Application Of Machine Learning Algorithms To Predict Heart Failure Dataset

Eziamaka Gozie Nwakile
Faculty of Engineering, Environment and Computing
MSc Data Science and Computational Intelligence
Coventry University
Coventry, United Kingdom
nwakilee@uni.coventry.ac.uk

Abstract— In this paper, the aim is to use machine learning classification algorithms such as Decision Tree, Random Forest, Logistic Regression, Support Vector Machine, K Nearest Neighbour and Naive Bayes to predict the heart failure dataset. Machine learning techniques were applied to the dataset for preprocessing, feature importance, encoding and scaling before the data was passed into the classification algorithms. The outcomes of the prediction and the achievement were acquired, compared and visualised with confusion metrics and ROC.

Keywords — classification, machine learning, cardiovascular disease(CVD), standardscalar, random forest, encoding

1. INTRODUCTION

According to the World Health Organisation, approximately 17.9million people died from Cardiovascular disease(CVD) in 2019. Globally, CVD is one of the main causes of death, accounting for 32% of all deaths worldwide. Nearly eighty percent of these deaths were caused by heart attack or strokes for people under 70years of age (WHO, 2021). There are four main types of CVD namely coronary heart disease, stroke, peripheral arterial disease and aortic disease. There is a rapid increase in the cases of heart diseases which is alarming and requires quick intervention. CVD is potentially preventable to a large extent by eating heart-healthy meals, maintaining a healthy weight, exercising for a minimum of 30minutes everyday and avoiding the use of tobacco products,(Mayoclinic, 2022) The early detection and management of cardiovascular disease among people at high risk for the disorder or who have already established cardiovascular disease (like hypertension, diabetes, hyperlipidaemia or already established disease) requires a machine learning model,(Kaggle,). It is paramount and of great concern to predict these kinds of diseases ahead of time in order to reduce the global mortality rate. The sensitivity of this issue means that the prediction of CVD has to be precise, efficient and accurate.

Machine learning which is a subset of Artificial Intelligence(AI) plays key roles in different facets of life such as the health sector, education and finance amongst others all over the world. The appropriate application of machine learning in advance helps health practitioners gain insights that will aid them in diagnosis and treatment of their patients. Diverse machine learning algorithms have been used to predict cardiovascular disease in the past but we aim to utilise various machine

learning classification algorithms to predict cardiovascular disease using the dataset obtained from Kaggle.

2. LITERATURE REVIEW

As technology evolves, machine learning has become a go-to place for analysis and predictions. This has widened the funnel of researchers who are now building new models and improving old ones for use by end-users.

In recent years, there has been a great deal of success in predicting the early stages of heart disease as shown by various researchers. On the wider range of the research according to the work done by Nagamani et al (2019), they used Naïve Bayes Decision Tree in predicting heart disease, diabetics, and breast cancer. The result for the intelligent heart disease prediction system was derived using confusion metrics.

By limiting the study to just heart disease prediction, Mohan Senthilkumar et al in 2019, where the researchers implemented a hybrid system in the machine learning niche to predict heart disease using the dataset from Cleveland. As a result of the preprocessing of their dataset, the age and sex features were removed since the author explained that these were personal information about the patients and therefore would not affect the prediction. Another feature of the dataset that they have identified as being important is the clinical records of the patients. During the design of the hybrid system proposed by the researchers, the author mentioned that they used a combination of Random Forest and Linear Method to approach the problem, and they named their model Hybrid Random Forest Linear Method (HRFLM). The final output of the model is a classifier that predicts whether a patient has heart disease or not. The output was compared with other decision tree classifiers, it was discovered that the HRFLM combination did justice in yielding a better result than a single model.

Ali Liaqat et al (2019) built a model that has two algorithms based on linear Support Vector Machine (SVM). These models were named L1 and L2 regularisation. L1 was used for removing unnecessary features, while the L2 was used for the prediction. In the optimization, the author has stated that they used a hybrid grid search algorithm. The metrics returned were accuracy, specificity, correlation coefficient, sensitivity, and Receiver Operating Chart (ROC) chart. The prediction was done as an L1-linear SVM model stacked with an L2-linear SVM model which returned 91.1% testing accuracy on 30% testing data, and 84.05% training accuracy on 70% training data.

Furthermore, some researchers have been able to create intelligent heart disease prediction systems by using Neural Networks. A typical example is the work done by Palaniappan Sellappan et al who proposed using Naïve Bayes, Neural Network, and Decision Trees models to achieve a better prediction.

3. METHODS

The challenge we're attempting to tackle is a classification problem that involves predicting heart failure based on certain characteristics. The following classification algorithm is used in this paper to compare which algorithm best predicts the dataset:

3.1. *Decision Tree*

The Decision Tree algorithm is part of the supervised learning algorithms family. The decision tree approach, unlike other supervised learning algorithms, can also be used to solve regression and classification tasks. By learning simple decision rules inferred from past data, the aim of deploying a Decision Tree is to develop a training model that can be used to predict the class or value of the target variable (training

data). We start from the root of the tree when using Decision Trees that forecast a class label for a record. The values of the root attribute and the record's attribute are compared. We follow the branch that corresponds to that value and jump to the next node based on the comparison,(Nagesh Singh Chauha, 2022).

3.2. *Random Forest*

Random Forest is primarily used to solve classification issues. A forest, as we all know, is made up of trees, and more trees equals a more healthy forest. Similarly, the random forest method generates decision trees on data samples, then obtains predictions from each of them, and finally votes to determine the best option. It's an ensemble method that's superior to a single decision tree because it averages the results to reduce overfitting, (Nagesh Singh Chauha, 2022).

3.3. *Logistic Regression*

Logistic regression is a "supervised machine learning" approach that can be used to model the probability of a class or event. It's used when the data can be separated linearly and the result is binary or dichotomous. As a result, for binary classification problems, logistic regression is commonly used. Predicting an output variable that is discrete into two classes is referred to as binary classification. Yes/No, Pass/Fail are examples of binary classification.

3.4. *Support Vector Machine*

SVMs (support vector machines) are a sophisticated classification technique that also seeks to identify the best possible boundary between two groups. There are several hyperplanes that can perform this duty, but the goal is to locate the hyperplane with the biggest margin, or maximum distances between the two classes, so that if a new data point needs to be classified in the future, it can be done quickly, (Rohit Dwivedi, 2020)

3.5. *K Nearest Neighbour*

K Nearest Neighbours (KNN) is an easy-to-implement technique that is suitable for both classification and regression problems. KNN works by calculating the distances between a query and all of the instances in the data, picking the K closest examples to the query, and then voting for the most frequent label (in the case of classification) or averaging the labels (in the case of regression), (Onel Harrison, 2018)

3.6. *Naive Bayes*

Naive Bayes' is a classification method based on the assumption of predictor independence. A Naive Bayes classifier, in simple terms, assumes that the presence of one feature in a class is unrelated to the presence of any other feature. We infer that such values are obtained from a gaussian distribution unless the predictors have an uniform distribution and are not discrete, (Sunil, 2017)

4. PROBLEM AND DATA EXPLORATION

The exploratory data analysis (EDA) allows for exploration and understanding of data. In this section I will explain the steps carried out during the EDA.

The proposed system in evaluating and modelling of the dataset starts with the data collection, exploration and preprocessing (this includes statistical analysis, visualisation using bivariate and univariate analysis), and modelling (experimentation and prediction).

4.1. *Data collection*

The data used in this assessment which is the heart failure prediction dataset was sourced from Kaggle(Appendix 1). This dataset was derived from the combination of 5 different cardiovascular disease dataset to make up a total

of 918 observations and 12 attributes (features) as shown in (Table 1).

Table 1. Dataset dictionary

#	Features	Description	Data Type
0	Age	age of the patient [years]	int64
1	Sex	sex of the patient [M: Male, F: Female]	object
2	ChestPainType	chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]	object
3	RestingBP	resting blood pressure [mm Hg]	int64
4	Cholesterol	serum cholesterol [mm/dl]	int64
5	FastingBS	fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]	int64
6	RestingECG	resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]	object
7	MaxHR	maximum heart rate achieved [Numeric value between 60 and 202]	int64
8	ExerciseAngina	exercise-induced angina [Y: Yes, N: No]	object
9	Oldpeak	ST [Numeric value measured in depression]	float64
10	ST_Slope	the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]	object
11	HeartDisease	output class [1: heart disease, 0: Normal]	int64

The label for the diagnosis is the Heart Disease feature which contains 1s and 0s. 1s meaning that a person has or liable to have a heart disease and 0 means that a person doesn't have or would likely not have a heart disease.

The first ten instances of the data were previewed (fig 1), then the data features were summarised into several characteristics as shown in tables 1 and 2.

Fig.1. First ten instances in the dataset

```
#loading the dataset
df=pd.read_csv('heart.csv')
df.head(10)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
5	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0
6	45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
7	54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
8	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
9	48	F	ATA	120	284	0	Normal	120	N	0.0	Up	0

```
df.shape
```

```
(918, 12)
```

The objective of this paper is to predict early detection of heart disease given the same parameters in the dataset.

4.2. ***Preprocessing***

4.2.1. *Statistical Analysis*

The data preprocessing starts with the descriptive statistics analysis of the raw data. The datatype of the dataset is attributes where first checked (fig 2) to know what kind of data types are contained within the data, and how to handle each attribute.

Fig. 2. Datatypes of the dataset

```
df.info()
```

```
<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
```

The data statistical info was extracted using a single line of code to display in a tabular form the count, minimum value, maximum value, standard deviation, mean, and percentiles (25, 50, and 75). This is shown in fig 3.

Fig. 3: Statistical information of the dataset

```
df.describe()
# df.describe(include='all')
```

	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

The number of unique occurrence and missing values in each column was calculated and plotted in a tabular form (fig 4a), also the categorical data were grouped (fig 4b).

Fig. 4a. Statistical information in tabular form

	Name of Col	Num of Null	Dtype	N_unique
0	Age	0	int64	50
1	Sex	0	object	2
2	ChestPainType	0	object	4
3	RestingBP	0	int64	67
4	Cholesterol	0	int64	222
5	FastingBS	0	int64	2
6	RestingECG	0	object	3
7	MaxHR	0	int64	119
8	ExerciseAngina	0	object	2
9	Oldpeak	0	float64	53
10	ST_Slope	0	object	3
11	HeartDisease	0	int64	2

Fig. 4b. Grouped categorical data

Sex	
M	725.00
F	193.00

ChestPainType	
ASY	496.00
NAP	203.00
ATA	173.00
TA	46.00

FastingBS	
0	704.00
1	214.00

RestingECG	
Normal	552.00
LVH	188.00
ST	178.00

ExerciseAngina	
N	547.00
Y	371.00

ST_Slope	
Flat	460.00
Up	395.00
Down	63.00

HeartDisease	
1	508.00
0	410.00

4.2.2. *Visualisation*

The distribution of the data was plotted using histogram (fig 5a) and Kernel Density (KDE) plot (fig. 5b).

From the histogram plot it is observable that Resting Blood Pressure, maximum heart rate (MaxHR), and age are well distributed. Whereas the cholesterol and Oldpeak are poorly distributed.

Fig. 5a: Histogram plot of selected features

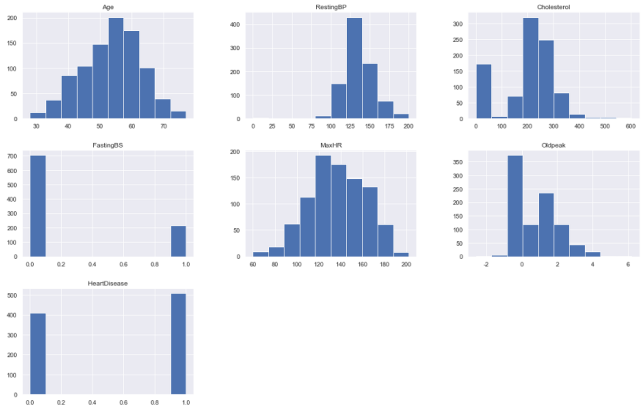
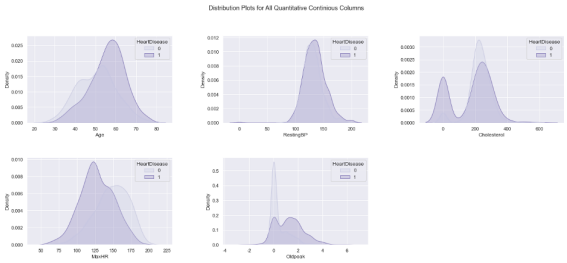


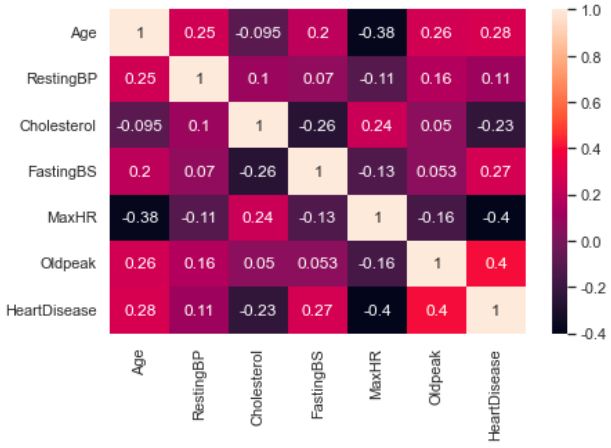
Fig. 5b: Kernel Density Plot



4.2.3. *Correlation*

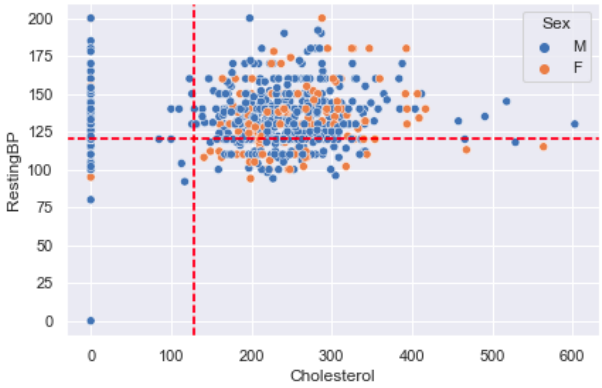
A correlation heatmap was also created using the correlation matrix(fig 6). It describes the relationship between two or more variables. These factors could be aspects of the raw data that were used to forecast our target variable. Correlation is a statistical method for determining how one variable moves or changes in connection to another. It's a bi-variate analysis metric that describes the relationship between two or more variables,(Amit Upadhyay, 2020).

Fig. 6: Correlation heatmap



After checking the correlation between the variables, a scattered plot for RestingBP and Cholesterol was plotted against sex to find out the sex(fig 7) that is most likely to have heart issues. From the result of the scattered plot, the Males are more likely to have a high cholesterol level with high restingBP.

Fig. 7: ScatteredPlot of RestingBP, Cholesterol and Sex



4.2.4. *Encoding*

The dataset was encoded to convert the labels into a numeric form to a readable machine format(fig 8). The reason for this is that the majority of machine learning algorithms are incapable of interpreting categorical input. As a result, they're frequently encoded in binary before being sent to the algorithm. An encoder is frequently used to do this.

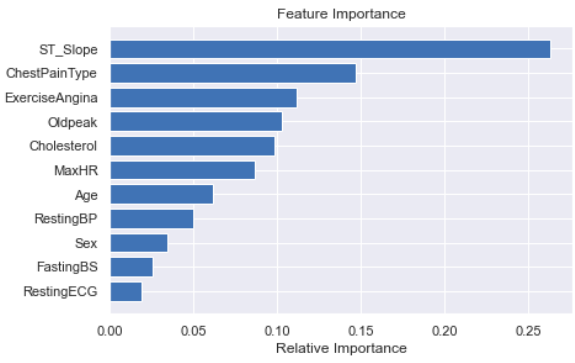
Fig. 8: Output of the encoded dataset

```
Sex
ChestPainType
RestingECG
ExerciseAngina
ST_Slope
<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    int64
 2   ChestPainType   918 non-null    int64
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    int64
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    int64
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    int64
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(11)
memory usage: 86.2 KB
```

4.2.5. *Feature Importance*

Feature importance was carried to find out the features that had more importance compared to the others as seen in fig 8. The reason for checking the feature importance of our dataset is to pick the most significant variables and eliminate redundant and irrelevant features. Feature selection improves the machine learning process and raises the predictive capacity of machine learning algorithms. It was observed carrying out this procedure that ST_Slope, ChestPain Type, ExerciseAngina, OldPeak and Cholesterol were the top 5 features amongst the 12 features. This also implies that these top 5 features play a significant role in determining if a person can have a heart disease or not.

Fig. 8: Feature Importance



4.2.6. *Test and Split*

Before our dataset was passed through our model, the dataset was split into 2 which is the train and test dataset. The reason for this is because there will not be enough data in the training dataset for the model to learn an appropriate mapping of inputs to outputs when the dataset is split into train and test sets. There will also be insufficient data in the test set to evaluate the model's performance appropriately. In general, this procedure is used to estimate the performance of machine learning algorithms. 70% of the dataset was used to train the model while 30% which is the remaining data was used to test the data.

4.2.7. *StandardScalar*

Our dataset was scaled after the test and train split procedure. This is due to the fact that when data is acquired, it includes features of various dimensions and scales. Different sizes of data features have a negative impact on a dataset's modelling. In terms of accuracy rates, it leads to a biassed outcome of predictions. As a result, prior to modelling, the data must be scaled. In my project, it was observed that when the data was modelled before scaling the performance of KNN was the poorest at 69% but after scaling using the standardscaler technique, the performance increased to 85%.

Standardisation is a scaling approach that converts the statistical distribution of the data into the following format in fig 9, making the data scale-free:

mean – 0 (zero)

standard deviation – 1

Fig. 9: KNN converter

$$z = \frac{x - \mu}{\sigma}$$

By this, the entire data set scales with a zero mean and unit variance, altogether

5. Classification Modelling

Having imported the necessary python libraries for our chosen algorithms, these algorithms were evaluated to see how they would perform with our dataset first before the pre-processing and the second after the pre-processing. From table 2, it was observed Random Forest, SVM and Logistics regression models did better than the other models. So, there was a need to use a StandardScaler to improve the accuracy of the models. In addition to the scaling, we hyper tuned some parameters of the model like the Random Forest manually to see if we can get the accuracy to exceed 90%. However, we could not but was able to increase its accuracy by 1% that is from 88% to 89% using an n-estimator of 500 and random-state of 1. SVM had the second best accuracy with an accuracy of 88% when we used a linear kernel for the model followed by Logistic Regression with an accuracy of 87%. For the KNN model, we used the 3 closest data to the datapoint, that is n_neighbors=3 because during the visualisation of the dataset, we checked for outliers in the datasets. It is from the result of the visualisation for checking for outliers in our dataset that brought about assigning the number of neighbours to the KNN model for better accuracy and precision. KNN's model accuracy increased by a little over 18%.

Table 2. Results of the classifiers before and after scaling

Classification Algorithms	Before Preprocessing	After Preprocessing
Decision Tree	**77%**	**76%**
Random Forest	**88%**	**89%**
Logistic Regression	**86%**	**87%**
Support Vector Machine(SVM)	**87%**	**88%**
KNN(K Nearest Neighbour)	**69%**	**85%**
Naive Bayes	**84%**	**87%**

6. Classification Result

Scikit-learn contains functions that are used in classification metrics such as precision, f1-score and recall. The results of the 6 algorithms used is this project are as follows:

Decision Tree

The results from the decision tree classifier shows that the accuracy of the model is 76%, precision at 84% and recall at 75%. The True Positives(TP) had a value of 86 and True Negative(TN) had 125. While the AUC from the ROC Curve is 0.76.

Fig 10: Classification report for decision tree classifier

```
Decision Tree Classifier
Accuracy: 0.76
Precision: 0.84
Recall: 0.75

              precision    recall  f1-score   support

           0       0.67      0.79      0.73       109
           1       0.84      0.75      0.79       167

    accuracy                           0.76       276
   macro avg       0.76      0.77      0.76       276
weighted avg       0.78      0.76      0.77       276
```

Fig 11: Confusion matrix for decision tree classifier

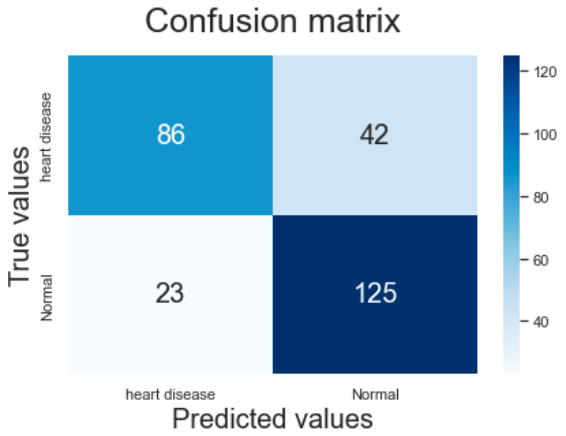
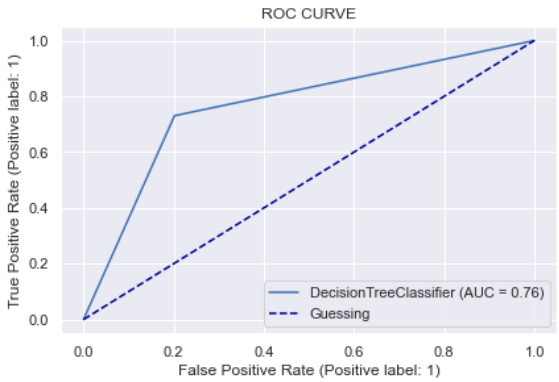


Fig 12: ROC Curve for decision tree classifier



Random Forest

The results from the random forest classifier shows that the accuracy of the model is 89%, precision at 92% and recall at 90%. The True Positives(TP) had a value of 96 and True Negative(TN) had 150. Getting 150 for TN is very good because in health related issues predictions have to be as accurate as possible. The AUC is 0.93.

Fig 13: Classification report for random forest classifier

```
Random Forest Classifier
Accuracy: 0.89
Precision: 0.92
Recall: 0.90

              precision    recall  f1-score   support

           0       0.85      0.88      0.86       109
           1       0.92      0.90      0.91       167

    accuracy                           0.89       276
   macro avg       0.88      0.89      0.89       276
weighted avg       0.89      0.89      0.89       276
```

Fig 14: Confusion matrix for random forest classifier

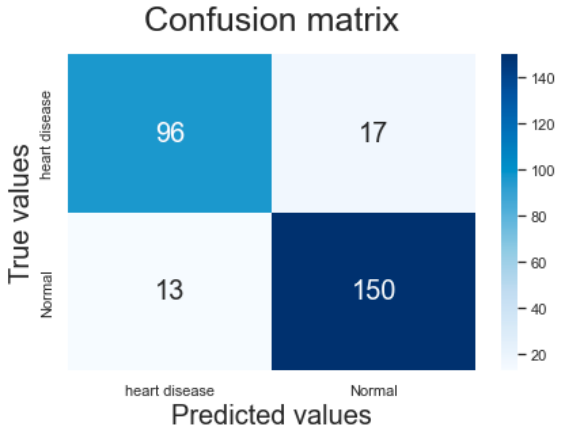
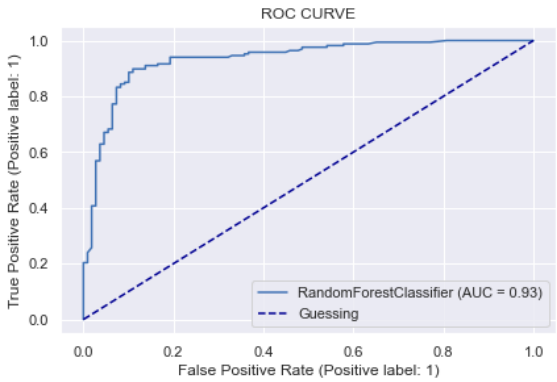


Fig 15: ROC Curve for random forest classifier



Logistic Regression

The results from the logistic regression classifier shows that the accuracy of the model is 87%, precision at 92% and recall at 86%. The True Positives(TP) had a value of 96 and True Negative(TN) had 144. While the AUC is 0.94 from the ROC Curve.

Fig 16: Classification report for logistic regression classifier

```
Logistic Regression
Accuracy: 0.87
Precision: 0.92
Recall: 0.86

              precision    recall  f1-score   support

           0       0.81      0.88      0.84       109
           1       0.92      0.86      0.89       167

    accuracy                           0.87       276
   macro avg       0.86      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276
```

Fig 17: Confusion matrix for logistic regression classifier

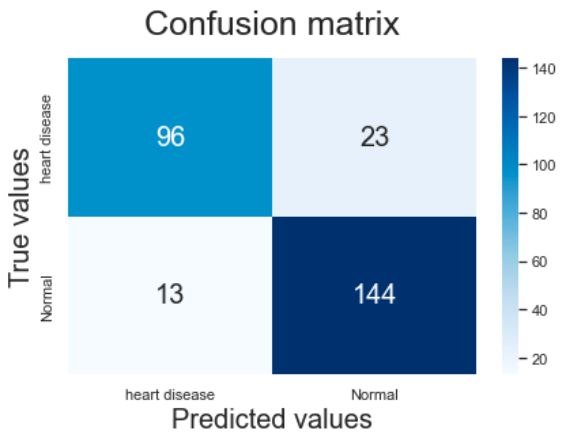
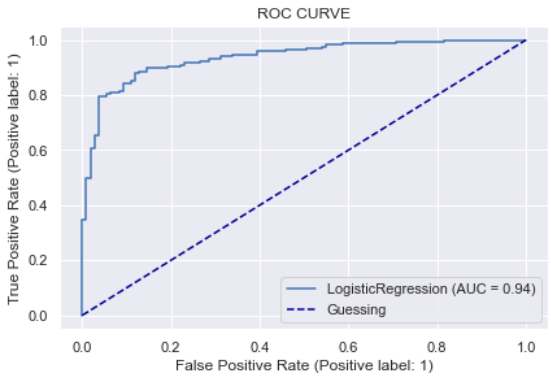


Fig 18: ROC Curve for logistic regression classifier



Support Vector Machine

The SVM classifier result shows that the accuracy of the model is 88%, precision at 91% and recall at 88%. The True Positives(TP) had a value of 95 and True Negative(TN) had 147. The AUC from the ROC curve is 0.93.

Fig 19: Classification report for SVM classifier

```
Support Vector Machines
Accuracy: 0.88
Precision: 0.91
Recall: 0.88

              precision    recall  f1-score   support

           0       0.83      0.87      0.85       109
           1       0.91      0.88      0.90       167

    accuracy                           0.88       276
   macro avg       0.87      0.88      0.87       276
weighted avg       0.88      0.88      0.88       276
```

Fig 20: Confusion matrix for SVM classifier

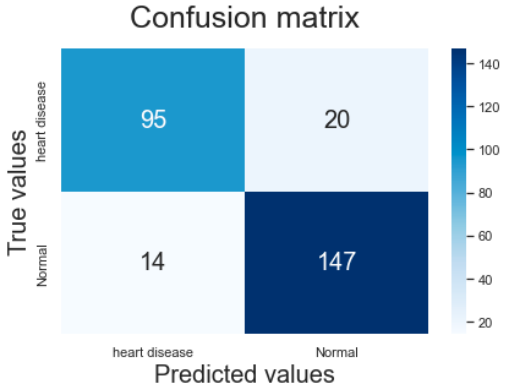
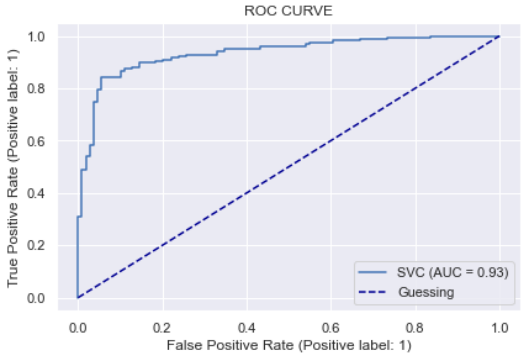


Fig 21: ROC Curve for SVM classifier



KNN(K Nearest Neighbour)

The results from the decision tree classifier shows that the accuracy of the model is 85%, precision at 84% and recall at 90%. The True Positives(TP) had a value of 94 and True Negative(TN) had 141. The ROC curve is 0.90.

Fig 22: Classification report for KNN classifier

```
K Nearest Neighbour
Accuracy: 0.85
Precision: 0.90
Recall: 0.84

              precision    recall  f1-score   support

           0       0.78      0.86      0.82       109
           1       0.90      0.84      0.87       167

    accuracy                           0.85       276
   macro avg       0.84      0.85      0.85       276
weighted avg       0.86      0.85      0.85       276
```

Fig 23: Confusion matrix for KNN classifier

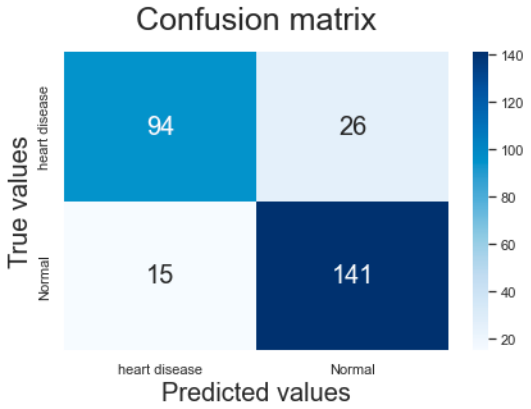
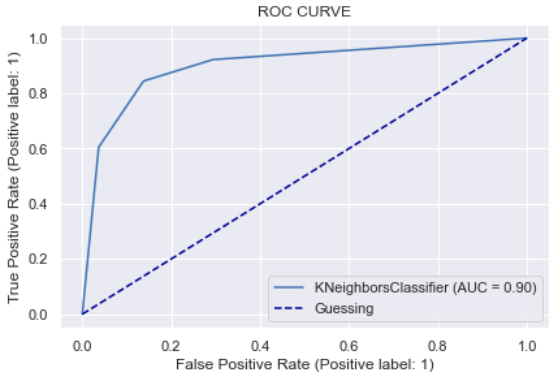


Fig 24: ROC Curve for KNN classifier



Naive Bayes

The results from the Naive Bayes classifier shows that the accuracy of the model is 87%, precision at 92% and recall at 86%. The True Positives(TP) had a value of 90 and True Negative(TN) had 144. The ROC curve is 0.92.

Fig 25: Classification report for Naive Bayes classifier

```
Naive
Accuracy: 0.87
Precision: 0.92
Recall: 0.86
```

	precision	recall	f1-score	support
0	0.81	0.88	0.84	109
1	0.92	0.86	0.89	167
accuracy			0.87	276
macro avg	0.86	0.87	0.87	276
weighted avg	0.87	0.87	0.87	276

Fig 26: Confusion matrix for Naive Bayes classifier

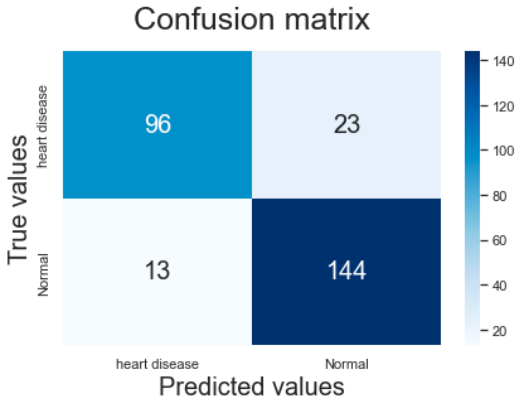
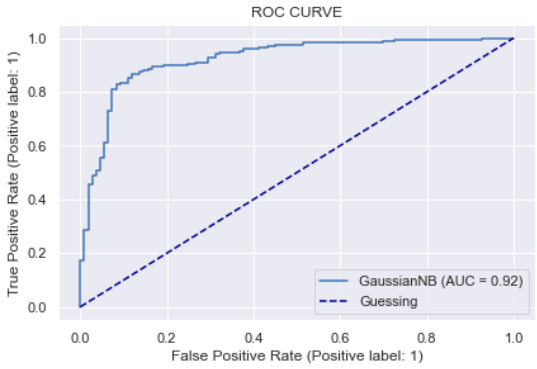


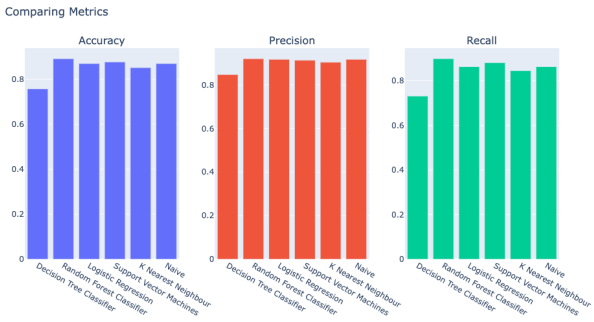
Fig 27: ROC Curve for Naive Bayes classifier



Comparison of the accuracy, precision and recall of the models

The accuracy, precision and recall of the models were plotted to have a visualised representation of their performance.

Fis. 28: Comparison metrics of the models



7. Discussion & Conclusions

The Random Forest Classifier with n_estimator = 900 was chosen as my best model to use at this time because the Area under the curve(AUC) was approximately 0.93, f1 score at approximately 0.91 and the accuracy was 89%. This is the prediction model I have chosen for this project at this time because It had better precision, accuracy and recall and was very good at picking out False Negatives(FN).

During the experimentation I checked the feature importance to see which of the features were important in the model learning process and which wasn't important. ST_Slope and ChestPain Type were the top two features that aided in the learning process while FastingBS and RestingECG had the least importance to the learning process.

8. Recommendation

According to the British Heart Foundation in March 2022, over 7.6 million people suffer from heart or circulatory disease in the United Kingdom with men having a higher rate of 4million and women over 3million cases.

Machine learning applied to the prediction of heart disease can be a useful tool for both the prediction of patient survival and detecting the most critical traits(or risk factors) that may lead to heart failure. Machine learning can be used not only for clinical prediction, but also to improve the performance and accuracy of health professionals

9. References

Ali, L., Niamat, A., Khan, J. A., Golilarz, N. A., Xingzhong, X., Noor, A., Nour, R., & Bukhari, S. A. C. (2019). An Optimized Stacked Support Vector Machines Based Expert System for the Effective Prediction of Heart Failure. *IEEE Access*, *7*(2169-3536), 54007–54014. https://doi.org/10.1109/ACCESS.2019.2909969

Bonthu, H. (2021, July 11). *An Introduction to Logistic Regression*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-logistic-regression/

Dwivedi, R. (2020, May 4). *How Does Support Vector Machine Algorithm Works In Machine Learning? | Analytics Steps*. Www.analyticssteps.com. https://www.analyticssteps.com/blogs/how-does-support-vector-machine-algorithm-works-machine-learning

Harrison, O. (2019, July 14). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761#:~:text=KNN%20works%20by%20finding%20the

https://www.facebook.com/kdnuggets. (2020a). *Decision Tree Algorithm, Explained - KDnuggets*. KDnuggets. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

https://www.facebook.com/kdnuggets. (2020b). *Random Forest® — A Powerful Ensemble Learning Algorithm - KDnuggets*. KDnuggets. https://www.kdnuggets.com/2020/01/random-forest-powerful-ensemble-learning-algorithm.html

Mayo Clinic Staff. (2019). *Top strategies to prevent heart disease*.

Mayo Clinic. https://www.mayoclinic.org/diseases-conditions/heart-disease/in-depth/heart-disease-prevention/art-20046502

Mohan, S., Thirumalai, C., & Srivastava, G. (2019). Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques. *IEEE Access*, *7*, 81542–81554. https://doi.org/10.1109/access.2019.2923707

Nagamani, T., Logeswari, S., & Gomathy, B. (2019). Heart Disease Prediction using Data Mining with Mapreduce Algorithm. In *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* (pp. 2278–3075). https://www.ijitee.org/wp-content/uploads/papers/v8i3/C2624018319.pdf

Palaniappan, S., & Awang, R. (2008). Intelligent Heart Disease Prediction System Using Data Mining Techniques. *IJCSNS International Journal of Computer Science and Network Security*, *8*(8), 343. http://paper.ijcsns.org/07_book/200808/20080849.pdf?origin%3Dpublication_detail

Sunil. (2017, September 11). *Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#:~:text=It%20is%20a%20classification%20technique

Upadhyay, A. (2020, August 9). *What Is Correlation?* Medium. https://medium.com/analytics-vidhya/what-is-correlation-4fe0c6fbed47

Using StandardScaler() Function to Standardize Python Data. (2020, October 26). JournalDev. https://www.journaldev.com/45025/standardscaler-function-in-python

Wikipedia Contributors. (2019, April 1). *Cardiovascular disease*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Cardiovascular_disease

World Health Organization. (2021, June 11). *Cardiovascular diseases (CVDs)*. Www.who.int. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)#:~:text=Cardiovascular%20diseases%20(CVDs)%20are%20the

10. Appendix

Dataset: https://www.kaggle.com/code/francescoaliatis/heart-failure-prediction-logistic-regression-roc/data

Github Repository to jupyter notebook containing the codes https://github.com/EziamakaNwakile/Heart-Failure-Dataset.git

Python codes for the project

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import seaborn as sns

from plotly import subplots
```

```
from plotly.offline import iplot, init_notebook_mode #visualization
import plotly.express as px #visualization
from plotly.subplots import make_subplots #visualization
import plotly.graph_objects as go #visualization
import plotly.offline as py

from sklearn import preprocessing
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import metrics
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# import lightgbm as ltb
import shap

# settings
import warnings
warnings.filterwarnings("ignore")
plt.rcParams['axes.unicode_minus'] = False
plt.style.use('fivethirtyeight')
sns.set(font_scale = 1)
pd.set_option('display.max_columns', None)

print("Import completed")

#loading the dataset
df=pd.read_csv('heart.csv')

df.head(10)

df.shape

df.info()

#Statistical Analysis
df.describe()

# df.isnull().sum()
df_info2 = pd.DataFrame(columns=['Name of Col', 'Num of Null', 'Dtype', 'N_unique'])

for i in range(0, len(df.columns)):
    df_info2.loc[i] = [df.columns[i],

df[df.columns[i]].isnull().sum(),
                   df[df.columns[i]].dtypes,
                   df[df.columns[i]].nunique()]

df_info2

from IPython.core.display import HTML

def multi_table(table_list):
    # Acceps a list of IpyTable objects and returns a table which contains each IpyTable in a cell
    return HTML(
        '<table><tr style="background-color:white;">' +
        ''.join(['<td>' + table._repr_html_() + '</td>' for table in table_list]) +
        '</tr></table>')

cat=['Sex','ChestPainType','FastingBS','RestingECG','ExerciseAngina','ST_Slope','HeartDisease']
df_nunique = {var: pd.DataFrame(df[var].value_counts()) for var in set(cat)}
multi = []
for i in cat:
```

```
multi.append(df_nunique[i].style.format('{:.2f}')
             .highlight_max(color = 'yellow')
             .highlight_min(color = 'turquoise'))
display(multi_table(multi))

for i in cat:
   df_groupby = {var: pd.DataFrame(df.groupby([var, i]).size()) for var in set(cat)}
   multi = []

#Univariate Analysis
df.hist(figsize=(18,12))

#Check the ration of male to female likely to have heart disease
print("Sex Ratio in Data")
print(df["Sex"].value_counts())

plt.pie(df["Sex"].value_counts(),
        labels = ["male", "female"],
        colors = ["lightblue","hotpink"],
        autopct='%1.1f%%',
        explode = [0,0.2],
        startangle = 180,
        shadow = True)

plt.legend()
plt.show()

fig,ax=plt.subplots(1,1,figsize=(10,10))
sns.kdeplot(data=df,x='Age',hue='ChestPainType',shade=True)
plt.show()

sns.pairplot(df,hue='HeartDisease')

sns.pairplot(df)

#Dividing the data into categorical, numerical, continous, and discrete

#to find out columns with categorical data so that i will know which ones to use encoding for
categorical_data=[f for f in df.columns if df[f].dtype=='O']
categorical_data

#to find out columns with numeric data
numerical_data=[f for f in df.columns if df[f].dtype!='O']
numerical_data

#to find out columns with continuos data for EDA and predictions
numerical_data_continuous=[f for f in numerical_data if df[f].nunique()>10]
numerical_data_continuous

#to find out columns with discrete variables
numerical_data_discrete=[f for f in numerical_data if df[f].nunique()<11]
numerical_data_discrete

columns = ['Age', 'Resting_Blood_Pressure', 'Cholesterol', 'Max_Heart_Rate', 'Previous_Peak']

plt.figure(figsize=(20,10))

for n, column in enumerate(numerical_data_continuous):
   ax = plt.subplot(2, 3, n + 1)
   sns.kdeplot(data = df, x = column, palette="Purples", shade=True, hue='HeartDisease')
   plt.grid('on')

plt.subplots_adjust(wspace=0.35, bottom=0.2, hspace=0.4)
plt.suptitle("Distribution Plots for All Quantitative Continious Columns")
plt.show()

plt.figure(figsize=(20,10))
```

```
for n, column in enumerate(numerical_data_continuous):
    ax = plt.subplot(2, 3, n + 1)
    sns.regplot(data = df, x = column, y='HeartDisease', color='blue', logistic=True, ci=None)
    plt.grid('on')

plt.subplots_adjust(wspace=0.35, bottom=0.2, hspace=0.4)
plt.suptitle("Plotting Regression Between each independent Continious Variables & dependent variable")
plt.show()

fig, axes = plt.subplots(nrows=2, ncols=3, figsize = (12, 7))
axes = axes.flatten()

for col, ax in zip(categorical_data, axes):
    ax = sns.countplot(data = df, x = col, ax = ax)
    ax.set_title(f"Distribution of {col}", fontsize=15)
    plt.subplots_adjust(hspace=0.5, wspace=0.3)

plt.show()

colors = ['chocolate', 'purple', 'magenta']

def distribution_plot(column, title):

    fig = plt.figure(figsize=(17, 7))
    grid = GridSpec(nrows=2, ncols=1, figure=fig)
    color = np.random.choice(colors, 1)[0]

    ax0 = fig.add_subplot(grid[0, :])
    ax0.set_title(f'Histogram and BoxPlot of {title}', y=1.05)
    sns.histplot(column, ax=ax0, color=color)
    print(f'Skewness of {title} =======>>> {np.round(column.skew(), 3)}')

    ax1 = fig.add_subplot(grid[1, :])
    plt.axis('off')
    sns.boxplot(x=column, ax=ax1, color=color)


# plot distribution graph for continuos data
for x in numerical_data_continuous:
    distribution_plot(df[x], x)

newDf = df.copy()
newDf['HeartDisease'].replace({0:'Healthy', 1:'At Risk'}, inplace = True)

plt.figure(figsize=(10,30))

for n, column in enumerate(categorical_data):
    ax = plt.subplot(5, 2, n + 1)
    sns.countplot(data=newDf, x=column, hue='HeartDisease', palette='Purples')
    plt.xticks(rotation=80)

    for s in ['top', 'right']:
        ax.spines[s].set_visible(False)

plt.subplots_adjust(wspace=0.50, hspace=1.2, bottom=0.2)
plt.show()

#creating a copy of dataset for the visualization
eda= df.copy()
#change values to make better visualizations
eda['Sex'] = np.where(eda['Sex'] == 'F', 'Female', 'Male')
eda['HeartDisease'] = np.where(eda['HeartDisease'] == 0, 'Normal', 'Heart Disease')
```

```
eda['ExerciseAngina'] = np.where(eda['ExerciseAngina'] == 'N', 'No', 'Yes')
eda["ChestPainType"].replace({'TA': 'Typical Angina', 'ATA': 'Atypical Angina', 'NAP': 'Non-Anginal Pain', 'ASY': 'Asymptomatic'}, inplace= True)
eda["ST_Slope"].replace({'Up': 'Upsloping', 'Down': 'Downsloping',}, inplace= True)

# function for barplots
def barplot_fx(x_col):
    new_df = px.data.tips(x_col)
    text_inplace = f'Heart failure by {x_col}'
    fig = px.histogram(eda, x="HeartDisease",
                color=x_col, barmode='group',
                color_discrete_sequence =colors, opacity=0.9,text_auto=True,
                height=450, width = 450)

fig.update_layout(title_text=text_inplace,title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
    fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,

marker_line_width=1,marker_line_color="black")
    fig.show()

for x in categorical_data:
    barplot_fx(x)

#plot a heatmap using the correlation
sns.heatmap(df.corr(), annot=True)

# scatter plot
sns.scatterplot(x='Cholesterol',y='RestingBP',hue='Sex',data=df)
plt.axvline(129, linestyle = '--', color = "red") #normal level for LDL
plt.axhline(120, linestyle = '--', color = "red") #normal resting BP
plt.show()

# convert some columns to binary
# I can use Label encoder or numpy

encode_df = df.copy()
cat_columns = [cname for cname in encode_df.columns if encode_df[cname].dtype == 'object']

encoder = preprocessing.LabelEncoder()

for col in cat_columns:
    print(col)
    encode_df[col] = encoder.fit_transform(encode_df[col])
    encode_df[col] = encode_df[col].astype('int64')
encode_df.info()

# feature selection
# RandomForestClassifier will be used if the variable is classifiable within yes and no
# RandomForestRegressor will be used if the variable is in a range of numbers

X = encode_df.drop(['HeartDisease'],axis=1)
y = encode_df.HeartDisease

rf_model = RandomForestClassifier(max_depth=7)
rf_model.fit(X,y)
train_features = X.columns
importances = rf_model.feature_importances_
indices = np.argsort(importances)
# Feature Importance
```

```
plt.title('Feature Importance')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [train_features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()

# after label encoding, perform get dumies / one-hot-encoding
X = pd.get_dummies(X, columns=categorical_data)

#Test and Training the dataset
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=1)

#checking the shape of the train and test data
X_train.shape, X_test.shape, y_train.shape, y_test.shape

#initiating standard scaler
scaler = preprocessing.StandardScaler()
#Rescale both sets using the trained scaler
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

def plot_conf_mat(model, X_test, y_test):
    y_pred = model.predict(X_test)
    matrix = confusion_matrix(y_pred, y_test)
    print(classification_report(y_test, y_pred))
    df_cm = pd.DataFrame(matrix, index = ['heart disease', 'Normal'],
                         columns = ['heart disease', 'Normal'])
    plt.figure(figsize = (6,4))
    sns.heatmap(df_cm,
                annot=True,
                cmap='Blues',
                fmt='.5g',
                annot_kws={"size": 20}).set_title('Confusion matrix', fontsize = 25, y=1.05)
    plt.xlabel('Predicted values', fontsize = 20)
    plt.ylabel('True values', fontsize = 20)
    plt.show()


#creating the parameters
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space, 'penalty': ['l1', 'l2']}

classifiers = {
    'Decision Tree Classifier': DecisionTreeClassifier(),
    'Random Forest Classifier': RandomForestClassifier(n_estimators=500, random_state=1),
    'Logistic Regression': LogisticRegression(solver='liblinear',max_iter=10000),
    'Support Vector Machines': SVC(kernel='linear', random_state=42),
    'K Nearest Neighbour': KNeighborsClassifier(n_neighbors=3, leaf_size=5, p=2),
    'Naive': GaussianNB()
}

summary = list()
for name, clf in classifiers.items():
    print(name)
    nada = clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    pred_accuracy = clf.score(X_test,y_test)
    pred_precision = metrics.precision_score(y_test, y_predict)
    pred_recall = metrics.recall_score(y_test, y_predict)

    print(f'Accuracy: {pred_accuracy:.2f}')
    print(f"Precision: {pred_precision:.2f}")
```

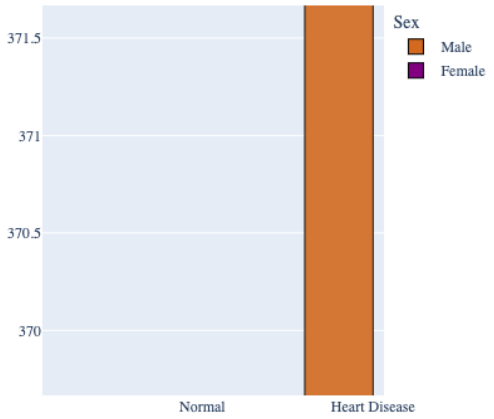
```
    print(f"Recall: {pred_recall:.2f}")
    print()
    plot_conf_mat(clf, X_test, y_test)
    print("\n")

    summary.append({
        'Accuracy':pred_accuracy,
        'Precision':pred_precision,
        'Recall':pred_recall,
        'name': name,
    })

summary = pd.DataFrame(summary)

fig = subplots.make_subplots(rows=1,
cols=3, subplot_titles=(
    'Accuracy', 'Precision', 'Recall'
))

trace0 = go.Bar(x=summary['name'],
y=summary['Accuracy'], name='Accuracy')
fig.append_trace(trace0, 1, 1)

trace1 = go.Bar(x=summary['name'],
y=summary['Precision'], name='Precision')
fig.append_trace(trace1, 1, 2)

trace2 = go.Bar(x=summary['name'],
y=summary['Recall'], name='Recall')
fig.append_trace(trace2, 1, 3)

fig['layout'].update(title='Comparing
Metrics')
fig['layout'].update(showlegend=False)

py.iplot(fig)

# Create Random Forest classifer object
clf =
RandomForestClassifier(n_estimators=900,
random_state=1)

# Train Random Forest Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(
y_test, y_pred))

from sklearn.metrics import
plot_roc_curve

for name, clf in classifiers.items():
    fig=plot_roc_curve(clf,X_test,y_test)
    #
fig=plot_roc_curve(LGR,X_test,y_test,ax=f
ig.ax_)
    #
fig=plot_roc_curve(clf_RF,X_test,y_test,ax
=fig.ax_)
    # Plot line with no predictive power
(baseline)
    plt.plot([0, 1], [0, 1], color='darkblue',
linestyle='--', label='Guessing')
    plt.title("ROC CURVE")
    plt.legend()
    plt.show()
```

Barplot for Sex, ChestPain, ExerciseAngina, HeartDisease and ST_Slope.

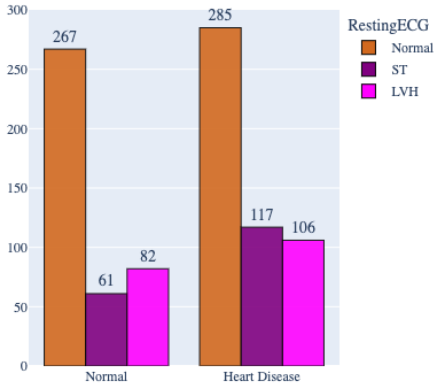
- The results from the barplots shows that: mean and median of "Age" is almost equal - Normally distributed

- mean and median of "RestingBP" is almost equal - Normally distributed
- mean of "Cholesterol" is lesser than its median - Left skewed
- mean and median of "MaxHR" is almost equal - Normally distributed
- mean of "Oldpeak" is greater than its median - Right skewed

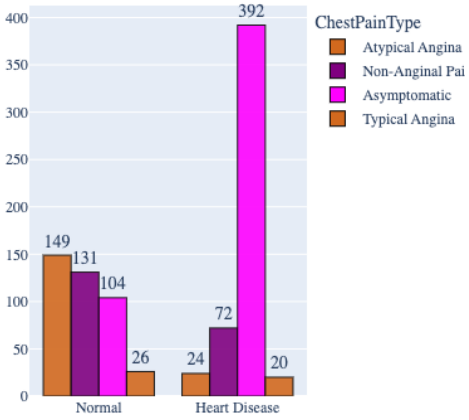
Heart failure by Sex



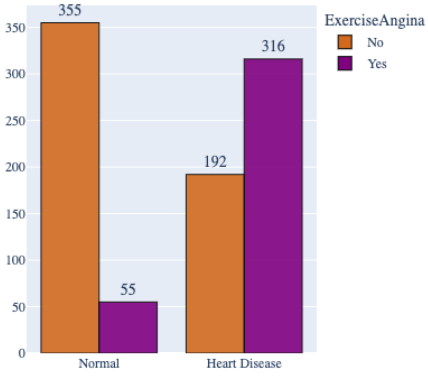
Heart failure by RestingECG



Heart failure by ChestPainType



Heart failure by ExerciseAngina



Heart failure by ST_Slope