape=(28,), activation='relu'),
         keras.layers.Dense(26, activation='relu'),
         keras.layers.Dense(1, activation='sigmoid')
     ])

     ann_model.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics=['accuracy'])

     history = ann_model.fit(X_train_final, y_train_binary,
                             validation_data=(X_valid_final, y_valid_binary),
                             epochs=200, verbose=False);
```

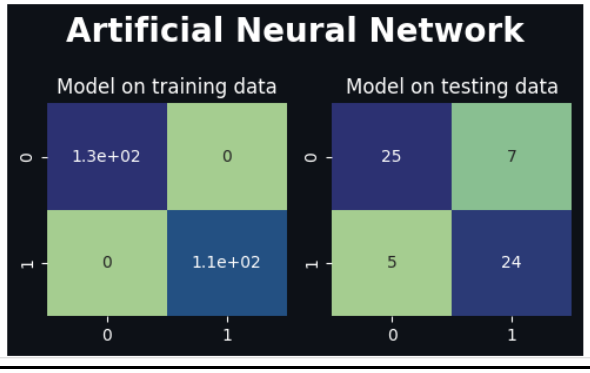
- Evaluating the ANN model on both the training and validation datasets:

```
[72] #Evaluating the ANN model on both the training and validation datasets

     #For the training data
     ann_train_preds = ann_model.predict(X_train_final, verbose=False)

     #For the testing data
     ann_test_preds = ann_model.predict(X_valid_final, verbose=False)

     #Convert continuous predictions to binary predictions
     ann_test_preds_binary = (ann_test_preds >= 0.5).astype(int)
     ann_train_preds_binary = (ann_train_preds >= 0.5).astype(int)

     print(classification_report(y_valid_binary, [int(y>0.5) for y in ann_test_preds]))
```

```
                precision    recall  f1-score   support

           0       0.83      0.78      0.81        32
           1       0.77      0.83      0.80        29

    accuracy                           0.80        61
   macro avg       0.80      0.80      0.80        61
weighted avg       0.81      0.80      0.80        61
```

- Getting the confusion matrix for both training and validation data:

```
fig, axes = plt.subplots(1, 2, figsize=(5, 3))
fig.suptitle("Artificial Neural Network")
#Confusion matrix for the training data
axes[0].set_title("Model on training data")
sns.heatmap(confusion_matrix(y_train_binary, ann_train_preds_binary), annot=True, cmap='crest', cbar=False, ax=axes[0])
#Confusion matrix for the testing data
axes[1].set_title("Model on testing data")
sns.heatmap(confusion_matrix(y_valid_binary, ann_test_preds_binary), annot=True, cmap='crest', cbar=False, ax=axes[1])
plt.tight_layout()
plt.show()
```



4.4 Support Vector Classification (SVC)

- Training the SVC model:

```
[74] #Training the SVC Model
     svc_model = SVC(kernel='linear', C=0.01, probability=True)
     svc_model.fit(X_train_final, y_train_binary);
```

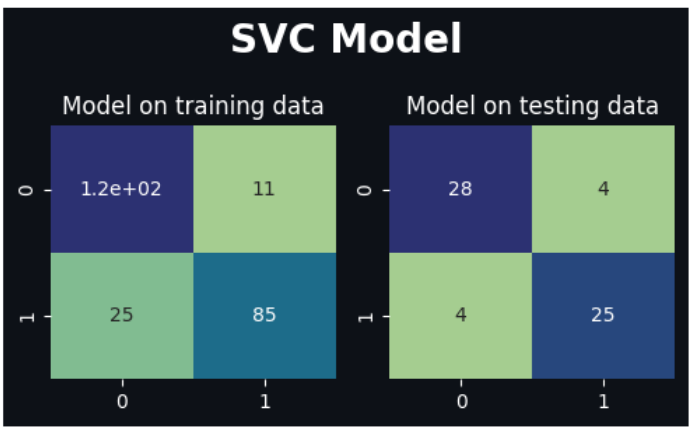
- Evaluating the SVC model on both the training and validation datasets:

```
#Evaluating the SVC model on both the training and validation datasets

#For the training data
svc_train_preds = svc_model.predict(X_train_final)

#For the testing data
svc_test_preds = svc_model.predict(X_valid_final)
print(classification_report(y_valid_binary, svc_test_preds))
```

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88        32
           1       0.86      0.86      0.86        29

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61
```

- Getting the confusion matrix for both training and validation data:

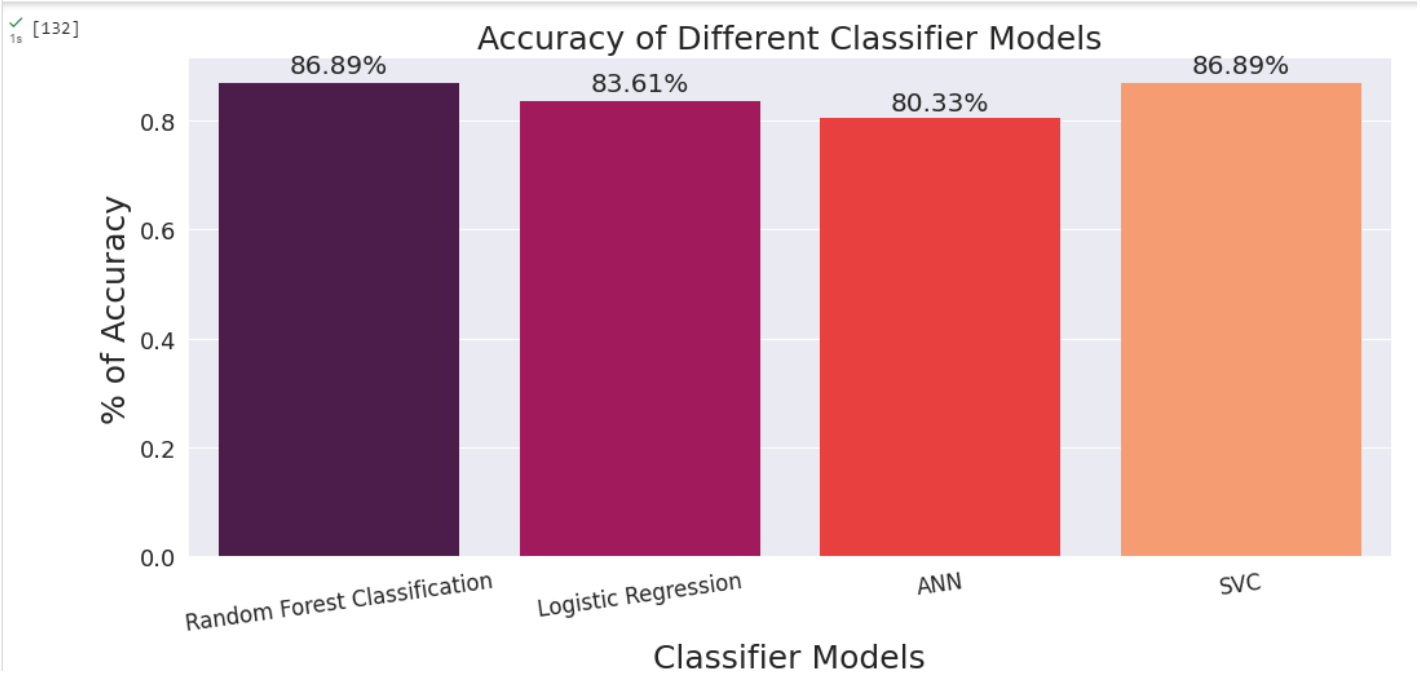
```
fig, axes = plt.subplots(1, 2, figsize=(5,3))
fig.suptitle("SVC Model")
#Confusion matrix for the training data
axes[0].set_title("Model on training data")
sns.heatmap(confusion_matrix(y_train_binary, svc_train_preds), annot=True, cmap='crest', cbar=False, ax=axes[0])
#Confusion matrix for the testing data
axes[1].set_title("Model on testing data")
sns.heatmap(confusion_matrix(y_valid_binary, svc_test_preds), annot=True, cmap='crest', cbar=False, ax=axes[1])
plt.tight_layout()
plt.show()
```



4.5 Model Comparison

After evaluating each model, the models were compared with each other to find out the models with the highest accuracy.

```
[132] plt.rcParams['figure.figsize']=12,5
      sns.set_style("darkgrid")
      ax = sns.barplot(x=model_list, y=acc_scores, palette = "rocket", saturation =1.5)
      plt.xlabel("Classifier Models", fontsize = 18 )
      plt.ylabel("% of Accuracy", fontsize = 18)
      plt.title("Accuracy of Different Classifier Models", fontsize = 18)
      plt.xticks(fontsize = 12, horizontalalignment = 'center', rotation = 8)
      plt.yticks(fontsize = 13)
      for p in ax.patches:
          width, height = p.get_width(), p.get_height()
          x, y = p.get_xy()
          ax.annotate(f'{height:.2%}', (x + width/2, y + height*1.02), ha='center', fontsize = 'x-large')
      plt.show()
```

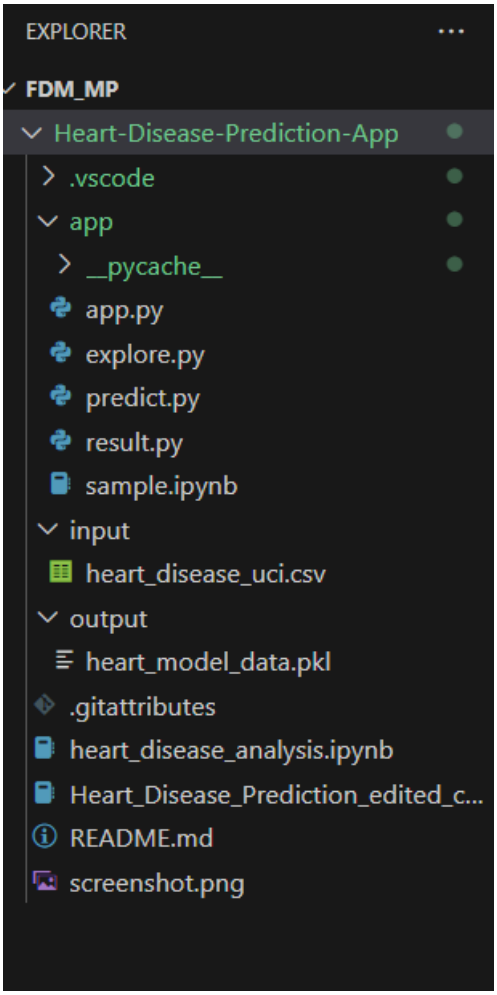


- Random Forest classifier and SVC models had the highest accuracy.

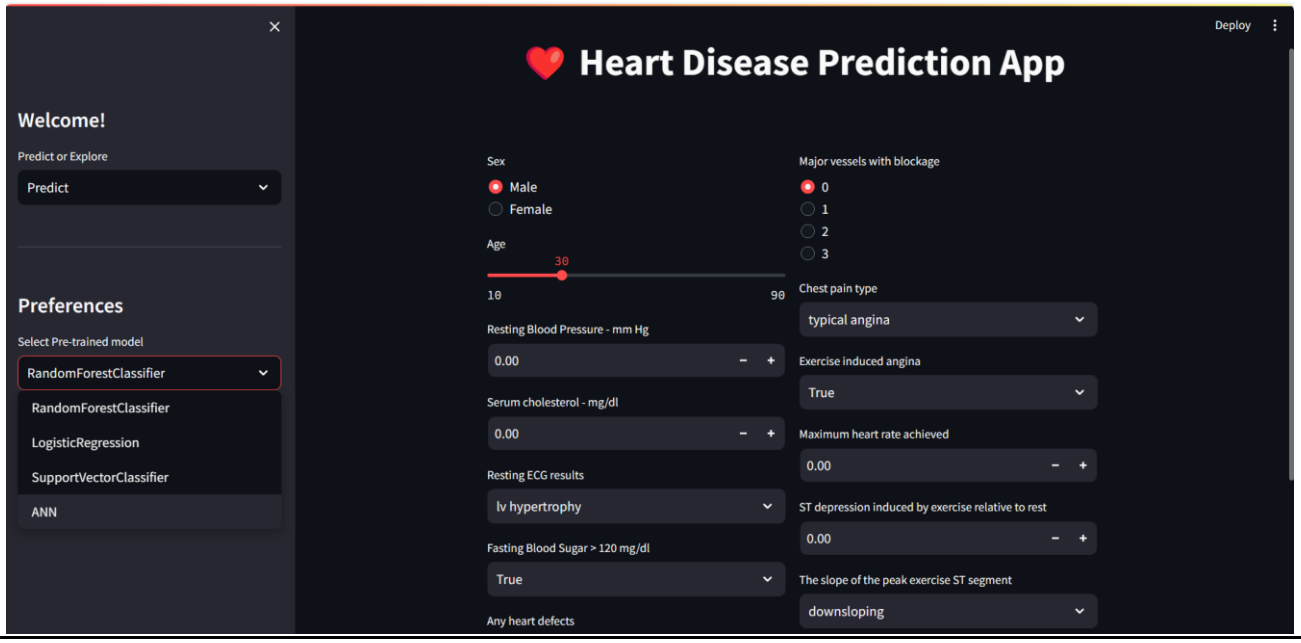
5. Web Application Implementation

Streamlit which is an open-source python framework was used to build the front end and to demonstrate the visualizations, user inputs and integrations with the developed models.

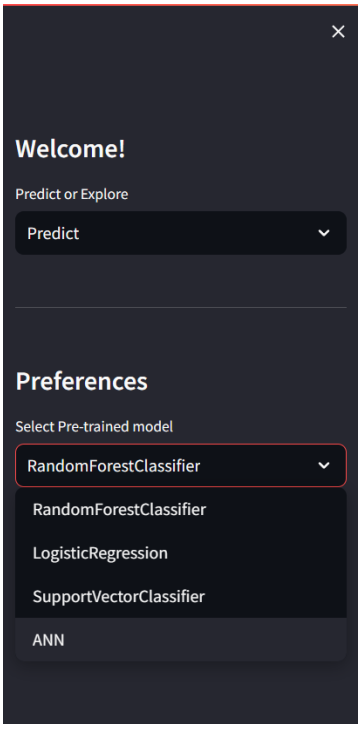
- Project Structure:



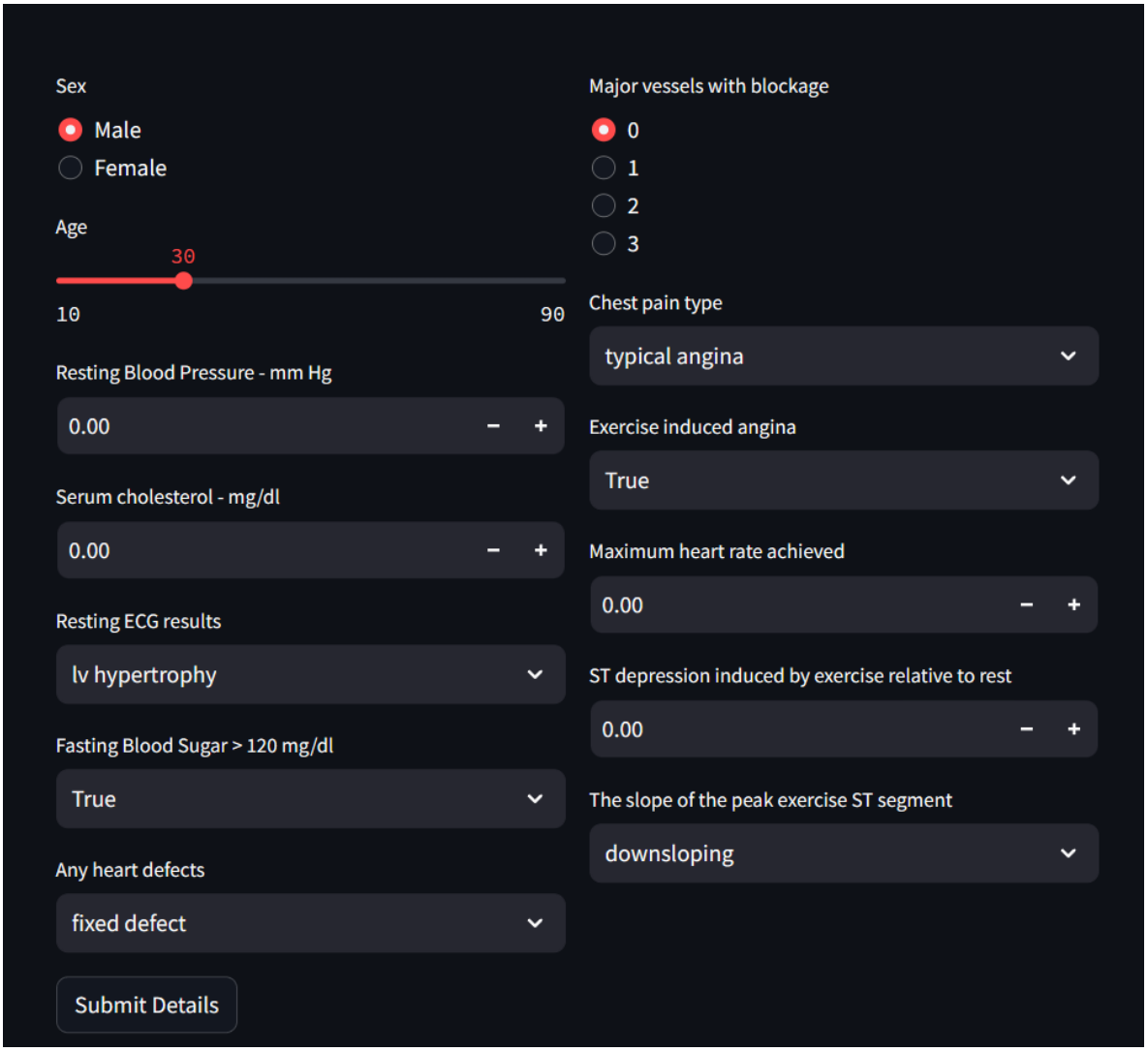
- UI implementation:



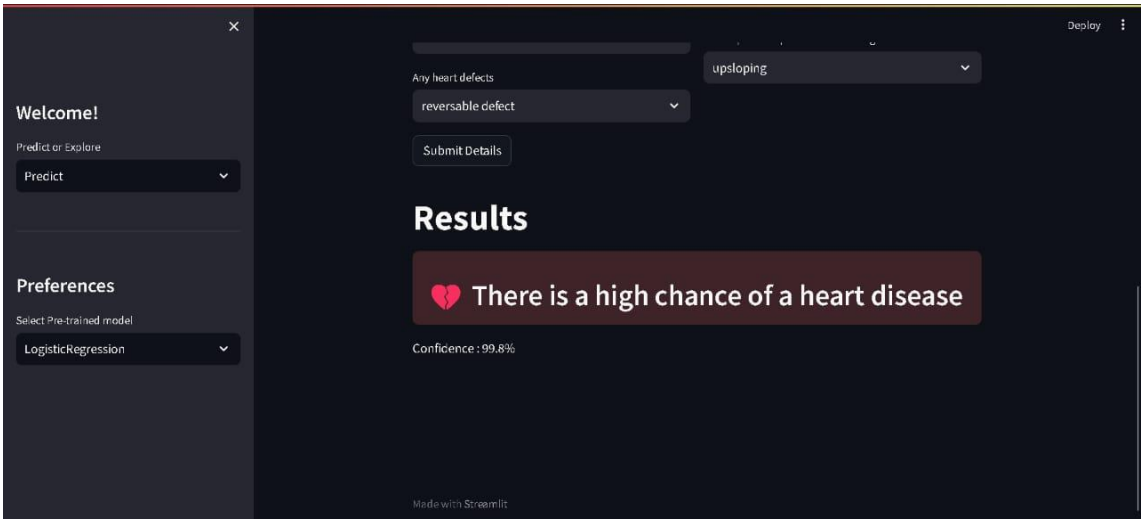
- Menu Panel:

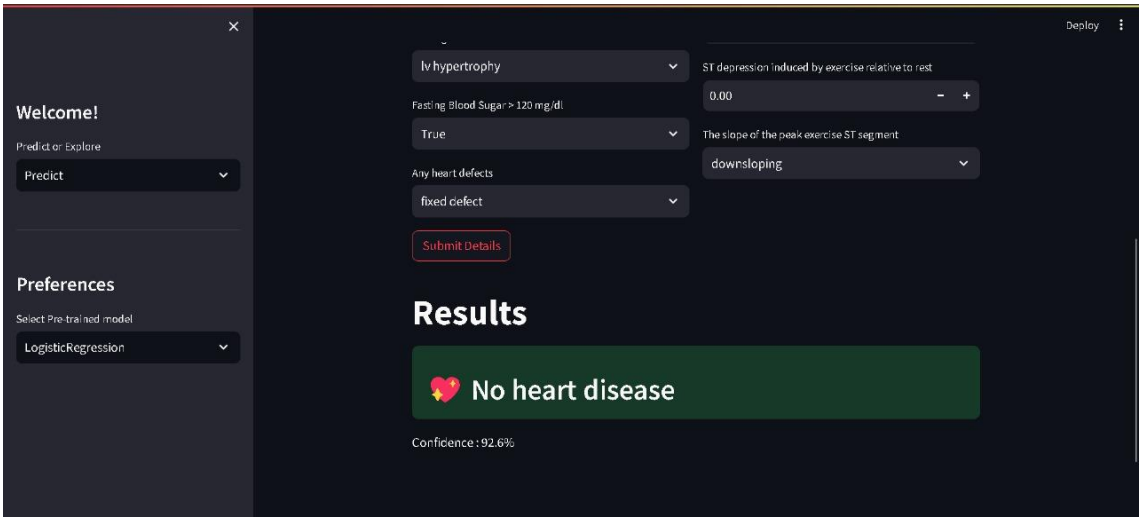


- Input Fields:



- Prediction Results:





Project Team and Workload

Name	Registration Number	Responsibilities
Fernando M.T.S	IT21064654	• Data Preprocessing • Develop and evaluate Logistic Regression model • Documentation
Hiroshan I	IT21034572	• EDA • Web Application development • Develop and evaluate Random Forest model
Shifan M.R.M	IT21002274	• Develop and evaluate ANN model
Vithanage C.V	IT21038396	• Develop and evaluate SVC model

References

[1] "Machine Learning Algorithms" [Online].
Available: https://www.geeksforgeeks.org/machine-learning-algorithms/

[2] "Data Preprocessing in Machine Learning: 7 Easy Steps To Follow," [Online].

Available: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/

[3] “Artificial Neural Network Tutorial" [Online].

Available: https://www.javatpoint.com/artificial-neural-network

[4] J. Huneycutt, "medium," [Online].

Available: https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652.

[5] "scikit-learn," [Online].

Available: https://scikit-learn.org/stable/modules/tree.html.