

UNIVERSITAT POLITÈCNICA DE CATALUNYA

SUPERVISED AND EXPERIENTIAL LEARNING

---

PRACTICAL WORK 2  
COMBINING MULTIPLE CLASSIFIERS

---

**Author:**

CLARA RIVADULLA DURÓ

*Spring Semester*

2022-23



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH

## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Methodology</b>	<b>2</b>
2.1	Algorithms . . . . .	2
2.1.1	CART . . . . .	2
2.1.2	Decision Forest . . . . .	4
2.1.3	Random Forest . . . . .	5
2.2	Selection and pre-processing of the data sets . . . . .	5
2.3	How to execute the code? . . . . .	7
<b>3</b>	<b>Results</b>	<b>8</b>
3.1	Iris Data Set . . . . .	11
3.2	Tic-Tac-Toe Endgame Data Set . . . . .	11
3.3	Abalone Data Set . . . . .	11
3.4	Mushroom Data Set . . . . .	12
<b>4</b>	<b>Conclusions</b>	<b>13</b>
<b>Appendices</b>		<b>14</b>
<b>A</b>	<b>Full Results</b>	<b>14</b>
A.1	Iris Data Set . . . . .	14
A.2	Tic-Tac-Toe Endgame Data Set . . . . .	22
A.3	Abalone Data Set . . . . .	30
A.4	Mushroom Data Set . . . . .	36

## 1 INTRODUCTION

The main purpose for the second practical work of the Supervised and Experiential Learning course is to implement, compare and validate two combinations of multiple classifiers: a *Decision Forest* and a *Random Forest*.

In this report, I present both algorithms and how they were implemented using the CART (Classification and Regression Trees) method as the base-learner for inducing the trees. The algorithms were implemented in Python and tested on 4 different data sets of varying sizes and types, including mixed, numeric, and categorical data.

In [section 2](#), I provide a brief overview of the algorithms, how data sets were selected and how to execute the code. In [section 3](#), I compare the performance of RF and DF on these data sets and provide insights into the strengths and weaknesses of each method. Finally, I conclude the work with a summary of my findings and some other thoughts on the project in [section 4](#).

## 2 METHODOLOGY

### 2.1 ALGORITHMS

#### 2.1.1 CART

CART (*Classification and Regression Trees*) is a variation of the **decision tree** algorithm that can handle both classification and regression tasks first introduced by Leo Breiman [1].

The algorithm works by splitting the data into two subsets based on a specific feature. This process is repeated recursively until a stopping criterion is met. The resulting tree can be used to classify new data based on the features that lead to the terminal nodes.

In a decision tree, the *impurity* measure refers to the measure of the homogeneity of the target variable within a node. In other words, impurity is a way to measure how well a split separates the data based on the target variable. The impurity measure we were meant to use in the implementation of CART is *Gini* (Figure 1). The value of  $Gini(X)$  lies between 0 and  $1 - \frac{1}{k}$ .

$$Gini(X) = 1 - \sum_{i=1}^k p_{x \in C_i}^2$$

Figure 1: *Gini*-index of impurity.

The baseline CART method used in the implementations of DF and RF is almost the same, except for the *bestSplit* function, which is defined differently for every method. Here's the pseudocode for the Decision Tree (CART) (Figure 2) and both *bestSplit* functions, one for DF (Figure 3) and one for RF (Figure 4).

**DecisionTree**(*max\_depth*, *min\_impurity*, *X\_train*, *y\_train*, *X\_test*)

1. Fit the model
  - 1.1. *classes*  $\leftarrow$  Unique classes in *y*
  - 1.2. *features*  $\leftarrow$  Number of features in *X\_train*
  - 1.3. *root*  $\leftarrow$  *growTree*(*X\_train*, *y\_train*) Grow the decision tree
2. *predictions*  $\leftarrow$  [*predict\_x*(*x*) for *x* in *X\_test*] Traverse the tree for every data instance in *X\_test* finding where the *node.feature\_idx* value falls (below or above the *node.threshold*, left or right)
3. Return the *predictions*

*growTree*(*X*, *y*, *depth*=0)

1. *node*  $\leftarrow$  new *Node*(*pred\_class*  $\leftarrow$  Most common class in *y*)
2. if *depth* < *max\_depth*:
  - 2.1. *feature\_idx*, *threshold*, *impurity*  $\leftarrow$  *bestSplit*(*X*, *y*)
  - 2.2. *left\_idxs*  $\leftarrow$  Select indices where instances are below or equal the *threshold* in the *feature\_idx*
  - 2.3. *right\_idxs*  $\leftarrow$  Select indices where instances are above the *threshold* in the *feature\_idx*
  - 2.4. Save the *feature\_idx*, the *threshold* and the *impurity* inside the *node*
  - 2.5. *node.left*  $\leftarrow$  *growTree*(*X*[*left\_idxs*], *y*[*left\_idxs*], *depth* + 1)
  - 2.6. *node.right*  $\leftarrow$  *growTree*(*X*[*right\_idxs*], *y*[*right\_idxs*], *depth* + 1)
3. Return the *node*

Figure 2: Pseudocode of the CART Decision Tree.

*bestSplitDF*(*X*, *y*)

1. If *len(y)*  $\leq$  1 return None, None, None
2. *best\_gini*  $\leftarrow$  Compute the Gini impurity of the parent node
3. *best\_threshold*, *best\_idx*  $\leftarrow$  None, None
4. If *best\_gini*  $\geq$  *min\_impurity*:
  - 4.1. Loop over every feature index *feature\_idx* in *features*
    - 4.1.1. *thresholds*  $\leftarrow$  Possible thresholds (unique values in *X* for feature *j*)
    - 4.1.2. Compute the midpoints of the *thresholds* if possible (numeric features only)
    - 4.1.3. For every *threshold* in *thresholds*:
      - 4.1.3.1. *left\_idxs*  $\leftarrow$  Select indices where instances are below or equal the *threshold* in the *feature\_idx*
      - 4.1.3.2. *right\_idxs*  $\leftarrow$  Select indices where instances are above the *threshold* in the *feature\_idx*
      - 4.1.3.3. *gini*  $\leftarrow$  Compute the Gini impurity of the current node
      - 4.1.3.4. If *gini* < *best\_gini*: *best\_gini*  $\leftarrow$  *gini*, *best\_idx*  $\leftarrow$  *feature\_idx*, *best\_threshold*  $\leftarrow$  *threshold*
  5. Return *best\_index*, *best\_threshold*, *best\_gini*

Figure 3: Pseudocode of the *bestSplit* function of a CART Decision Tree for the Decision Forest algorithm.

```

bestSplitRF(X, y)
1. If len(y) <= 1 return None, None, None
2. best_gini ← Compute the Gini impurity of the parent node
3. best_threshold, best_idx ← None, None
4. features_idxs ← Select F random feature indices
5. If best_gini >= min_impurity:
   5.1. Loop over every feature index feature_idx in features_idxs
       5.1.1. thresholds ← Possible thresholds (unique values in X for feature
              j)
       5.1.2. Compute the midpoints of the thresholds if possible (numeric
              features only)
       5.1.3. For every threshold in thresholds:
              5.1.3.1. left_idxs ← Select indices where instances are
                     below or equal the threshold in the feature_idx
              5.1.3.2. right_idxs ← Select indices where instances are
                     above the threshold in the feature_idx
              5.1.3.3. gini ← Compute the Gini impurity of the current
                     node
              5.1.3.4. If gini < best_gini: best_gini ← gini, best_idx ←
                     feature_idx, best_threshold ← threshold
6. Return best_index, best_threshold, best_gini

```

Figure 4: Pseudocode of the *bestSplit* function of a CART Decision Tree for the Random Forest algorithm.

On the other hand, although it would've been nice to implement a post-pruning technique (such as *Cost Complexity Pruning*), unfortunately I have not been able to make it work. However, some **pre-pruning** techniques have been considered (minimum number of data points in a leaf node, maximum depth and minimum impurity allowed).

### 2.1.2 DECISION FOREST

A *Decision Forest* (DF), proposed by *Tin Kam Ho* in 1998 [2], is an ensemble method that uses multiple Decision Trees. Every Decision Tree in a DF is trained with the same original training set but choosing F random features each time, and the final prediction is made by combining the predictions of all trees. Here's the pseudocode of my implementation of the algorithm, that uses the *Decision Tree* algorithm shown previously (Figure 2) along with the *bestSplitDF* function (Figure 3).

```

DecisionForest(max_depth, min_impurity, NT, F, X_train, y_train, X_test)
trees  $\leftarrow$  []
predictions  $\leftarrow$  []
1. Repeat NT times:
    1.1. Select F features randomly from all features in X_train
    1.2. tree  $\leftarrow$  DecisionTree(max_depth, min_impurity)
    1.3. Fit the tree with the subset of X_train containing the randomly selected
        F features and y_train
    1.4. Append the tree to trees along with the indices of the randomly selected
        features
2. Predict and append the predictions of X_test with every tree in trees
    (considering only the stored feature indices for each tree) to predictions
3. Return the final prediction (taking the most common prediction for every data
    instance in X_test)

```

Figure 5: Pseudocode of the implemented Decision Forest algorithm.

### 2.1.3 RANDOM FOREST

A *Random Forest* (RF), proposed by *Leo Breiman* in 2001 [3], is also an ensemble learning method that combines multiple *Decision Trees* to improve the accuracy and robustness of the predictions. With RF, numerous Decision Trees are created, each trained on a different subset of the training data (taking a bootstrapped sampling of the original training set) and using a randomly selected subset of features for each node splitting. As in DF, the final prediction is made by combining the predictions of all the trees. Here's the pseudocode of my implementation of the algorithm, that uses the *Decision Tree* algorithm shown previously (Figure 2) along with the *bestSplitRF* function (Figure 4).

```

RandomForest(max_depth, min_impurity, NT, F, X_train, y_train, X_test)
trees  $\leftarrow$  []
predictions  $\leftarrow$  []
1. Repeat NT times:
    1.1. bootstrap_X, bootstrap_y  $\leftarrow$  Bootstrap the X_train and y_train sets
        randomly
    1.2. tree  $\leftarrow$  DecisionTree(max_depth, min_impurity, F)
    1.3. Fit the tree with bootstrap_X and bootstrap_y
    1.4. Append the tree to trees
2. Predict and append the predictions of X_test with every tree in trees
    (considering only the stored feature indices for each tree) to predictions
3. Return the final prediction (taking the most common prediction for every data
    instance in X_test)

```

Figure 6: Pseudocode of the implemented Random Forest algorithm.

## 2.2 SELECTION AND PRE-PROCESSING OF THE DATA SETS

The data sets have been chosen according to the restrictions given: "choose 3 different data sets, one of small size ( $\#instances \leq 500$ ), one of medium size ( $500 < \#instances \leq 2000$ ) and one of large size ( $\#instances > 2000$ )".

However, I found it necessary to use 4 data sets for testing purposes: 1 small with numeric

features only, 1 medium with categorical features only, 1 large with mixed (numeric and categorical) features + 1 large with categorical features only (because the mixed one gave too bad results, as we'll see in [section 3](#)). Each data set is described next, but an overall comparison can also be found in [Table 1](#).

So, taking all of this into consideration, the selected data sets for testing the algorithm are:

1. **Iris Data Set [4] (Small)** It consists of *150 instances* of iris flowers, where each sample is described by *4 features*: sepal length, sepal width, petal length, and petal width, all measured in centimeters. The data set is commonly used for classification tasks, where the goal is to predict the species of iris flower. There are three classes in the data set: Iris Setosa, Iris Versicolor, and Iris Virginica, with 50 samples in each class.
2. **Tic-Tac-Toe Endgame Data Set [5] (Medium)** It contains *958 instances* representing all possible endgame scenarios for the classic game of Tic-Tac-Toe. The data set has *9 attributes* representing each square on the board, with three possible values: x (representing the player who put an "X" on the square), o (representing the player who put an "O" on the square), and b (representing a blank square). The goal of the data set is to predict the class label for each instance, which represents the outcome of the game: 'positive' or 'negative'.
3. **Abalone Data Set [6] (Large)** It consists of measurements of physical characteristics of abalone, a type of marine snail, and their age, which is determined by cutting the shell and counting the number of rings. The data set contains *4177 instances*, where each sample is described by *8 features*, including sex, length, diameter, height, weight of the whole abalone, weight of the meat, weight of the shell, and number of rings. The data set is often used for regression tasks, where the goal is to predict the age of an abalone based on its physical characteristics. The Abalone data set is also used for classification tasks, where the goal is to predict the sex of an abalone based on its physical characteristics.
4. **Mushroom Data Set [7] (Large)** It contains information about various characteristics of mushrooms, including their physical attributes, habitat, and edibility. The data set has *8124 instances* and *23 attributes*, including a class label indicating whether each mushroom is 'edible' or 'poisonous'.

Data set	# Classes	# Instances	# Attributes	# Numeric attr.	# Categorical attr.
Iris	3	150	4	All	None
Tic-Tac-Toe Endgame	2	958	9	None	All
Abalone	28	4177	8	7	1
Mushroom	2	8124	22	None	All

Table 1: Summarized description of every data set.

### 2.3 HOW TO EXECUTE THE CODE?

Before running the code for the first time, you must change the `path` written in the 12th line of the `main.py` file to the one where you've stored the folder containing the unzipped project. Once the path is changed, you must follow the next steps:

1. Open the folder containing the code of the project (*source*) in the terminal

```
cd <root_folder_of_project>/source
```

2. Create a virtual environment using Python

```
python3 -m venv venv/
```

3. Open the virtual environment

```
source venv/bin/activate
```

4. Install the required dependencies

```
pip install -r requirements.txt
```

5. Run the main file of the project

```
python main.py
```

### 3 RESULTS

In this section, I'm going to discuss the results obtained for every data set, emphasizing the best and the worst of each one, both with the Decision Forest and the Random Forest model, as well as with the baseline Decision Tree.

All the resulting accuracies, for all data sets, method (Decision Forest or Random Forest) and possible combinations of parameters (number of trees, NT, and number of randomly chosen features, F), can be found in [Table 2](#). Additionally, a few plots are included in this section so that these results can be easily and visually analyzed ([Table 3](#)).

The full output of the program with all the accuracies, the computation times and the 3 most relevant features for each model and data set can be found in [Appendix A](#), organized by data set.

The NT values tried with every data set and method are: 1, 10, 25, 50, 75, 100. The F values for Decision Forest are:  $\text{int}(\frac{m}{4})$ ,  $\text{int}(\frac{m}{2})$ ,  $\text{int}(3 * \frac{m}{4})$ ,  $\text{Runif}(1, M)$ , and the F values for Random Forest: 1, 2,  $\text{int}(\log_2(M+1))$ ,  $\text{int}(\sqrt{m})$ ,  $m$ , being  $m$  the total number of features. The maximum depth of the trees is always 100 because incrementing it didn't have any effect on the results, and decrementing it a lot could make them even worse. The minimum impurity accepted is always  $1e - 7$ .

Iris				Tic Tac Toe Endgame				Abalone				Mushroom			
method	NT	F	accuracy	method	NT	F	accuracy	method	NT	F	accuracy	method	NT	F	accuracy
DF	1	4	100.0	RF	10	9	99.48	RF	50	1	27.87	DF	1	20	100.0
DF	10	2	100.0	RF	75	3	99.48	RF	75	1	27.63	DF	1	22	100.0
DF	10	3	100.0	RF	100	3	99.48	RF	100	1	26.91	DF	10	11	100.0
DF	10	4	100.0	RF	25	9	98.96	RF	100	2	26.91	DF	10	16	100.0
DF	25	4	100.0	RF	50	3	98.96	RF	10	1	25.84	DF	10	20	100.0
DF	50	3	100.0	RF	75	2	98.96	RF	75	3	25.72	DF	10	22	100.0
DF	50	4	100.0	RF	75	9	98.96	RF	25	2	25.48	DF	25	11	100.0
DF	75	4	100.0	RF	100	9	98.96	RF	50	3	25.24	DF	25	16	100.0
DF	100	4	100.0	RF	25	3	98.44	RF	25	1	25.0	DF	25	20	100.0
RF	25	2	100.0	RF	50	9	98.44	DF	75	2	24.76	DF	25	22	100.0
RF	25	4	100.0	DF	75	8	96.88	DF	75	4	24.76	DF	50	5	100.0
RF	50	4	100.0	RF	50	2	96.88	RF	10	2	24.64	DF	50	11	100.0
RF	75	2	100.0	RF	100	2	96.88	DF	25	4	24.52	DF	50	16	100.0
RF	75	4	100.0	DF	100	8	96.35	DF	100	4	24.16	DF	50	20	100.0
RF	100	1	100.0	DF	10	9	95.83	RF	25	3	24.16	DF	50	22	100.0
RF	100	2	100.0	DF	25	8	95.83	DF	25	2	23.21	DF	75	5	100.0
RF	100	4	100.0	DF	50	8	95.83	DF	75	6	23.21	DF	75	11	100.0
DF	1	3	96.67	DF	75	9	95.83	DF	25	6	22.85	DF	75	16	100.0
DF	25	3	96.67	DF	100	9	95.83	DF	50	2	22.73	DF	75	20	100.0
DF	50	2	96.67	DF	10	8	95.31	DF	50	6	22.49	DF	75	22	100.0
DF	75	1	96.67	DF	25	9	95.31	DF	100	6	22.13	DF	100	5	100.0
DF	75	2	96.67	DF	50	9	95.31	DF	10	6	21.77	DF	100	11	100.0
DF	75	3	96.67	DF	100	6	94.79	DF	10	2	21.65	DF	100	16	100.0
DF	100	3	96.67	RF	1	9	94.79	RF	10	3	21.65	DF	100	20	100.0
RF	1	1	96.67	DF	1	9	94.27	DF	10	4	21.41	DF	100	22	100.0
RF	1	2	96.67	DF	75	6	93.23	RF	1	1	20.33	RF	1	22	100.0

RF	1	4	96.67	RF	10	3	93.23	DF	1	4	19.74	RF	10	2	100.0
RF	10	1	96.67	DF	50	6	92.19	RF	1	2	19.62	RF	10	4	100.0
RF	10	4	96.67	RF	25	2	91.15	RF	1	3	18.78	RF	10	22	100.0
RF	25	1	96.67	DF	1	8	90.62	DF	1	2	16.99	RF	25	2	100.0
RF	50	1	96.67	RF	10	2	90.62	RF	50	2	25.12	RF	25	4	100.0
RF	50	2	96.67	DF	25	6	90.1	RF	100	3	25.12	RF	25	22	100.0
RF	75	1	96.67	RF	50	1	86.46	DF	50	4	23.8	RF	50	2	100.0
RF	10	2	93.33	RF	75	1	85.94	DF	100	2	23.8	RF	50	4	100.0
DF	1	2	90.0	DF	10	6	85.42	RF	75	2	25.6	RF	50	22	100.0
DF	25	2	90.0	DF	75	4	83.85	DF	1	6	19.02	RF	75	2	100.0
DF	100	2	90.0	DF	50	4	81.77					RF	75	4	100.0
DF	25	1	80.0	DF	25	4	81.25					RF	75	22	100.0
DF	50	1	80.0	DF	10	4	80.73					RF	100	2	100.0
DF	100	1	80.0	RF	25	1	80.73					RF	100	4	100.0
DF	10	1	76.67	RF	10	1	79.17					RF	100	22	100.0
DF	1	1	53.33	RF	1	3	78.12					DF	25	5	99.82
				RF	100	1	78.12					RF	1	4	99.73
				DF	100	4	77.08					DF	10	5	99.65
				DF	1	6	76.04					DF	1	16	99.38
				RF	1	2	72.92					RF	50	1	98.49
				DF	10	2	72.4					DF	1	11	98.41
				DF	25	2	70.31					RF	100	1	95.75
				DF	50	2	70.31					RF	1	2	94.95
				DF	75	2	70.31					RF	75	1	94.42
				DF	100	2	70.31					RF	1	1	93.71
				DF	1	2	69.79					RF	25	1	93.45
				DF	1	4	68.75					RF	10	1	90.79
				RF	1	1	62.5					DF	1	5	84.68

Table 2: Accuracies obtained for every data set, method and combination of parameters (number of trees, NT, and number of randomly chosen features, F).

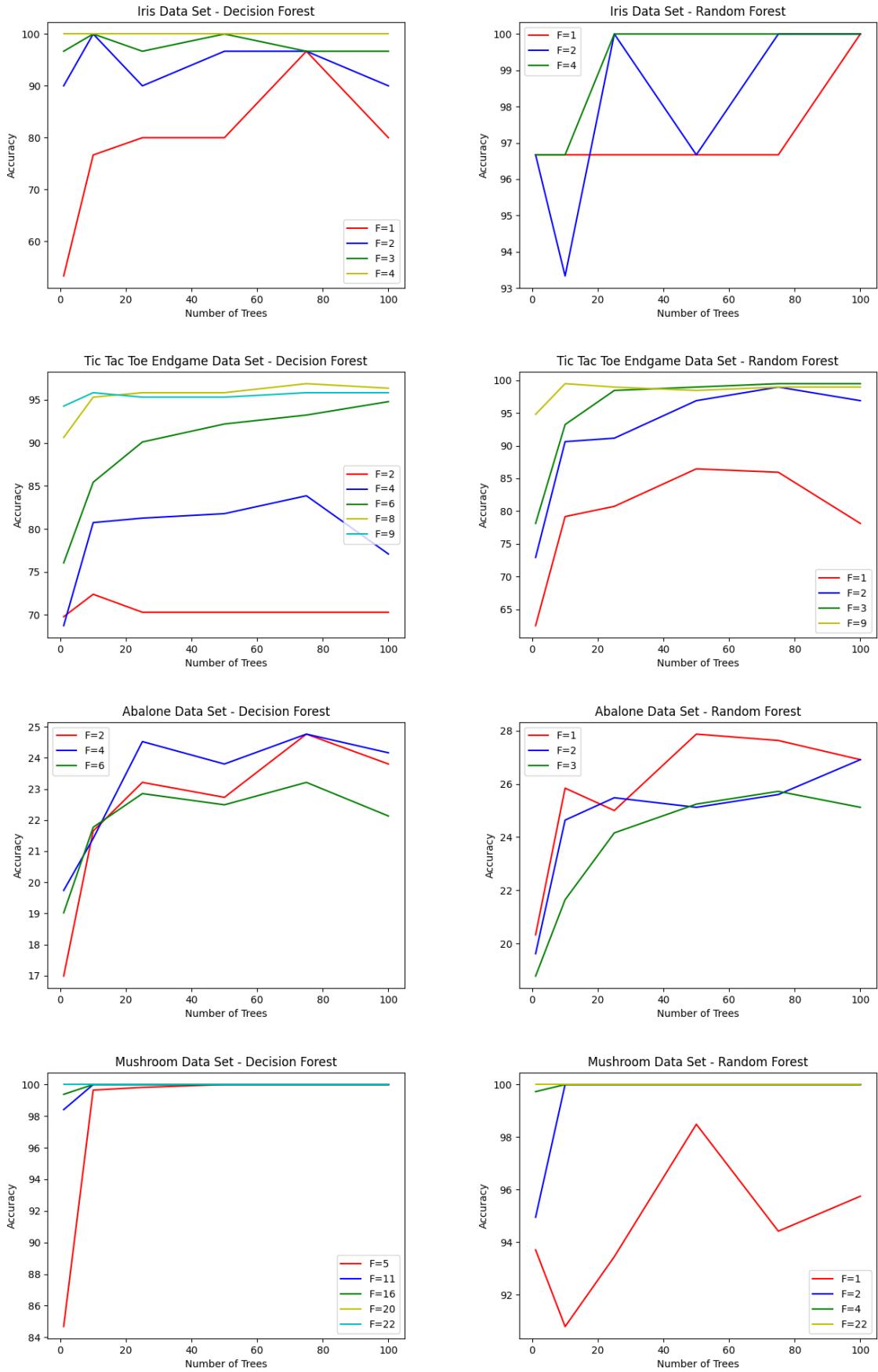


Table 3: Plots of the resulting accuracies for every number of trees, number of features (F), method (Decision Forest and Random Forest) and data set.

### 3.1 IRIS DATA SET

As we can see in [Table 3](#), the *Iris Data Set* gives the most outstanding results, along with the *Mushroom Data Set* (see [subsection 3.4](#)). With the Decision Forest method, accuracies improve as the number of features increases. The more features we take into consideration, the best results we get. It happens similarly with Random Forest, except that as the number of trees increases up to a certain point (NT=100, for example), the accuracies achieved are maximum no matter how many features we're looking at. With DF, accuracies range from 53.33% to 100%; with RF, accuracies are not inferior to 93.33% ([Table 2](#)).

If we have a look at the output where we can see the 3 most relevant features for every model ([subsection A.1](#)), we can see that the feature that appears more times in the 1st place is the 'petal.width' (26 times), followed by the 'petal.length' (23 times). This could mean that the petal's information of a flower determines its species to a greater extent.

Nonetheless, we can't say that using DF or RF is beneficial for this specific data set, as we already got a perfect accuracy with a single Decision Tree and all features. And faster, in fact (see [subsection A.1](#)).

### 3.2 TIC-TAC-TOE ENDGAME DATA SET

Regarding the *Tic-Tac-Toe Endgame Data Set*, the range of accuracies is now higher with RF (the lower accuracy is 62.5%, with F=1 and NT=1, and the higher is 99.48%, with F=3 and NT<sub>i</sub>=75 or F=9 and NT=10, as seen in [Table 2](#)). Although the fact that the lower accuracy is with RF is probably due to not having tried DF with F=1. More or less, for both methods, the accuracy rises as the number of feature does. However, we can say that we need a lower number of trees and features with RF, and that not even with higher values does the DF method achieve the same results.

The most relevant feature is 'V7' (it appears 14 times in the first place in the output [subsection A.2](#)), followed by 'V9' (it appears 13 times).

In this case, using a combination of classifiers such as RF does improve the accuracy. With a single Decision Tree, the accuracy was of a 95.31%, and with RF we've been able to increase it to almost the maximum.

### 3.3 ABALONE DATA SET

The *Abalone Data Set* is by far the worst of them all. It could be because it's the only mixed data set (which made me have second thoughts about my implementation, but I really couldn't find the mistake), or just because it's not so optimal for classification tasks. However, we can see that, once again, RF performed better than DF, giving a maximum accuracy of the 27.87%

with  $NT=50$  and  $F=1$  (against the 24.76% of DF with  $NT=75$  and  $F=2$ ), as we can see in [Table 2](#).

The most relevant features are 'Shell\_weight' (with 15 appearances in the first place) and 'Whole\_weight' (with 7 appearances in the first place).

The accuracies improved considerably with respect to fitting a single Decision Tree (which gave an accuracy of the 20.69%), but the time it takes to fit and predict the models increased as well (quite a lot), as we can see in [subsection A.3](#).

### 3.4 MUSHROOM DATA SET

Finally, the results of the *Mushroom Data Set* were very satisfactory, achieving a perfect accuracy with many configurations of parameters ([Table 2](#)). As we can see in [Table 3](#), with only 2 features and just a few trees (10), RF can predict perfectly. DF requires at least 11 features to achieve the same results with 10 trees. Nonetheless, this is no surprise, as we would already get 100% of accuracy with a single DT that considers all features, and it would even take less time to do so (see [subsection A.4](#)). The most relevant feature is undoubtedly the 'odor', with 39 appearances in the first place of the 3 most relevant ones.

## 4 CONCLUSIONS

After implementing a Decision Forest and a Random Forest algorithm, and after putting them to test with the 4 selected data sets, we can conclude that:

- The best results were obtained with the *Iris* and the *Mushroom* data sets, achieving perfect accuracies for both of them.
- A Random Forest (RF) and a Decision Forest (DF) improve the accuracies in comparison to a single Decision Tree (DT) with the *Tic-Tac-Toe Endgame* and the *Abalone* data sets. The accuracies achieved with a DT were already maximum with the other two data sets.
- RF seems to perform better than DF does, needing a smaller number of features (F) and trees (NT) to achieve best results.
- The *Abalone* data set has performed terribly with all the models, but we can still optimize the accuracy if we use the RF method.
- RF and DF can worsen the results with respect to a single DT, if parameters are not chosen correctly.
- Some features have definitely more relevance than others when using ensemble methods such as DF and RF.

# Appendices

## A FULL RESULTS

### A.1 IRIS DATA SET

```
*****  
Iris Data Set  
*****  
Accuracy DT: 100.0% Time elapsed: 0.04s  
-----  
Accuracy DF | NT=1 | F=1: 53.33% Time elapsed: 0.02s  
1. sepal.width  
2. sepal.length  
3. petal.length  
-----  
Accuracy DF | NT=1 | F=2: 90.0% Time elapsed: 0.02s  
1. petal.width  
2. sepal.width  
3. sepal.length  
-----  
Accuracy DF | NT=1 | F=3: 96.67% Time elapsed: 0.03s  
1. petal.length  
2. sepal.length  
3. sepal.width  
-----  
Accuracy DF | NT=1 | F=1: 53.33% Time elapsed: 0.01s  
1. sepal.width  
2. sepal.length  
3. petal.length  
-----  
Accuracy DF | NT=1 | F=4: 100.0% Time elapsed: 0.03s  
1. petal.length  
2. petal.width  
3. sepal.length  
-----  
Accuracy RF | NT=1 | F=1: 96.67% Time elapsed: 0.03s  
1. petal.length  
2. sepal.length  
3. petal.width
```

```
-----  
Accuracy RF | NT=1 | F=2: 100.0% Time elapsed: 0.02s
```

```
1. petal.width  
2. petal.length  
3. sepal.length
```

```
-----  
Accuracy RF | NT=1 | F=2: 93.33% Time elapsed: 0.02s
```

```
1. petal.width  
2. petal.length  
3. sepal.length
```

```
-----  
Accuracy RF | NT=1 | F=2: 96.67% Time elapsed: 0.02s
```

```
1. petal.width  
2. sepal.length  
3. petal.length
```

```
-----  
Accuracy RF | NT=1 | F=4: 96.67% Time elapsed: 0.03s
```

```
1. petal.width  
2. petal.length  
3. sepal.length
```

```
-----  
Accuracy DF | NT=10 | F=1: 76.67% Time elapsed: 0.22s
```

```
1. sepal.width  
2. petal.width  
3. sepal.length
```

```
-----  
Accuracy DF | NT=10 | F=2: 100.0% Time elapsed: 0.26s
```

```
1. petal.length  
2. petal.width  
3. sepal.length
```

```
-----  
Accuracy DF | NT=10 | F=3: 100.0% Time elapsed: 0.29s
```

```
1. petal.length  
2. petal.width  
3. sepal.length
```

```
-----  
Accuracy DF | NT=10 | F=1: 76.67% Time elapsed: 0.14s
```

```
1. sepal.length  
2. sepal.width
```

```
3. petal.length
-----
Accuracy DF | NT=10 | F=4: 100.0% Time elapsed: 0.38s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=10 | F=1: 96.67% Time elapsed: 0.11s
1. sepal.length
2. petal.width
3. petal.length
-----
Accuracy RF | NT=10 | F=2: 100.0% Time elapsed: 0.18s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=10 | F=2: 96.67% Time elapsed: 0.15s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=10 | F=2: 93.33% Time elapsed: 0.19s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy RF | NT=10 | F=4: 96.67% Time elapsed: 0.27s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy DF | NT=25 | F=1: 76.67% Time elapsed: 0.31s
1. sepal.width
2. sepal.length
3. petal.width
-----
Accuracy DF | NT=25 | F=2: 90.0% Time elapsed: 0.58s
1. petal.length
```

```
2. sepal.length
3. petal.width
-----
Accuracy DF | NT=25 | F=3: 96.67% Time elapsed: 0.7s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy DF | NT=25 | F=1: 80.0% Time elapsed: 0.33s
1. sepal.length
2. petal.length
3. sepal.width
-----
Accuracy DF | NT=25 | F=4: 100.0% Time elapsed: 0.88s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=25 | F=1: 96.67% Time elapsed: 0.27s
1. petal.width
2. sepal.length
3. petal.length
-----
Accuracy RF | NT=25 | F=2: 96.67% Time elapsed: 0.38s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=25 | F=2: 96.67% Time elapsed: 0.45s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy RF | NT=25 | F=2: 100.0% Time elapsed: 0.44s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy RF | NT=25 | F=4: 100.0% Time elapsed: 0.84s
```

```
1. petal.length
2. petal.width
3. sepal.width
-----
Accuracy DF | NT=50 | F=1: 96.67% Time elapsed: 0.85s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy DF | NT=50 | F=2: 96.67% Time elapsed: 1.46s
1. petal.length
2. sepal.length
3. petal.width
-----
Accuracy DF | NT=50 | F=3: 100.0% Time elapsed: 1.78s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy DF | NT=50 | F=1: 80.0% Time elapsed: 0.72s
1. sepal.width
2. sepal.length
3. petal.length
-----
Accuracy DF | NT=50 | F=4: 100.0% Time elapsed: 1.97s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=50 | F=1: 96.67% Time elapsed: 0.56s
1. petal.width
2. sepal.width
3. petal.length
-----
Accuracy RF | NT=50 | F=2: 100.0% Time elapsed: 0.81s
1. petal.width
2. petal.length
3. sepal.length
```

```
Accuracy RF | NT=50 | F=2: 100.0% Time elapsed: 0.74s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy RF | NT=50 | F=2: 96.67% Time elapsed: 0.75s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=50 | F=4: 100.0% Time elapsed: 1.27s
1. petal.length
2. petal.width
3. sepal.width
-----
Accuracy DF | NT=75 | F=1: 76.67% Time elapsed: 0.93s
1. sepal.length
2. sepal.width
3. petal.width
-----
Accuracy DF | NT=75 | F=2: 96.67% Time elapsed: 1.6s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy DF | NT=75 | F=3: 96.67% Time elapsed: 2.04s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy DF | NT=75 | F=1: 96.67% Time elapsed: 0.85s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy DF | NT=75 | F=4: 100.0% Time elapsed: 2.53s
1. petal.width
2. petal.length
3. sepal.length
```

```
-----  
Accuracy RF | NT=75 | F=1: 96.67% Time elapsed: 0.89s  
1. petal.length  
2. sepal.width  
3. petal.width  
-----  
Accuracy RF | NT=75 | F=2: 100.0% Time elapsed: 1.24s  
1. petal.width  
2. petal.length  
3. sepal.length  
-----  
Accuracy RF | NT=75 | F=2: 100.0% Time elapsed: 1.11s  
1. petal.width  
2. petal.length  
3. sepal.length  
-----  
Accuracy RF | NT=75 | F=2: 100.0% Time elapsed: 1.13s  
1. petal.length  
2. petal.width  
3. sepal.length  
-----  
Accuracy RF | NT=75 | F=4: 100.0% Time elapsed: 2.0s  
1. petal.length  
2. petal.width  
3. sepal.length  
-----  
Accuracy DF | NT=100 | F=1: 80.0% Time elapsed: 1.14s  
1. sepal.length  
2. petal.width  
3. petal.length  
-----  
Accuracy DF | NT=100 | F=2: 90.0% Time elapsed: 2.39s  
1. sepal.length  
2. petal.width  
3. petal.length  
-----  
Accuracy DF | NT=100 | F=3: 96.67% Time elapsed: 3.21s  
1. petal.width  
2. petal.length
```

```
3. sepal.length
-----
Accuracy DF | NT=100 | F=1: 80.0% Time elapsed: 1.41s
1. petal.length
2. sepal.length
3. sepal.width
-----
Accuracy DF | NT=100 | F=4: 100.0% Time elapsed: 3.67s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=100 | F=1: 100.0% Time elapsed: 1.07s
1. petal.width
2. petal.length
3. sepal.width
-----
Accuracy RF | NT=100 | F=2: 100.0% Time elapsed: 1.53s
1. petal.length
2. petal.width
3. sepal.length
-----
Accuracy RF | NT=100 | F=2: 100.0% Time elapsed: 1.51s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=100 | F=2: 100.0% Time elapsed: 1.51s
1. petal.width
2. petal.length
3. sepal.length
-----
Accuracy RF | NT=100 | F=4: 100.0% Time elapsed: 2.6s
1. petal.width
2. petal.length
3. sepal.width
```

## A.2 TIC-TAC-TOE ENGAME DATA SET

```
*****  
Tic Tac Toe Endgame Data Set  
*****  
Accuracy DT: 95.31% Time elapsed: 0.13s  
-----  
Accuracy DF | NT=1 | F=2: 69.79% Time elapsed: 0.01s  
1. V7  
2. V9  
3. V1  
-----  
Accuracy DF | NT=1 | F=4: 68.75% Time elapsed: 0.08s  
1. V6  
2. V4  
3. V1  
-----  
Accuracy DF | NT=1 | F=6: 76.04% Time elapsed: 0.16s  
1. V1  
2. V2  
3. V4  
-----  
Accuracy DF | NT=1 | F=8: 90.62% Time elapsed: 0.15s  
1. V9  
2. V1  
3. V4  
-----  
Accuracy DF | NT=1 | F=9: 94.27% Time elapsed: 0.13s  
1. V9  
2. V1  
3. V3  
-----  
Accuracy RF | NT=1 | F=1: 62.5% Time elapsed: 0.03s  
1. V8  
2. V2  
3. V6  
-----  
Accuracy RF | NT=1 | F=2: 72.92% Time elapsed: 0.08s  
1. V5  
2. V6
```

3. V2

---

Accuracy RF | NT=1 | F=3: 86.98% Time elapsed: 0.07s

1. V1
  2. V9
  3. V5
- 

Accuracy RF | NT=1 | F=3: 78.12% Time elapsed: 0.08s

1. V9
  2. V2
  3. V4
- 

Accuracy RF | NT=1 | F=9: 94.79% Time elapsed: 0.11s

1. V1
  2. V3
  3. V7
- 

Accuracy DF | NT=10 | F=2: 72.4% Time elapsed: 0.11s

1. V4
  2. V6
  3. V5
- 

Accuracy DF | NT=10 | F=4: 80.73% Time elapsed: 0.69s

1. V9
  2. V1
  3. V4
- 

Accuracy DF | NT=10 | F=6: 85.42% Time elapsed: 1.84s

1. V6
  2. V3
  3. V1
- 

Accuracy DF | NT=10 | F=8: 95.31% Time elapsed: 1.75s

1. V9
  2. V7
  3. V8
- 

Accuracy DF | NT=10 | F=9: 95.83% Time elapsed: 1.43s

1. V9

2. V1

3. V3

-----  
Accuracy RF | NT=10 | F=1: 79.17% Time elapsed: 0.34s

1. V7

2. V4

3. V1

-----  
Accuracy RF | NT=10 | F=2: 90.62% Time elapsed: 0.76s

1. V8

2. V9

3. V3

-----  
Accuracy RF | NT=10 | F=3: 94.27% Time elapsed: 0.83s

1. V5

2. V9

3. V7

-----  
Accuracy RF | NT=10 | F=3: 93.23% Time elapsed: 1.12s

1. V3

2. V9

3. V7

-----  
Accuracy RF | NT=10 | F=9: 99.48% Time elapsed: 1.37s

1. V1

2. V3

3. V9

-----  
Accuracy DF | NT=25 | F=2: 70.31% Time elapsed: 0.31s

1. V2

2. V7

3. V8

-----  
Accuracy DF | NT=25 | F=4: 81.25% Time elapsed: 1.98s

1. V8

2. V2

3. V4

-----  
Accuracy DF | NT=25 | F=6: 90.1% Time elapsed: 5.55s

1. V8

2. V4

3. V1

-----

Accuracy DF | NT=25 | F=8: 95.83% Time elapsed: 5.06s

1. V7

2. V1

3. V9

-----

Accuracy DF | NT=25 | F=9: 95.31% Time elapsed: 4.78s

1. V9

2. V1

3. V7

-----

Accuracy RF | NT=25 | F=1: 80.73% Time elapsed: 0.72s

1. V6

2. V9

3. V5

-----

Accuracy RF | NT=25 | F=2: 91.15% Time elapsed: 1.94s

1. V7

2. V5

3. V3

-----

Accuracy RF | NT=25 | F=3: 98.44% Time elapsed: 2.23s

1. V5

2. V1

3. V9

-----

Accuracy RF | NT=25 | F=3: 98.44% Time elapsed: 2.13s

1. V9

2. V3

3. V5

-----

Accuracy RF | NT=25 | F=9: 98.96% Time elapsed: 2.98s

1. V7

2. V1

3. V9

-----

Accuracy DF | NT=50 | F=2: 70.31% Time elapsed: 0.55s

1. V2
  2. V3
  3. V4
- 

Accuracy DF | NT=50 | F=4: 81.77% Time elapsed: 3.67s

1. V7
  2. V4
  3. V2
- 

Accuracy DF | NT=50 | F=6: 92.19% Time elapsed: 10.39s

1. V4
  2. V8
  3. V6
- 

Accuracy DF | NT=50 | F=8: 95.83% Time elapsed: 8.89s

1. V2
  2. V8
  3. V1
- 

Accuracy DF | NT=50 | F=9: 95.31% Time elapsed: 6.61s

1. V9
  2. V1
  3. V7
- 

Accuracy RF | NT=50 | F=1: 86.46% Time elapsed: 1.29s

1. V2
  2. V9
  3. V1
- 

Accuracy RF | NT=50 | F=2: 96.88% Time elapsed: 3.35s

1. V7
  2. V9
  3. V5
- 

Accuracy RF | NT=50 | F=3: 100.0% Time elapsed: 3.93s

1. V7
2. V9
3. V5

-----  
Accuracy RF | NT=50 | F=3: 98.96% Time elapsed: 4.13s

1. V7
  2. V5
  3. V9
- 

Accuracy RF | NT=50 | F=9: 98.44% Time elapsed: 5.61s

1. V7
  2. V9
  3. V1
- 

Accuracy DF | NT=75 | F=2: 70.31% Time elapsed: 0.81s

1. V2
  2. V7
  3. V9
- 

Accuracy DF | NT=75 | F=4: 83.85% Time elapsed: 5.0s

1. V7
  2. V6
  3. V4
- 

Accuracy DF | NT=75 | F=6: 93.23% Time elapsed: 14.1s

1. V8
  2. V2
  3. V6
- 

Accuracy DF | NT=75 | F=8: 96.88% Time elapsed: 15.96s

1. V2
  2. V9
  3. V8
- 

Accuracy DF | NT=75 | F=9: 95.83% Time elapsed: 10.94s

1. V9
  2. V1
  3. V3
- 

Accuracy RF | NT=75 | F=1: 85.94% Time elapsed: 2.15s

1. V7
2. V4

3. V1

-----  
Accuracy RF | NT=75 | F=2: 98.96% Time elapsed: 5.57s

1. V5

2. V9

3. V3

-----  
Accuracy RF | NT=75 | F=3: 100.0% Time elapsed: 6.5s

1. V5

2. V1

3. V9

-----  
Accuracy RF | NT=75 | F=3: 99.48% Time elapsed: 6.44s

1. V5

2. V9

3. V7

-----  
Accuracy RF | NT=75 | F=9: 98.96% Time elapsed: 9.87s

1. V9

2. V3

3. V7

-----  
Accuracy DF | NT=100 | F=2: 70.31% Time elapsed: 1.26s

1. V7

2. V1

3. V6

-----  
Accuracy DF | NT=100 | F=4: 77.08% Time elapsed: 9.06s

1. V6

2. V7

3. V2

-----  
Accuracy DF | NT=100 | F=6: 94.79% Time elapsed: 30.43s

1. V4

2. V6

3. V8

-----  
Accuracy DF | NT=100 | F=8: 96.35% Time elapsed: 22.23s

1. V1

2. V2

3. V8

---

Accuracy DF | NT=100 | F=9: 95.83% Time elapsed: 21.84s

1. V9

2. V1

3. V3

---

Accuracy RF | NT=100 | F=1: 78.12% Time elapsed: 6.24s

1. V9

2. V2

3. V3

---

Accuracy RF | NT=100 | F=2: 96.88% Time elapsed: 7.35s

1. V5

2. V9

3. V3

---

Accuracy RF | NT=100 | F=3: 99.48% Time elapsed: 9.13s

1. V5

2. V1

3. V3

---

Accuracy RF | NT=100 | F=3: 99.48% Time elapsed: 8.55s

1. V5

2. V7

3. V1

---

Accuracy RF | NT=100 | F=9: 98.96% Time elapsed: 12.75s

1. V7

2. V1

3. V9

---

### A.3 ABALONE DATA SET

```
*****  
Abalone Data Set  
*****  
Accuracy DT: 20.69% Time elapsed: 39.63s  
-----  
Accuracy DF | NT=1 | F=2: 16.99% Time elapsed: 15.43s  
1. Whole_weight  
2. Length  
3. Sex  
-----  
Accuracy DF | NT=1 | F=4: 19.74% Time elapsed: 20.57s  
1. Shell_weight  
2. Diameter  
3. Whole_weight  
-----  
Accuracy DF | NT=1 | F=6: 19.02% Time elapsed: 24.56s  
1. Shell_weight  
2. Diameter  
3. Shucked_weight  
-----  
Accuracy RF | NT=1 | F=1: 20.33% Time elapsed: 3.59s  
1. Viscera_weight  
2. Diameter  
3. Shell_weight  
-----  
Accuracy RF | NT=1 | F=2: 18.54% Time elapsed: 5.75s  
1. Height  
2. Viscera_weight  
3. Diameter  
-----  
Accuracy RF | NT=1 | F=3: 18.78% Time elapsed: 10.53s  
1. Whole_weight  
2. Shell_weight  
3. Viscera_weight  
-----  
Accuracy RF | NT=1 | F=2: 19.62% Time elapsed: 6.98s  
1. Diameter  
2. Viscera_weight
```

### 3. Length

---

Accuracy DF | NT=10 | F=2: 21.65% Time elapsed: 82.84s

1. Shell\_weight
  2. Viscera\_weight
  3. Whole\_weight
- 

Accuracy DF | NT=10 | F=4: 21.41% Time elapsed: 169.82s

1. Shell\_weight
  2. Height
  3. Length
- 

Accuracy DF | NT=10 | F=6: 21.77% Time elapsed: 232.89s

1. Shell\_weight
  2. Diameter
  3. Shucked\_weight
- 

Accuracy RF | NT=10 | F=1: 25.84% Time elapsed: 31.43s

1. Viscera\_weight
  2. Whole\_weight
  3. Diameter
- 

Accuracy RF | NT=10 | F=2: 23.8% Time elapsed: 60.02s

1. Diameter
  2. Shell\_weight
  3. Viscera\_weight
- 

Accuracy RF | NT=10 | F=3: 21.65% Time elapsed: 82.67s

1. Height
  2. Whole\_weight
  3. Diameter
- 

Accuracy RF | NT=10 | F=2: 24.64% Time elapsed: 60.97s

1. Whole\_weight
  2. Height
  3. Viscera\_weight
- 

Accuracy DF | NT=25 | F=2: 23.21% Time elapsed: 178.31s

1. Diameter

```
2. Height
3. Shell_weight
-----
Accuracy DF | NT=25 | F=4: 24.52% Time elapsed: 379.22s
1. Shell_weight
2. Viscera_weight
3. Diameter
-----
Accuracy DF | NT=25 | F=6: 22.85% Time elapsed: 791.15s
1. Shell_weight
2. Viscera_weight
3. Diameter
-----
Accuracy RF | NT=25 | F=1: 25.0% Time elapsed: 192.61s
1. Sex
2. Length
3. Shell_weight
-----
Accuracy RF | NT=25 | F=2: 26.2% Time elapsed: 269.33s
1. Length
2. Whole_weight
3. Diameter
-----
Accuracy RF | NT=25 | F=3: 24.16% Time elapsed: 259.73s
1. Whole_weight
2. Diameter
3. Shucked_weight
-----
Accuracy RF | NT=25 | F=2: 25.48% Time elapsed: 150.52s
1. Shucked_weight
2. Shell_weight
3. Height
-----
Accuracy DF | NT=50 | F=2: 22.73% Time elapsed: 428.52s
1. Length
2. Shell_weight
3. Whole_weight
-----
Accuracy DF | NT=50 | F=4: 23.8% Time elapsed: 807.57s
```

```
1. Shell_weight
2. Height
3. Viscera_weight
-----
Accuracy DF | NT=50 | F=6: 22.49% Time elapsed: 1108.32s
1. Shell_weight
2. Diameter
3. Viscera_weight
-----
Accuracy RF | NT=50 | F=1: 27.87% Time elapsed: 147.38s
1. Whole_weight
2. Shell_weight
3. Diameter
-----
Accuracy RF | NT=50 | F=2: 25.0% Time elapsed: 298.98s
1. Height
2. Shell_weight
3. Diameter
-----
Accuracy RF | NT=50 | F=3: 25.24% Time elapsed: 418.03s
1. Viscera_weight
2. Whole_weight
3. Height
-----
Accuracy RF | NT=50 | F=2: 25.12% Time elapsed: 295.99s
1. Viscera_weight
2. Diameter
3. Shell_weight
-----
Accuracy DF | NT=75 | F=2: 24.76% Time elapsed: 540.83s
1. Viscera_weight
2. Shucked_weight
3. Shell_weight
-----
Accuracy DF | NT=75 | F=4: 24.76% Time elapsed: 1200.77s
1. Shell_weight
2. Diameter
3. Height
```

Accuracy DF | NT=75 | F=6: 23.21% Time elapsed: 1785.23s

1. Shell\_weight
  2. Diameter
  3. Shucked\_weight
- 

Accuracy RF | NT=75 | F=1: 27.63% Time elapsed: 222.54s

1. Height
  2. Diameter
  3. Whole\_weight
- 

Accuracy RF | NT=75 | F=2: 24.16% Time elapsed: 442.73s

1. Length
  2. Diameter
  3. Height
- 

Accuracy RF | NT=75 | F=3: 25.72% Time elapsed: 632.9s

1. Diameter
  2. Shell\_weight
  3. Height
- 

Accuracy RF | NT=75 | F=2: 25.6% Time elapsed: 443.8s

1. Whole\_weight
  2. Viscera\_weight
  3. Height
- 

Accuracy DF | NT=100 | F=2: 23.8% Time elapsed: 745.71s

1. Shell\_weight
  2. Diameter
  3. Whole\_weight
- 

Accuracy DF | NT=100 | F=4: 24.16% Time elapsed: 1517.49s

1. Shell\_weight
  2. Height
  3. Diameter
- 

Accuracy DF | NT=100 | F=6: 22.13% Time elapsed: 2350.91s

1. Shell\_weight
2. Height
3. Diameter

-----  
Accuracy RF | NT=100 | F=1: 26.91% Time elapsed: 288.94s

1. Whole\_weight
  2. Shucked\_weight
  3. Viscera\_weight
- 

Accuracy RF | NT=100 | F=2: 25.96% Time elapsed: 585.23s

1. Height
  2. Shell\_weight
  3. Diameter
- 

Accuracy RF | NT=100 | F=3: 25.12% Time elapsed: 838.02s

1. Diameter
  2. Shell\_weight
  3. Viscera\_weight
- 

Accuracy RF | NT=100 | F=2: 26.91% Time elapsed: 571.3s

1. Shell\_weight
  2. Whole\_weight
  3. Length
-

#### A.4 MUSHROOM DATA SET

```
*****  
Mushroom Data Set  
*****  
Accuracy DT: 100.0% Time elapsed: 0.5s  
-----  
Accuracy DF | NT=1 | F=5: 84.68% Time elapsed: 0.13s  
1. habitat  
2. stalk-root  
3. ring-number  
-----  
Accuracy DF | NT=1 | F=11: 98.41% Time elapsed: 0.24s  
1. spore-print-color  
2. gill-size  
3. stalk-shape  
-----  
Accuracy DF | NT=1 | F=16: 99.38% Time elapsed: 0.46s  
1. spore-print-color  
2. habitat  
3. population  
-----  
Accuracy DF | NT=1 | F=20: 100.0% Time elapsed: 0.55s  
1. spore-print-color  
2. gill-size  
3. odor  
-----  
Accuracy DF | NT=1 | F=22: 100.0% Time elapsed: 0.55s  
1. population  
2. spore-print-color  
3. gill-size  
-----  
Accuracy RF | NT=1 | F=1: 93.71% Time elapsed: 0.07s  
1. stalk-color-below-ring  
2. ring-type  
3. gill-spacing  
-----  
Accuracy RF | NT=1 | F=2: 94.95% Time elapsed: 0.11s  
1. odor  
2. population
```

```
3. habitat
-----
Accuracy RF | NT=1 | F=4: 100.0% Time elapsed: 0.13s
1. ring-type
2. spore-print-color
3. habitat
-----
Accuracy RF | NT=1 | F=4: 99.73% Time elapsed: 0.14s
1. bruises
2. cap-color
3. stalk-shape
-----
Accuracy RF | NT=1 | F=22: 100.0% Time elapsed: 0.6s
1. population
2. spore-print-color
3. gill-size
-----
Accuracy DF | NT=10 | F=5: 99.65% Time elapsed: 2.58s
1. population
2. odor
3. spore-print-color
-----
Accuracy DF | NT=10 | F=11: 100.0% Time elapsed: 3.99s
1. cap-color
2. odor
3. habitat
-----
Accuracy DF | NT=10 | F=16: 100.0% Time elapsed: 4.76s
1. odor
2. spore-print-color
3. population
-----
Accuracy DF | NT=10 | F=20: 100.0% Time elapsed: 5.8s
1. odor
2. habitat
3. ring-number
-----
Accuracy DF | NT=10 | F=22: 100.0% Time elapsed: 5.47s
1. odor
```

```
2. gill-size
3. ring-number
-----
Accuracy RF | NT=10 | F=1: 90.79% Time elapsed: 0.5s
1. stalk-root
2. gill-attachment
3. odor
-----
Accuracy RF | NT=10 | F=2: 100.0% Time elapsed: 1.12s
1. stalk-color-below-ring
2. spore-print-color
3. stalk-root
-----
Accuracy RF | NT=10 | F=4: 100.0% Time elapsed: 1.59s
1. spore-print-color
2. odor
3. stalk-color-below-ring
-----
Accuracy RF | NT=10 | F=4: 100.0% Time elapsed: 1.37s
1. odor
2. spore-print-color
3. population
-----
Accuracy RF | NT=10 | F=22: 100.0% Time elapsed: 5.53s
1. odor
2. gill-size
3. population
-----
Accuracy DF | NT=25 | F=5: 99.82% Time elapsed: 5.04s
1. odor
2. cap-color
3. cap-shape
-----
Accuracy DF | NT=25 | F=11: 100.0% Time elapsed: 9.84s
1. odor
2. population
3. spore-print-color
-----
Accuracy DF | NT=25 | F=16: 100.0% Time elapsed: 12.19s
```

```
1. odor  
2. habitat  
3. ring-number
```

---

```
Accuracy DF | NT=25 | F=20: 100.0% Time elapsed: 13.41s
```

```
1. odor  
2. gill-size  
3. population
```

---

```
Accuracy DF | NT=25 | F=22: 100.0% Time elapsed: 15.47s
```

```
1. odor  
2. gill-size  
3. population
```

---

```
Accuracy RF | NT=25 | F=1: 93.45% Time elapsed: 1.14s
```

```
1. cap-shape  
2. cap-surface  
3. cap-color
```

---

```
Accuracy RF | NT=25 | F=2: 100.0% Time elapsed: 2.3s
```

```
1. cap-color  
2. habitat  
3. odor
```

---

```
Accuracy RF | NT=25 | F=4: 100.0% Time elapsed: 4.31s
```

```
1. spore-print-color  
2. odor  
3. stalk-root
```

---

```
Accuracy RF | NT=25 | F=4: 100.0% Time elapsed: 4.03s
```

```
1. odor  
2. spore-print-color  
3. habitat
```

---

```
Accuracy RF | NT=25 | F=22: 100.0% Time elapsed: 13.17s
```

```
1. odor  
2. population  
3. gill-size
```

---

```
Accuracy DF | NT=50 | F=5: 100.0% Time elapsed: 11.29s
1. population
2. odor
3. stalk-color-below-ring
-----
Accuracy DF | NT=50 | F=11: 100.0% Time elapsed: 20.05s
1. odor
2. spore-print-color
3. population
-----
Accuracy DF | NT=50 | F=16: 100.0% Time elapsed: 24.64s
1. odor
2. spore-print-color
3. population
-----
Accuracy DF | NT=50 | F=20: 100.0% Time elapsed: 27.8s
1. odor
2. population
3. gill-size
-----
Accuracy DF | NT=50 | F=22: 100.0% Time elapsed: 30.4s
1. odor
2. gill-size
3. ring-number
-----
Accuracy RF | NT=50 | F=1: 98.49% Time elapsed: 2.66s
1. cap-shape
2. cap-surface
3. cap-color
-----
Accuracy RF | NT=50 | F=2: 100.0% Time elapsed: 6.58s
1. odor
2. gill-color
3. spore-print-color
-----
Accuracy RF | NT=50 | F=4: 100.0% Time elapsed: 8.22s
1. odor
2. spore-print-color
3. population
```

```
-----  
Accuracy RF | NT=50 | F=4: 100.0% Time elapsed: 8.32s  
1. odor  
2. spore-print-color  
3. habitat  
-----  
Accuracy RF | NT=50 | F=22: 100.0% Time elapsed: 29.32s  
1. odor  
2. gill-size  
3. population  
-----  
Accuracy DF | NT=75 | F=5: 100.0% Time elapsed: 16.4s  
1. odor  
2. spore-print-color  
3. cap-shape  
-----  
Accuracy DF | NT=75 | F=11: 100.0% Time elapsed: 31.95s  
1. odor  
2. spore-print-color  
3. population  
-----  
Accuracy DF | NT=75 | F=16: 100.0% Time elapsed: 39.75s  
1. odor  
2. population  
3. spore-print-color  
-----  
Accuracy DF | NT=75 | F=20: 100.0% Time elapsed: 40.97s  
1. odor  
2. gill-size  
3. population  
-----  
Accuracy DF | NT=75 | F=22: 100.0% Time elapsed: 44.76s  
1. odor  
2. gill-size  
3. ring-number  
-----  
Accuracy RF | NT=75 | F=1: 94.42% Time elapsed: 4.36s  
1. cap-shape  
2. cap-surface
```

3. cap-color

-----  
Accuracy RF | NT=75 | F=2: 100.0% Time elapsed: 7.78s

1. odor

2. spore-print-color

3. cap-color

-----  
Accuracy RF | NT=75 | F=4: 100.0% Time elapsed: 13.16s

1. odor

2. spore-print-color

3. stalk-shape

-----  
Accuracy RF | NT=75 | F=4: 100.0% Time elapsed: 12.9s

1. odor

2. spore-print-color

3. population

-----  
Accuracy RF | NT=75 | F=22: 100.0% Time elapsed: 42.09s

1. odor

2. gill-size

3. population

-----  
Accuracy DF | NT=100 | F=5: 100.0% Time elapsed: 20.83s

1. odor

2. cap-color

3. spore-print-color

-----  
Accuracy DF | NT=100 | F=11: 100.0% Time elapsed: 41.43s

1. odor

2. spore-print-color

3. habitat

-----  
Accuracy DF | NT=100 | F=16: 100.0% Time elapsed: 56.96s

1. odor

2. spore-print-color

3. population

-----  
Accuracy DF | NT=100 | F=20: 100.0% Time elapsed: 54.98s

1. odor

```
2. gill-size
3. spore-print-color
-----
Accuracy DF | NT=100 | F=22: 100.0% Time elapsed: 55.76s
1. odor
2. gill-size
3. ring-number
-----
Accuracy RF | NT=100 | F=1: 95.75% Time elapsed: 5.02s
1. cap-shape
2. cap-surface
3. cap-color
-----
Accuracy RF | NT=100 | F=2: 100.0% Time elapsed: 9.92s
1. odor
2. spore-print-color
3. population
-----
Accuracy RF | NT=100 | F=4: 100.0% Time elapsed: 17.13s
1. odor
2. spore-print-color
3. habitat
-----
Accuracy RF | NT=100 | F=4: 100.0% Time elapsed: 17.43s
1. odor
2. spore-print-color
3. habitat
-----
Accuracy RF | NT=100 | F=22: 100.0% Time elapsed: 60.6s
1. odor
2. gill-size
3. population
```

## REFERENCES

- [1] L. Breiman, *Classification and Regression Trees*. 1984.
- [2] T. K. Ho, *The Random Subspace Method for Constructing Decision Forests*. 1998.
- [3] L. Breiman, *Random Forests*. 2001.
- [4] *UCI Machine Learning Repository: Iris Data Set*.
- [5] *UCI Machine Learning Repository: Tic-tac-toe Endgame Data Set*.
- [6] *UCI Machine Learning Repository: Abalone Data Set*.
- [7] *UCI Machine Learning Repository: Mushroom Data Set*.