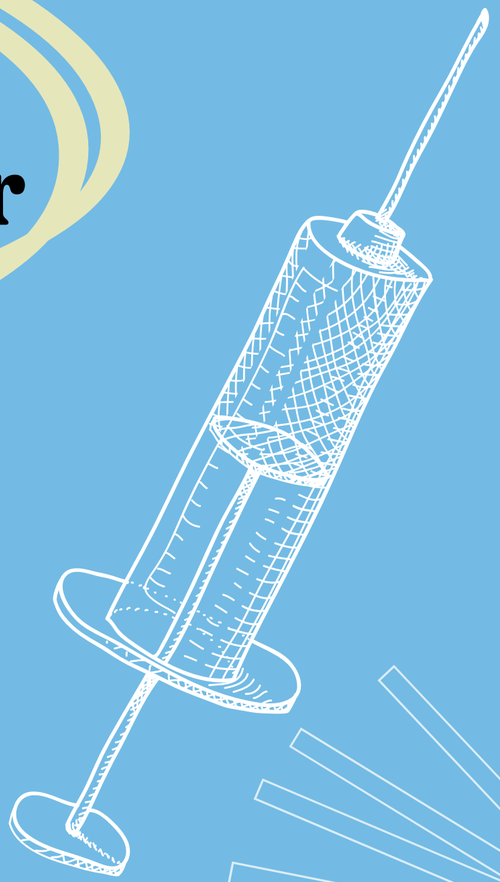
Early Detection, Better Outcomes: Machine Learning In Cardiovascular Care



PRESENTED BY

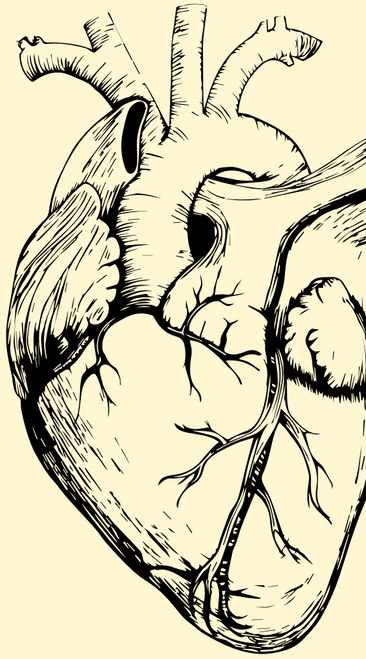
ARSHIYA

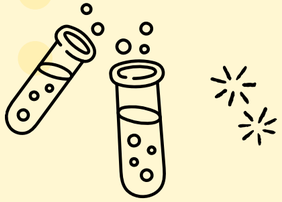
&

MEHAR

Content

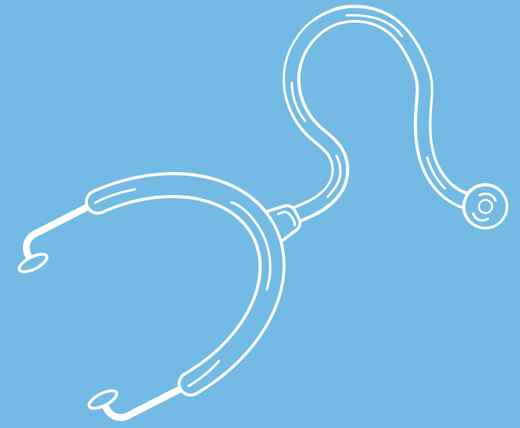
- Problem Statement
- Statistical Summary
- Basic EDA
- Cleaning Data
- Feature Selection
- Modelling
- Conclusion





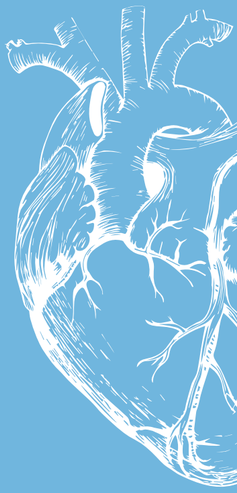
Problem Statement

Developing a Personalised Risk Assessment for Early Detection of Heart Disease



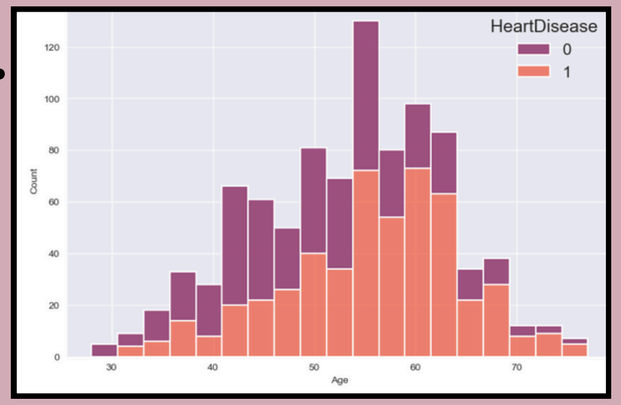
Statistical Summary

- The dataset that we have used is Heart Failure Prediction Dataset from Kaggle
- There are 918 observations and 12 fields in the dataset.
- The response variable is **HeartDisease.**
- There are 6 categorical data out of which HeartDisease and FastingBS are numeric in nature.
- There are 4 numeric variables.

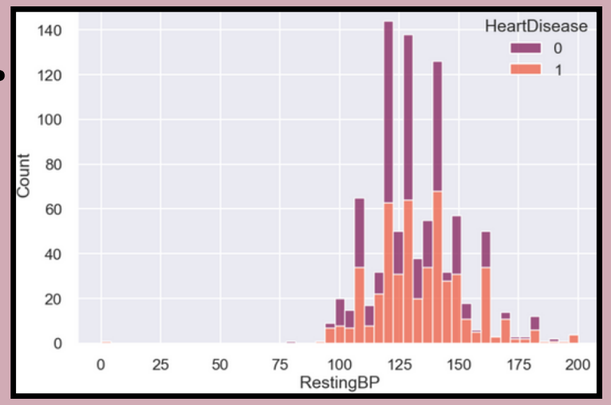


Bi-Variate EDA

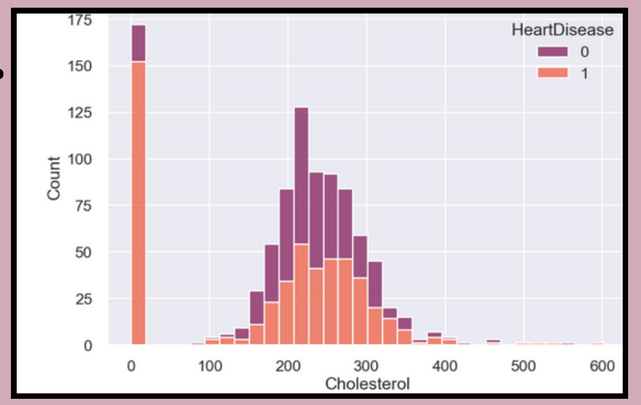
Numerical Analysis with Heart Disease

1. 

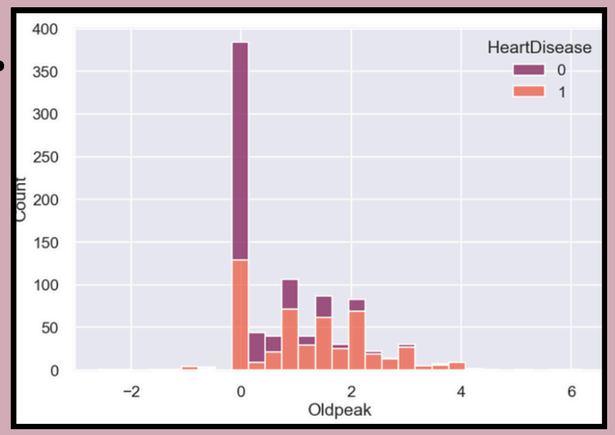
Age and Heart Disease

2. 

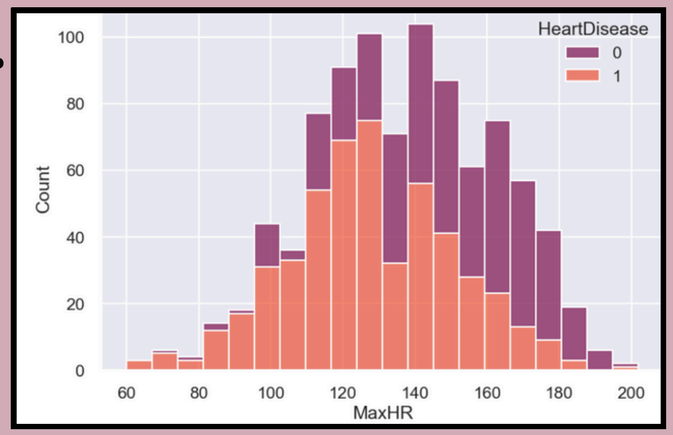
Resting BP and Heart Disease

3. 

Cholestrol and Heart Disease

4. 

OldPeak and Heart Disease

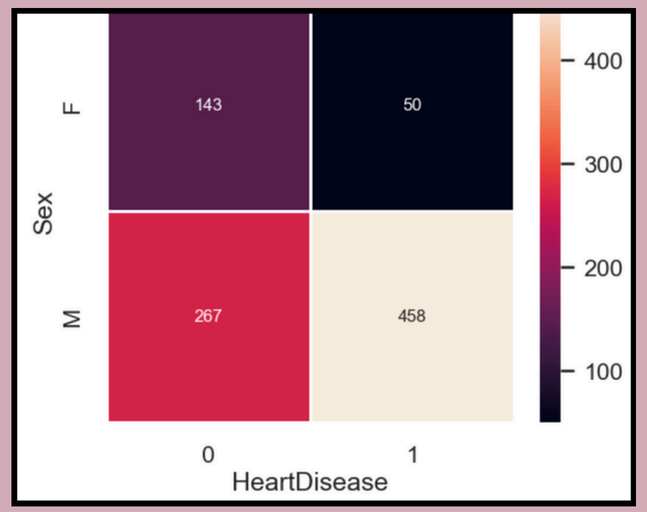
5. 

MaxHR and Heart Disease

Bi-Variate EDA

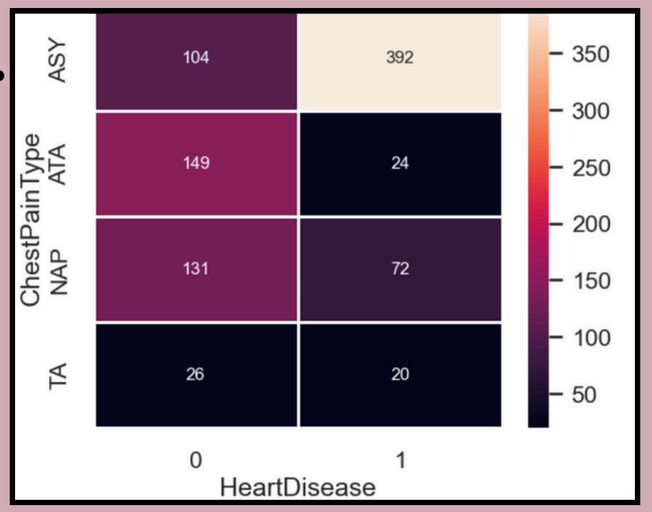
Categorical Analysis with Heart Disease

1.



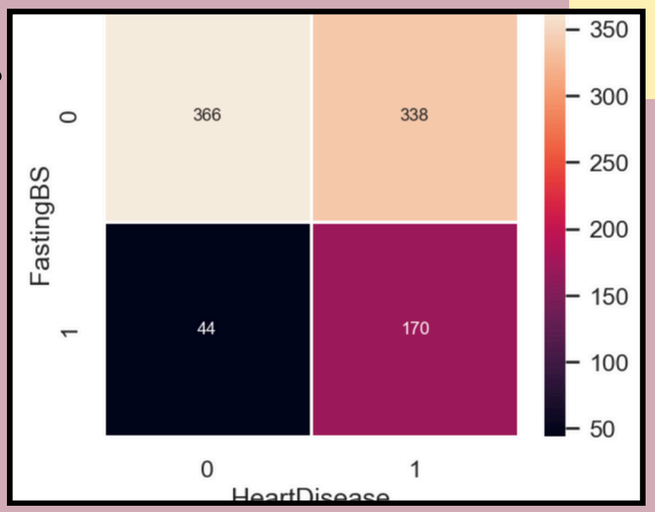
Sex and Heart Disease

2.



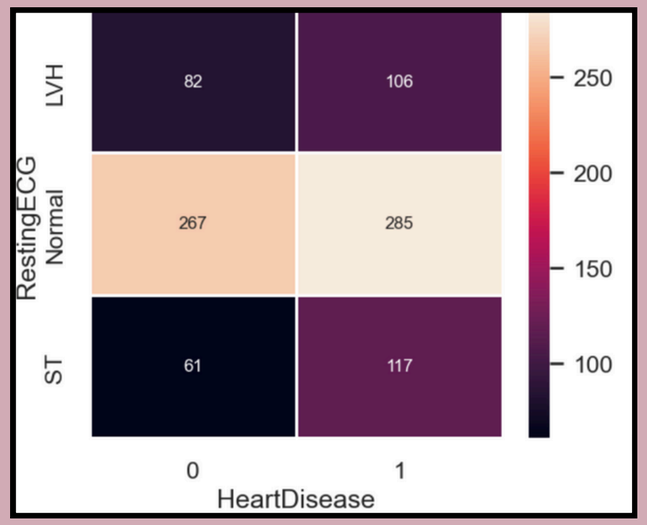
Chest Pain Type and Heart Disease

3.



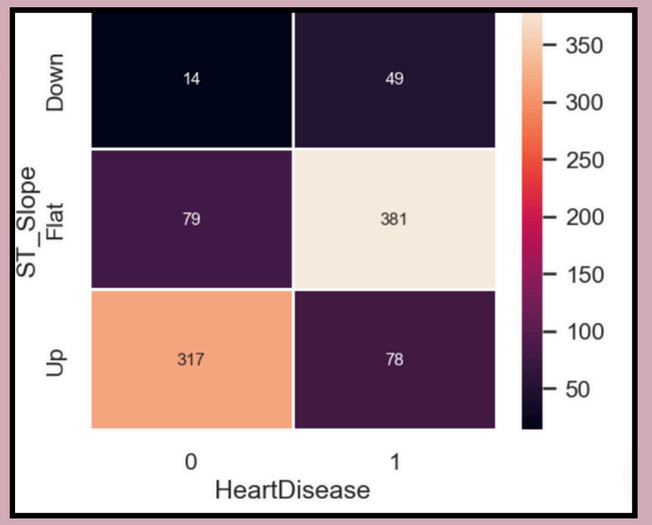
FastingBS and Heart Disease

4.



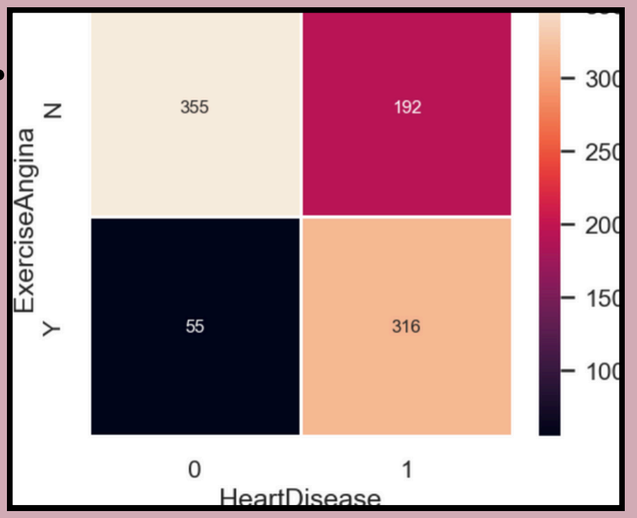
RestingECG and Heart Disease

5.



ST_Slope and Heart Disease

6.



ExerciseAngina and Heart Disease

Multi-Variate EDA

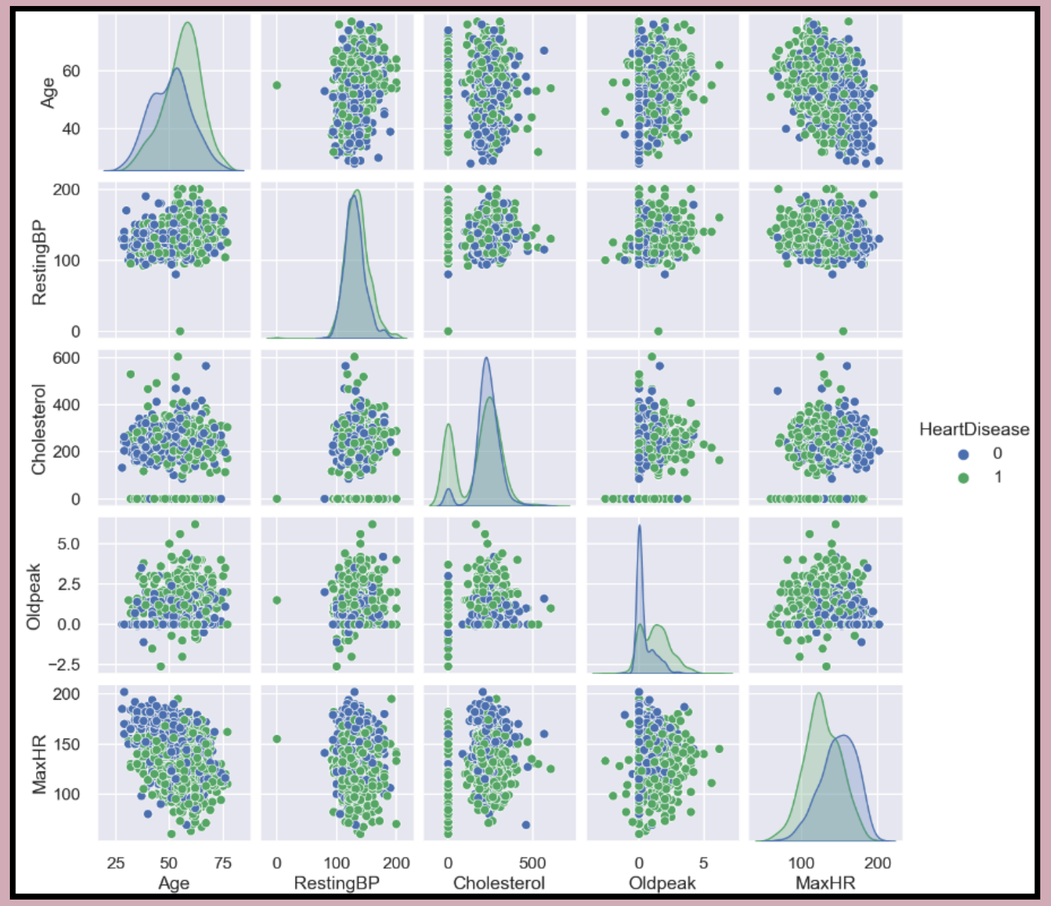
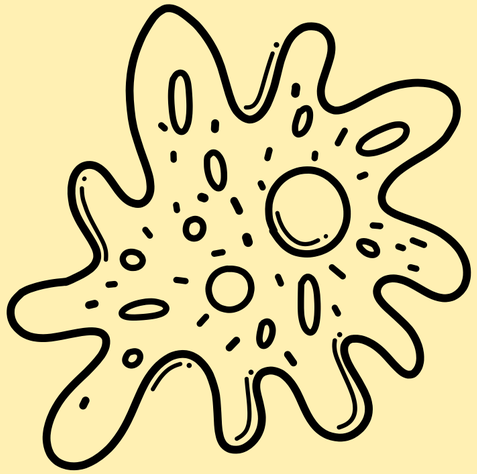
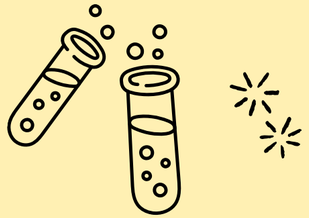


figure 3.2 Correlation between Variables



Cleaning

- Removing Outliers
- Impute Missing Data



0 Values Masked as Non-Null Values

figure 4.1 (a.1) Information from the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

Suggests that there are no Non-Null Values

figure 4.1 (a.2) Description of the dataset

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Impossible for a person to have no Cholesterol and Resting Blood Pressure!

0 Values Masked as Non-Null Values

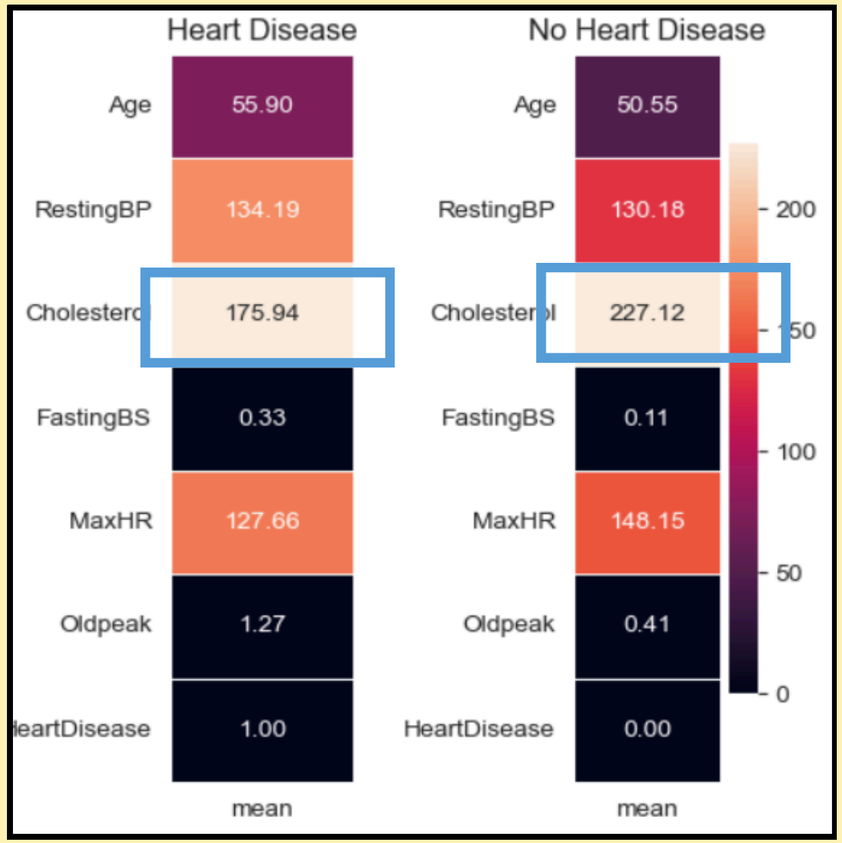


figure 4.1 (b) Average of Data points respective to their columns for persons with and without heart disease

How can a person without heart disease have higher levels of cholesterol compared to a person with heart disease?

Removing Outliers


```python
def outlierRemove(dataframe, column):
    # Convert the column to numeric, coercing errors to NaN
    dataframe[column] = pd.to_numeric(dataframe[column], errors='coerce')

    # Calculate upper and lower whiskers
    q1 = dataframe[column].quantile(0.25)
    q3 = dataframe[column].quantile(0.75)
    iqr = q3 - q1
    upper_whisker = q3 + 1.5 * iqr
    lower_whisker = q1 - 1.5 * iqr

    # Replace outliers with np.nan
    #dataframe.loc[dataframe[column] > upper_whisker, column] = np.nan
    dataframe.loc[dataframe[column] < lower_whisker, column] = np.nan

# outlierRemove(heartData, "Cholesterol")
```

figure 4.1 (c.1) Function to Remove Outliers before the first Whisker

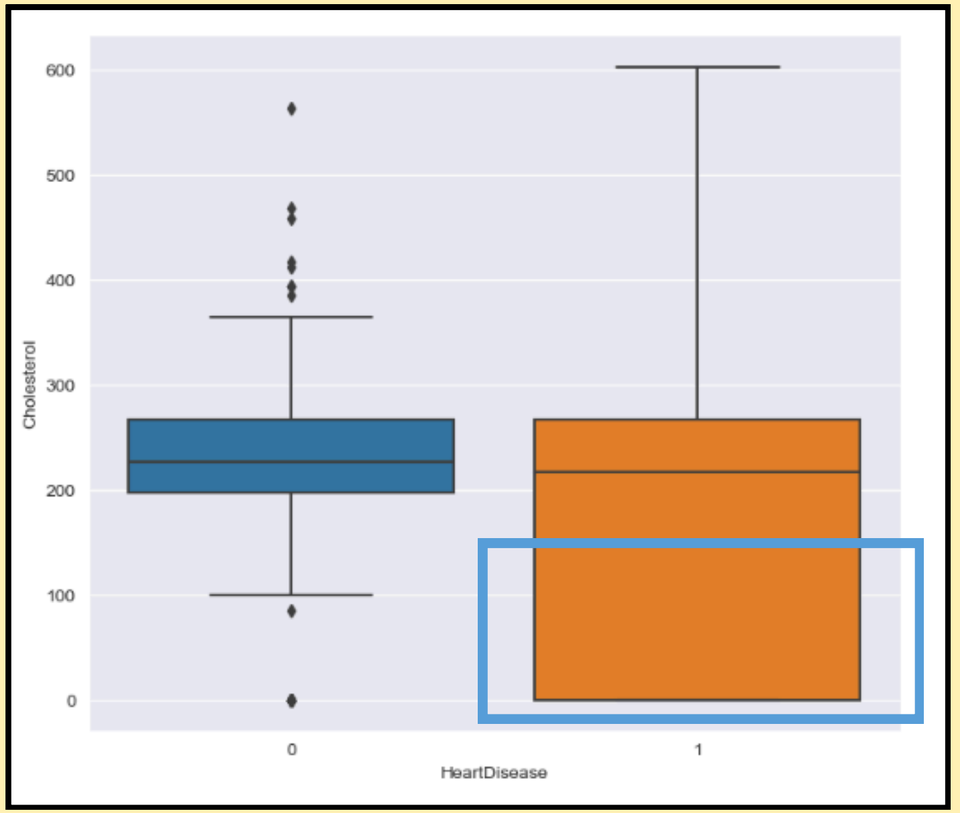


figure 4.1 (c.2) As you can see the distribution is skewed and inaccurate because a person cannot just have o Cholesterol levels

Impute Missing Data

	Number of records missing	Percentage of missing records
Cholesterol	172	0.187364
RestingBP	2	0.002179
Age	0	0.000000
Sex	0	0.000000
ChestPainType	0	0.000000
FastingBS	0	0.000000
RestingECG	0	0.000000
MaxHR	0	0.000000
ExerciseAngina	0	0.000000
Oldpeak	0	0.000000
ST_Slope	0	0.000000
HeartDisease	0	0.000000

figure 4.1 (d.1) Missing data from removing the outliers

The average value of the variable will replace the missing value for a record with heart disease based on whether heart disease is present or not

```
def replaceWithMean(var):
    meanWithDisease = heartData.loc[heartData['HeartDisease'] == 1, var].mean()
    meanWithoutDisease = heartData.loc[heartData['HeartDisease'] == 0, var].mean()

    print(meanWithDisease)
    print(meanWithoutDisease)

    heartData.loc[heartData['HeartDisease'] == 1, var] = heartData.loc[heartData['HeartDisease'] == 1, var].fillna(meanWithDisease)
    heartData.loc[heartData['HeartDisease'] == 0, var] = heartData.loc[heartData['HeartDisease'] == 0, var].fillna(meanWithoutDisease)

# replaceWithMean("Cholesterol")
```

figure 4.1 (d.2) Impute Missing data with mean corresponding to column and heart disease indication

Average Reconsilidation

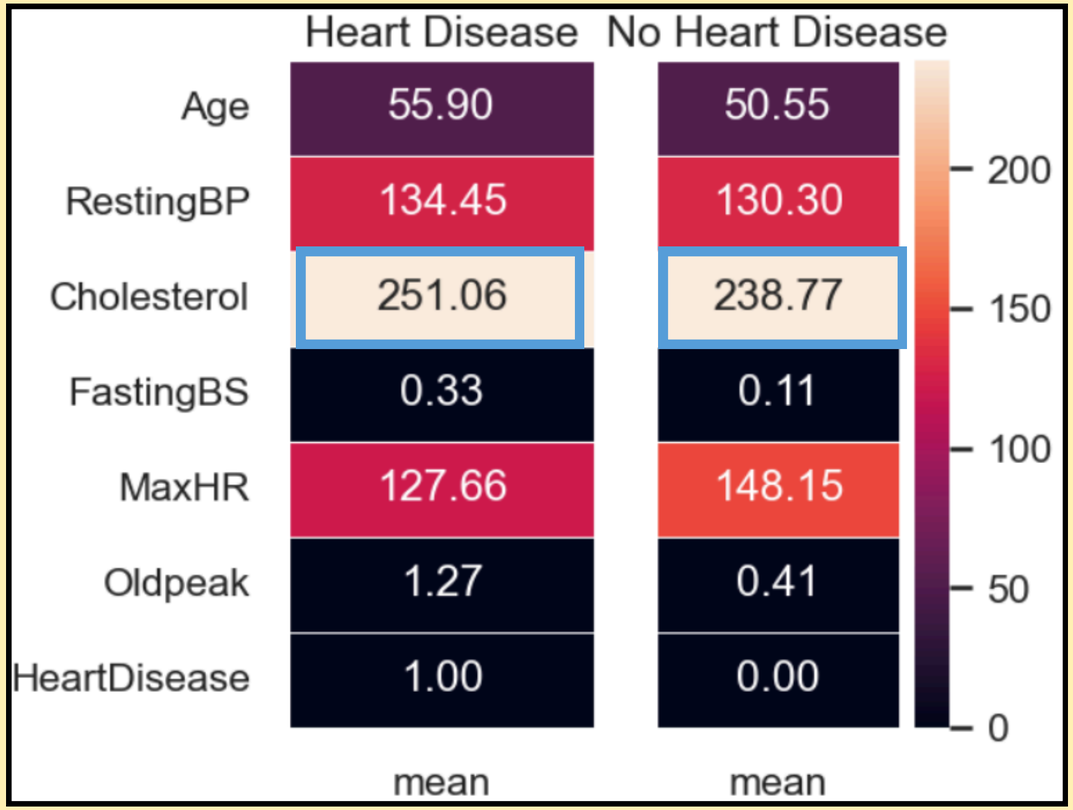
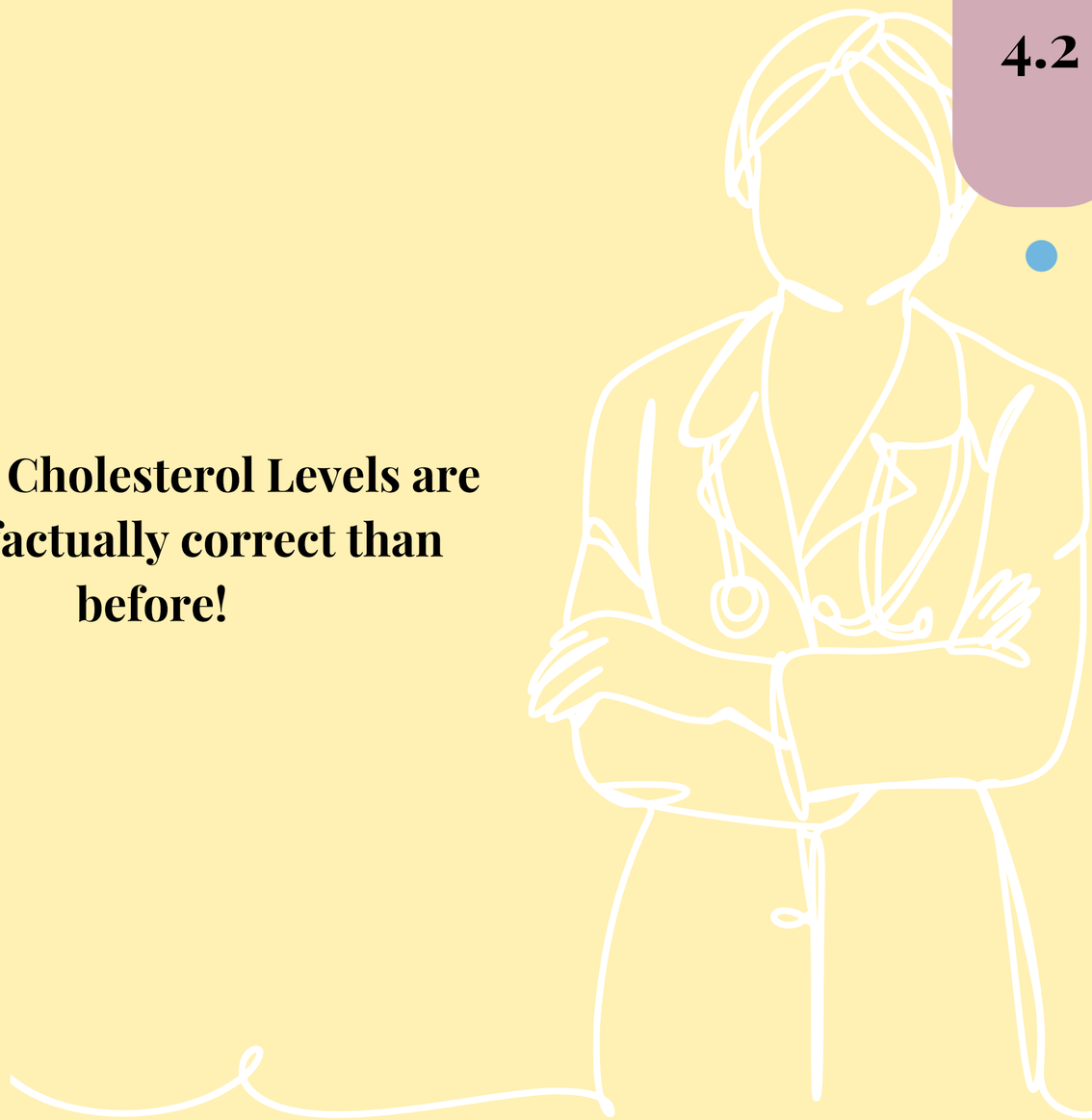


figure 4.1 (e) Missing data from removing the outliers

Average Cholesterol Levels are more factually correct than before!



Feature Selection

- Label Encoding, Normalization & Standardization
- Chi Squared Test
- ANOVA Test

Label Encoding


```
le = LabelEncoder()
df1 = heartData.copy(deep = True)

df1['Sex'] = le.fit_transform(df1['Sex'])
df1['ChestPainType'] = le.fit_transform(df1['ChestPainType'])
df1['RestingECG'] = le.fit_transform(df1['RestingECG'])
df1['ExerciseAngina'] = le.fit_transform(df1['ExerciseAngina'])
df1['ST_Slope'] = le.fit_transform(df1['ST_Slope'])
```

figure 5.1a (a) Using LabelEncoder, a sklearn library to label encode

ChestPainType
1
2
1
0
2

figure 5.1a (c) How it looked on the dataset where the index ranged from 0-4 in the order mentioned before

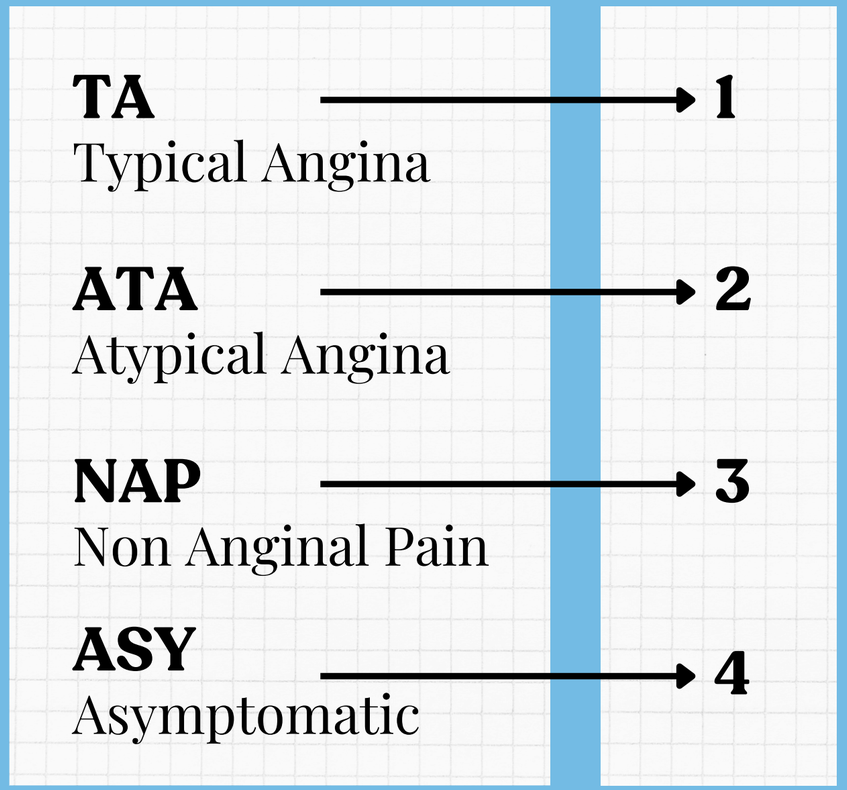


figure 5.1a (b) Example of how label encoding works

Normalization and Standardization

5.1b

```
mms = MinMaxScaler() # Normalization
ss = StandardScaler() # Standardization

df1['Oldpeak'] = mms.fit_transform(df1[['Oldpeak']])
df1['Oldpeak'] = ss.fit_transform(df1[['Oldpeak']])
df1['Age'] = ss.fit_transform(df1[['Age']])
df1['RestingBP'] = ss.fit_transform(df1[['RestingBP']])
df1['Cholesterol'] = ss.fit_transform(df1[['Cholesterol']])
df1['MaxHR'] = ss.fit_transform(df1[['MaxHR']])
```

figure 5.1b Using the sklearn library MinMaxScaler and StandardScaler to scale the data points for ease of comparison on correlation

- MinMaxScaler to ensure data points lay in the positive axis
- StandardScalar to ensure that the mean is standardized

Correlation

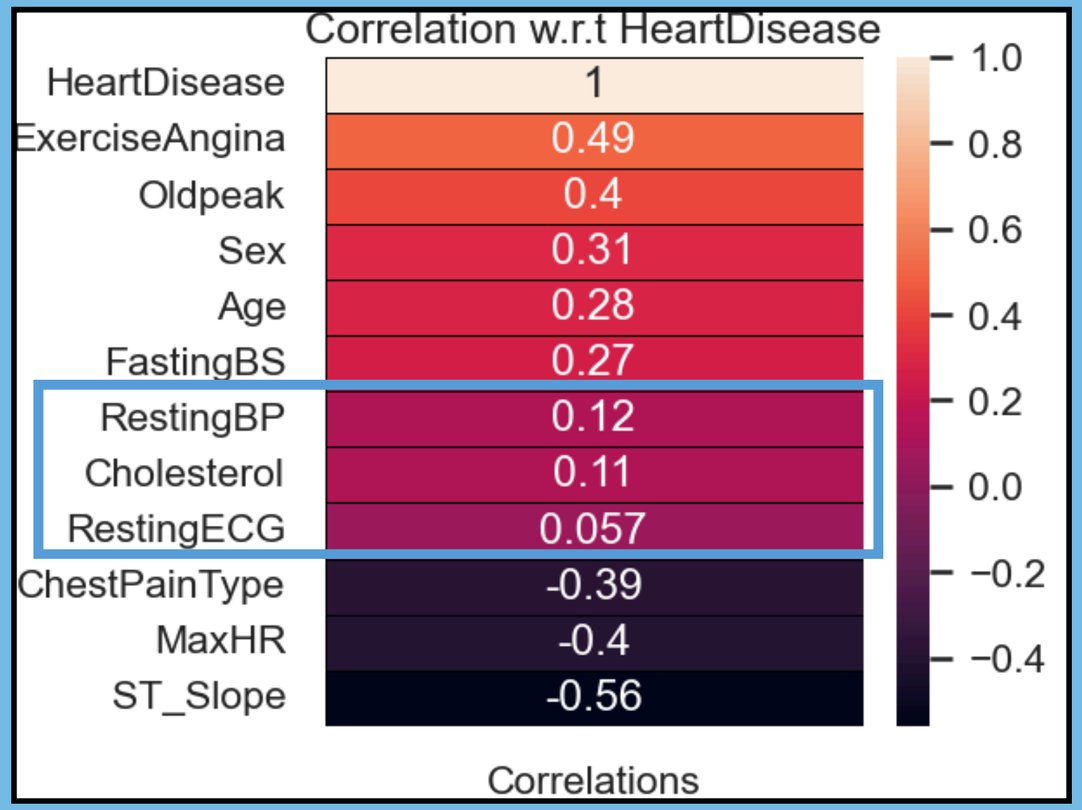


figure 5.1c Correlation with Heart Disease

Correlation of 0 is something we do not want

Chi Squared Test


```python
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

categorical_features = ["Sex", "ChestPainType", "FastingBS", "RestingECG" ,"ExerciseAngina" ,"ST_Slope" ,"HeartDisease"]

features = df1.loc[:,categorical_features[:-1]]
target = df1.loc[:,categorical_features[-1]]

best_features = SelectKBest(score_func = chi2,k = 'all')
fit = best_features.fit(features,target)

featureScores = pd.DataFrame(data = fit.scores_,index = list(features.columns),columns = ['Chi Squared Score'])

plt.subplots(figsize = (5,5))
sns.heatmap(featureScores.sort_values(ascending = False,by = 'Chi Squared Score'),annot = True,
            cmap = "rocket",linewidths = 0.4,linecolor = 'black',fmt = '.2f');
plt.title('Selection of Categorical Features');
```

figure 5.2(a) Sklearn library to do the Chi-squared test on categorical data

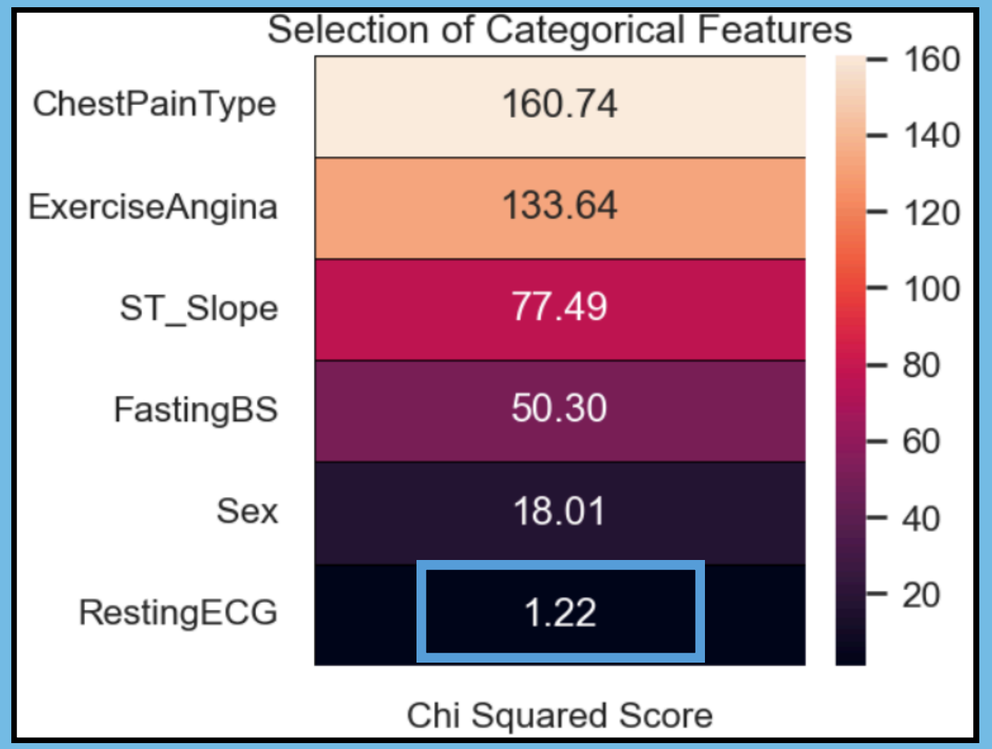


figure 5.2(b) Chi Squared Score with Chest Pain Type being the best indicator of heart disease and Resting ECG being the worst

Resting ECG

ANOVA Test


```
from sklearn.feature_selection import f_classif

numerical_features = ['Age', 'RestingBP', 'Cholesterol', 'Oldpeak', 'MaxHR']

features = df1.loc[:,numerical_features]
target = df1.loc[:,categorical_features[-1]]

best_features = SelectKBest(score_func = f_classif,k = 'all')
fit = best_features.fit(features,target)

featureScores = pd.DataFrame(data = fit.scores_,index = list(features.columns),columns = ['ANOVA Score'])

plt.subplots(figsize = (5,5))
sns.heatmap(featureScores.sort_values(ascending = False,by = 'ANOVA Score'),annot = True,
            cmap = "rocket",linewidths = 0.4,linecolor = 'black',fmt = '.2f');
plt.title('Selection of Numerical Features');
```

figure 5.3(a) Sklearn library to do the ANOVA test on numeric data

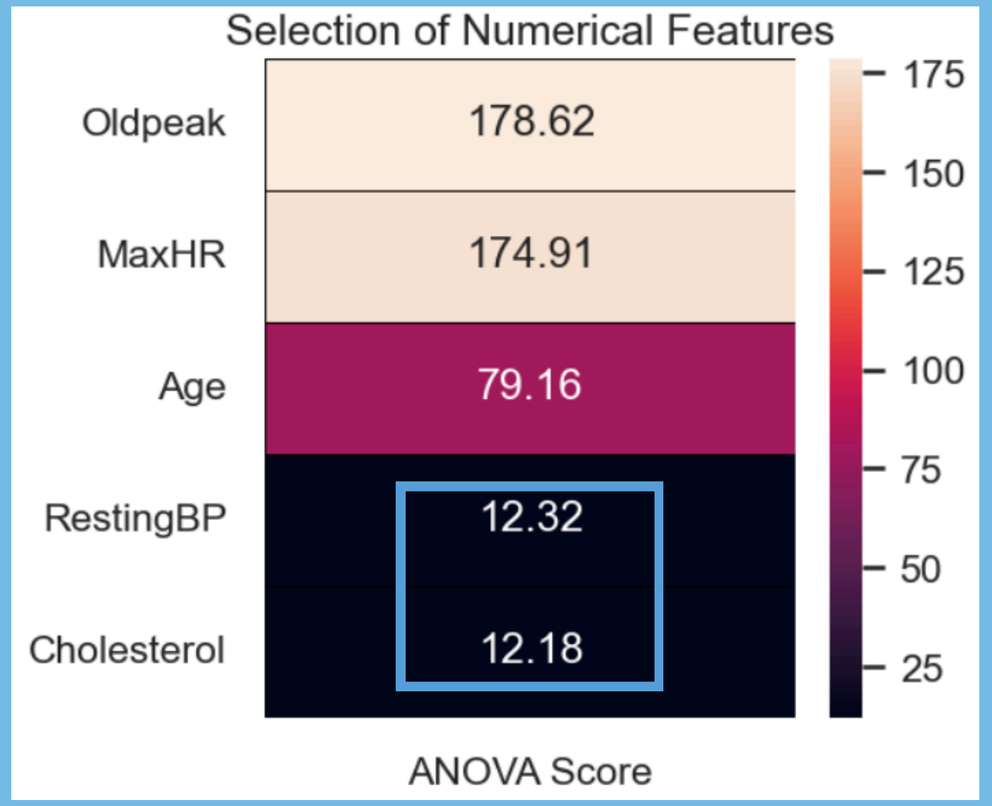


figure 5.3(b) ANOVA test score with Chest Pain Type being the best indicator of heart disease and Resting ECG being the worst

Resting Blood Pressure

Cholesterol Levels

Models Used

Multi Layer Perceptron Classifier

Logistic Regression

Random Forest

Heart Disease

Multi-Layer Perceptron

6.1

MLP

Handling Categorical Data

```
categoricalHeartData = pd.DataFrame(heartData[['Sex', 'ChestPainType', 'FastingBS', 'ExerciseAngina', 'ST_Slope']])
# Concatenate dummy variables for categorical features
heartDataLog = pd.concat([heartData, pd.get_dummies(heartData[categoricalHeartData.columns])], axis=1)
heartDataLog.drop(categoricalHeartData.columns, axis=1, inplace=True) # remove the original categorical column
```

```
heartDataLog.replace({False: 0, True: 1}, inplace=True)
heartDataLog.head()
```

	Age	Cholesterol	MaxHR	Oldpeak	HeartDisease	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA	ChestPainType_NAP	Chest
0	40	289.0	172	0.0	0	0	1	0	1	0	
1	49	180.0	156	1.0	1	1	0	0	0	1	
2	37	283.0	98	0.0	0	0	1	0	1	0	
3	48	214.0	108	1.5	1	1	0	1	0	0	
4	54	195.0	122	0.0	0	0	1	0	0	1	

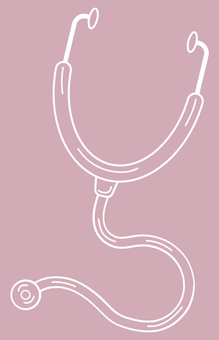


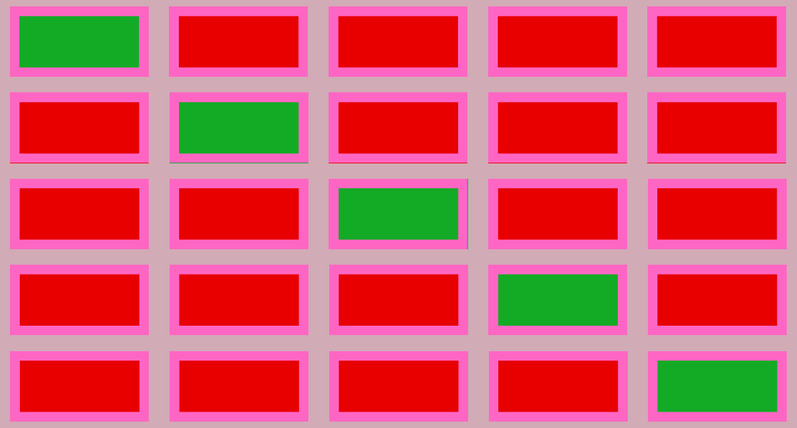
figure 6.1 (a) One Hot Encoding Categorical Data so it is machine readable

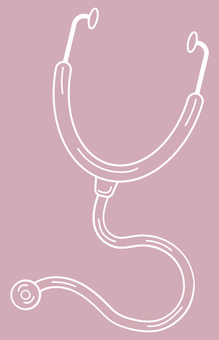
MLP

Stratified K-Fold

```python
kf=model_selection.StratifiedKFold(n_splits=5, random_state=6, shuffle=True) # train test split

scaler = StandardScaler()

for fold, (train_idx, test_idx) in enumerate(kf.split(X=heartDataLog, y=y)):
    xtrain_fold, xtest_fold = X.iloc[train_idx], X.iloc[test_idx]
    ytrain_fold, ytest_fold = y[train_idx], y[test_idx]

    # Fit StandardScaler on the training data and transform both train and test data
    xtrain_fold_scaled = scaler.fit_transform(xtrain_fold)
    xtest_fold_scaled = scaler.transform(xtest_fold)
```

figure 6.1 (b) Using Stratified K Fold for train and test split





Test Data

Train Data

MLP Results Analysis

figure 6.1 (c) Accuracy score, recall, AUC ROC score, classification report containing precision, recall, and f1-score

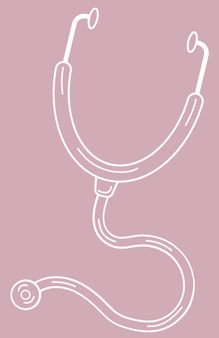
```
--------------------------------
Fold 1:
Score: 0.78
Recall: 0.82
AUC: 0.81
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.72      0.74        82
           1       0.79      0.82      0.80       102

    accuracy                           0.78       184
   macro avg       0.78      0.77      0.77       184
weighted avg       0.78      0.78      0.78       184

--------------------------------
```

```
--------------------------------
Fold 2:
Score: 0.77
Recall: 0.78
AUC: 0.83
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.76      0.75        82
           1       0.80      0.78      0.79       102

    accuracy                           0.77       184
   macro avg       0.77      0.77      0.77       184
weighted avg       0.77      0.77      0.77       184

--------------------------------
```

```
--------------------------------
Fold 3:
Score: 0.83
Recall: 0.85
AUC: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.79      0.80        82
           1       0.84      0.85      0.84       102

    accuracy                           0.83       184
   macro avg       0.82      0.82      0.82       184
weighted avg       0.83      0.83      0.83       184

--------------------------------
```

```
--------------------------------
Fold 4:
Score: 0.83
Recall: 0.83
AUC: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.83      0.81        82
           1       0.86      0.83      0.84       101

    accuracy                           0.83       183
   macro avg       0.83      0.83      0.83       183
weighted avg       0.83      0.83      0.83       183

--------------------------------
```

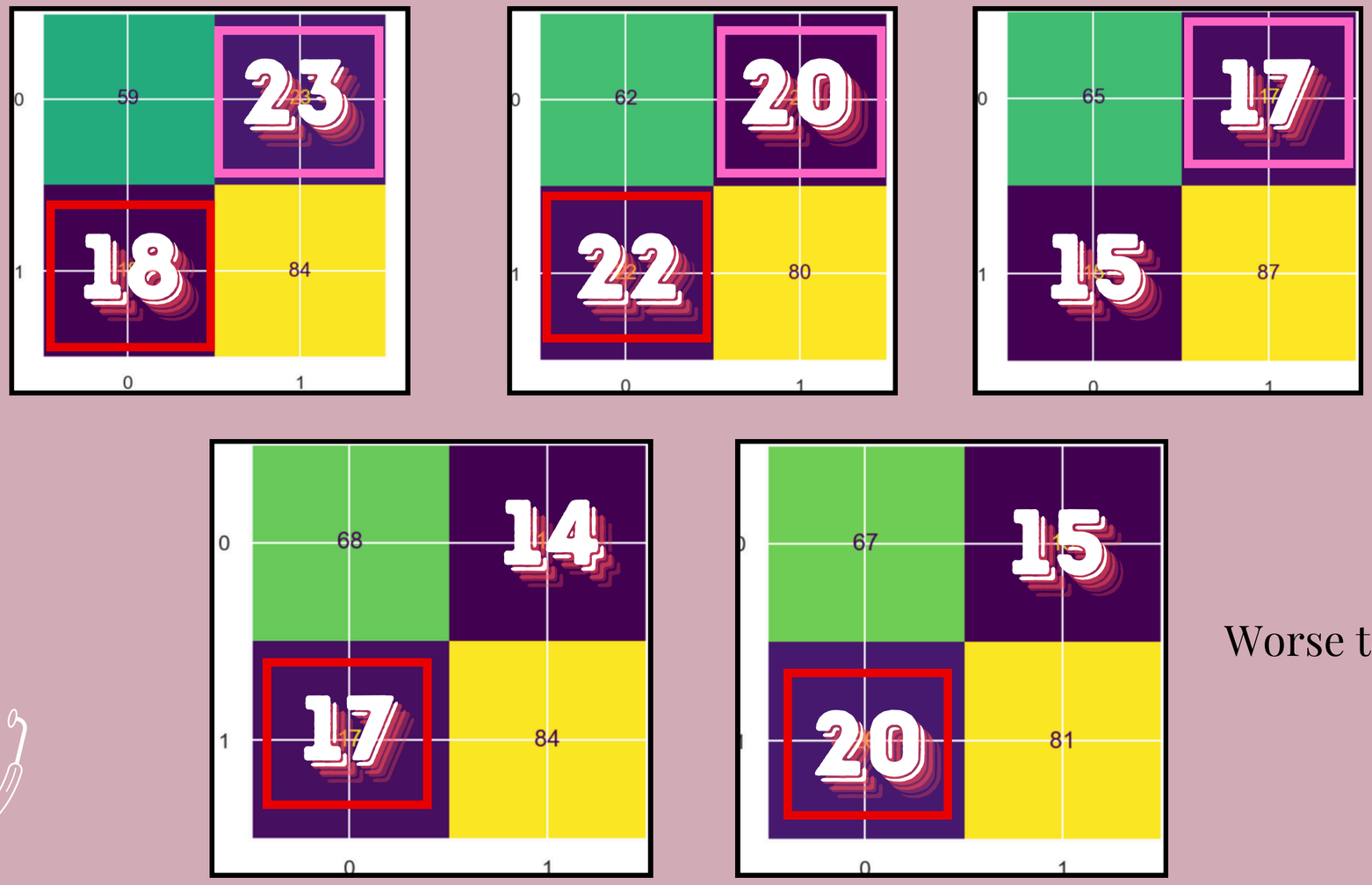
```
--------------------------------
Fold 5:
Score: 0.81
Recall: 0.8
AUC: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.82      0.79        82
           1       0.84      0.80      0.82       101

    accuracy                           0.81       183
   macro avg       0.81      0.81      0.81       183
weighted avg       0.81      0.81      0.81       183

--------------------------------
```



Average Score of 80.4%

MLP Results Analysis

figure 6.1 (c) Confusion matrix showing TP, FP, TN, & FN



False Positives

False Negatives

Worse to have higher False Negatives!

Logistic Regression

6.2

Logistic Regression

6.2

Stratified K-Fold & Training the Model

```
kf=model_selection.StratifiedKFold(n_splits=5, random_state=6, shuffle=True)
for fold , (trn_,val_) in enumerate(kf.split(X=heartDataLog,y=y)):

    X_train=heartDataLog.loc[trn_,feature_col_nontree]
    y_train=heartDataLog.loc[trn_,target]

    X_valid=heartDataLog.loc[val_,feature_col_nontree]
    y_valid=heartDataLog.loc[val_,target]

    #print(pd.DataFrame(X_valid).head())
    ro_scaler=StandardScaler()
    X_train=ro_scaler.fit_transform(X_train)
    X_valid=ro_scaler.transform(X_valid)


    clf=LogisticRegression()
    clf.fit(X_train,y_train)
```

figure 6.2 (a) Using Stratified K Fold for train and test split & Training the model

Logistic Regression Results Analysis

6.2

figure 6.2 (b) Classification report containing precision, recall, and f1-score

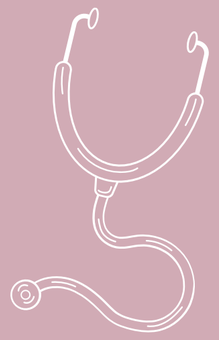
```
The fold is : 0 :
              precision    recall  f1-score   support

           0       0.85      0.80      0.83        82
           1       0.85      0.88      0.87       102

    accuracy                           0.85       184
   macro avg       0.85      0.84      0.85       184
weighted avg       0.85      0.85      0.85       184

The accuracy for Fold 1 : 0.8436154949784792
```

```
The fold is : 1 :
              precision    recall  f1-score   support

           0       0.80      0.84      0.82        82
           1       0.87      0.83      0.85       102

    accuracy                           0.84       184
   macro avg       0.83      0.84      0.84       184
weighted avg       0.84      0.84      0.84       184

The accuracy for Fold 2 : 0.8373983739837398
```

```
The fold is : 2 :
              precision    recall  f1-score   support

           0       0.86      0.78      0.82        82
           1       0.84      0.90      0.87       102

    accuracy                           0.85       184
   macro avg       0.85      0.84      0.84       184
weighted avg       0.85      0.85      0.85       184

The accuracy for Fold 3 : 0.8412242945958871
```

```
The fold is : 3 :
              precision    recall  f1-score   support

           0       0.86      0.80      0.83        82
           1       0.85      0.89      0.87       101

    accuracy                           0.85       183
   macro avg       0.85      0.85      0.85       183
weighted avg       0.85      0.85      0.85       183

The accuracy for Fold 4 : 0.8479835788456894
```

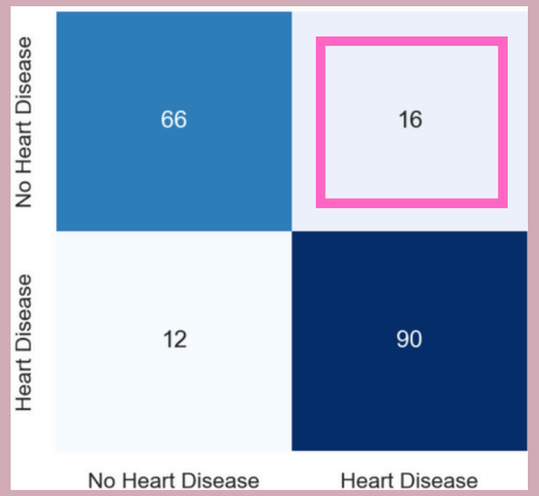
```
The fold is : 4 :
              precision    recall  f1-score   support

           0       0.91      0.85      0.88        82
           1       0.89      0.93      0.91       101

    accuracy                           0.90       183
   macro avg       0.90      0.89      0.89       183
weighted avg       0.90      0.90      0.90       183

The accuracy for Fold 5 : 0.8921758029461482
```

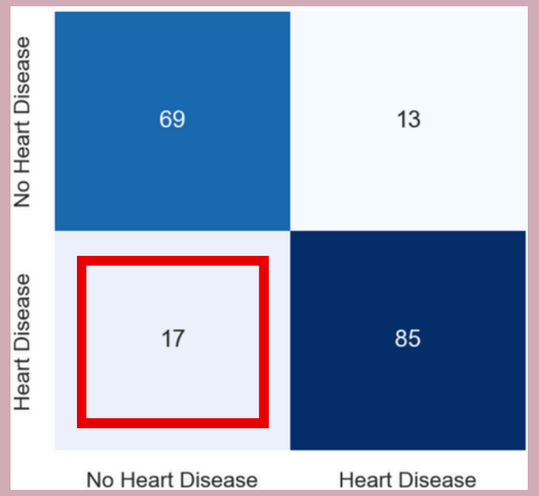


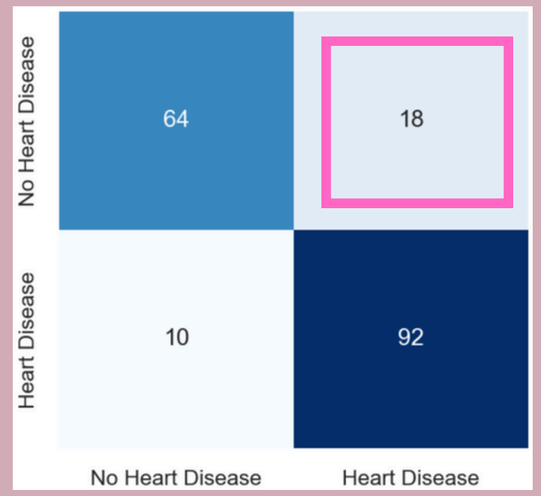
Average Score of 85%

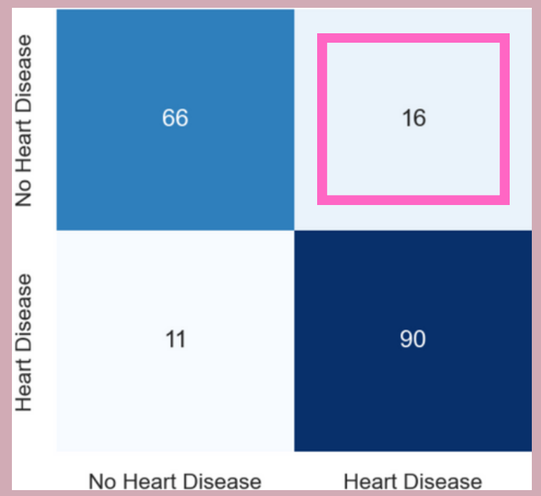
Logistic Regression Results Analysis

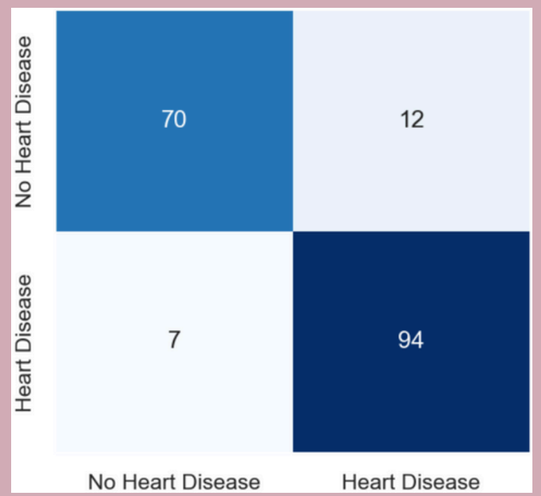
figure 6.2 (c) Confusion matrix showing TP, FP, TN, & FN











False Positives

False Negatives

Lesser False Negatives!

Actual Data Points

Predicted Data Points

Model 3

RANDOM FOREST

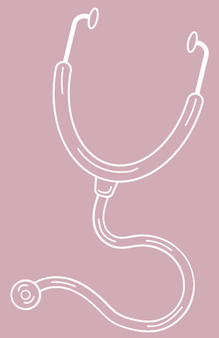
Random Forest

Predictors Used

- Age
- Cholesterol
- Oldpeak
- MaxHR

Model Code- 100 Trees

```
# Create a Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100)
# Train the classifier
rf_classifier.fit(X_train, y_train)
```

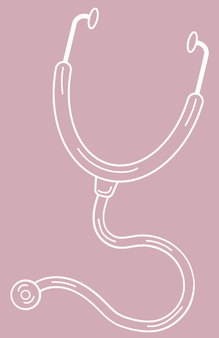


Random Forest Result

Accuracy for the Test Data: 0.79

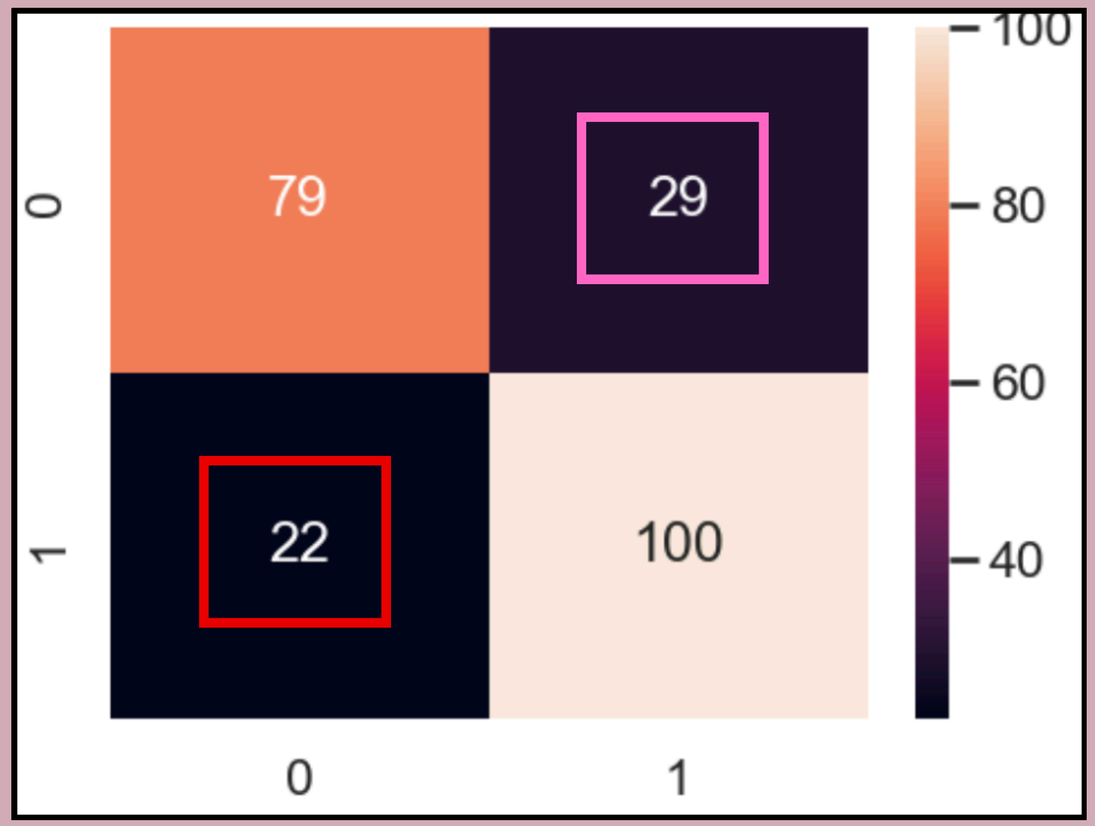
Classification Report:

	precision	recall	f1-score	support
0	0.83	0.67	0.74	104
1	0.77	0.89	0.82	126
accuracy			0.79	230
macro avg	0.80	0.78	0.78	230
weighted avg	0.80	0.79	0.79	230



Score: 80%

Random Forest Result

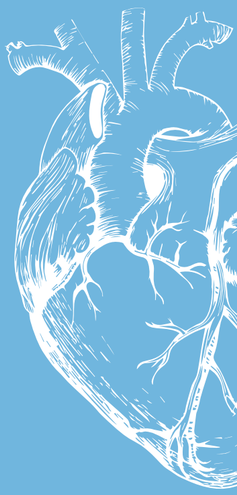


Confusion matrix showing TP, FP, TN, & FN

False Positives

False Negatives

Conclusion



FACTORS CONTRIBUTING TO HEART DISEASE

- Exercise Induced Angina
- Old Peak
- Gender
- Age
- Fasting Blood Sugar
- Cholesterol Levels
- Chest Pain Type Experienced
- Maximum Heart Rate
- ST Slope

BEST MODEL TO PREDICT HEART DISEASE

The model with the highest accuracy was Logistic Regression

WHAT CAN BE DONE TO IDENTIFY EARLY RISK OF HEART DISEASE

- Understand and take care of your heart as you get older
- Men are more likely to get heart disease than women
- Experiencing certain types of chest pain, exercise-induced angina, and high levels of fasting blood sugarputs a person more at risk of heart disease

Thank you !

we luv Scuds


#SETHI MEHAR#
#ARSHIYA JAWAHAR ALI SIRAJUDEEN#


#ARSHIYA JAWAHAR ALI SIRAJUDEEN#