

**Heart Attack Analysis and Prediction**

**By**

**Parva Chowdhury: 210911178**

# Index

**1.Introduction ……………………………………..……..02**

**2.Methodology ………………………………….……..,…02**

**3.Results & Discussions ………………….………...……09**

**4.Conclusions ………………...……………..………..…13**

**5.References …………………………………….……..…13**

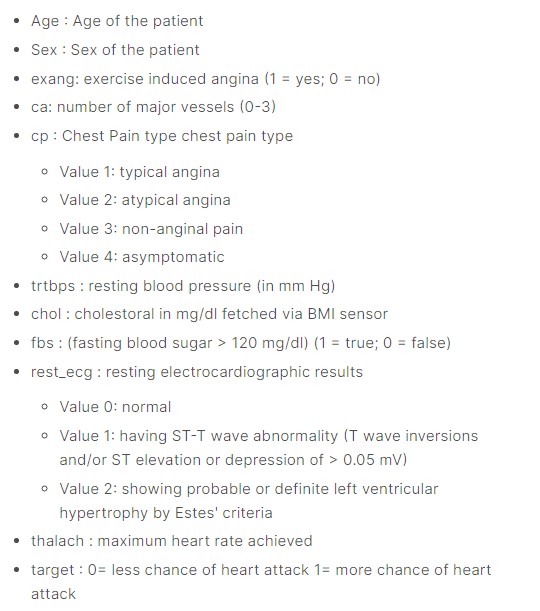
## Introduction

One of the main causes of the rising death rate is heart disease. One of the most significant areas to benefit from extensive information & analytics is healthcare. To forecast and treat the high death rate from heart attacks, medical data extraction is becoming increasingly important. Every day, terabytes of data are generated. To prevent making bad therapeutic judgments that have unfavourable effects, quality services are required. The cost of clinical testing can be reduced by the hospitals using suitable decision support systems. Hospital information systems are now used by hospitals to manage patient data. The vast volume of data produced by the healthcare sector is not efficiently utilized. To cut costs and anticipate heart disease, a novel strategy is required. To determine which data mining techniques are efficient and accurate, this paper will review several research projects on heart attack finding and categorization using various data mining techniques.

By using several data mining techniques like correlation matrix, KMeans clustering, Decision Tree classifier and Naive Bayes, we came up with a fast, efficient and fairly accurate model to predict if someone is at risk of a heart attack.

## Methodology

The dataset we’ve chosen to go ahead with contains the data of 303 patients and has a wide range of attributes which can help us analyze the data and make appropriate predictions. The attributes are as follows.



The data set mentioned can be found in Kaggle at:

<https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>

The data set has both continuous and non continuous attributes. They are as follows:

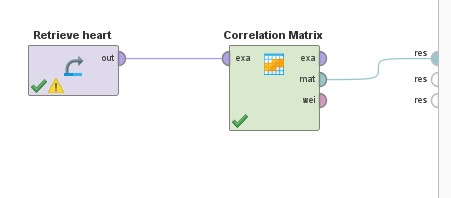
NonContinuousColumns:'sex','exng','caa','cp','fbs','restecg','slp',' thall'

Continuous Columns: ["age","trtbps","chol","thalachh","oldpeak"]

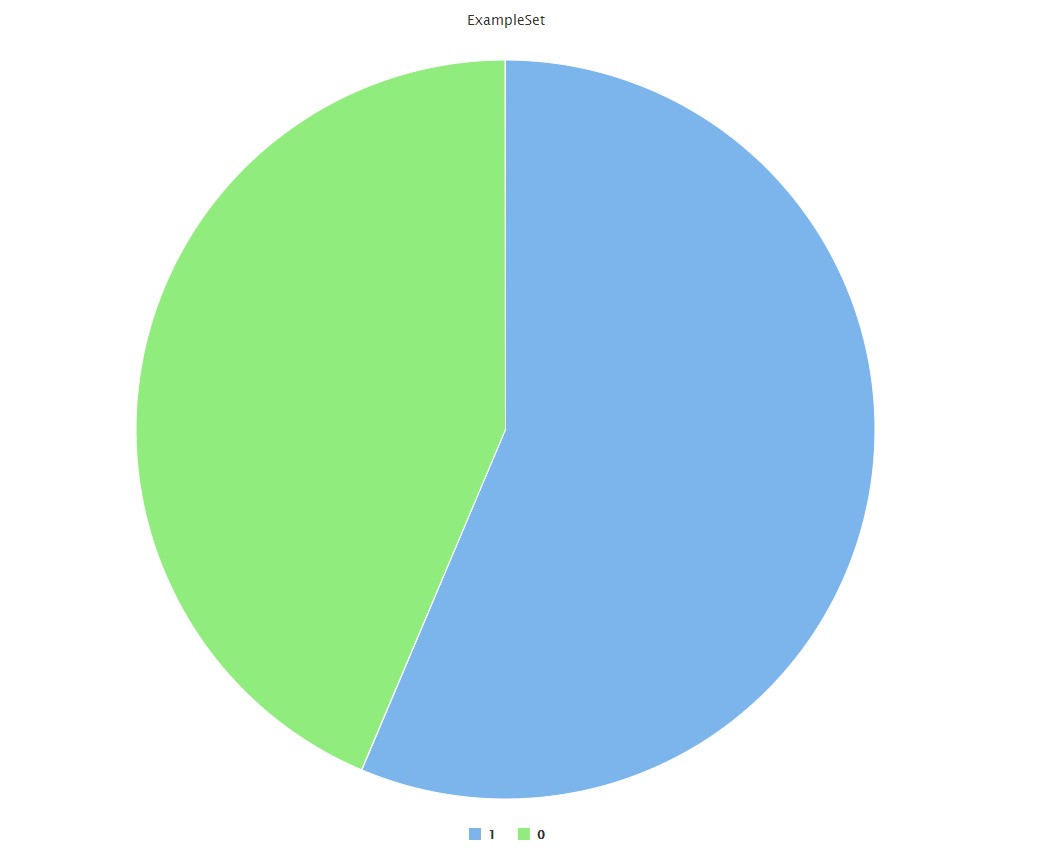
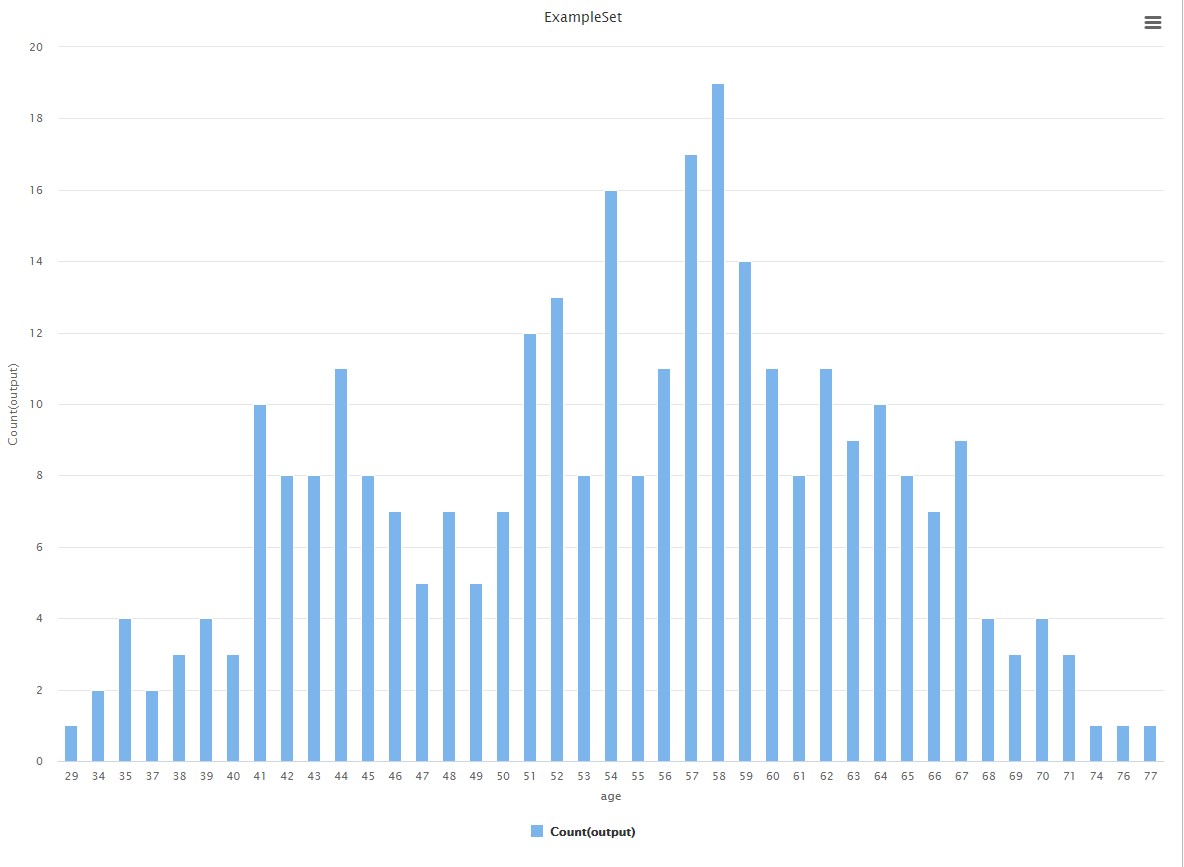
### Data Preprocessing

* There are no NULL values in the data, so there is no need to replace anything with the mean value
* The Outliers in the data are very less. (Checked using Rapid Miner)
* There is no linear correlation between continuous variables. (Checked by using Rapid Miner Correlation Matrix)





By general sense we might think that older people might have higher chances for heart attack but on seeing the data it says otherwise. Most heart attacks happened in the middle aged people.



Male vs Female Positve Outcome, again almost equal results. (some bias because there were more male records in the dataset

Adding the csv data to a dataframe using Pandas.

df1 = df

df1 = pd.get\_dummies(df1, columns = cat\_cols, drop\_first = **True**) X = df1.drop(['output'],axis=1) y = df1[['output']] scaler = RobustScaler() X[con\_cols] = scaler.fit\_transform(X[con\_cols]) print("The first 5 rows of X are") X.head()

We have chosen to go ahead with 3 algorithms because they are quite intuitive and different from one another

### 1) Decision Tree

We’ll also be using a popular and accurate classifier to predict classes for our dataset. For the above-mentioned, we are using the Decision Trees. It is a supervised learning algo used for classification of tuples.

The algorithm focuses on reducing the entropy of the entire dataset by splitting labelled attributes into their classes, starting from the attribute which has the highest entropy and going to the attribute which has the highest entropy in the reduced dataset. Thus, we calculate the entropy/information gain of each attribute to decide the root node and then split the data according to their unique labels, and then proceed to split the reduced data further based on the remaining attributes.

For performing the above, we use a custom decision tree code written from scratch in python .

The above attributes are used to construct the decision tree using the below code. (After importing all the necessary libraries and retrieving the dataset, the following snippet fetches us the decision tree) The code is as follows:

**from** sklearn **import** tree dt = DecisionTreeClassifier(random\_state = 42) dt.fit(X\_train, y\_train)

y\_pred = dt.predict(X\_test) text\_representation = tree.export\_text(dt) print(text\_representation)

print("The test accuracy score of Decision Tree is ", accuracy\_score(y\_test, y\_pred))

After observing the output, we can classify any new tuple based on their class in other attributes. This is how we implement decision tree on our dataset and classify them in different classes. After appropriate tree pruning we get the following tree

### 2) Random Forest Algorithm

It is the most potent and well-known algorithm in ML. It is a part of supervised computer learning.It is applied to classification and regression issues. The steps in the random forest method are as follows:

* Information is gathered
* Decision trees are built using various samples
* It uses the decision tree average.

Although it can handle datasets with categorical variables, it is slower than a single decision tree. ignores values that are missing.

Utilising Random Forest offers the primary benefit of having higher accuracy and lower volatility than using a single model. Choice matrix is used. Based on the parameters in the dataset, data pre-processing is completed. The quantity of trees directly affects the accuracy of the outcome, thus if there are more trees, the accuracy will be higher. Using this classifier, the model is now being trained. We obtain a predicted outcome from each tree as well as an independently computed amount of entropy. The most voted prediction result is selected.

rf = RandomForestClassifier() rf.fit(X\_train, y\_train) y\_pred = dt.predict(X\_test) print("The test accuracy score of Random Forest is ", accuracy\_score(y\_test, y\_pred))

Now we find the attribute weight of random forest using decision tree criterion:

**Gain Ratio Accuracy Information Gain**



#### 3) Support Vector Machine (SVM)

Support Vector Machine is very well known precision and supervised learning technique. It is used mostly for MLClassification issues.

In order to quickly classify following data points, the SVM method aims to define the optimum decision boundary or line that may split n-dimensional spaces into groups.

This ideal decision boundary is referred to as a "hyperplane".

Advantages:

* Support vector machines' benefits include:
* efficient in high-dimensional environments.
* Especially useful in situations where the number of dimensions exceeds the number of samples.
* It is also computationally efficient since it only uses a portion of the training points (known as support vectors) in the decision function.
* Different Parameters may be given for the decision function, making it flexible.

Before Tuning

|  |
| --- |
| **from** sklearn **import** tree dt = DecisionTreeClassifier(random\_state = 42) dt.fit(X\_train, y\_train)  y\_pred = dt.predict(X\_test) text\_representation = tree.export\_text(dt) print(text\_representation)  print("The test accuracy score of Decision Tree is ", accuracy\_score(y\_test, y\_pred)) |

Accuracy: 86.89%

After Tuning

Takes some hit on computational costs

|  |
| --- |
| svm = SVC()  parameters = {"C":np.arange(1,10,1),'gamma':[0.00001,0.00005, 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5]} searcher = GridSearchCV(svm, parameters)  searcher.fit(X\_train, y\_train) print("The best params are :", searcher.best\_params\_) print("The best score is :", searcher.best\_score\_)  y\_pred = searcher.predict(X\_test) print("The test accuracy score of SVM after hyper-parameter tuning is ", accuracy\_score(y\_test, y\_pred)) |

Accuracy: 90.166%

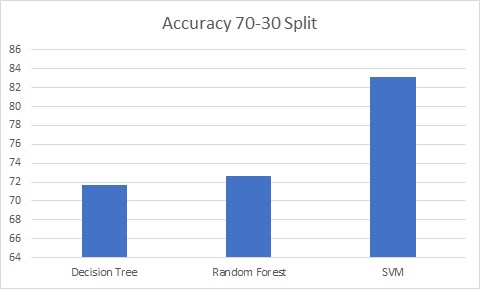
## Results and Discussions

The best accuracies obtained are as follows:

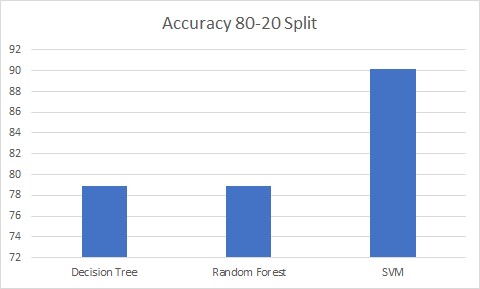
1. **Support vector machine algorithm is 90.166%.**
2. **Decision tree algorithm is 78.8666%.**
3. **Random Forest tree algorithm is 78.877%.**

### Accuracies obtained with different Training - Testing Splits

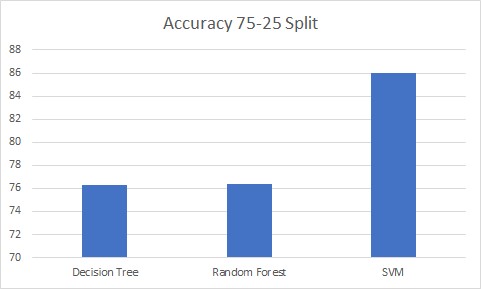
**70-30 Split**



**80-20 Split**



**75-25 Split**



### Accuracies Obtained with Multiple Parameter Combinations (Decision Tree and Random Forest)

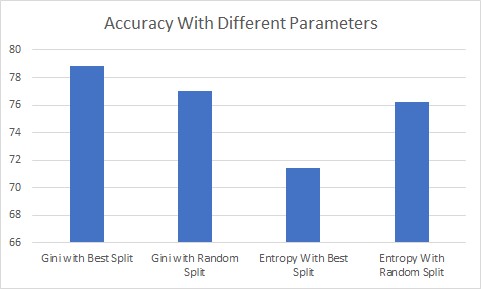
Decision Tree With Parameters Accuracy

Gini with Best Split 78.86

Gini with Random Split 77

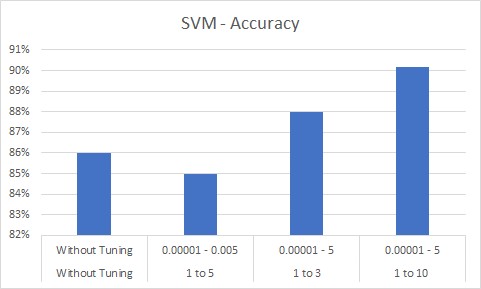
Entropy With Best Split 71.433

Entropy With Random Split 76.23



### Accuracies Obtained with Multiple Parameter Combinations (SVM Hyper Tuning)

|  |  |  |
| --- | --- | --- |
| C range | Gamma Ranges | Accuracy |
| Without Tuning | Without Tuning | 86% |
| 1 to 5 | 0.00001 - 0.005 | 85% |
| 1 to 3 | 0.00001 - 5 | 88% |
| 1 to 10 | 0.00001 - 5 | 90.16% |



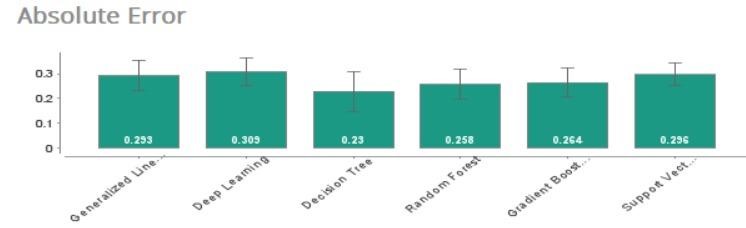
### Confusion Matrix Obtained

**Decision Tree (Best Case)**

|  |  |
| --- | --- |
| **25** | **4** |
| **9** | **23** |
| **Random Forest (Default)** |  |
| **25** | **4** |
| **9** | **23** |
| **SVM (Linear Kernel) (No Tuning)** |  |
| **26** | **3** |
| **5** | **27** |
| **SVM (RBF Kernel) Hyper Parameter Tuning** |  |
| **27** | **2** |
| **7** | **25** |

On comparing all the three models we can easily say that SVM is far more accurate than the remaining models

### Comparision with other algorithms with Rapid Miner AutoModel



Our hyper tuned SVM Model gave a better accuracy then Rapid Miner AutoModel

## Conclusion

These learning methods divide the provided information into various categories and, if an unknown sample is provided as input, forecasts the risk of heart disease. Medical students can use the system as a training tool. Additionally, it will be beneficial for doctors. We have created a generalised system, so in the future we may use it to analyse various datasets by just changing the name of the dataset file provided for the training module.

## References

**1)**[**https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-predicti on-dataset**](https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset) **: kaggle dataset link**

**2)** [**Heart Attack Probability Analysis Using Machine Learning**](https://ieeexplore.ieee.org/document/9663631)

A. A. Shanbhag, C. Shetty, A. Ananth, A. S. Shetty, K. Kavanashree Nayak and B. R. Rakshitha,

"Heart Attack Probability Analysis Using Machine Learning

**3)** [**Cluster Based Association Rule Mining For Heart**](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.299.3658&rep=rep1&type=pdf)[**Attack Prediction**](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.299.3658&rep=rep1&type=pdf)

*Akhil, Jabbar & Dr, P. & Deekshatulu, B.L.. (2011). Cluster based Association rule mining for Heart Attack Prediction. 32. 196-201.*

Remaining References are presented in Literature Survey.