Mathematical Analysis and Comparison of Machine Learning Algorithms

Submitted by:

Mrinalini Singh (16JE001958)

Apoorva S. Chouhan (16JE002048)

FINAL YEAR B. TECH. PROJECT PRESENTATION

MOTIVATION

Machine learning is a popular topic of research and application. It focuses on developing algorithms to “teach” a computer how to learn patterns and correlations in data. This field aims to automate problem tasks such as classification, clustering, and others. More advanced algorithms are used for facial recognition, sentiment analysis, and reinforcement learning and so on.

Often, people working with these algorithms treat them as black-boxes, but in order to completely understand an algorithm, it is important to have a mathematical understanding of them, it is important to understand the nuances of various algorithms having similar principles. The motivation of this project was to understand and to explain the mathematical principles of two of the most widely used algorithms – decision trees and naïve bayes classifiers. We tried to explain the working using simple examples, to compare them, and discuss the best field of application for each.

DATASETS USED

- Play tennis dataset
  - “Play Tennis” dataset is used to explain the mathematical approach of algorithms. This is a popular dataset which depicts if a person will play tennis or not based on weather conditions.
- Iris dataset
  - The iris dataset consists of data of 3 species of the flower.
- Bank note authentication dataset
  - The bank note authentication dataset consists of values of various parameters obtained after performing wavelet transform on images of authentic and counterfeit bank notes.

Decision Tree Classifier

Explanation using tennis dataset and testing classification efficiency study based on iris and bank note dataset classification using decision tree classifier.

Introduction

- Decision tree learning is one of the most common and powerful algorithms for classification. It is easy to understand the intuition behind the algorithm even without good knowledge of mathematics.
- It is robust to noisy data and falls under the category of supervised learning algorithms.
- With advancements in the first algorithm, decision trees can work with continuous data, and also be used for regression. For the scope of this project, we have focused specifically on CART type decision trees.
- A tree can be *"learned"* by splitting the data set into subsets based on attribute values. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is terminated when all the subsets at a node have the same value as the target variable, or when splitting no longer improves the quality of predictions.
- The final hypothesis can simply be understood as a set of if-then-else rules. This can simply be represented in the form of a flowchart.

Mathematics Involved

Decision trees are constructed using a top-down approach, by choosing a variable at each step that best splits the set of instances. Different algorithms use different metrics for measuring *"best split"*. These generally measure the homogeneity of the target variable within the subsets (also called nodes). These metrics are applied to each candidate subset, and the resulting values are combined to provide a measure of the quality of the split.

The metrics used to decide the best split are*: Information Gain* and *Gini Index*. For the purpose of this presentation, the algorithm has been explain using *Gini Index* as the measurement of purity (or impurity).

Mathematics Involved (Continued…)

Gini Index

Gini index or *gini impurity* is a measure of inequality in instances. Having values between 0 and 1, it measures how often a randomly chosen instance from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. Gini index of value 0 means instances are perfectly homogeneous whereas, Gini index of value 1 means maximal inequality among elements. If a subset is homogeneous this means that all the instances are from the same class.

Mathematics Involved (Continued…)

Gini index is mathematically defined as the sum of the square of the probabilities of each class,

$$Gini(A = v) = 1 - \sum_{i=1}^{c} p_{i,v}^2$$

$$Gini\ Index(A) = \sum_{j=1}^{a} p_j \cdot Gini(A = v_j)$$

where c represents different classes, a represents number possible values of v, v are the different possible values of A, is the proportion of instances belonging to each class for the same value v of attribute A, and is the proportion of instances of subsets having the same v. This metric is used in CART, to evaluate the split of subset or node. The attribute with the lowest value of *gini index* is taken as the root node for that particular tree, or if one split has already taken place, it will be the root node of the subsequent sub-tree. This will be clearer in the next section, as the evaluation is explained with an example.

Illustration on Tennis Dataset

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Rainy	Hot	High	Weak	No
2	Rainy	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Sunny	Mild	High	Weak	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Sunny	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Rainy	Mild	High	Weak	No
9	Rainy	Cool	Normal	Weak	Yes
10	Sunny	Mild	Normal	Weak	Yes
11	Rainy	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Sunny	Mild	Hot	Strong	No

The tennis dataset.

Illustration on Tennis Dataset (Continued…)

- The algorithm begins by calculating the values of *gini index* of various attributes of the "Play Tennis" dataset from the table on the previous slide. Outlook, temperature, humidity and wind are the various attributes or features of the data.
- **Outlook**
  - It can take three values - sunny, overcast and rainy. Summarising the decision for outlook features,
    - *Gini (Outlook = Overcast)* = $1 - (4/4)^2 - (0/4)^2 = 0$
    - *Gini (Outlook = Sunny)* = $1 - (2/5)^2 - (3/5)^2 = 0.48$
    - *Gini (Outlook = Rainy)* = $1 - (3/5)^2 - (2/5)^2 = 0.48$
  - *Gini Index (Outlook)* = $(4/14)0 + (5/14)0.48 + (5/140.48) = 0.342$

Outlook\Class	Yes	No	Total
Overcast	4	0	4
Sunny	2	3	5
Rainy	3	2	5

Illustration on Tennis Dataset (Continued…)

- Similarly, *Gini index* values for other attributes are calculated as
  - *Gini Index (Temperature)* = 0.439
  - *Gini Index (Humidity)* = 0.367
  - *Gini Index (Wind)* = 0.428

Temperature\Class	Yes	No	Total
Hot	2	2	4
Mild	4	2	6
Cool	3	1	4

Wind\Class	Yes	No	Total
Strong	3	3	6
Weak	6	2	8
Wind\Class	Yes	No	Total

Humidity\Class	Yes	No	Total
High	3	4	7
Normal	6	1	7

Illustration on Tennis Dataset (Continued…)

Now, since the value of *gini index* for the attribute 'Outlook' is minimum, it forms the root node of the tree, and the dataset is divided into three subsets - one with outlook = sunny, one with outlook = overcast and the last one with outlook = rainy.

Attribute	Gini Index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

Illustration on Tennis Dataset (Continued..)

This forms the first node for splitting in the decision tree.

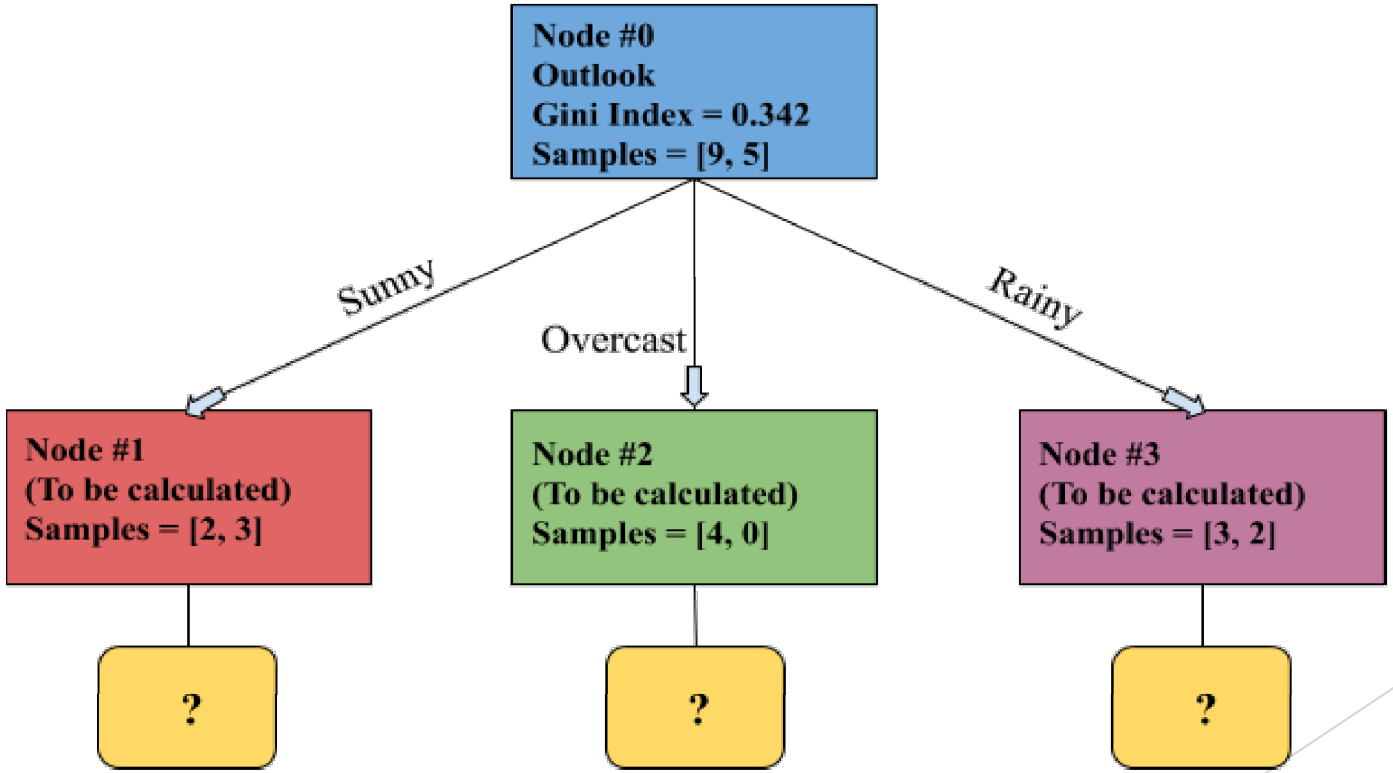


Illustration on Tennis Dataset (Continued..)

Next, *gini index* values are calculated again for the subsets now formed. The data for each leaf node is each table here.

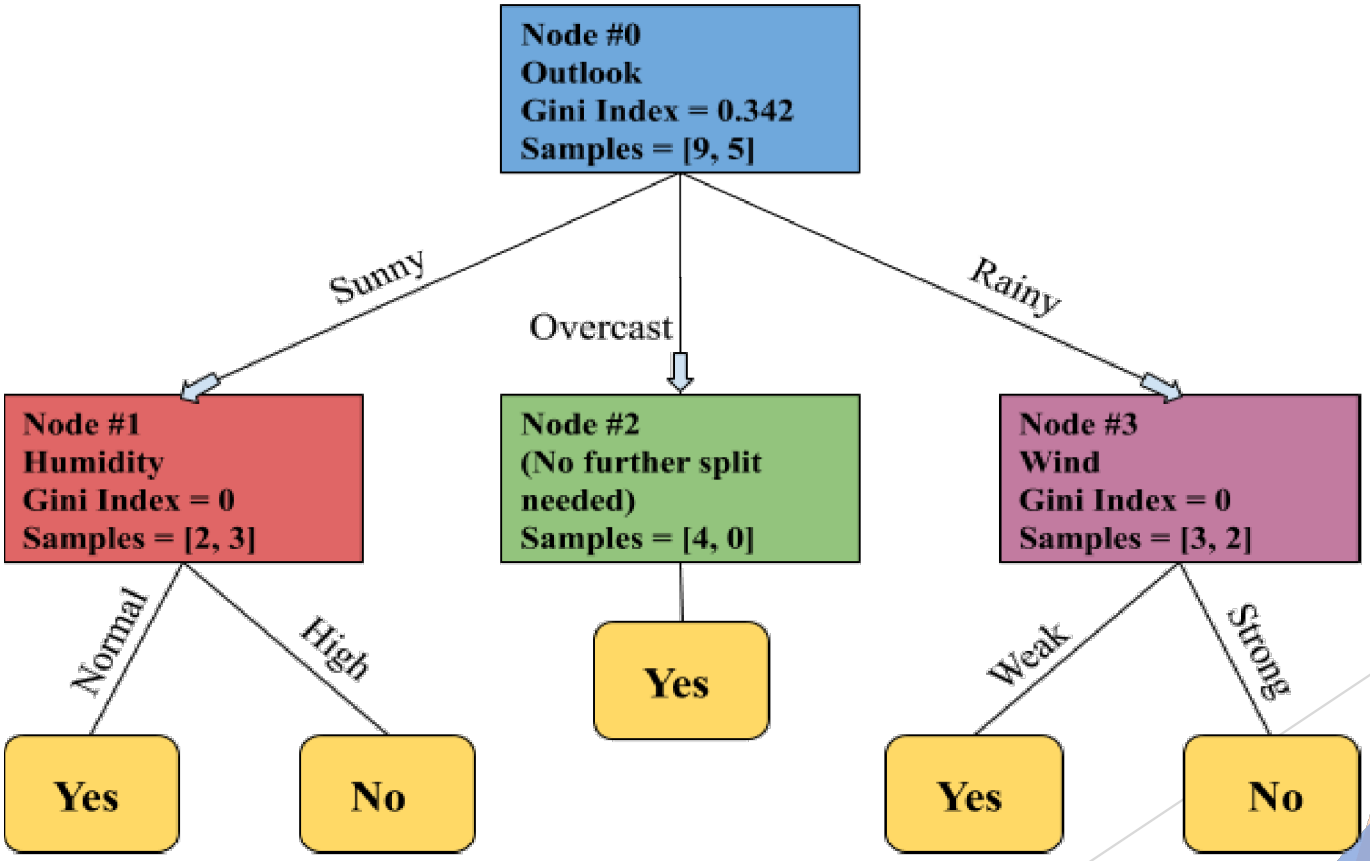
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
4	Sunny	Mild	High	Weak	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Sunny	Cool	Normal	Strong	No
10	Sunny	Mild	Normal	Weak	Yes
14	Sunny	Mild	Hot	Strong	No

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Rainy	Hot	High	Weak	No
2	Rainy	Hot	High	Strong	No
8	Rainy	Mild	High	Weak	No
9	Rainy	Cool	Normal	Weak	Yes
11	Rainy	Mild	Normal	Strong	Yes

Illustration on Tennis Dataset (Continued..)

The final decision tree is formed as:



Pseudo Code

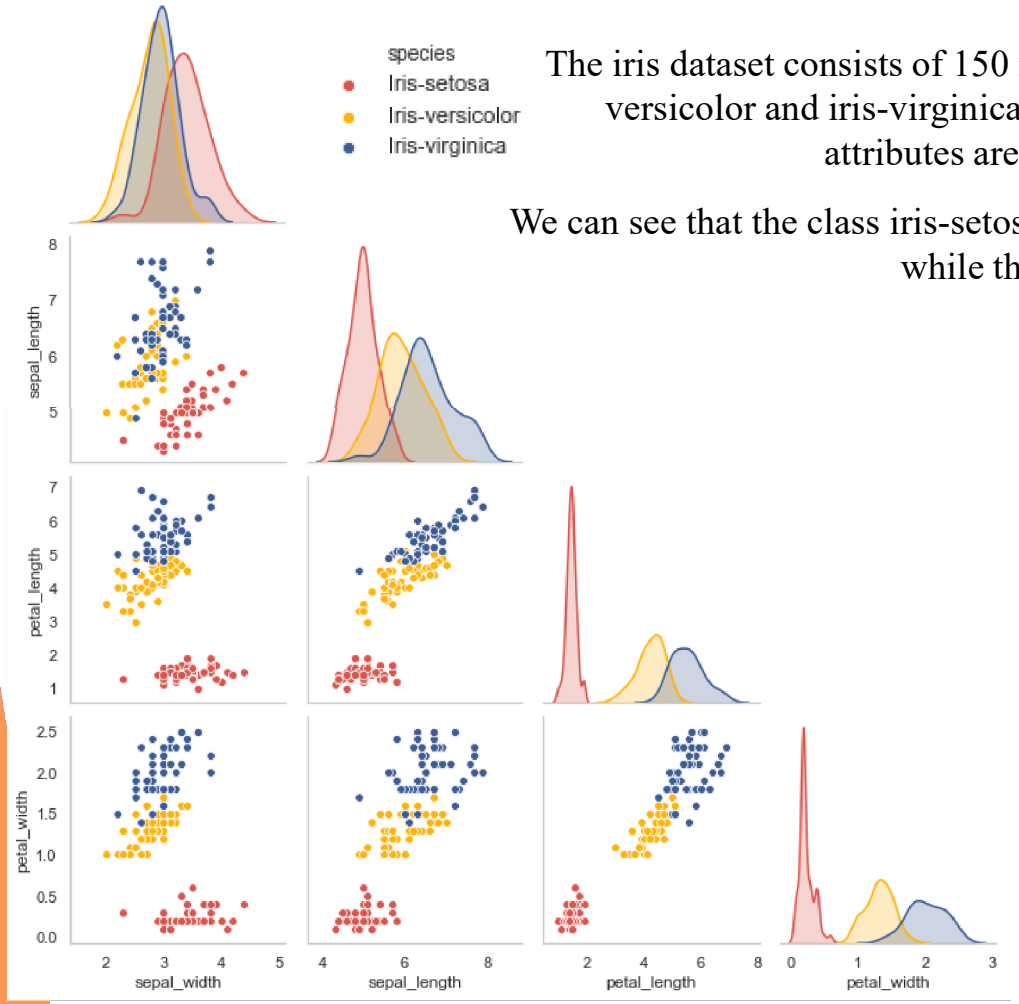
GenerateDecisionTree (Set of instances S, List of attributes A):

1. Create a node *N*;
2. If all the samples of the set belong to the same class *C*, then, label *N* with *C; terminate;*
3. If *A* is empty, label *N* with the most common class *C* of *S*; terminate;
4. G = 1; *a;*
5. For attribute a_i in *A*:
6. g = *GiniIndex(S,* a_i)
7. If g < G:

   $a = a_i$

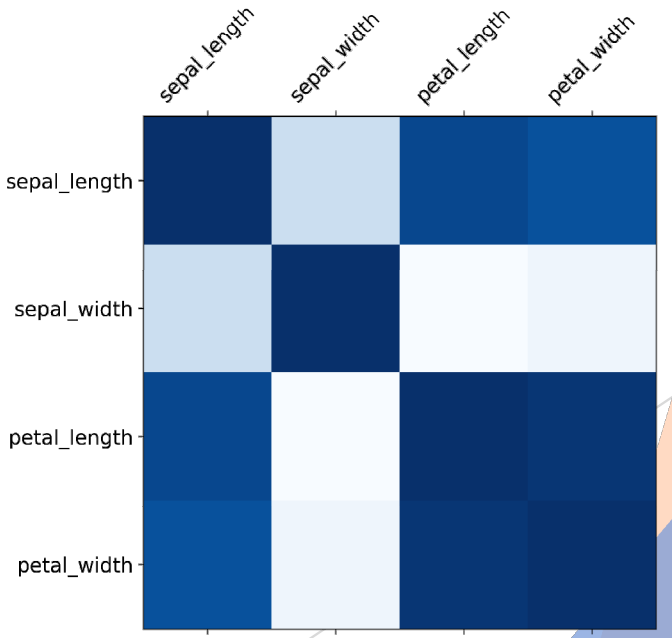
   G = g
8. For each value *v* of *a*:
9. Create a branch from *N*, with condition $a = v$
10. Let S_v be the subset of *S*, for which $a = v$
11. If S_v is empty, attach a leaf node with *C* as the most commonclass in *S*,
12. Else, attach node generated by *GenerateDecisionTree (*S_v*, A-a)*
13. Return node N;

Performance on Iris Dataset

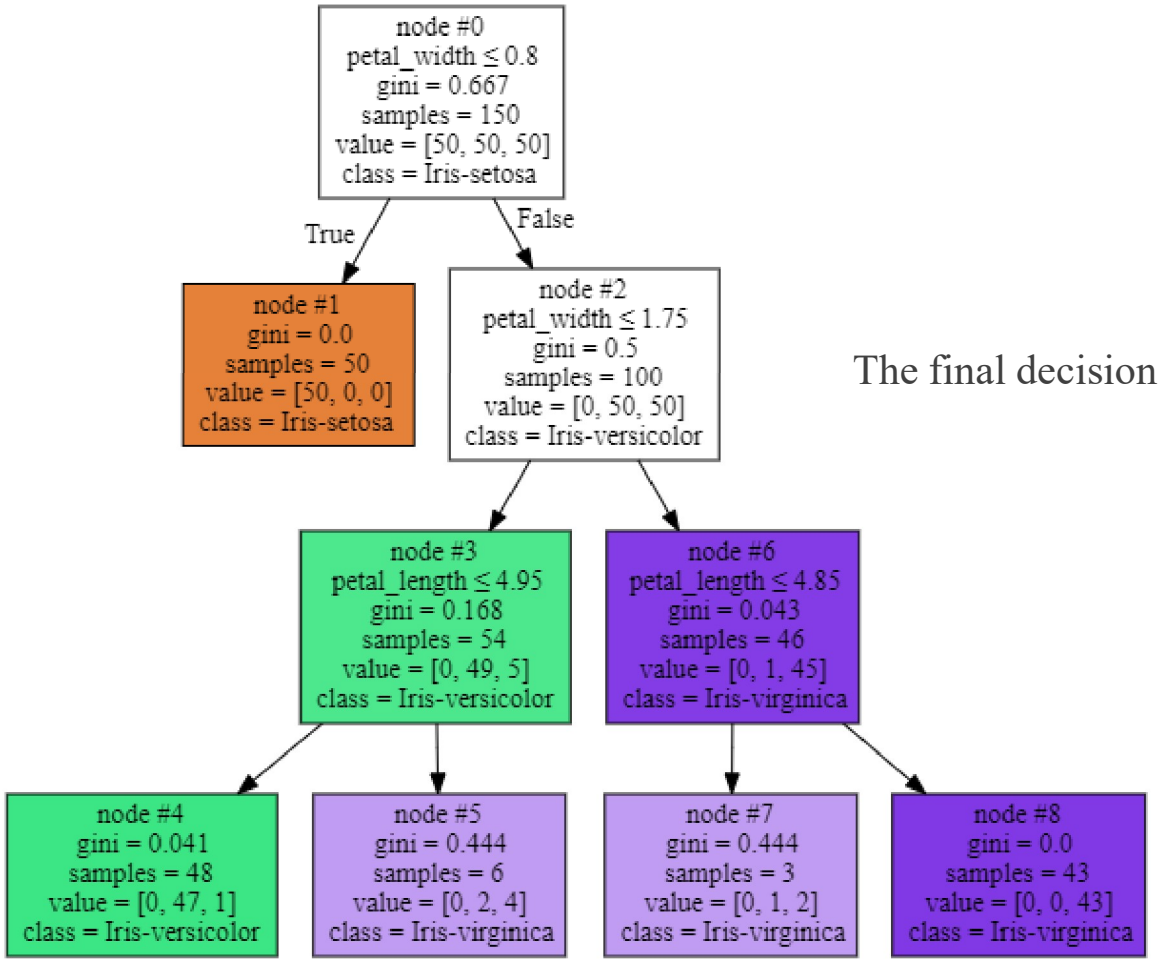


The iris dataset consists of 150 instances belonging to 3 classes – iris-setosa, iris-versicolor and iris-virginica. The pair plots and correlation matrix for various attributes are in figures on the left and the bottom respectively.

We can see that the class iris-setosa is linearly separable from the other two classes, while they are not linearly separable between themselves.



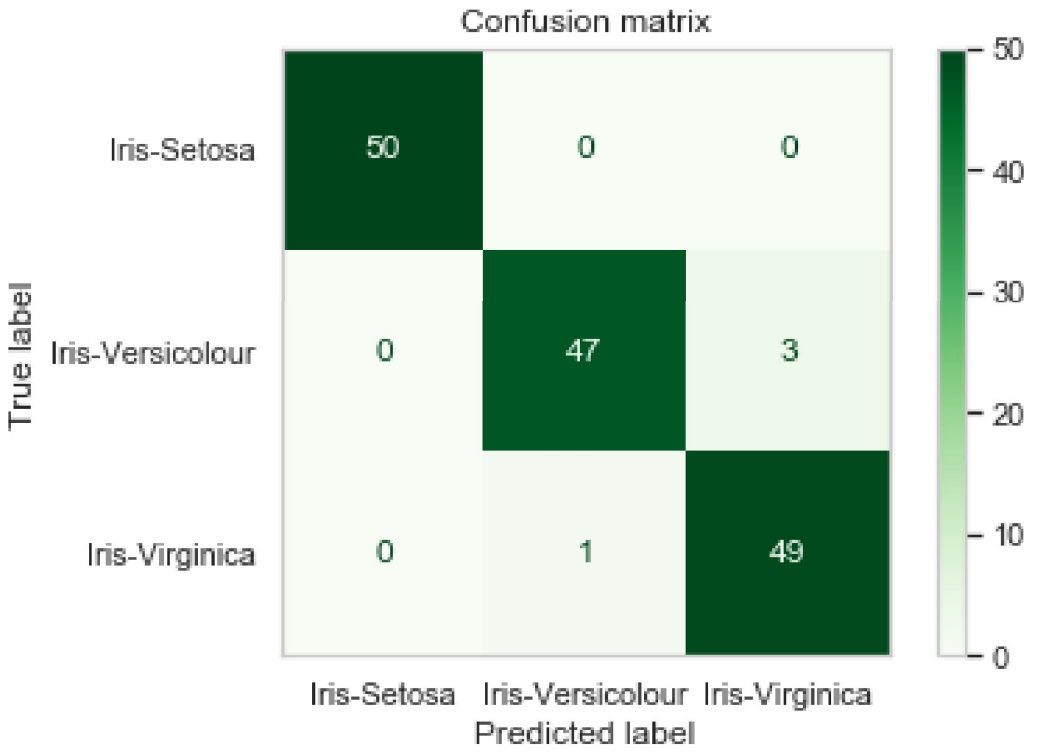
Performance on Iris Dataset (Continued…)



The final decision tree flowchart for iris.

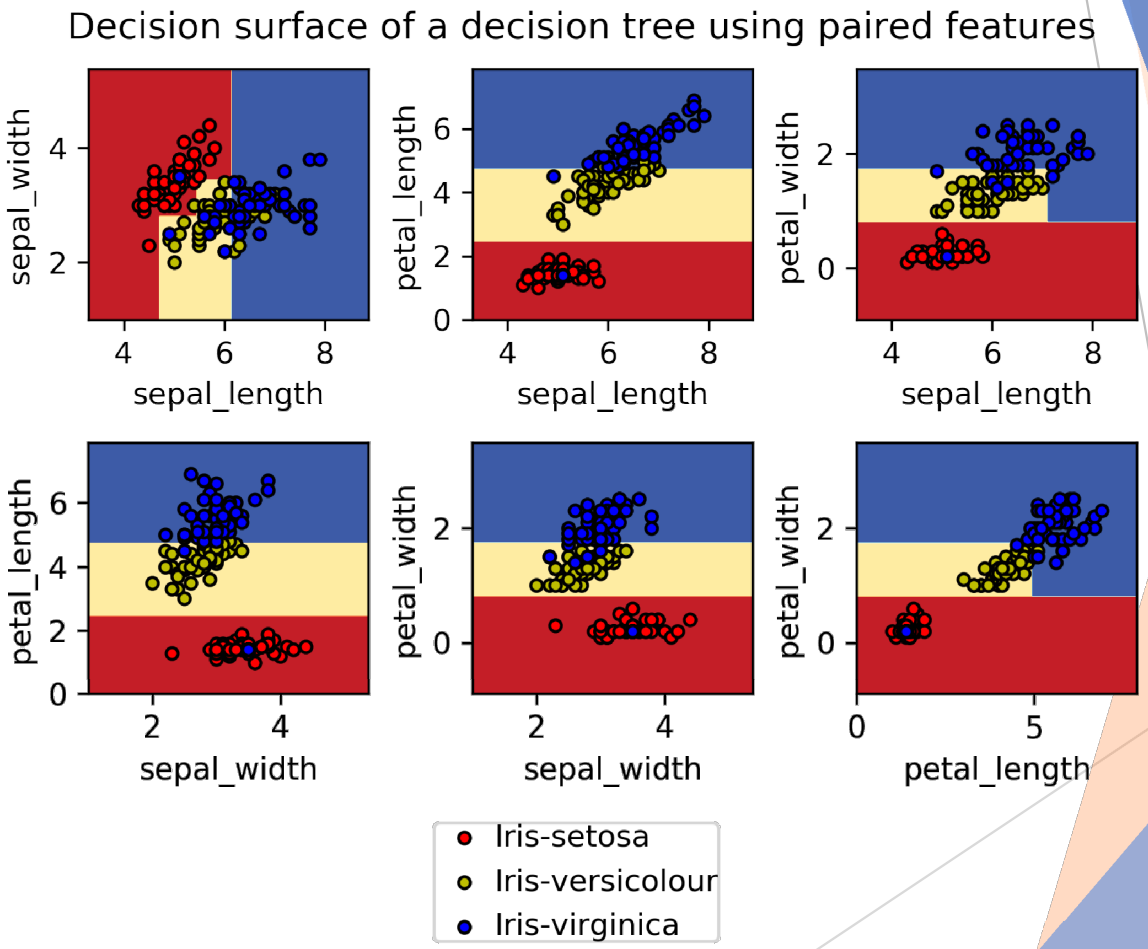
Performance on Iris Dataset (Continued…)

Confusion matrix when the model is fitted on 100% data.

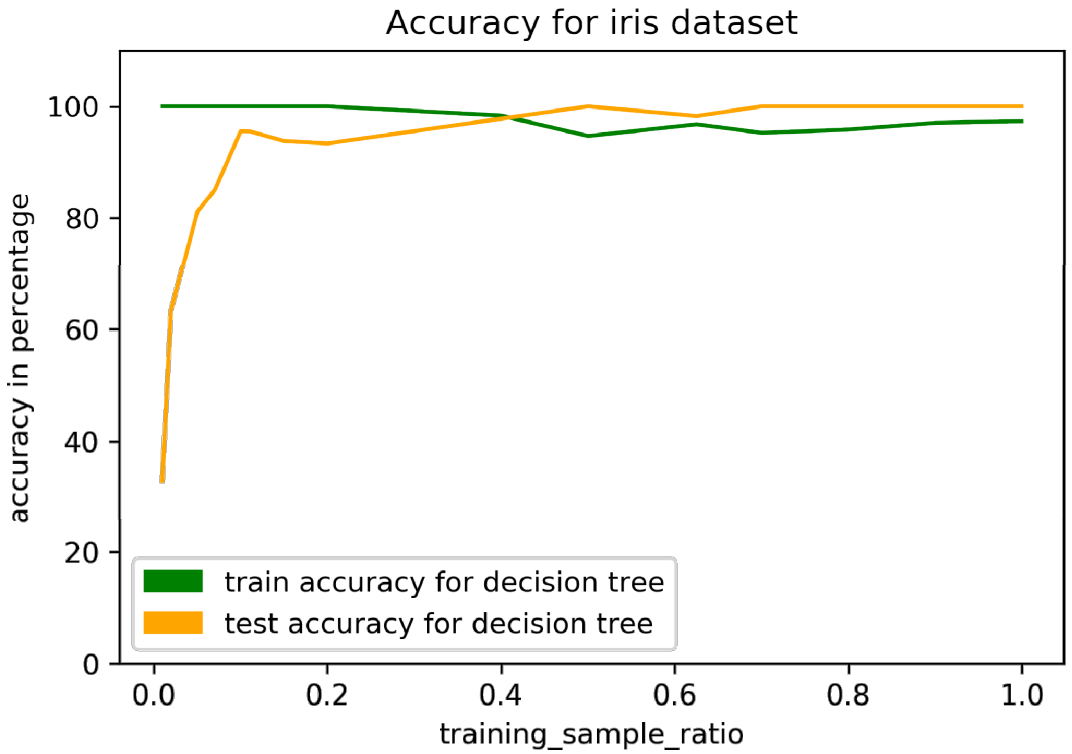


Performance on Iris Dataset (Continued...)

Decision boundary, using pairs of features when the model is fitted on 100% data.



Performance on Iris Dataset (Continued...)

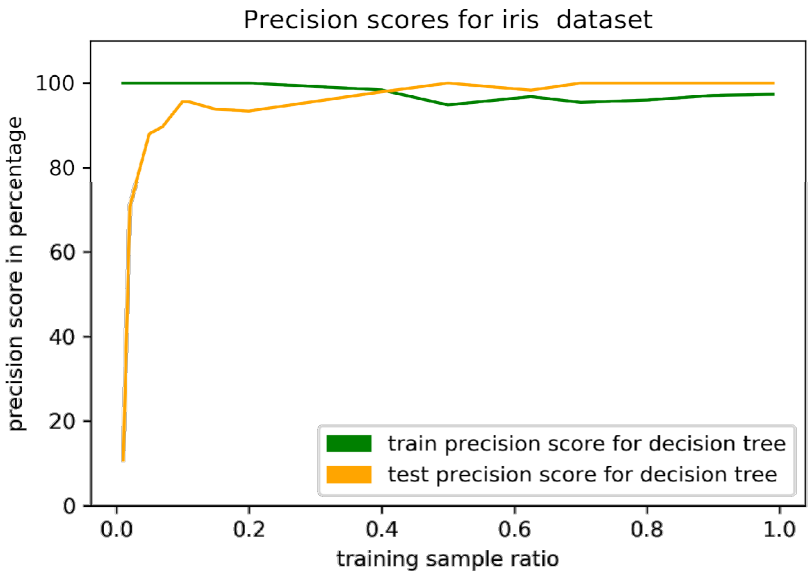


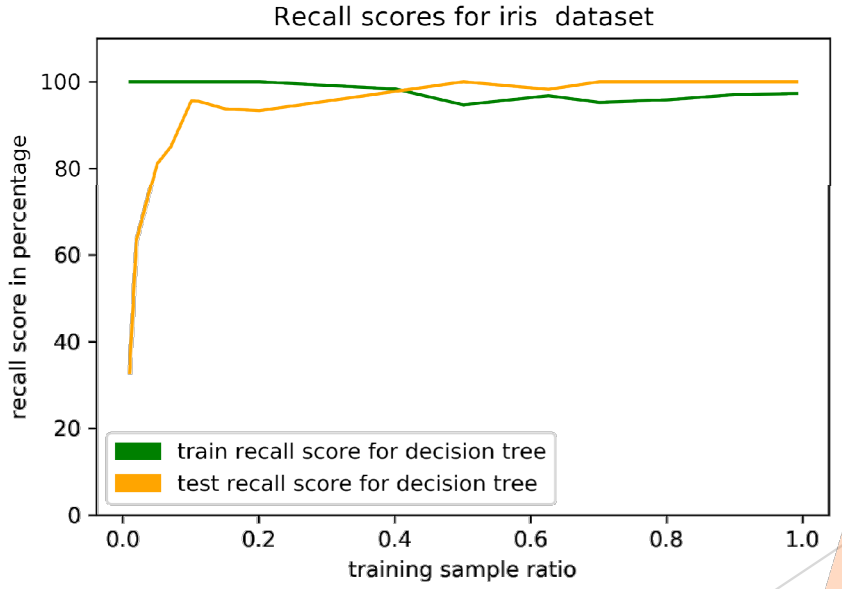
Accuracy on training and testing data, with different size of fragment of data used for fitting the model.

Accuracy when the entire data is fitted = 97.33%

Performance on Iris Dataset (Continued…)

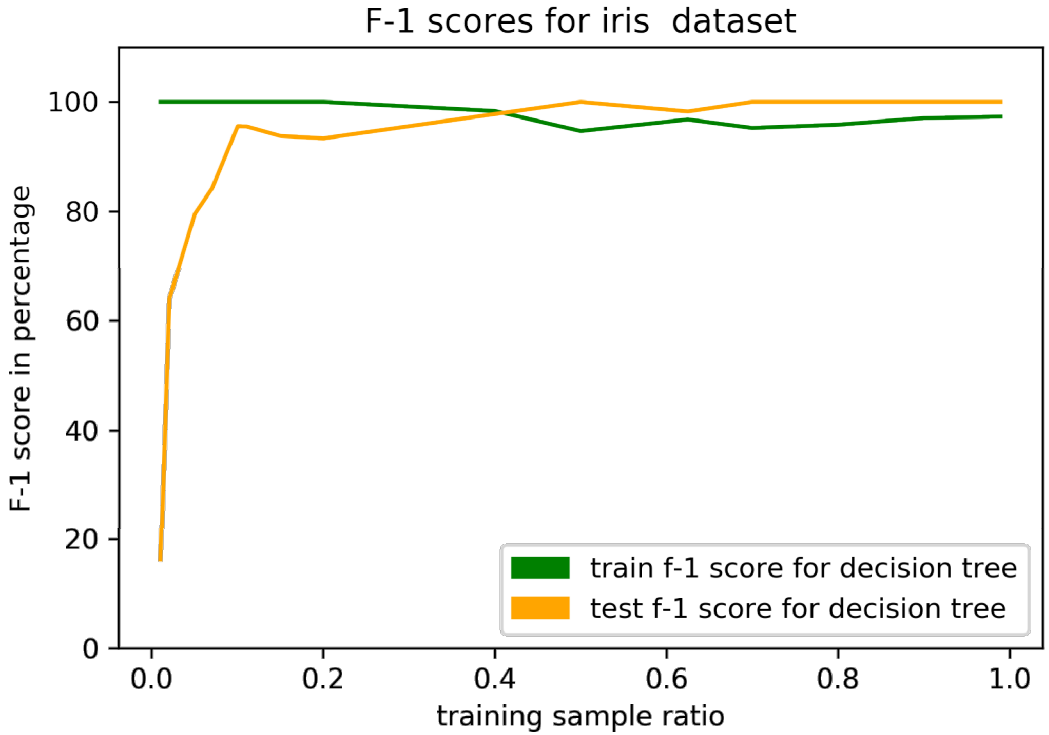
Precision and recall scores on training and testing data, with different size of fragment of data used for fitting the model.





When the entire data is fitted, precision score = 97.38% and recall score = 97.33%

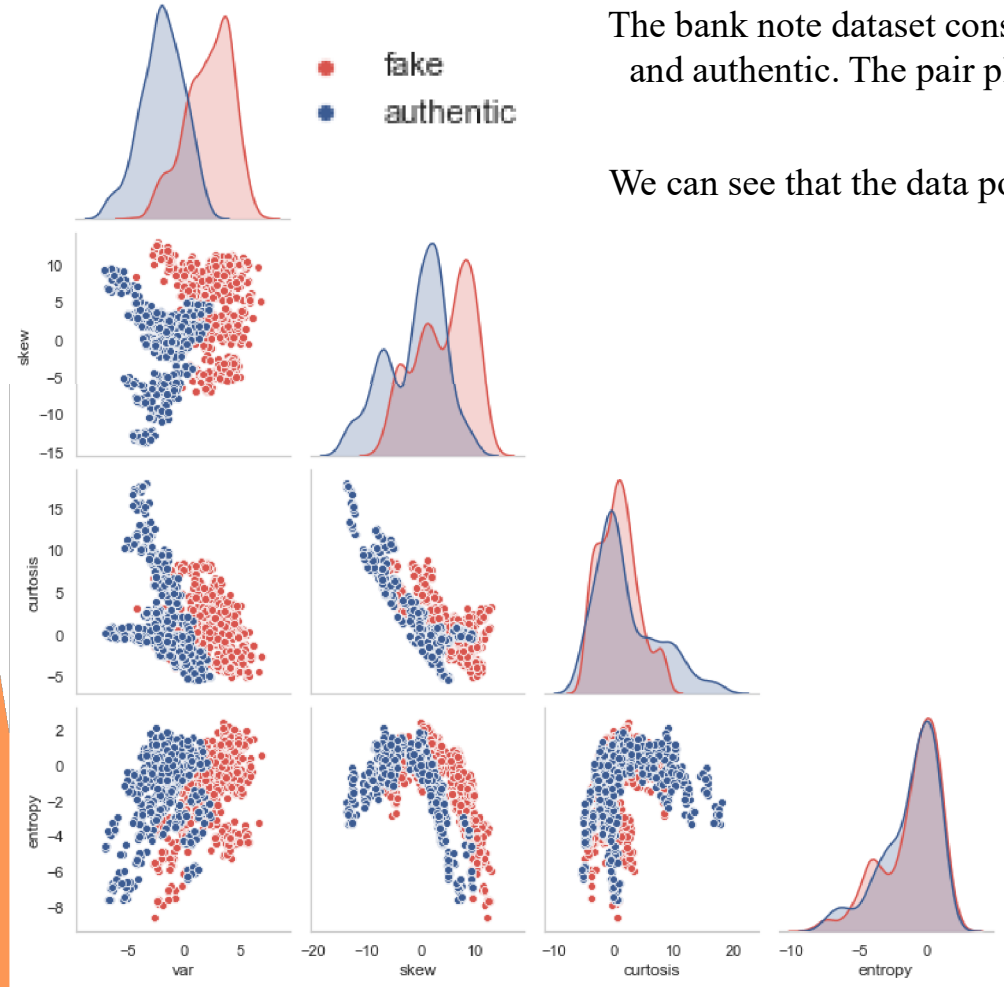
Performance on Iris Dataset (Continued...)



F-1 score on training and testing data, with different size of fragment of data used for fitting the model.

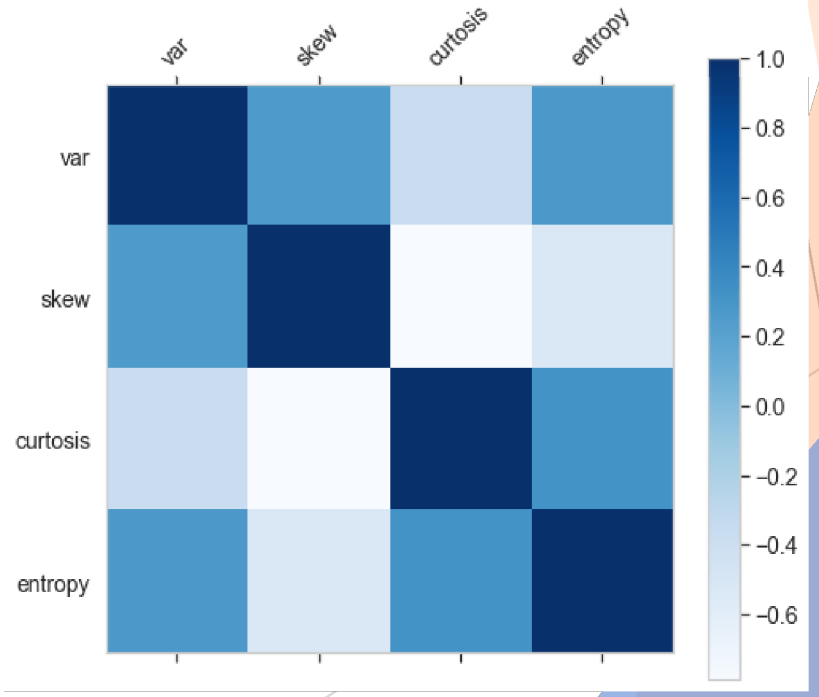
F-1 score when the entire data is fitted = 97.33%

Performance on Bank Note Dataset



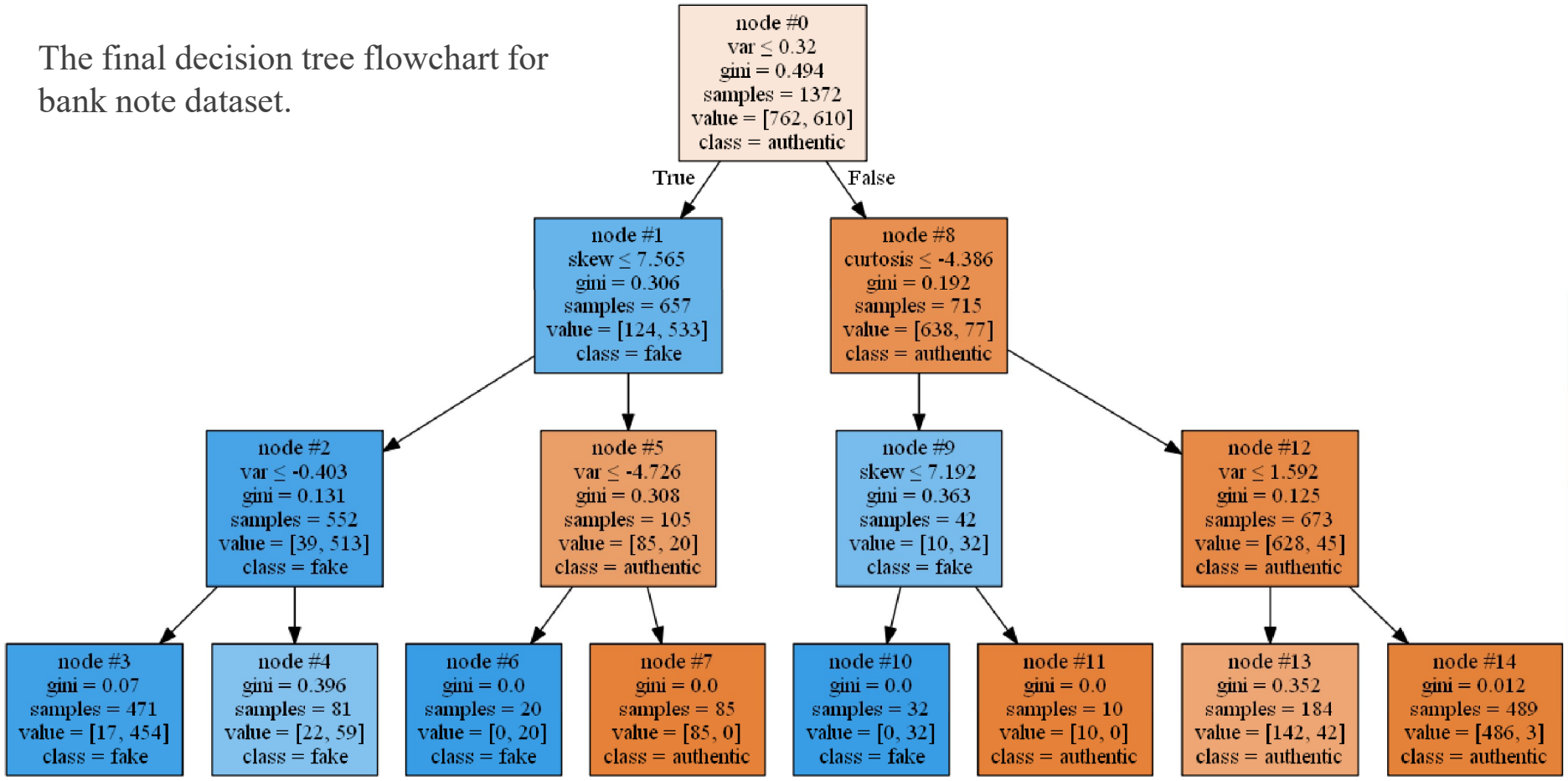
The bank note dataset consists of 1372 instances belonging to 2 classes – fake and authentic. The pair plots and correlation matrix for various attributes are in figures on the left and the bottom respectively.

We can see that the data points are not linearly separable between themselves.



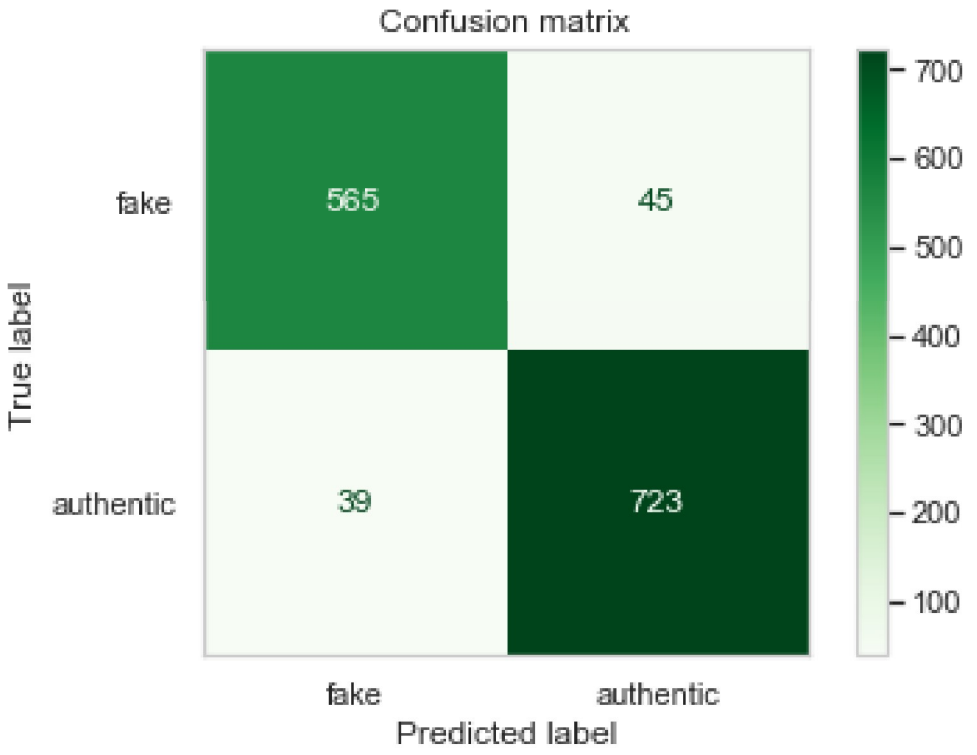
Performance on Bank Note Dataset (Continued…)

The final decision tree flowchart for bank note dataset.



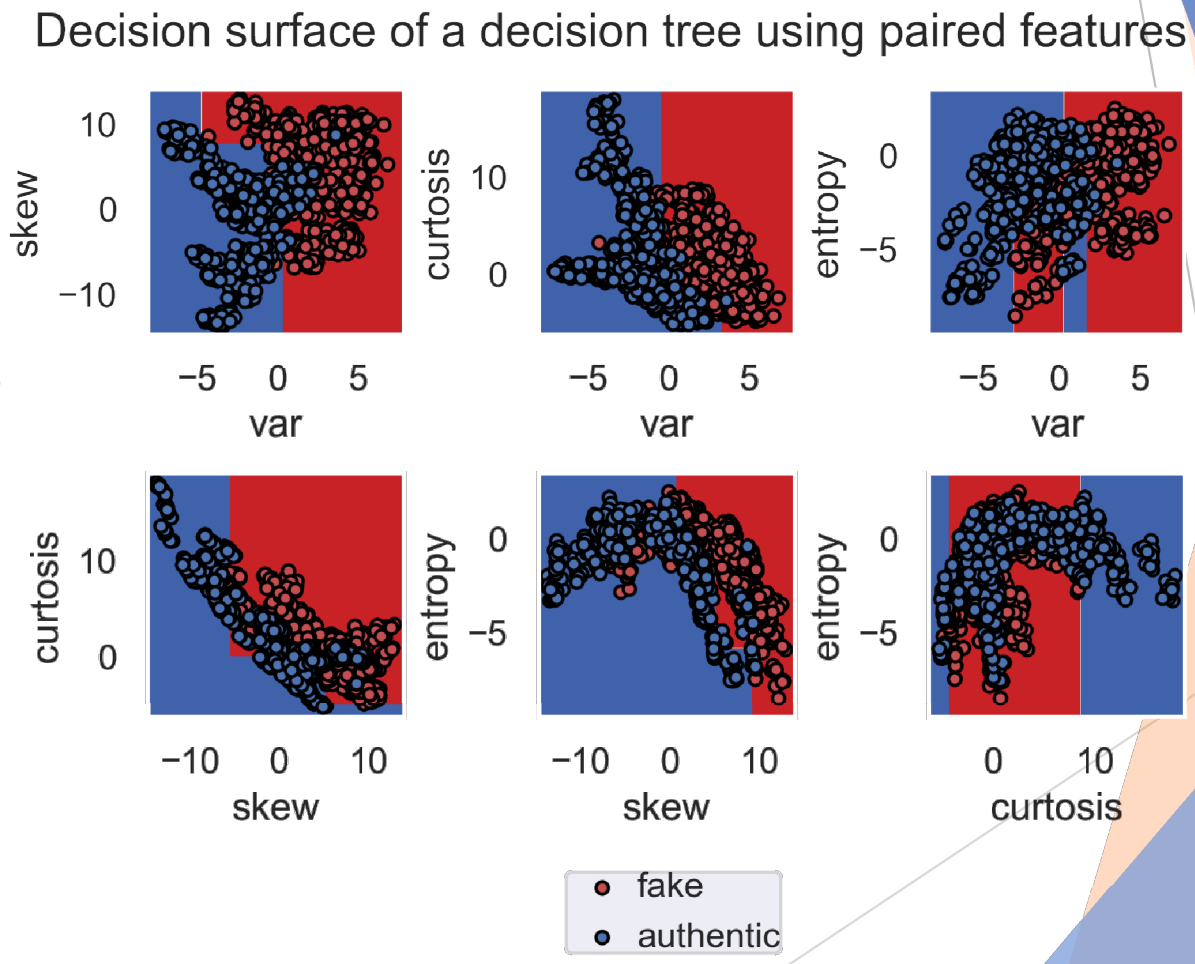
Performance on Bank Note Dataset (Continued…)

Confusion matrix when the model is fitted on 100% data.

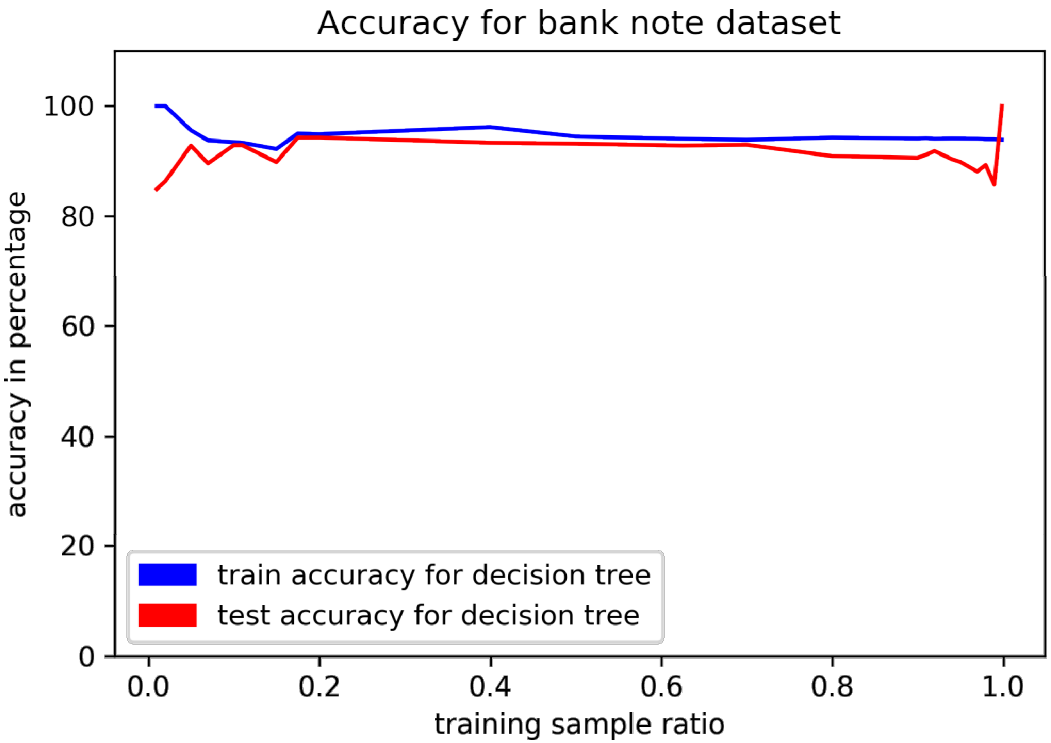


Performance on Bank Note Dataset (Continued…)

Decision boundary, using pairs of features when the model is fitted on 100% data.



Performance on Bank Note Dataset (Continued…)

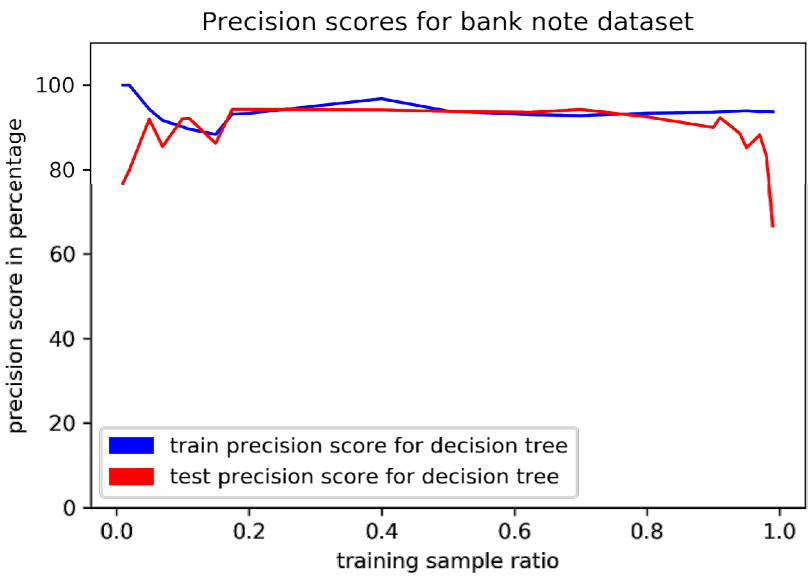


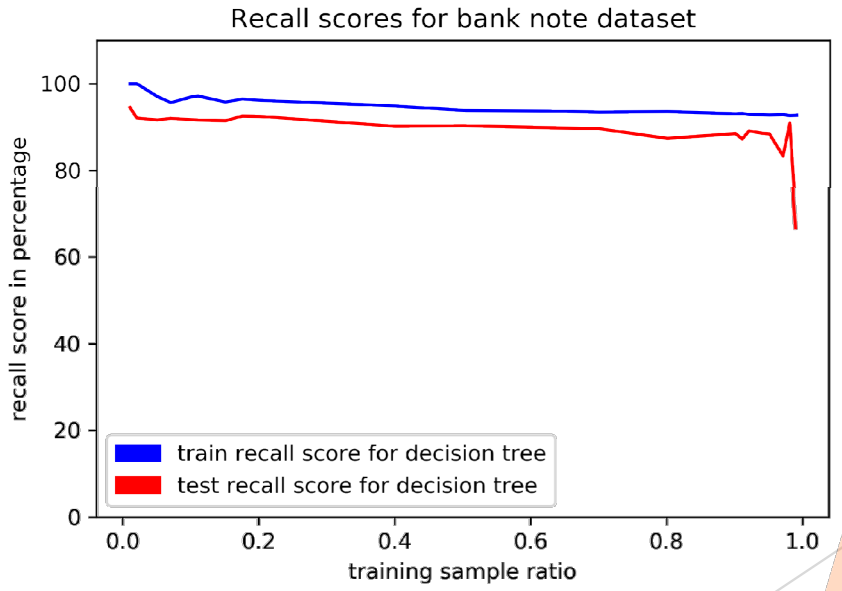
Accuracy on training and testing data, with different size of fragment of data used for fitting the model.

Accuracy when the entire data is fitted = 93.87%

Performance on Bank Note Dataset (Continued…)

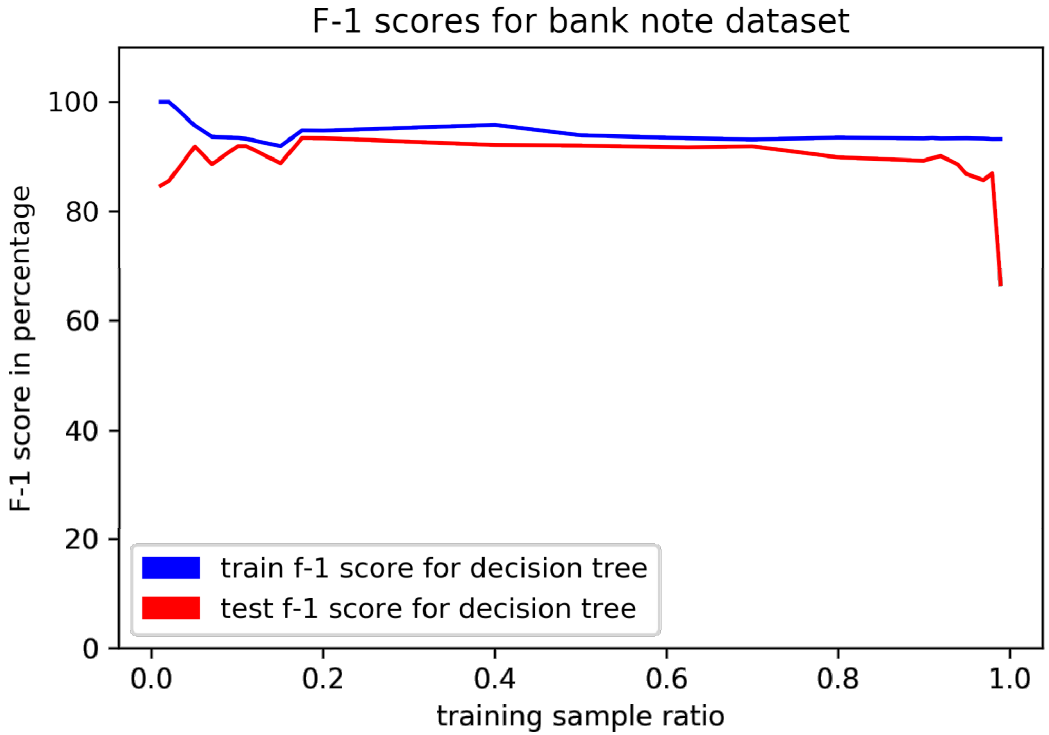
Precision and recall scores on training and testing data, with different size of fragment of data used for fitting the model.





When the entire data is fitted, precision score = 93.54% and recall score = 92.62%

Performance on Bank Note Dataset (Continued…)



F-1 score on training and testing data, with different size of fragment of data used for fitting the model.

F-1 score when the entire data is fitted = 93.08%

Discussion

- Decision Tree algorithm is excellent for distinct type data – for categorical and ordinal type variables. In order to create models containing continuous data, the data values are bucketed into intervals.
- The algorithm mentioned in this project cannot handle missing values, so it is important to use thoroughly clean data for this.
- Hyperparameters include the maximum depth of the tree, minimum number of instances in leaf nodes, minimum weighted fraction of the sum total of weights required to be at a leaf node (samples have equal weight when sample weight is not provided, this is to deal with class imbalance), minimum impurity decrease (a node will be split if this split induces a decrease of the impurity greater than or equal to this value), class weights (again, to handle class imbalance), and random state.
- Decision trees tend to overfit the data so for practical applications, it is necessary to split the data into training and test sets in order to evaluate the performance of the algorithm. Overfitting can be prevented by fine-tuning the various hyperparameters and by a method called tree-pruning. Tree-pruning involves “clipping off” the bottom of the tree to result in greater accuracy on training as well as test data.

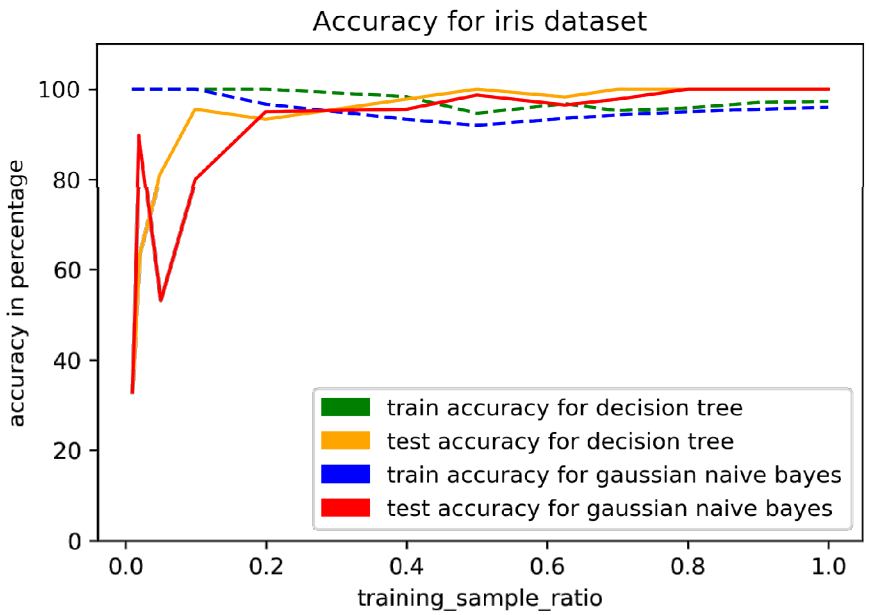
Conclusion

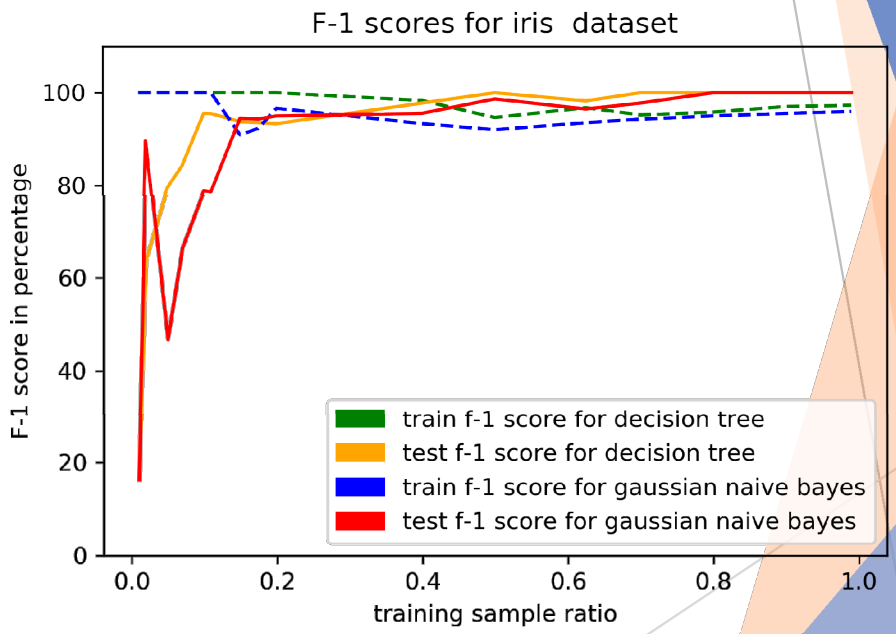
Overall, decision tree is a simple, yet powerful algorithm and is used by academicians and industry researchers in initial stages for exploring data. In some cases, its advanced version is used for final application too, giving good results.

Thank you!

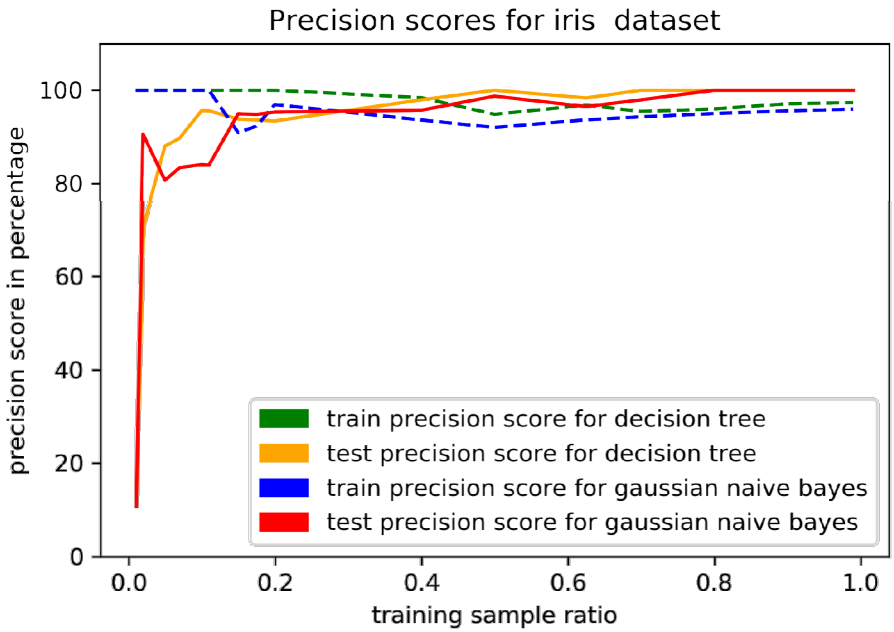
Appendix

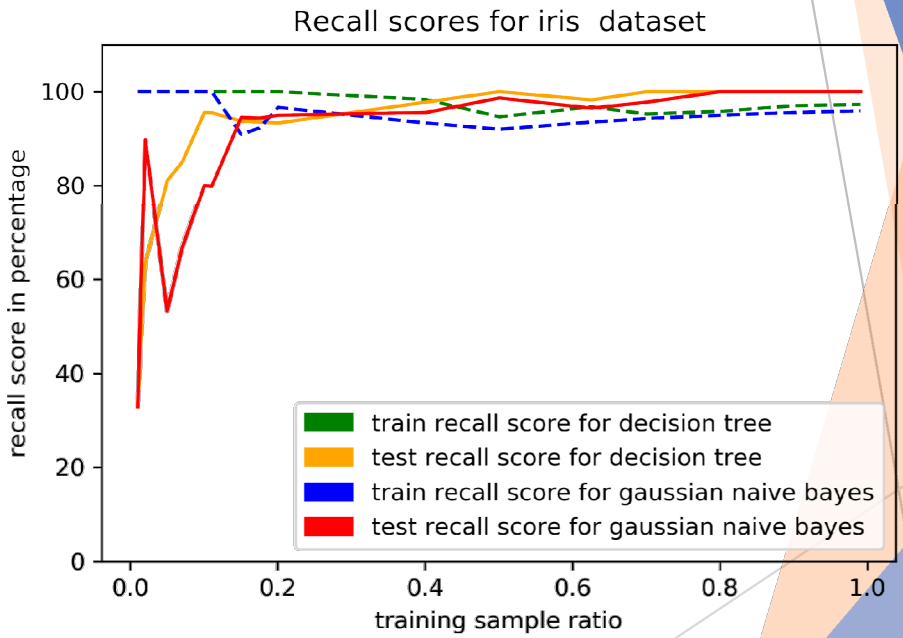
- Comparison of the algorithms on iris dataset on various parameters:





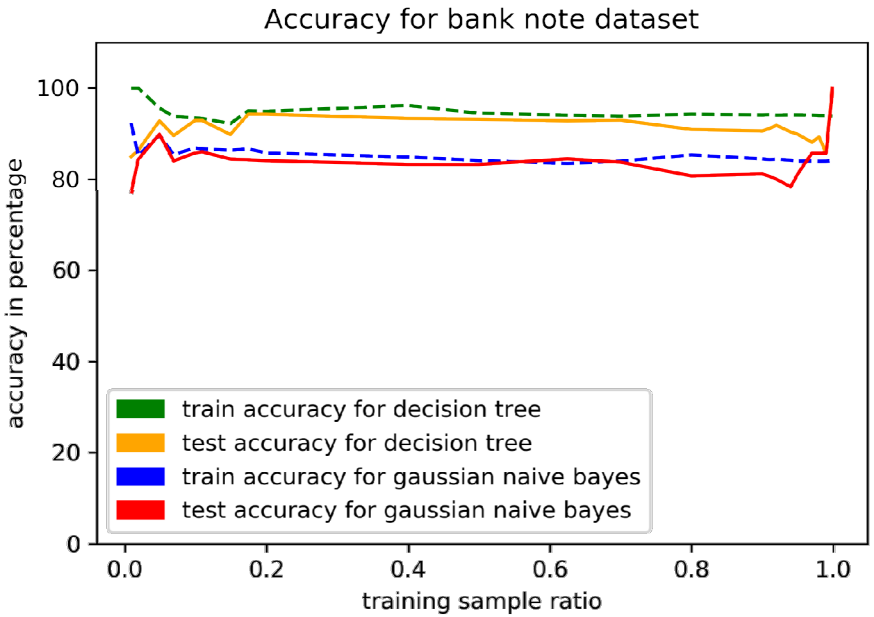
Appendix (Continued...)

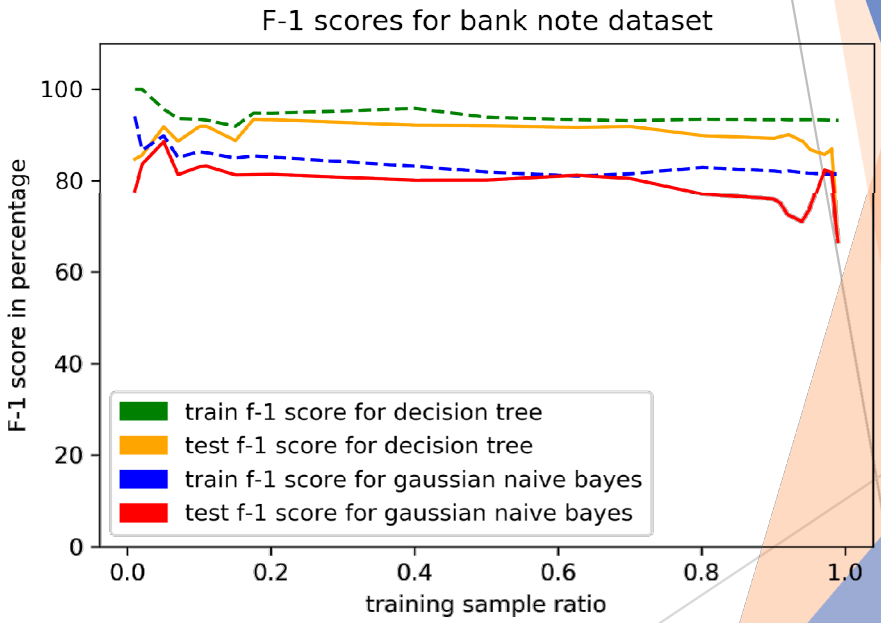




Appendix (Continued...)

- Comparison of the algorithms on bank note dataset on various parameters:





Appendix (Continued...)