CLASSIFIER ALGORITHMS:

# **Correlation Matrix**

Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables (e.g. height and weight)

Store numeric variables in cname variable

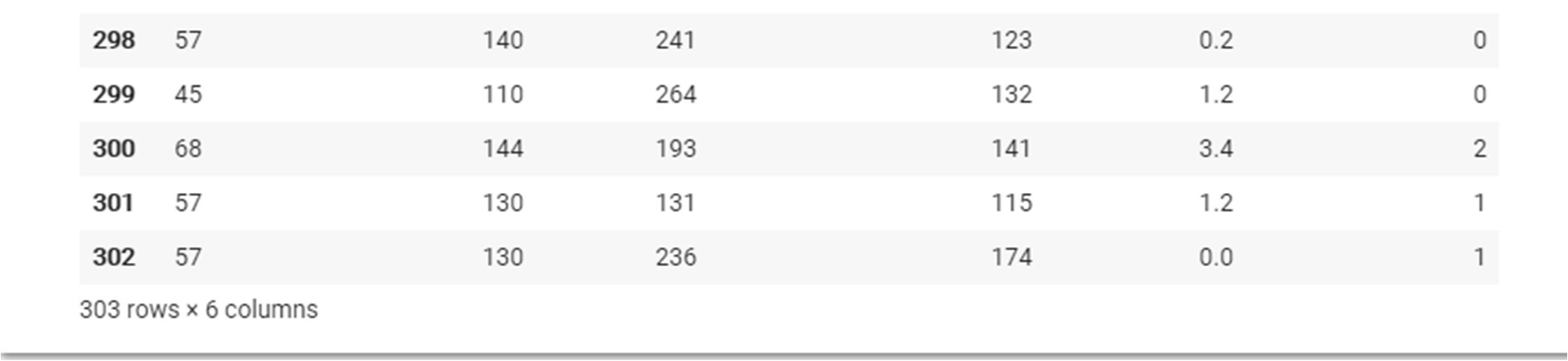
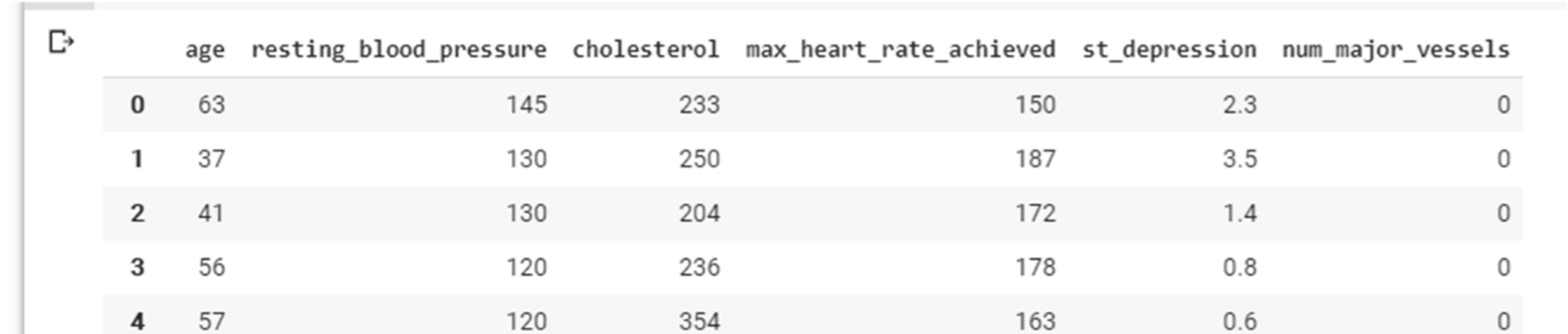


Let’s plot the Correlation plot



Figure 28 Correlation Matrix

**Correlation Analysis:**

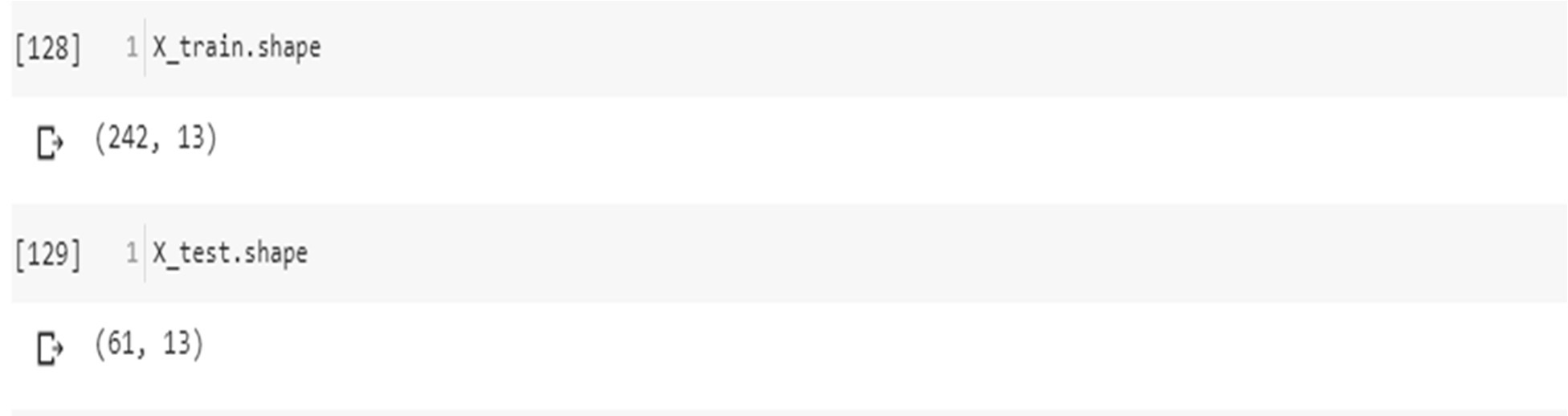


There is no single feature that has a very high correlation with our target value. Also, some of the features have a negative correlation with the target value and some have positive.

**Data Preparation**

Total Among 303 data’s randomly 242 are chosen for Training and 61 are chosen for Testing

.



# ***Modelling and predicting with Machine Learning***

The main goal of the entire project is to predict heart disease occurrence with the highest accuracy. In order to achieve this, we will test several classification algorithms. This section includes all results obtained from the study and introduces the best performer according to accuracy metric. We have chosen several algorithms typical for solving supervised learning problems throughout classification methods.

Modelling and predicting with Machine Learning

The main goal of the entire project is to predict heart disease occurrence with the highest accuracy. In order to achieve this, we will test several classification algorithms. This section includes all results obtained from the study and introduces the best performer according to accuracy metric. We have chosen several algorithms typical for solving supervised learning problems throughout classification methods.

First of all, let's equip ourselves with a handy tool that benefits from the cohesion of SciKit Learn library and formulate a general function for training our models. The reason for displaying accuracy on both, train and test sets, is to allow us to evaluate whether the model overfits or underfits the data (so-called bias/variance tradeoff).



***Logistic Regression***

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Types of logical regression:

* Binary (Pass/Fail)
* Multi (Cats, Dogs, Sheep)

Sigmoid function

|  |  |
| --- | --- |
| S(z)=1/1+e^−z |  |
|  |

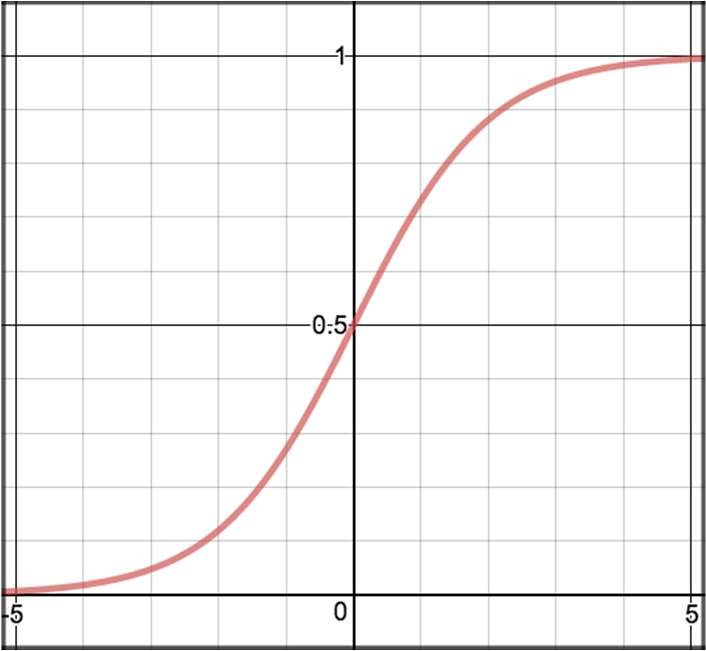


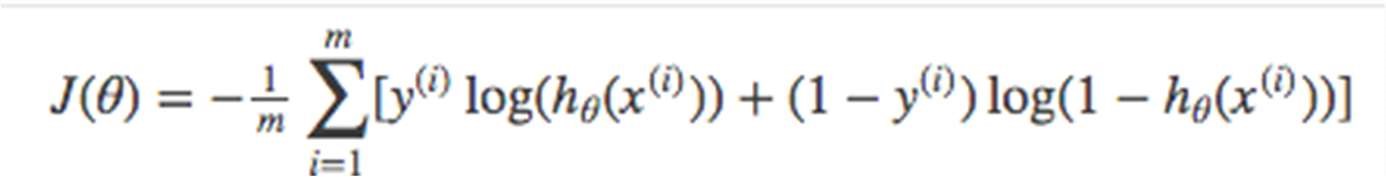
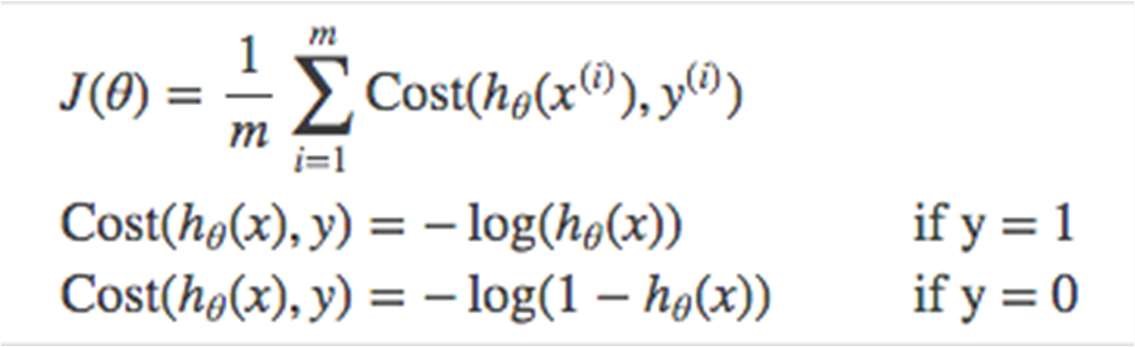
Figure 29 sigmoid function

Decision Boundary

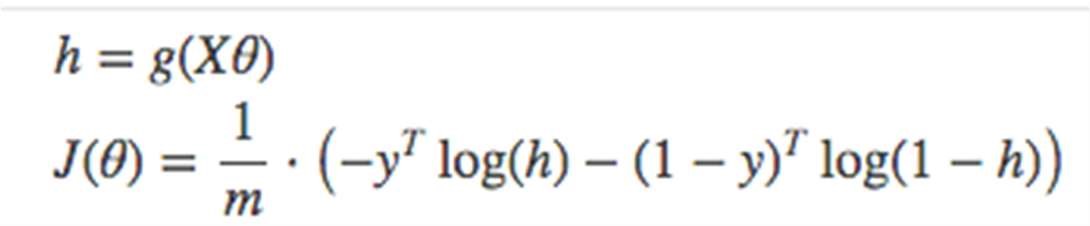
|  |  |  |
| --- | --- | --- |
|  | p≥0.5, class=1 p<0.5, class=0 |  |
|  |

Figure 30 Decision Boundary

Cost Function



Vectorized cost function



For Multiclass - Instead of y=0,1 we will expand our definition so that y=0,1...n. Basically we re-run binary classification multiple times, once for each class.

Procedure -

1

.

Divide the problem into n+1 binary classification problem (+1 because the index starts at

0).

2

.

For each class…

3

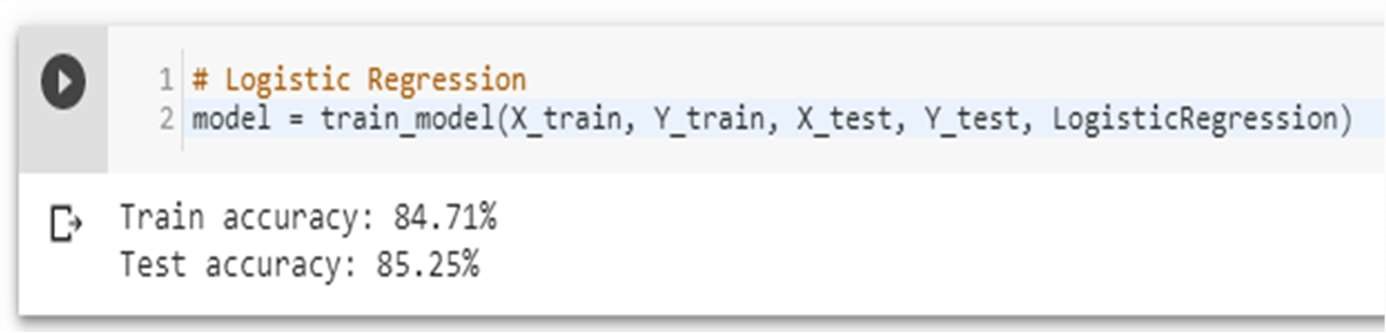
.

Predict the probability the observations are in that single class.

4

.

prediction = <math>max (probability of the classes)



Accuracy score of Logistic Regression is: 85.25%

Confusion Matrix

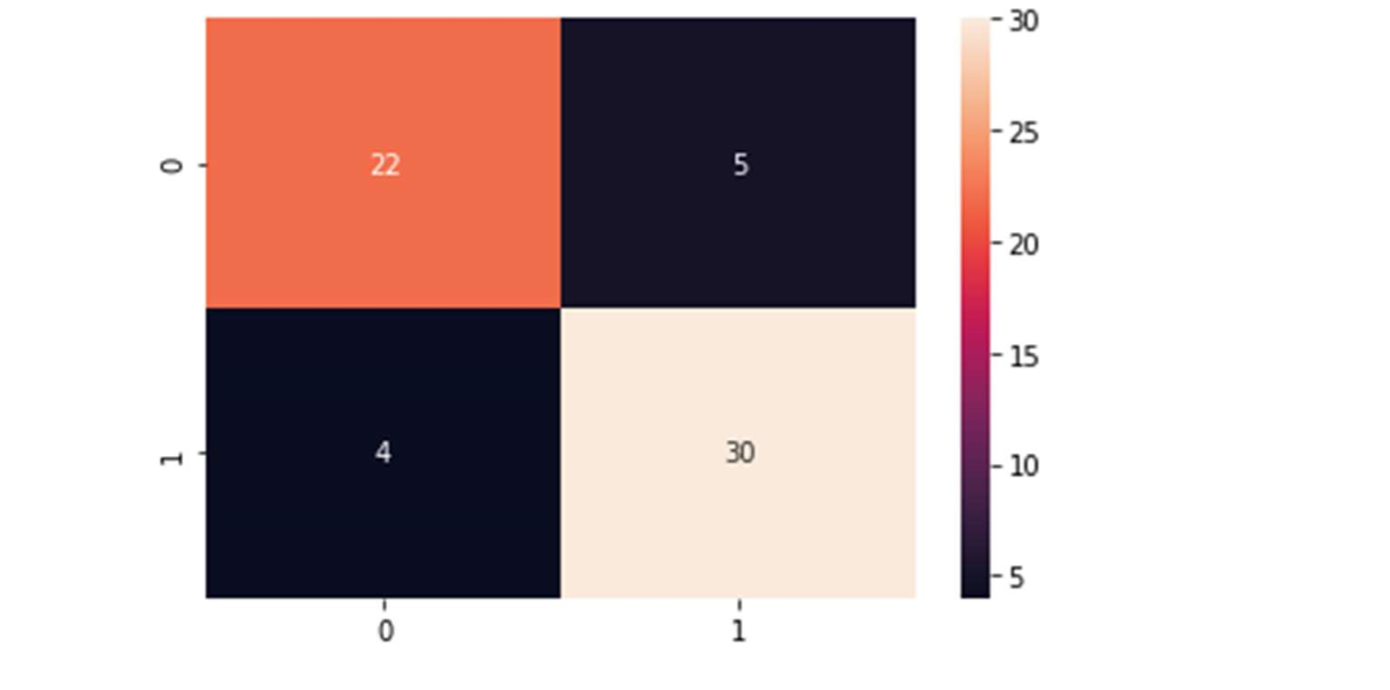
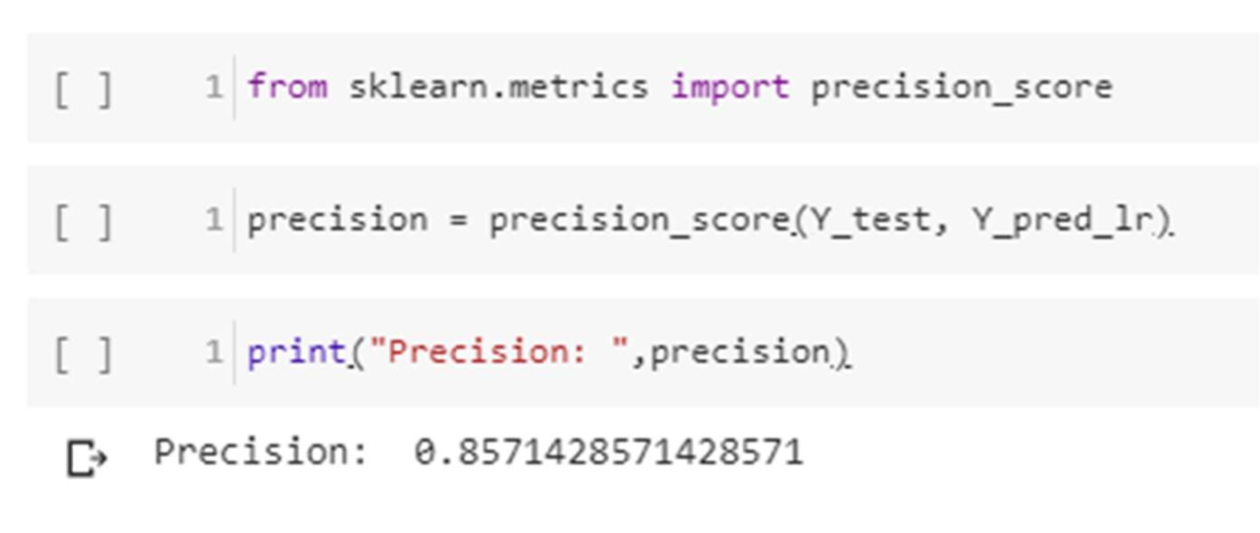


Figure 31 logistic regression confusion matrix

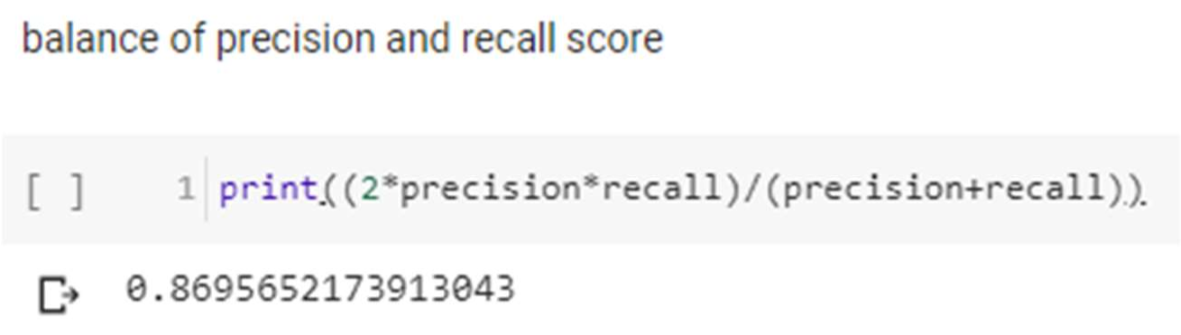
precision score



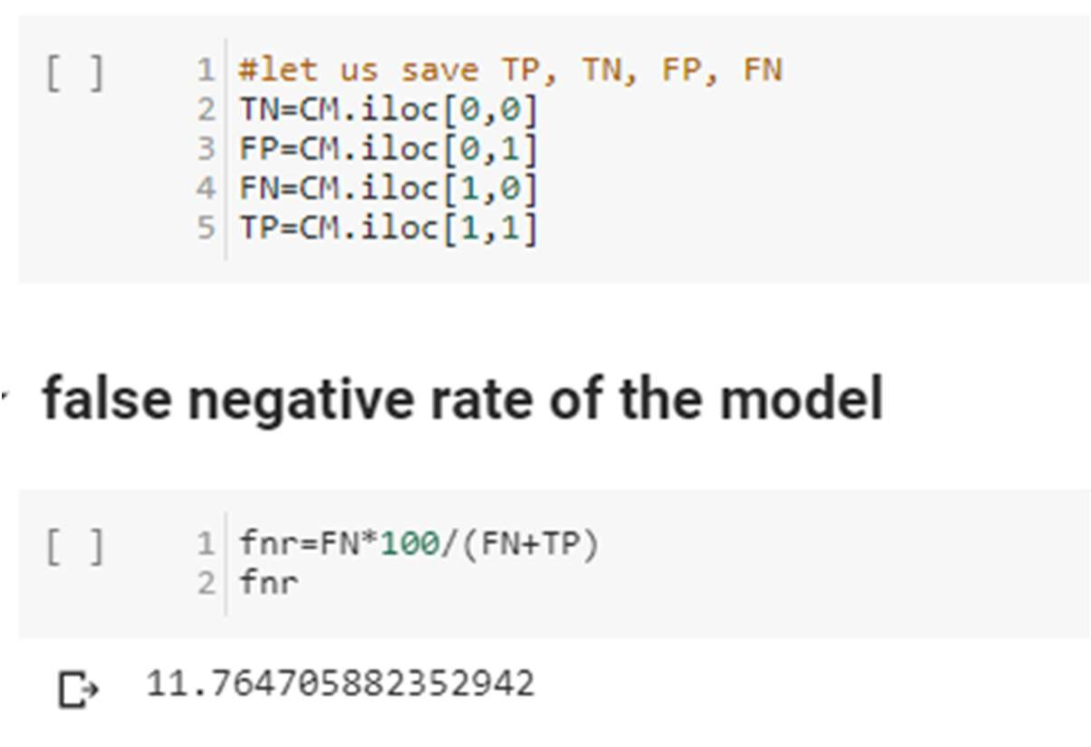
Recall



F-Score



false negative



***Random Forest***

Random Forest is a supervised learning algorithm. Random forest can be used for both classification and regression problems, by using random forest regressor we can use random forest on regression problems. But we have used random forest on classification in this project so we will only consider the classification part.

Random Forest pseudocode

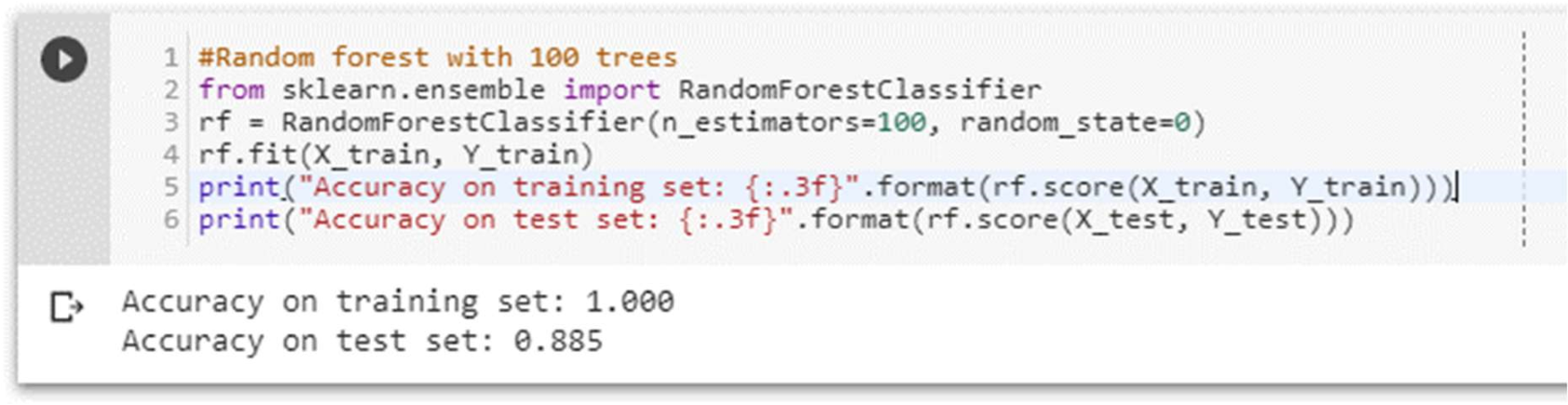
1. Randomly select “k” features from total “m” features.

Where k << m

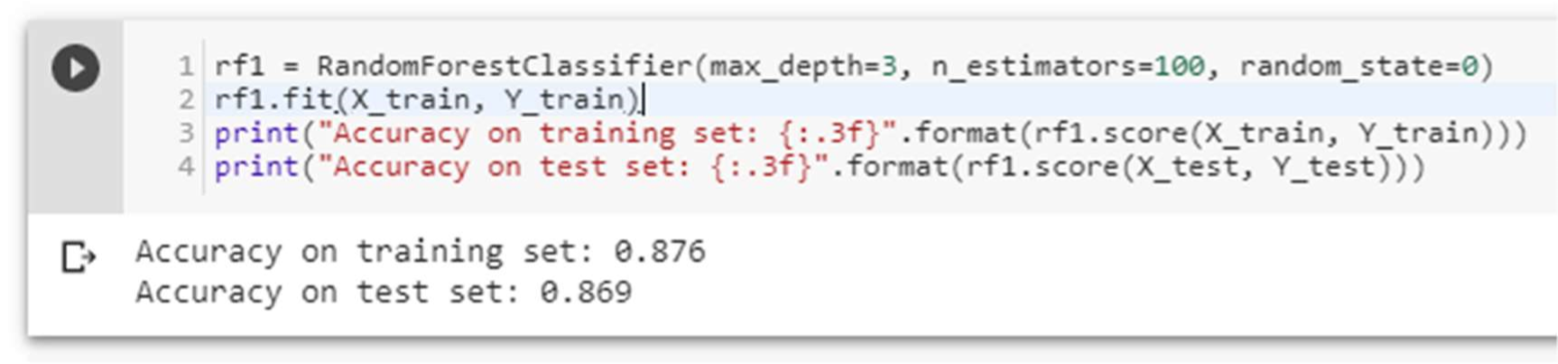
1. Among the “k” features, calculate the node “d” using the best split point.
2. Split the node into daughter nodes using the best split.
3. Repeat 1 to 3 steps until “l” number of nodes has been reached.
4. Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

Random forest prediction pseudocode

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target.
3. Consider the high voted predicted target as the final prediction from the random forest algorithm.



Pruning the depth of the trees



Accuracy score of Random Forest is 86.9%

Confusion Matrix

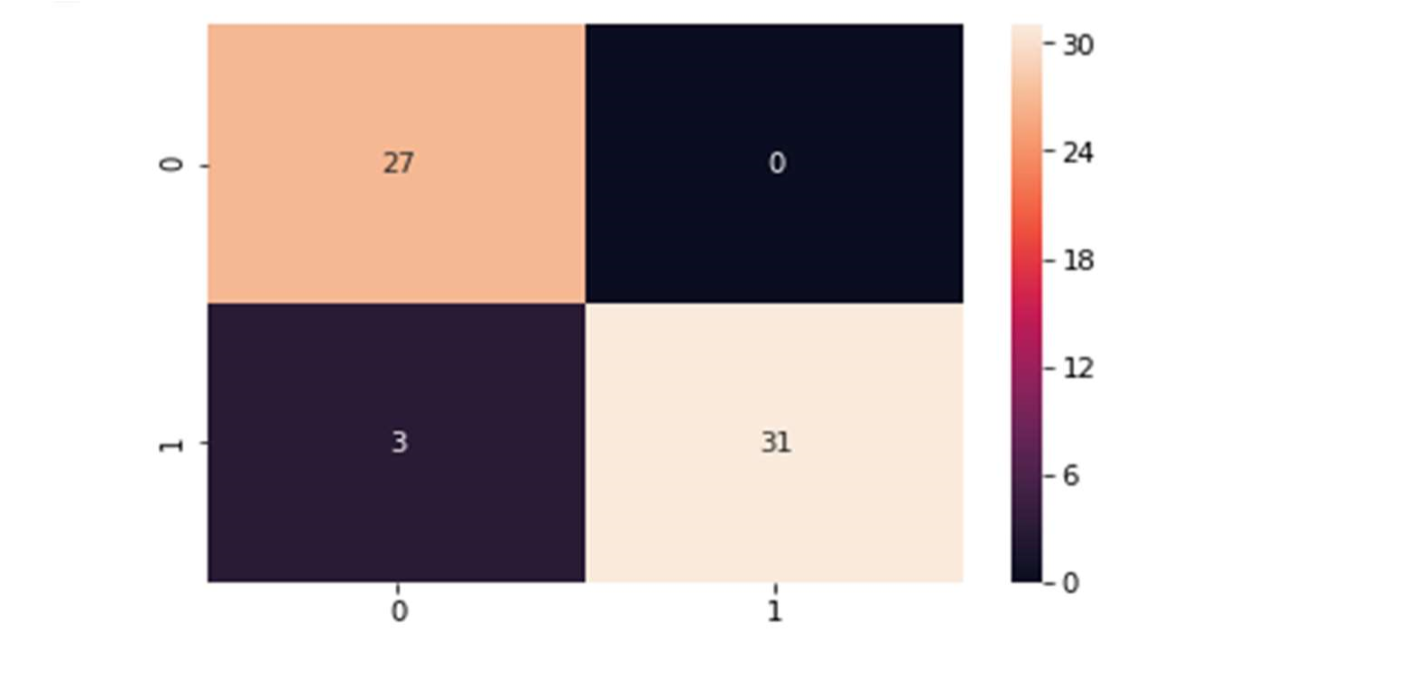
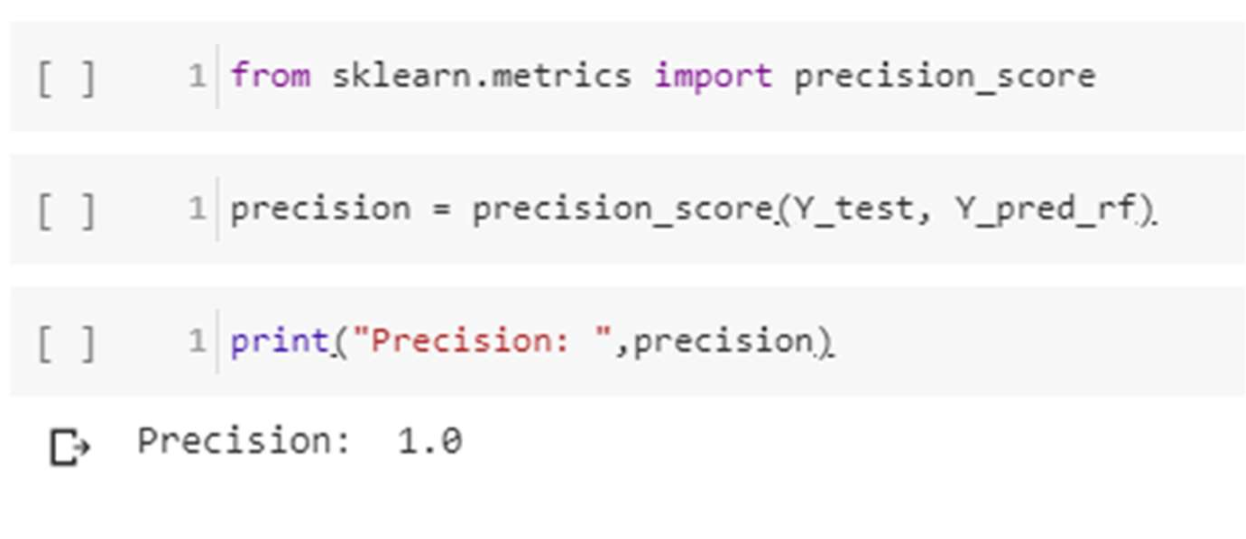
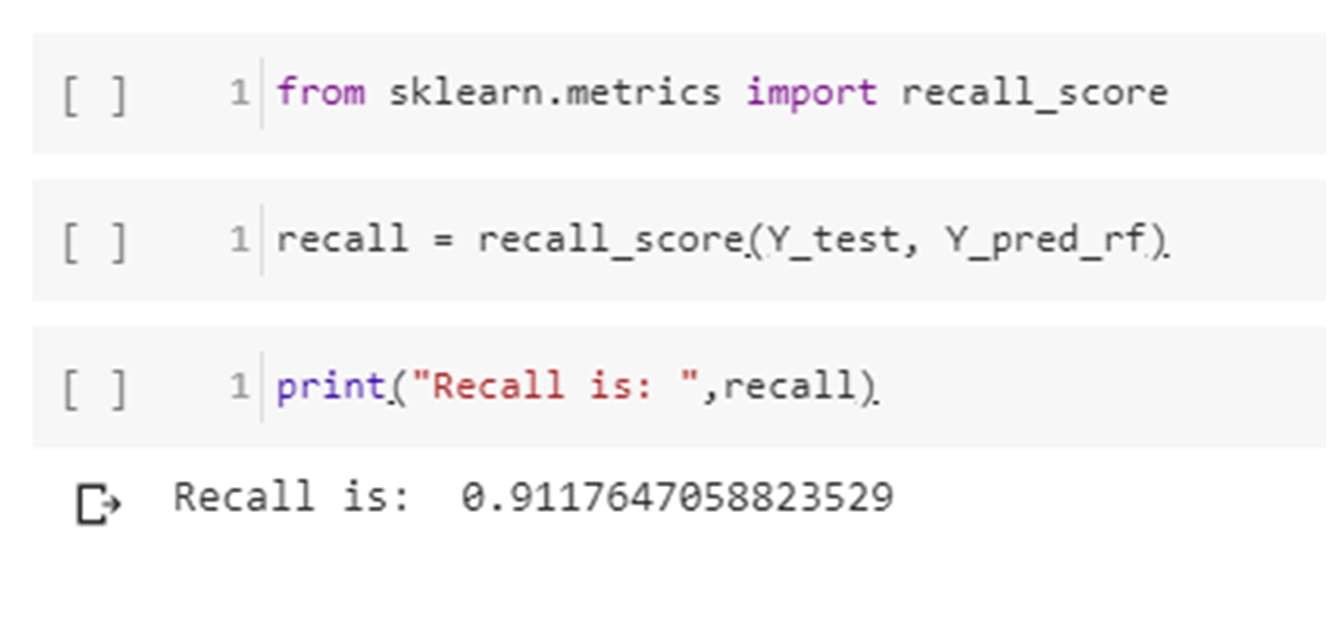


Figure 32 Random forest confusion matrix

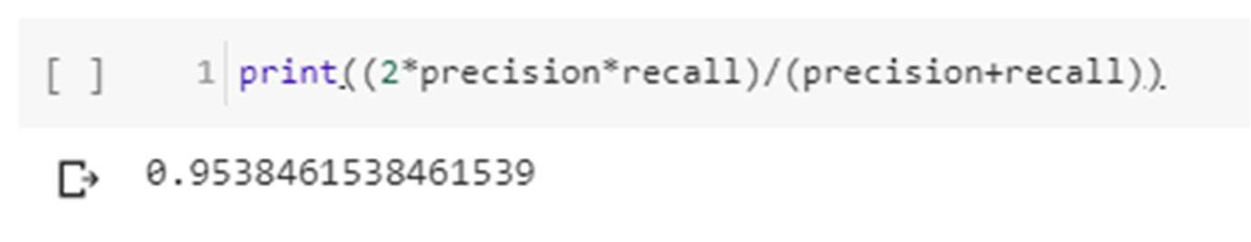
Precision score



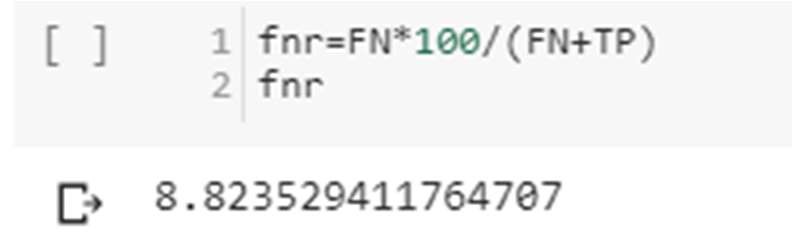
Recall



F-Score



False negative rate



***Naïve Bayes***

Bayes’ Theorem is stated as:

P(h|d) = (P(d|h) \* P(h)) / P(d)

* P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
* P(d|h) is the probability of data d given that the hypothesis h was true.
* P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
* P(d) is the probability of the data (regardless of the hypothesis).

we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h). After calculating the posterior probability for a number of different hypotheses, we will select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the (MAP) hypothesis.

This can be written as:

MAP(h) = max(P(h|d))

or

MAP(h) = max((P(d|h) \* P(h)) / P(d))

or

MAP(h) = max(P(d|h) \* P(h))

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize. Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation, and we could drop it so that we end up with:

MAP(h) = max(P(d|h))

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P (d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) \* P(d2|H) and so on. This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

MAP(h) = max(P(d|h) \* P(h))

Gaussian Naïve Bayes

mean(x) = 1/n \* sum(x)

Where n is the number of instances and x are the values for an input variable in your training data. We can calculate the standard deviation using the following equation:

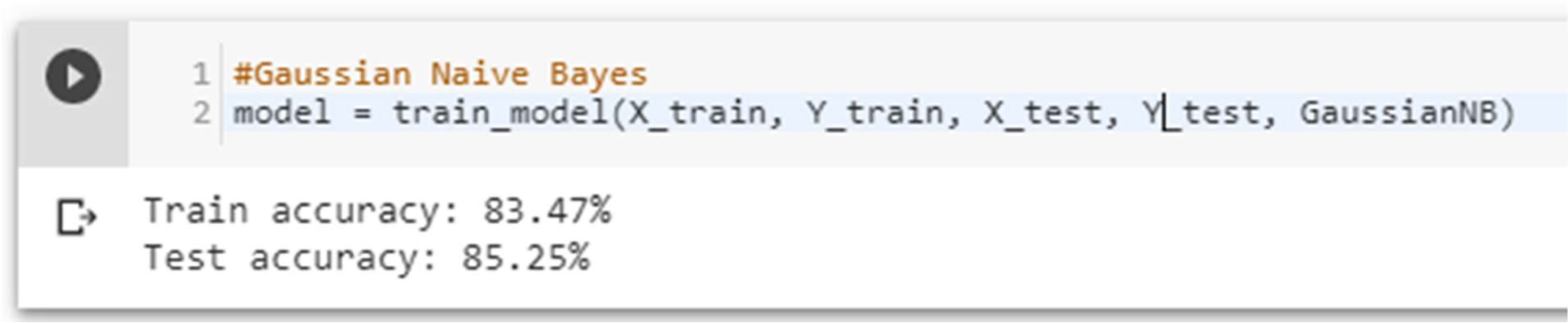
standard deviation(x) = sqrt (1/n \* sum(xi-mean(x)^2))

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the i’th instance and mean(x) is described above, and ^2 is the square. Gaussian PDF with a new input for the variable, and in

return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

pdf (x, mean, sd) = (1 / (sqrt (2 \* PI) \* sd)) \* exp (-((x-mean^2)/(2\*sd^2)))

Where pdf(x) is the Gaussian Probability Density Function (PDF), sqrt () is the square root, mean and sd are the mean and standard deviation calculated above, Pi is the numerical constant, exp () is the numerical constant e or Euler’s number raised to power and x is the input value for the input variable.



Confusion matrix

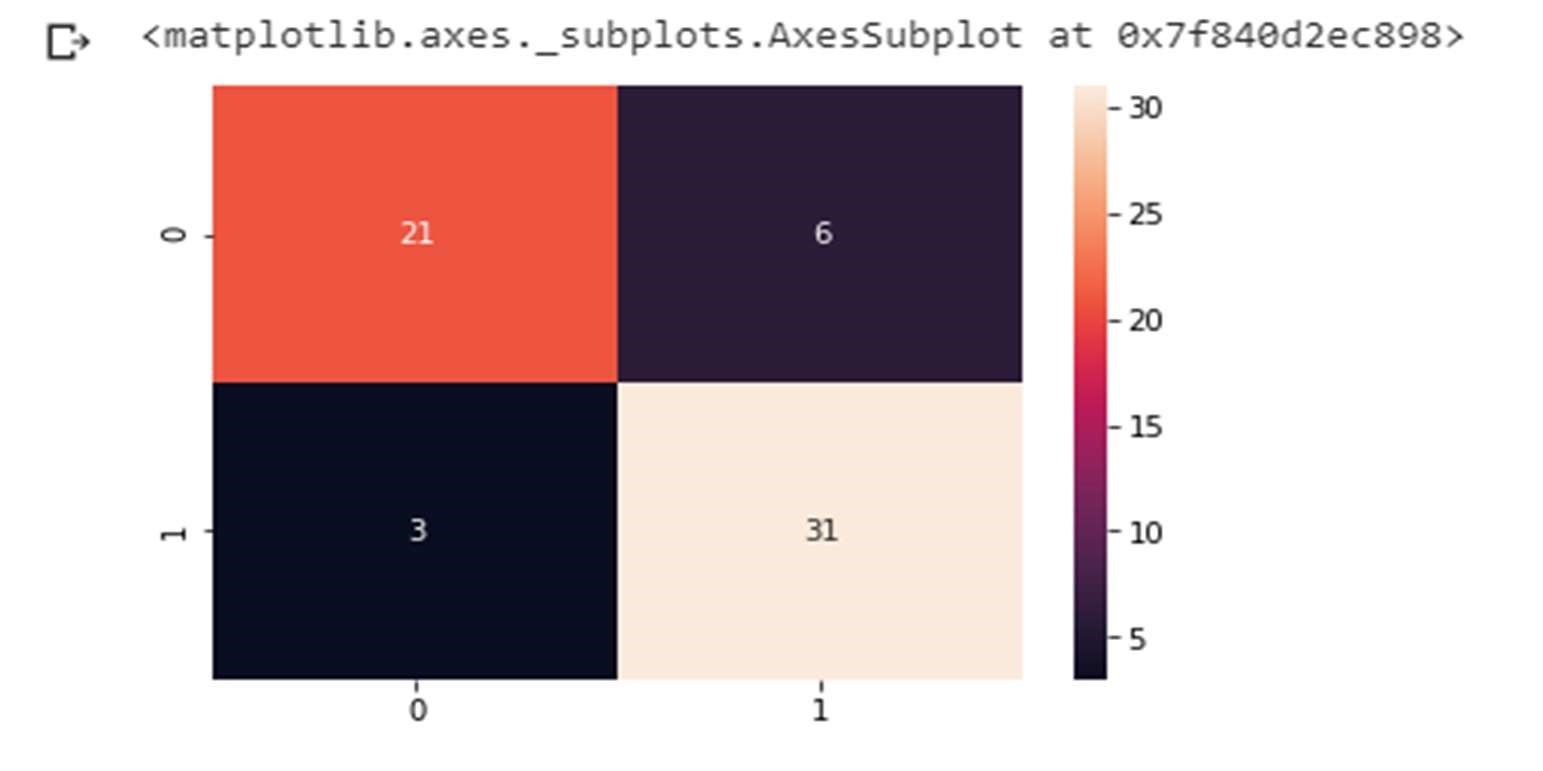
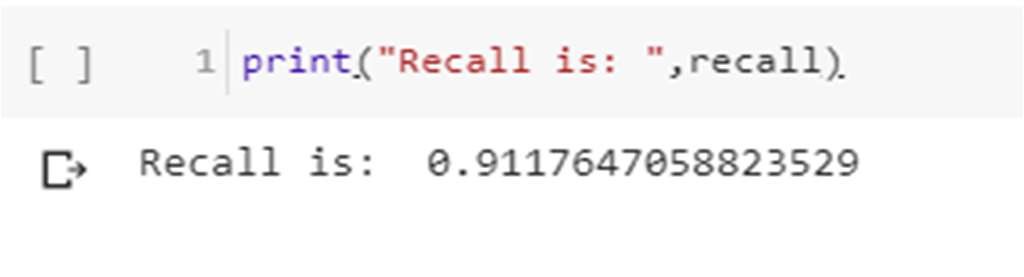


Figure 33 Naive Bayes confusion matrix

Precision score



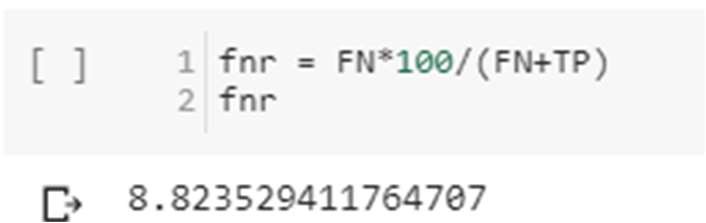
Recall



F-Score



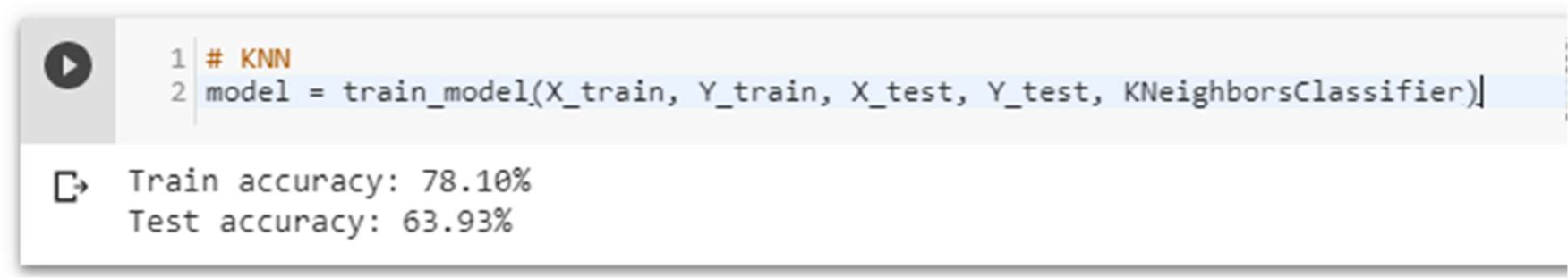
False negative



***K-Nearest Neighbour***

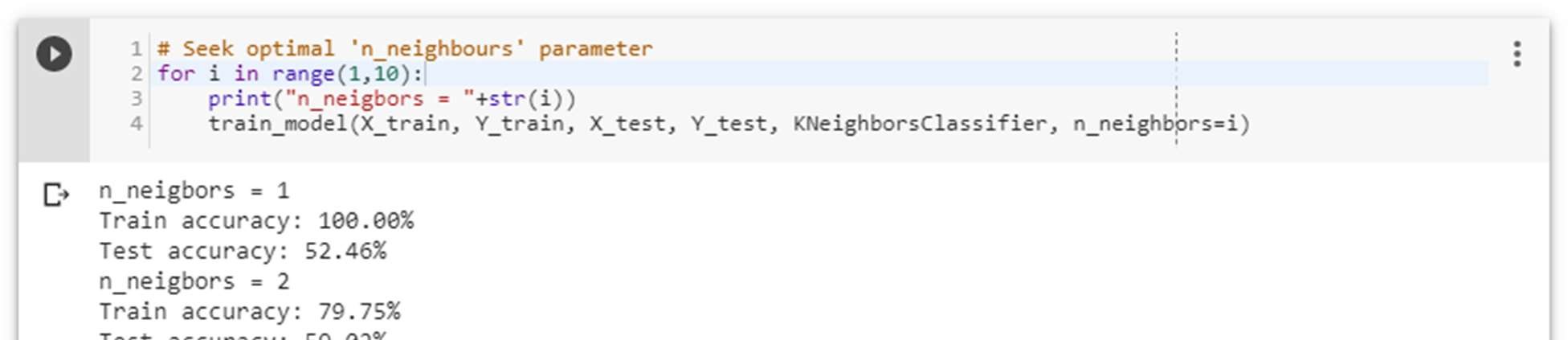
We can implement a KNN model by following the below steps:

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
   * Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
   * Sort the calculated distances in ascending order based on distance values
   * Get top k rows from the sorted array
   * Get the most frequent class of these rows
   * Return the predicted class



Using different inputs –

It turns out that the value of n = 8 is optimal



Confusion matrix

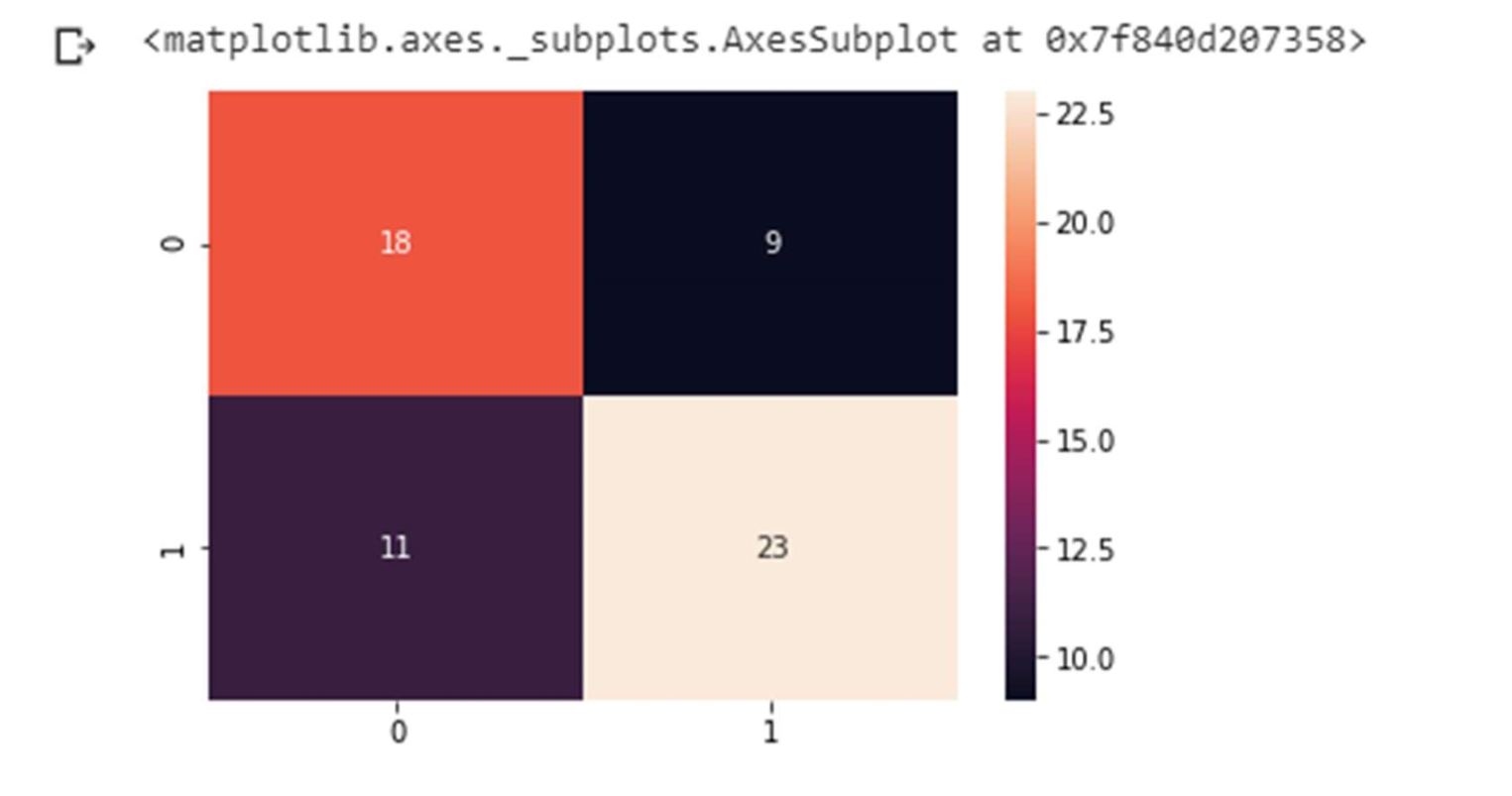
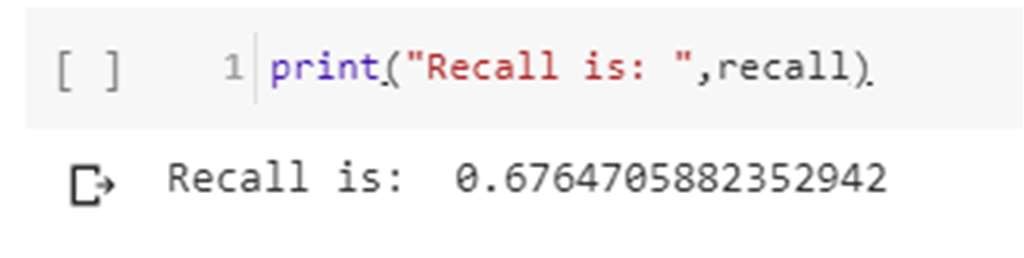


Figure 34 KNN confusion matrix

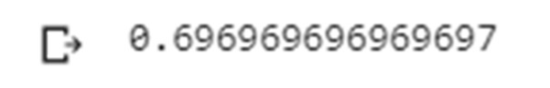
Precision score



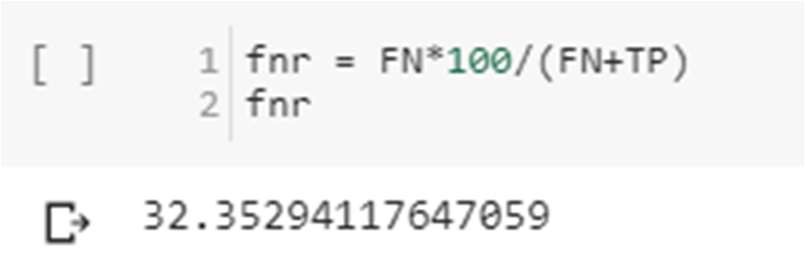
Recall



F-Score



False negative



***Decision Tree***

pseudocode

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Assumptions while creating Decision Tree

* At the beginning, the whole training set is considered as the root.
* Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
* Records are distributed recursively on the basis of attribute values.
* Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

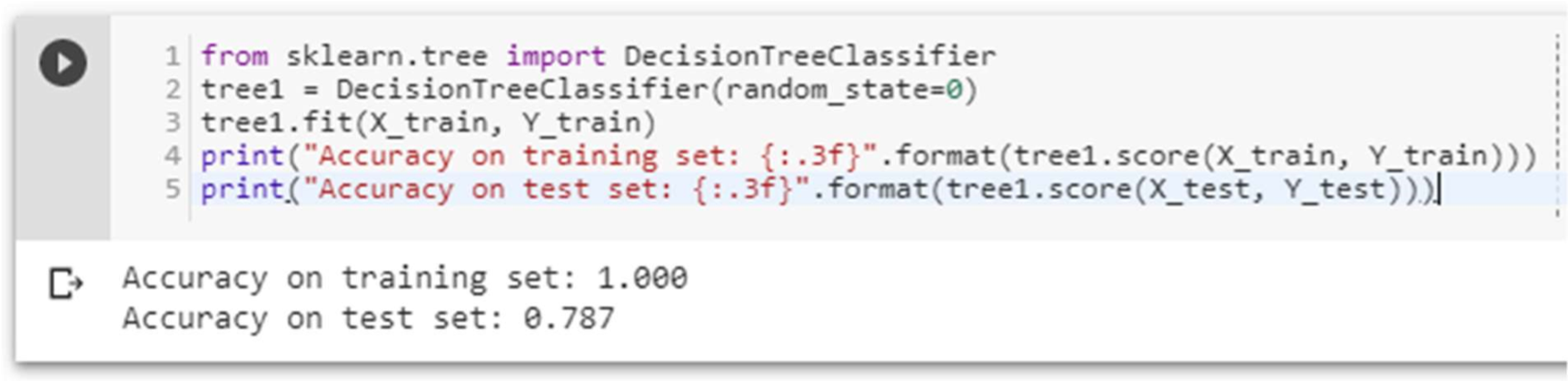
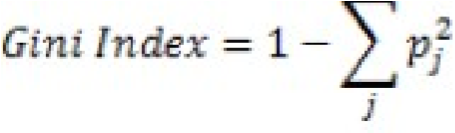
The popular attribute selection measures

* Information gain
* Gini index

Attribute selection method - A dataset consists of “n” attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can’t solve the issue. If we follow a random approach, it may give us bad results with low accuracy. For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criterion like information gain, Gini index, etc. These criterions will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e., the attribute with a high value (in case of information gain) is placed at the root. While using information Gain as a criterion, we assume attributes to be categorical, and for

Gini index, attributes are assumed to be continuous.[132]

Gini Index - Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred.



The accuracy on the training set is 100%, while the test set accuracy is much worse. This is an indicative that the tree is overfitting and not generalizing well to new data. Therefore, we need to apply pre-pruning to the tree. We set max depth=3, limiting the depth of the tree decreases overfitting. This leads to a lower accuracy on the training set, but an improvement on the test set.

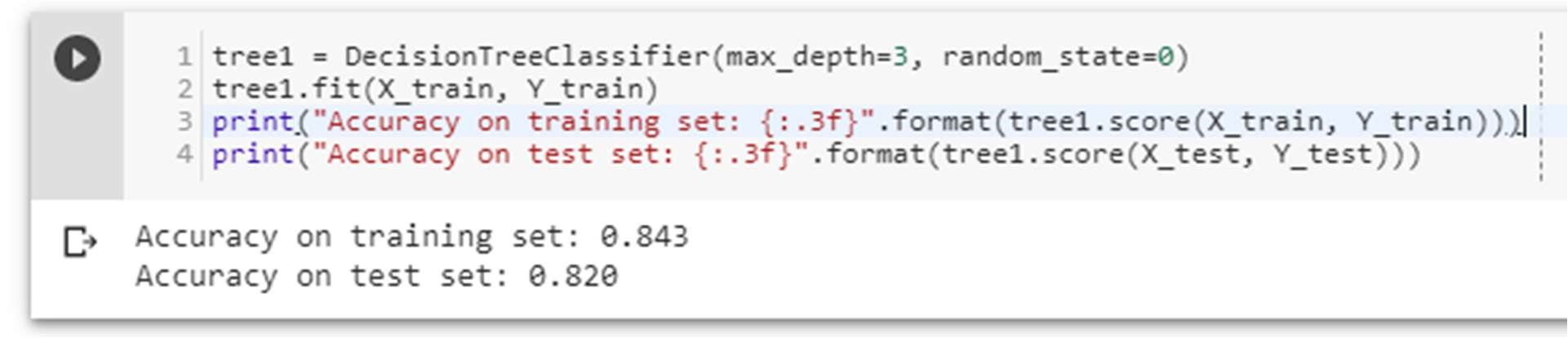
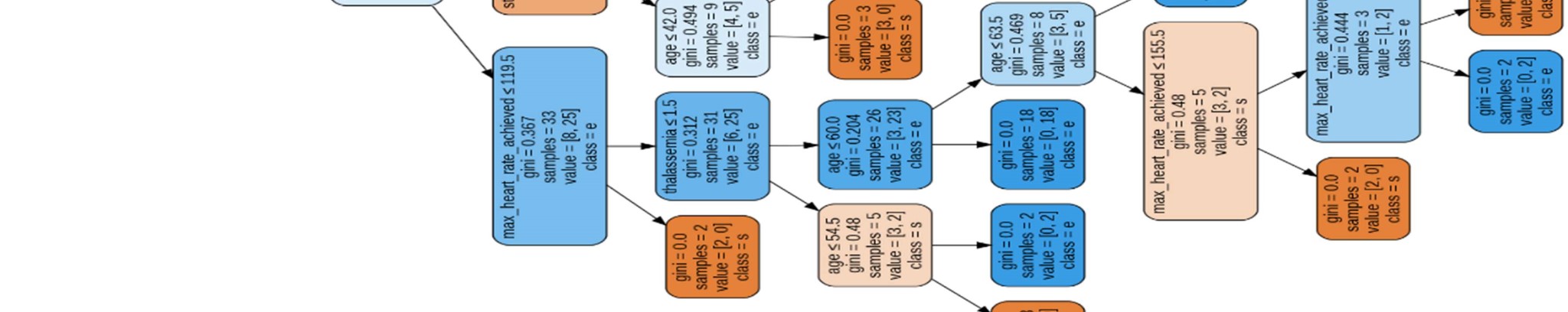
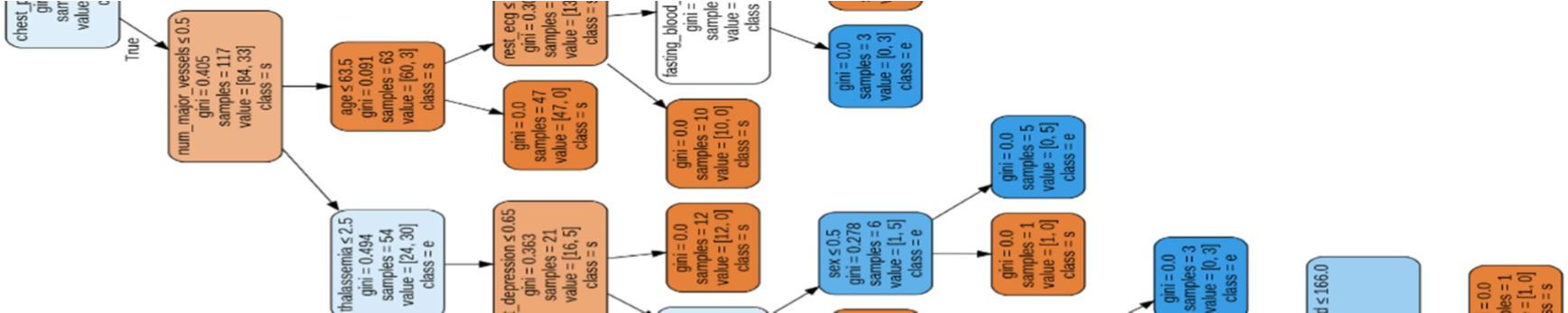
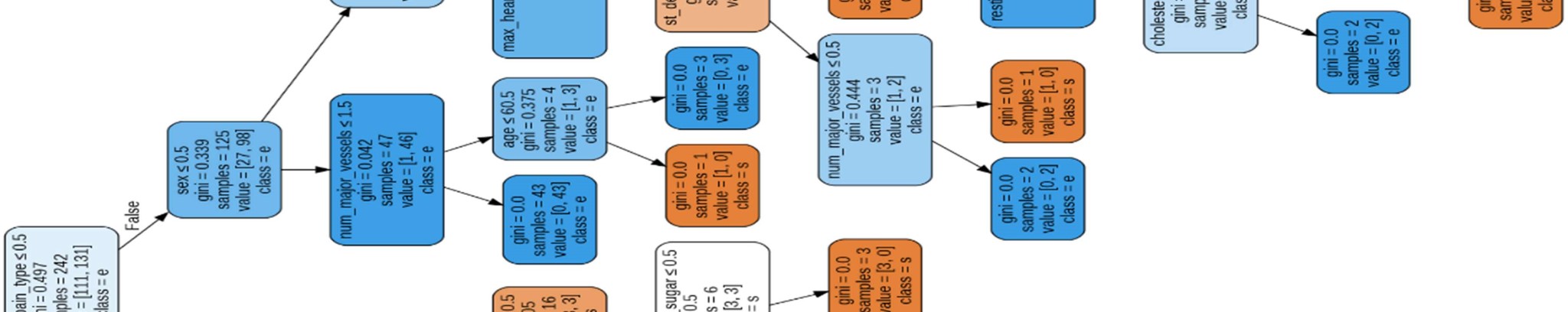
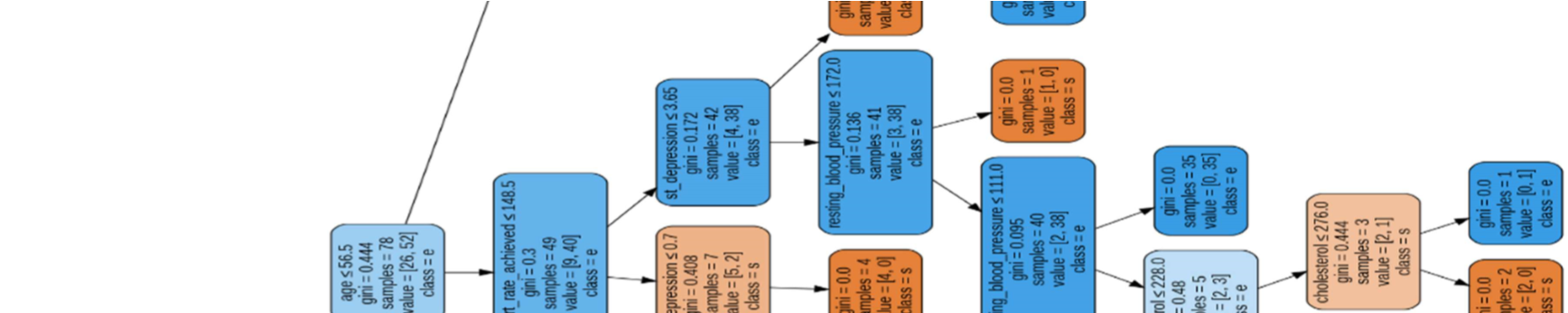
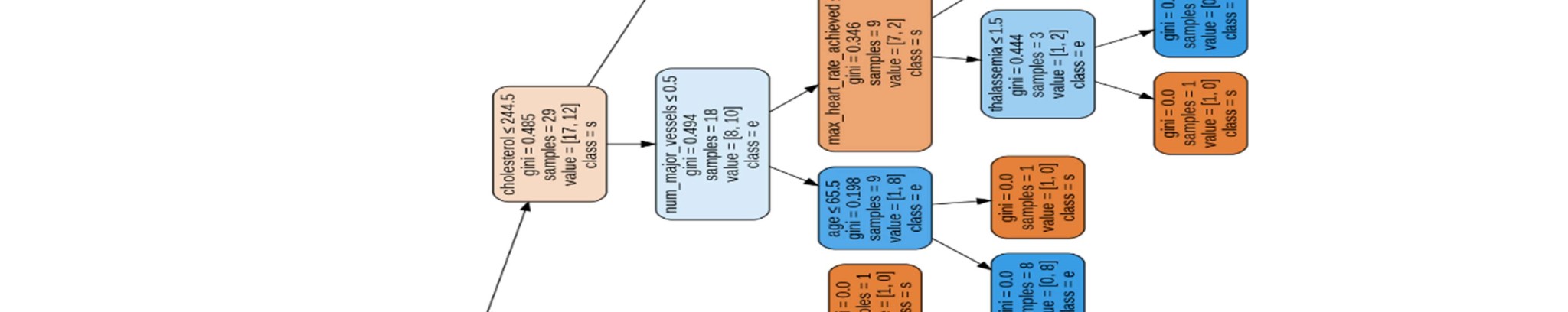
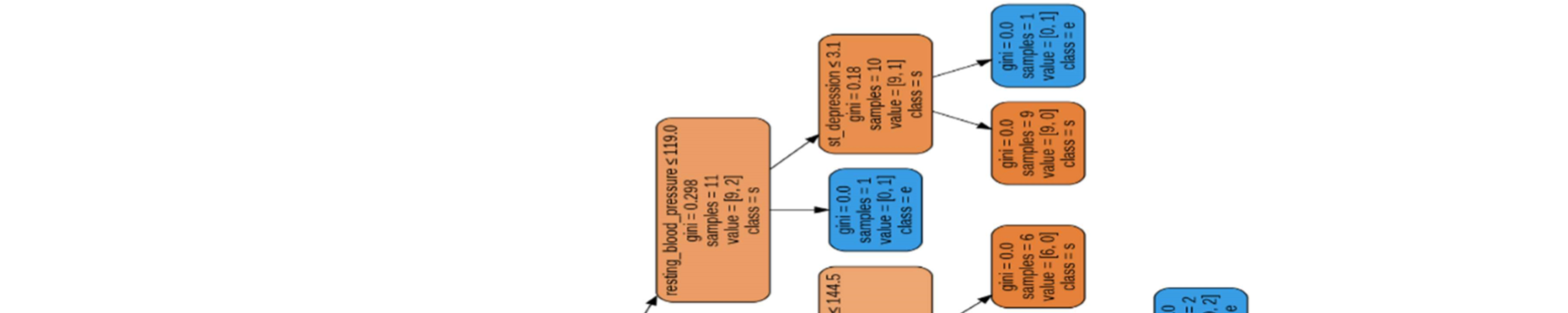


Figure 35

Decision tree visualization



Confusion matrix

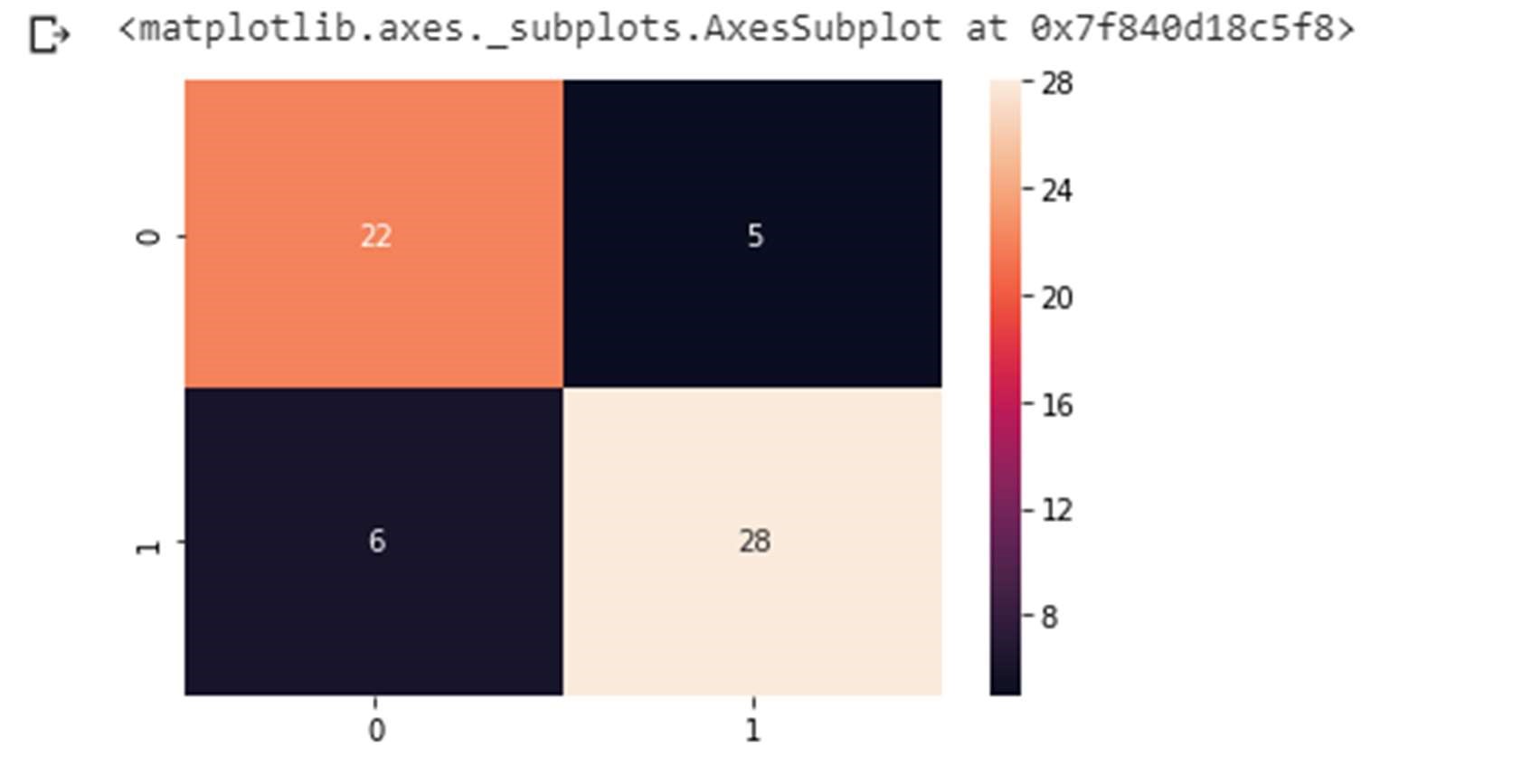
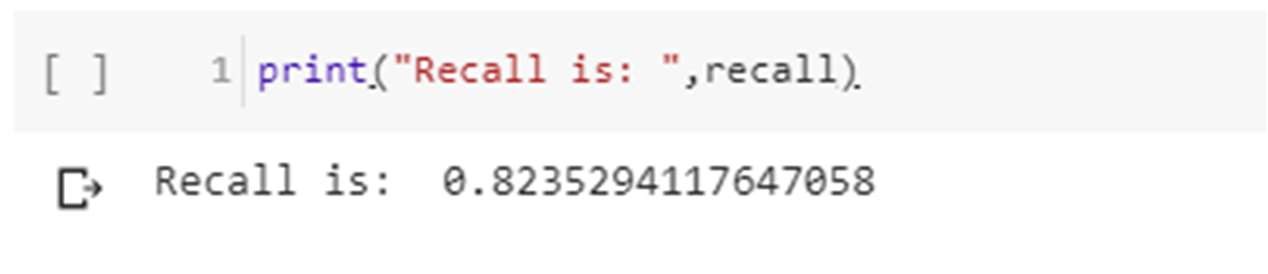


Figure 36 decision tree confusion matrix

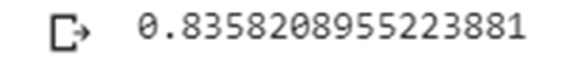
Precision score



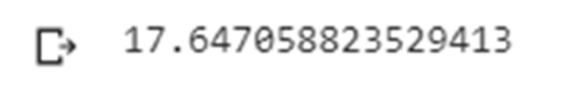
Recall



F-Score



False negative rate



***Result***

