

A Project Report on

Heart Disease Classification Model

Team Members:

Rahul Boggavarapu

Vineeth Kolli

Sai Sri Teja Inuganti

Manish Reddy

Sathwik Kumar Rudroju

Vamsi Thorlikonda

Introduction:

You have just been hired as a Data Scientist at a Hospital with an alarming number of patients coming in reporting various cardiac symptoms. A cardiologist measures vitals & hands you this data to perform Data Analysis and predict whether certain patients have heart disease. We would like to make a Machine Learning algorithm where we can train our AI to learn & improve from experience. Thus, we would want to classify patients as either positive or negative for heart disease. We have a data which classified if patients have heart disease or not according to features in it. We will try to use this data to create a model which tries predicting if a patient has this disease or not. We will use logistic regression (classification) algorithm.

- Predict whether a patient should be diagnosed with heart disease. This is a binary outcome.
Positive (+) = 1, patient diagnosed with heart disease
Negative (-) = 0, patient not diagnosed with heart disease
- Experiment with various Classification Models & see which yields greatest accuracy.
- Examine trends & correlations within our data.
- Determine which features are most important to Positive/Negative Heart Disease diagnosis.

It is well known that the libraries available in Python for data loading, management, and building models, such as Pandas, NumPy, and Scikit-Learn, help build robust data science applications. However, when dealing with medical data in data science, data privacy and protection are important parameters that cannot be ignored. The training data and learning models must comply with HIPAA laws for any AI project to be helpful in the real world.

With Python programming language, cybersecurity professionals can efficiently accomplish these aspects in healthcare projects. Beyond data science, Python is also widely used for decoding and sending packets, network scanning, accessing servers, analysing web traffic, and port scanning. Python packages are also robust in conducting a series of cybersecurity tasks, such as malware analysis and scanning in real-time.

Problem Statement:

Develop a heart disease classification model that accurately predicts the presence or absence of heart disease in patients based on their medical attributes and historical data. The model should analyze a set of input features, including age, gender, blood pressure, cholesterol levels, blood sugar levels, and other relevant factors, to provide a reliable assessment of the likelihood of heart disease.

The model's performance should be measured in terms of its accuracy, precision, recall, and F1-score. The primary objective is to create a highly accurate and robust classification model that can assist healthcare professionals in making informed decisions and identifying patients at high risk of heart disease.

Hypothesis Generation:

The hypotheses form the basis for exploring the relationship between various factors and the likelihood of heart disease. By analysing a dataset and evaluating these hypotheses, we can identify significant predictors and develop an accurate heart disease classification model. Some of the hypotheses are:

- Age: Older individuals are more likely to have heart disease due to the cumulative effects of aging on the cardiovascular system.
- Gender: There may be differences in the prevalence and manifestation of heart disease between males and females, with males being more susceptible.
- Blood Pressure: Higher blood pressure levels (systolic and diastolic) may indicate an increased risk of heart disease as hypertension is a known risk factor.
- Cholesterol Levels: Elevated levels of total cholesterol, low-density lipoprotein (LDL) cholesterol, and triglycerides may contribute to the development of heart disease.
- Blood Sugar Levels: High blood sugar levels, especially in individuals with diabetes, may be associated with an increased risk of heart disease.
- Family History: A family history of heart disease might suggest a genetic predisposition to the condition.
- Smoking: Smoking is a well-known risk factor for heart disease, and individuals who smoke may be more likely to develop the condition.

- Obesity: Excess body weight, particularly obesity, can increase the risk of heart disease due to its association with high blood pressure, diabetes, and abnormal cholesterol levels.
- Exercise: Regular physical activity and exercise may lower the risk of heart disease by improving cardiovascular health.

Data Description:

We have a data which classified if patients have heart disease or not according to features in it. We will try to use this data to create a model which tries predicting if a patient has this disease or not. We will use logistic regression (classification) algorithm.

Given data set contains

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital)
5. chol - serum cholesterol in mg/dl
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg - resting electrocardiographic results.
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest
11. slope - the slope of the peak exercise ST segment
12. ca - number of major vessels (0-3) colored by flourosopy
13. thal - 3 = normal; 6 = fixed defect; 7 = reversable defect
14. target - have disease or not (1=yes, 0=no)

Our data has 3 types of data:

Continuous: which is quantitative data that can be measured.

Ordinal Data: Categorical data that has an order to it (0,1,2,3, etc)

Binary Data: data whose unit can take on only two possible states (0 & 1)

Exploratory Data Analysis (EDA):

We will use Python to explore the data to gain a better understanding of the features and target variable. We will also analyse the data to summarize their main characteristics, using various visualization techniques.

This Python 3 environment comes with many helpful analytics libraries installed.

Importing Necessary Libraries:

Plotting Libraries:

```
import NumPy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

Metrics for Classification technique:

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

Scaler:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV, train_test_split
```

Model building:

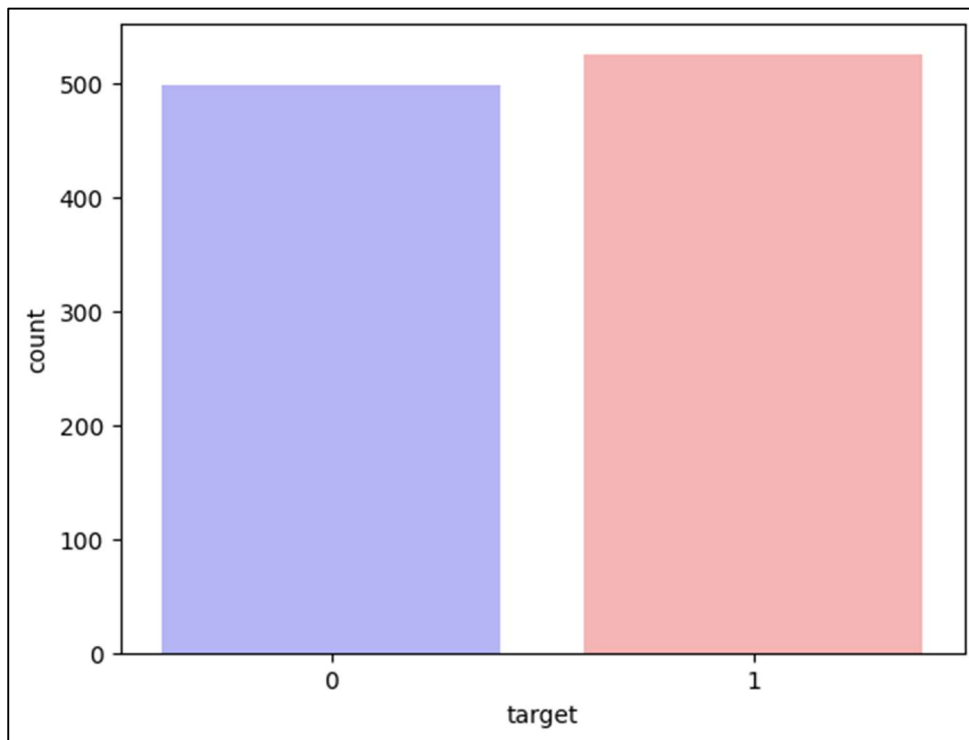
```
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SV
```

By plotting a graph between target and count to find the no of males and females it looks like

Code:

```
sns.countplot(x="target", data=df, palette="bwr")  
plt.show()
```

Graph:



To find percentage of male patients and female patients

Code:

```
countNoDisease = len(df[df.target == 0])  
countHaveDisease = len(df[df.target == 1])  
print("Percentage of Patients not having Heart Disease: {:.2f}%".format((countNoDisease /  
(len(df.target))*100)))  
print("Percentage of Patients having Heart Disease: {:.2f}%".format((countHaveDisease /  
(len(df.target))*100)))
```

Output:

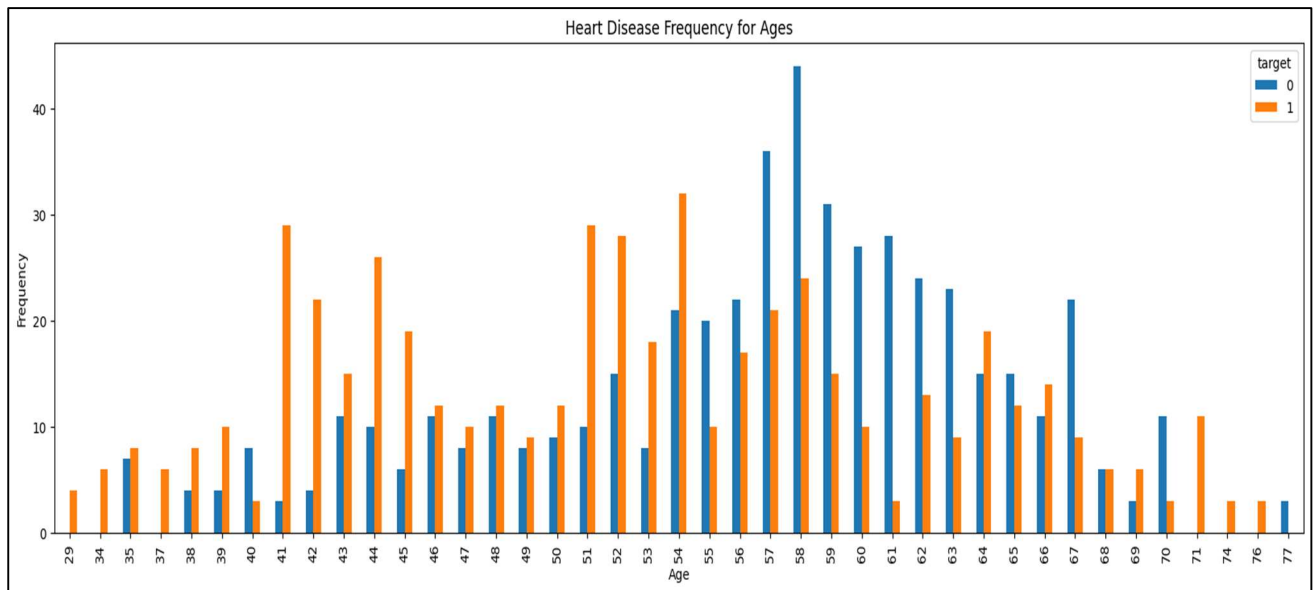
Percentage of Patients not having Heart Disease: 48.68%

Percentage of Patients having Heart Disease: 51.32%

Frequency of ages that the heart disease occurs

Code:

```
pd.crosstab(df.age,df.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('heartDiseaseAndAges.png')
plt.show()
```

Graph:

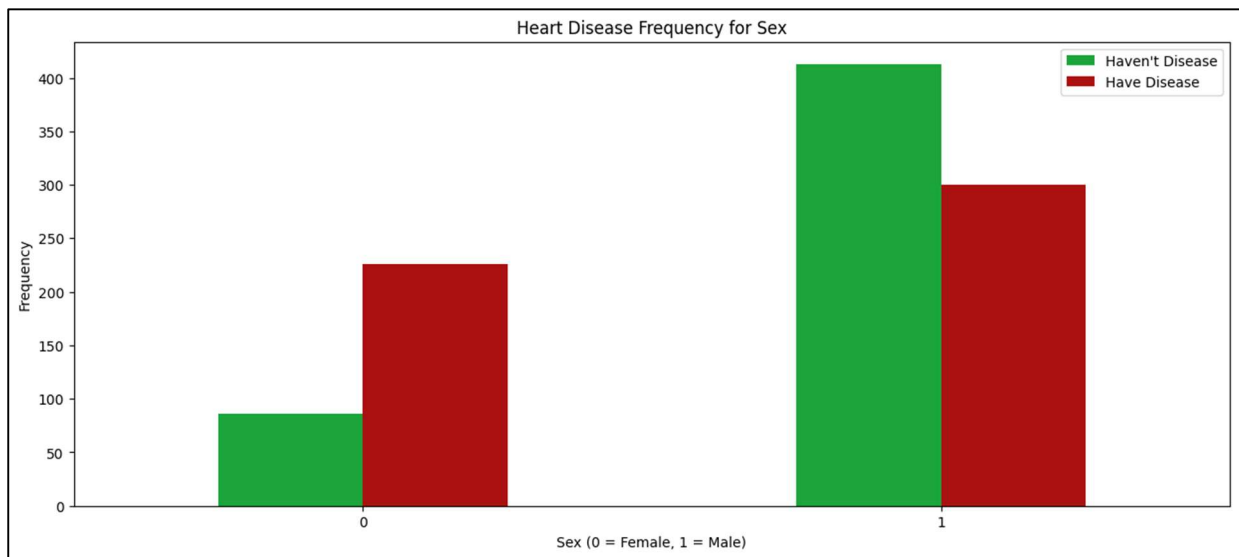
From the given dataset and by observing the graph we can tell that the heart disease will occur at the age of 58 mostly.

We can also classify the heart disease frequency for both male and female

Code:

```
pd.crosstab(df.sex,df.target).plot(kind="bar",figsize=(15,6),color=['#1CA53B','#AA1111'])
plt.title('Heart Disease Frequency for Sex')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.xticks(rotation=0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency')
plt.show()
```

Graph:

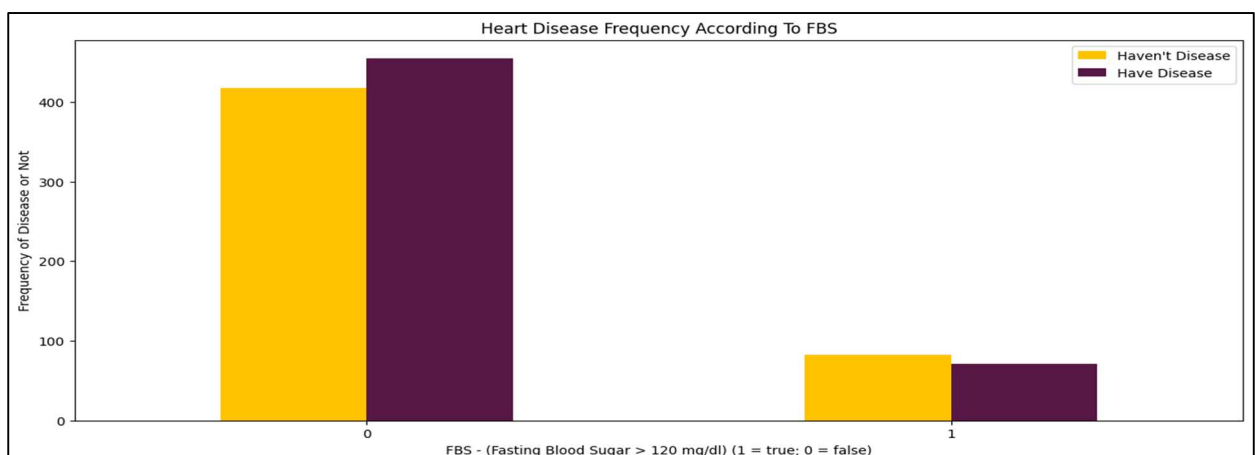


We can also classify based on fasting blood sugar if we have more than the 120 then they may have chances to get heart disease.

Code:

```
pd.crosstab(df.fbs,df.target).plot(kind="bar",figsize=(15,6),color=['#FFC300','#581845' ])
plt.title('Heart Disease Frequency According To FBS')
plt.xlabel('FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
plt.xticks(rotation = 0)
plt.legend(['Haven't Disease', 'Have Disease'])
plt.ylabel('Frequency of Disease or Not')
plt.show()
```

Graph:



Algorithms:

We can use different algorithms to classify and visualise the data and we can predict which algorithm is best based on the accuracy.

- Since 'cp', 'thal' and 'slope' are categorical variables we'll turn them into dummy variables.

Logistic Regression:

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.

Creating Model for Logistic Regression:

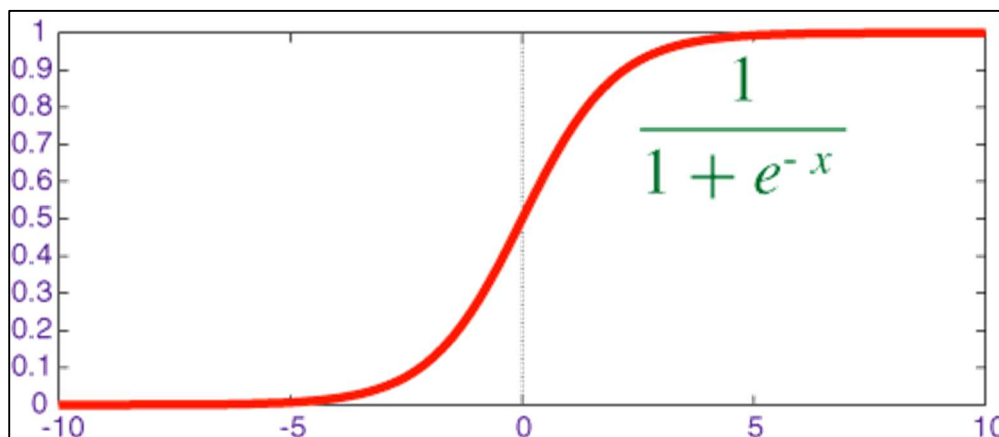
We can use sklearn library or we can write functions ourselves. Let's them both. Firstly, we will write our functions after that we'll use sklearn library to calculate score.

```
y = df.target.values
x_data = df.drop(['target'], axis = 1)
```

Normalize data:

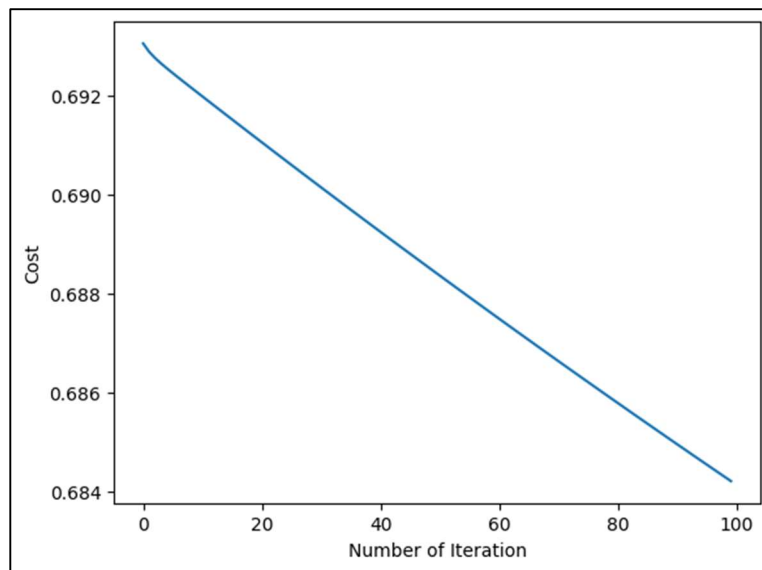
```
x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data))
```

Sigmoid Function:



```
def sigmoid(z):
    y_head = 1/(1+ np.exp(-z))
    return y_head
```

By using logistic regression, we get a n accuracy of 63.41%.



Manuel Test Accuracy: 63.41%

Sklearn Logistic Regression:

Basically, it measures the relationship between the categorical dependent variable and one or more independent variables by estimating the probability of occurrence of an event using its logistics function. sklearn. linear_model. Logistic Regression is the module used to implement logistic regression.

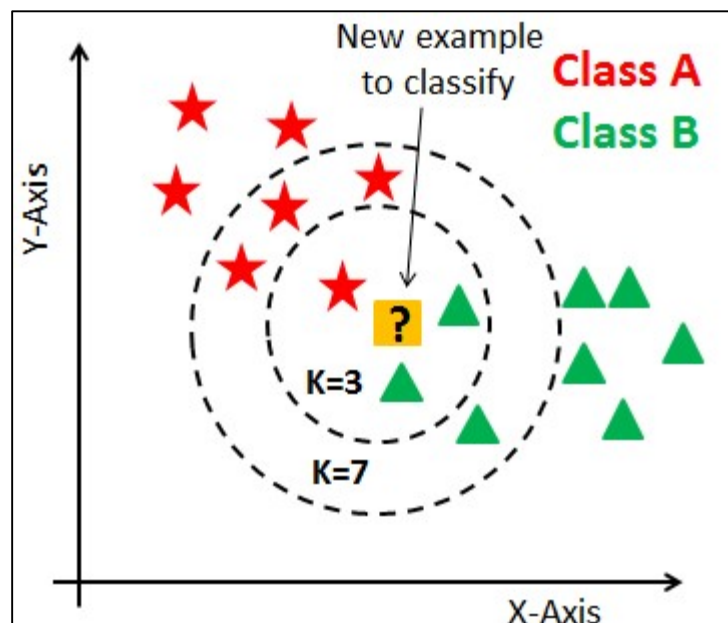
Code:

```
accuracies = {}  
lr = LogisticRegression()  
lr.fit(x_train.T,y_train.T)  
acc = lr.score(x_test.T,y_test.T)*100  
accuracies['Logistic Regression'] = acc  
print("Test Accuracy {:.2f}%".format(acc))
```

By using Sklearn Logistic Regression we get an accuracy of 72.68%.

K-Nearest Neighbour (KNN) Classification:

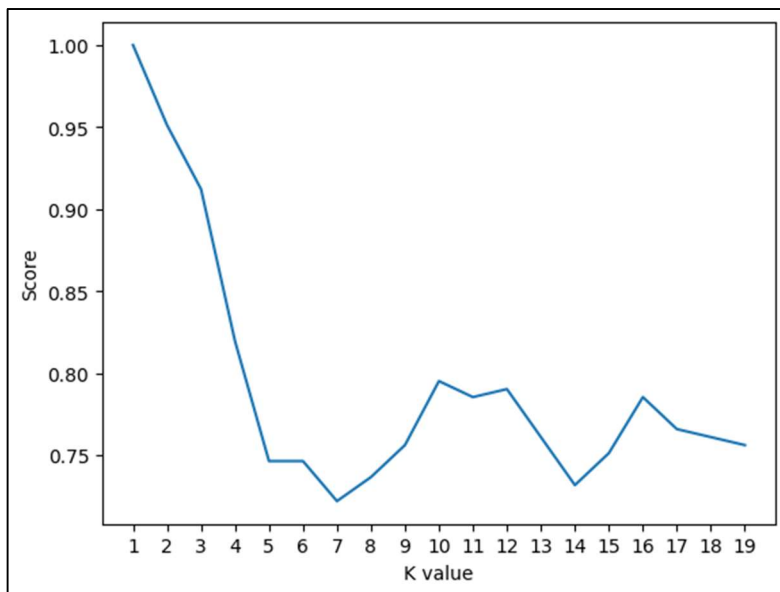
K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories-KNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm. KNN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.



Code:

```
# try to find best k value
scoreList = []
for i in range(1,20):
    knn2 = KNeighborsClassifier(n_neighbors = i) # n_neighbors means k
    knn2.fit(x_train.T, y_train.T)
    scoreList.append(knn2.score(x_test.T, y_test.T))
plt.plot(range(1,20), scoreList)
plt.xticks(np.arange(1,20,1))
plt.xlabel("K value")
plt.ylabel("Score")
plt.show()
acc = max(scoreList)*100
```

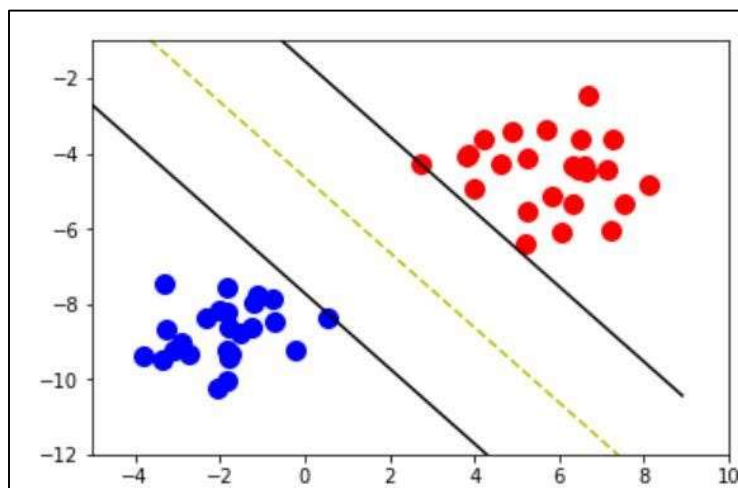
```
accuracies['KNN'] = acc
print("Maximum KNN Score is {:.2f}%".format(acc))
```



Maximum KNN Score is 100.00%.

Support Vector Machine (SVM) Algorithm:

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships.



Code:

```

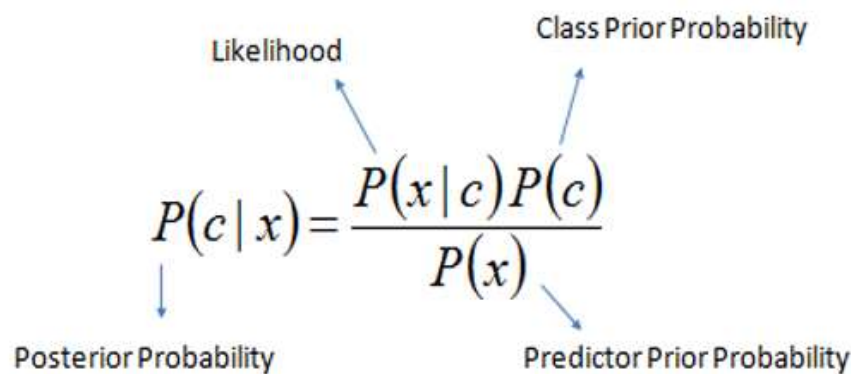
from sklearn.svm import SVC
svm = SVC(random_state = 1)
svm.fit(x_train.T, y_train.T)
acc = svm.score(x_test.T, y_test.T)*100
accuracies['SVM'] = acc
print("Test Accuracy of SVM Algorithm: {:.2f}%".format(acc))

```

By using SVM algorithm test accuracy will be 74.63%.

Naïve Bayes Algorithm:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.



The diagram shows the formula for Posterior Probability: $P(c | x) = \frac{P(x | c)P(c)}{P(x)}$. Arrows point from the terms to their respective labels: $P(c | x)$ points to 'Posterior Probability', $P(x | c)$ points to 'Likelihood', $P(c)$ points to 'Class Prior Probability', and $P(x)$ points to 'Predictor Prior Probability'.

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Code:

```

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train.T, y_train.T)
acc = nb.score(x_test.T, y_test.T)*100
accuracies['Naive Bayes'] = acc

```

```
print("Accuracy of Naive Bayes: {:.2f}%".format(acc))
```

By using Naïve bayes algorithm accuracy will be 88.29%.

Decision Tree Algorithm:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. The decisions or the test are performed based on features of the given dataset.

Code:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train.T, y_train.T)
acc = dtc.score(x_test.T, y_test.T)*100
accuracies['Decision Tree'] = acc
print("Decision Tree Test Accuracy {:.2f}%".format(acc))
```

By using decision tree algorithm accuracy will be 100%.

Random Forest Classification:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in MLAs the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Code:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 1000, random_state = 1)
```

```
rf.fit(x_train.T, y_train.T)
acc = rf.score(x_test.T, y_test.T)*100
accuracies['Random Forest'] = acc
print("Random Forest Algorithm Accuracy Score : {:.2f}%".format(acc))
```

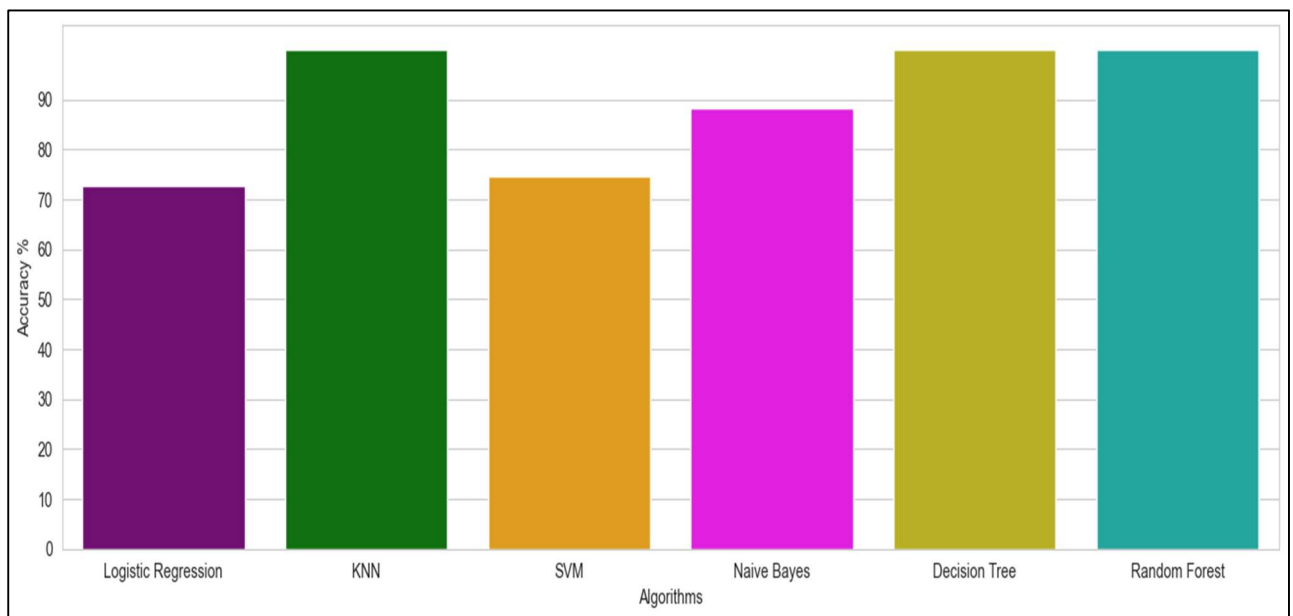
By using Random Forest classification accuracy will be 100%.

Comparing all algorithms:

Code:

```
colors = ["purple", "green", "orange", "magenta", "#CFC60E", "#0FBBAE"]
sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,100,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
plt.show()
```

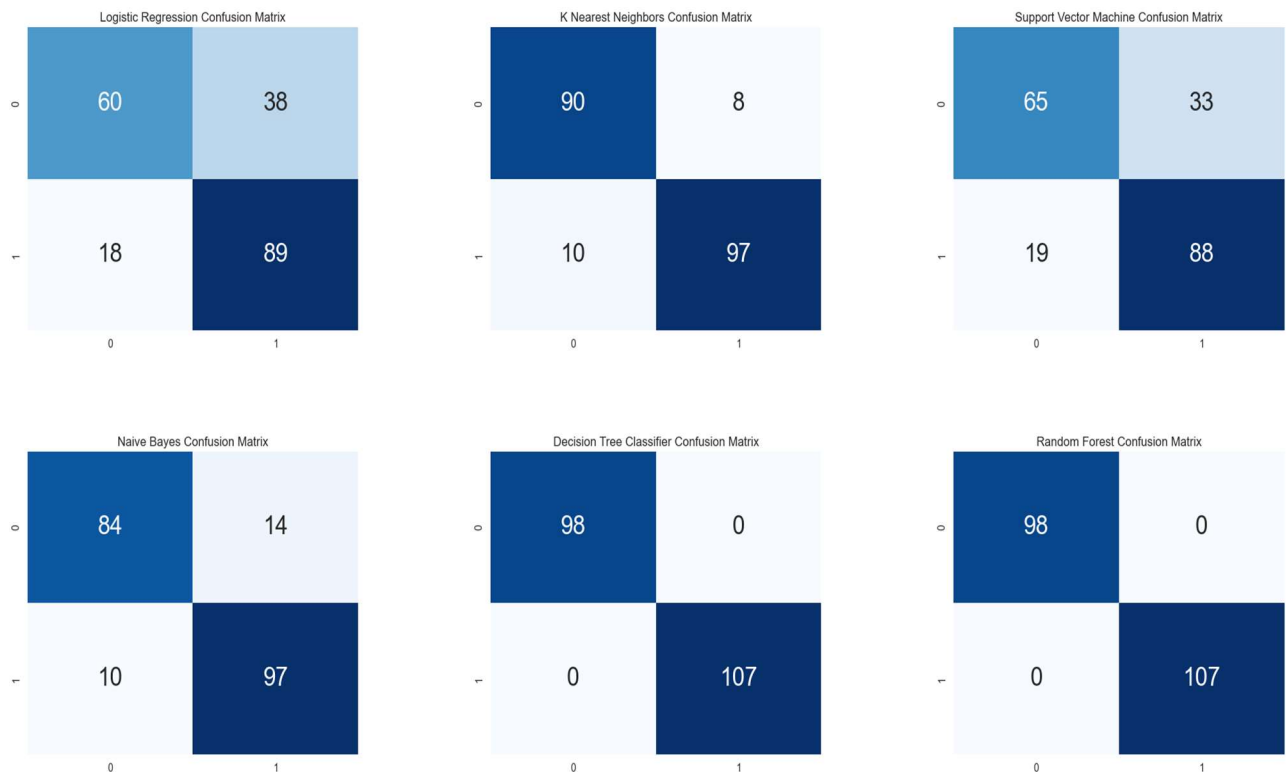
Graph:



By comparing all the algorithms, we can select KNN, Decision tree, Random Forest because they got 100% accuracy.

Confusion Matrix:

Confusion Matrixes



Business recommendation:

Implementing a heart disease classification model can bring several benefits to both businesses and society. Here are some ways in which the project can have a positive impact:

- **Improved Healthcare Decision-making:** The heart disease classification model can aid healthcare professionals in accurately diagnosing heart diseases. By using machine learning algorithms to analyze patient data, doctors can make better-informed decisions about treatment plans and interventions, leading to improved patient outcomes and reduced medical errors.
- **Early Detection and Prevention:** The model can help identify potential cases of heart disease at an early stage, enabling early intervention and preventive measures. Early detection can significantly reduce the risk of complications and mortality, leading to better overall public health.

- **Cost Savings for Healthcare Providers and Patients:** By enabling early detection and appropriate treatment, the model can potentially reduce the overall healthcare costs associated with heart diseases. This benefits both healthcare providers and patients, as it may lead to fewer hospitalizations and expensive medical procedures.
- **Personalized Medicine:** Machine learning algorithms can analyze vast amounts of patient data to identify patterns and correlations that might not be evident to human practitioners. This can lead to the development of personalized treatment plans tailored to each patient's specific needs, optimizing the effectiveness of therapies and medications.
- **Medical Research and Insights:** The data collected and analyzed by the heart disease classification model can contribute to medical research and provide valuable insights into heart diseases' underlying causes and risk factors. This, in turn, can drive advancements in medical knowledge and the development of new treatments.
- **Public Health Management:** Aggregated data from the heart disease classification model can be used by public health authorities to track and monitor trends in heart disease prevalence and distribution. This information can be used to implement targeted public health campaigns, create awareness, and allocate resources efficiently.
- **Increased Access to Healthcare:** With advancements in technology, the heart disease classification model can be deployed in various settings, including remote or underserved areas with limited access to healthcare facilities. This can expand the reach of healthcare services and improve health outcomes for vulnerable populations.
- **Competitive Advantage for Healthcare Providers:** Healthcare institutions that adopt advanced technologies like AI-driven heart disease classification models can gain a competitive edge in the industry. Patients are more likely to choose providers with advanced diagnostic capabilities and a track record of better outcomes.
- **Ethical Considerations:** It is crucial to address ethical considerations surrounding the use of AI in healthcare, such as data privacy, bias, and transparency. By developing and implementing responsible AI solutions, businesses can build trust with patients and society, enhancing their reputation and long-term sustainability.
- **Continuous Learning and Improvement:** As the heart disease classification model is deployed and used, it can continue to learn and improve its accuracy through feedback and updated data. This iterative learning process can lead to a more refined and robust model over time, further benefiting both businesses and society.

Conclusion:

Out of the 13 features we examined, the top 4 significant features that helped us classify between a positive & negative Diagnosis were chest pain type (cp), maximum heart rate achieved (thalach), number of major vessels (ca), and ST depression induced by exercise relative to rest (old peak). Our machine learning algorithm can now classify patients with heart disease. Now we can properly diagnose patients & get them the help they need to recover. By diagnosing detecting these features early, we may prevent worse symptoms from arising later. Our K-Nearest Neighbour (KNN) Classification yields the highest accuracy, 100%. Any accuracy above 70% is considered good but be careful because if your accuracy is extremely high, it may be too good to be true.