AdaBoost and Random Tree Forest "A Comparative Approach"

Jonas Perolini
School of Engineering (STI)
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: jonas.perolini@epfl.ch

Raphael Uebersax
School of Engineering (STI)
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: raphael.uebersax@epfl.ch

***Abstract*—This reports contrasts two classification algorithms: AdaBoost and Random Tree Forest (RTF). It investigates the performances of each algorithm on a real-world dataset depending on the choice of hyperparameters and other data specific applications. More precisely, the sensitivity to noisy, mislabeled, sparse, missing or incomplete training samples as well as imbalanced training datasets are discussed for both classifiers. Performances are also evaluated contrasted after implementing pre-processing methods such as upsampling the minority class using GMMs to balance the dataset or replacing missing values using KNN-regressor. Additionally, time complexity at training as well as a testing are looked further into as a function of the hyperparameters. Besides, to facilitate the analysis, the algorithms are also tested on a 2D toy dataset allowing for visualisation of the final classification of each classifier. Experiments showed that both algorithms obtained outstanding results on the binary classification task when trained on balanced or complete datasets. However, RTF tends to outperform AdaBoost on challenging ones with small, noisy and mislabeled training sets by its ability to generalize better and overfit less the training data. The performance gap is partly explained by the randomness introduced when growing a decision tree in RTF.**

I. Introduction

In our digitized world, data classification is omnipresent. The use of machine learning classification algorithms is crucial in a wide range of applications, including image recognition, speech recognition, DNA sequences classifications and many more. This report aims at analysing and contrasting two classification algorithms: AdaBoost and Random Forest.

It is structured as follows. First, both algorithms are briefly introduced in section I. Then, the methodology followed for different testing scenarios on both a real-world and a toy dataset is introduced in section II. The results are then presented in section III before being analysed and discussed in depth in section IV. Finally, section V concludes the report.

a) AdaBoost: AdaBoost is an iterative process that computes a weighted sum over several simple estimators for the final classification. Initially, each training sample is given a weight which is usually assigned uniformly. Then, a generation of weak learners is introduced. A weak learner is a simple model that is easy to implement and fast to train. Besides, the misclassification error of the weak learners is required to be less than 50% (i.e. slightly better than random guessing for binary classification). Then, the performance of each weak learner is evaluated and the one minimizing a predefined error is chosen and saved as an estimator. This process is repeated iteratively each time generating a new generation of weak learners. To avoid redundancy in the next generation of weak learners, more importance is given to the samples that were misclassified by the previous estimator by increasing their weights. This allows the next generation to focus on the weaknesses of its predecessor and as the process goes on, classifying correctly the samples that were initially harder to classify. Finally, the weighted sum over each estimator is used as the final prediction giving more relative weight to the best performing ones.

The two main hyperparameters of AdaBoost are the weak learners' model and the number of estimators used for the final classification.

AdaBoost is arguably slow at training as it fits a generation of weak learners at each iteration. The computational cost of the algorithm is O($M \cdot w$) where M is the number of estimators and w is the time needed to fit and evaluate the generation weak learners. At testing, for n points of p dimension, the computational cost of AdaBoost is O($p \cdot n$) which is significantly faster than at training.

b) Random Tree Forest: Random Tree Forest (RTF) is a supervised learning method for classification that uses multiple decision Trees at training and combines them such that they operate as an ensemble. This enables the algorithm to get an accurate and stable class prediction. The randomness and low correlation between trees is key in this method. While growing the trees, RTF searches for the best feature among a random subset of features such that the trees result in significantly different structures. This is mainly due to the fact that decision trees are very sensitive to the data they are trained on. This process is known as bagging and results in a wide diversity of trees. The introduced randomness significantly reduces the tendency of the algorithm to overfit the training data.

In this report, the influence of three relevant RTF hyperparameters will be discussed: the number of estimators, the maximum number of features to consider when looking for the best split and the maximum depth of a tree. However, it is important to keep in mind that there exists many other

hyperparameters such as the criterion use to assess the quality of the split or the minimum number of samples required to split a node which will not be further analysed.

According to [1], the time complexity of a single decision tree of depth d for n points of p dimensions (features) is $O(d{\cdot}p{\cdot}n{\cdot}log(n))$. It follows that for the worst case scenario where the depth of each tree is assumed to be $log(n)$, the computational cost of RTF at training is $O(T \cdot p \cdot n \cdot log(n)^2)$. Here, T is the number of trees. It is worth noting that since Random Forest relies on building several independent trees, it is simple to obtain a parallel and faster implementation of the algorithm. At testing, the computational cost of RTF is simply $O(p \cdot n)$.

II. Methods

In this section, the different implementations of AdaBoost and RTF are presented. Then the metrics used to quantitatively compare said algorithms are introduced. Finally, different scenarios in which the algorithms will be tested are detailed.

A. Algorithm setup

In this report, the Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) algorithm from the open-source python library *Scikit-learn* is used. The SAMME algorithm was proposed in [1] and extends the classic AdaBoost as it allows for multiclass classification. However, for binary classification, the SAMME algorithm is equivalent to the classic AdaBoost. To facilitate the comparison with Random Tree Forest, it was decided to use decision trees with a single root and two leaves called stumps as weak learners. For n points each of dimension p, the SAMME algorithm using a decision stump as weak learner (which is a decision tree of depth $d = 1$) has a complexity of O($M \cdot p \cdot n \cdot log(n)$) where M is the number of estimators used for the final classification. Finally, *Scikit-learn*'s implementation of Random Tree Forest will be used.

Henceforth, except where otherwise indicated, *Scikit-learn*'s default hyperparameters are used for both algorithms.

B. Performance assessment

In order to assess the performance of the two algorithms in different scenarios, the following metrics are combined.

1) Accuracy: Accuracy measures the proportion of correct classification on the test set. This metric is very useful to assess the overall performance of the machine learning algorithm but can also yield misleading results if the dataset is imbalanced. It can be computed using: True Positive (TP), True Negative (TN), False Positive (FN) and False Negative (FN) samples.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

2) Confusion matrix: Confusion matrices provide information on which classes are confused with which. It is a powerful metric when dealing with imbalanced datasets as it allows for a more in-depth analysis than accuracy. For binary classification, the confusion matrix is two by two:

$$M = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \tag{2}$$

3) F-measure: The F-measure is widely used in classifcation tasks as it finds a tradeoff between classifying correctly all datapoints of the same class and making sure that each class contains points of only one class. The F-measure variance at testing across different folds of crossvalidation also evaluates the sensitivity of the classifier to the training set and to its hyperparameters. It can be computed using the precision: $P = \frac{TP}{TP+FP}$ and the recall: $R = \frac{TP}{TP+FN}$

$$F = \frac{2PR}{P + R} \tag{3}$$

C. Scenarios

In order to compare and contrast AdaBoost and Random Tree Forest, the algorithms' performance and robustness in binary classification will be tested in different key scenarios. First, AdaBoost and RTF will be quantitatively compared and contrasted using the benchmark dataset: "Shill Bidding" found in the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Shill+Bidding+Dataset. The dataset is composed of 6321 data points each of nine dimensions. The datapoints are distributed into two highly imbalanced classes, namely Shill bidder (the minority class) and normal bidder.

The quantitative metrics are computed through 10-fold repetitions as introduced in the EPFL Applied Machine Learning Course [2]. At each fold repetition, the dataset is randomly split into training and testing sets with a ratio of 0.3 (i.e. 70% for training and 30% for testing). Then, in order to visualize the results, the algorithms will also be compared using a 3D version of the Shill Bidding dataset. The dimension reduction of the data set is achieved by pre-processing it with Principal Component Analysis (PCA) and only keeping three dimensions. It is important to note that this manipulation is not intended to boost the performance of the classification, as the three dimensions only explain roughly 70 % of the variance of the data, but rather to extract relevant features and visualize the results. For each fold repetition, the PCA projection matrix is computed on the training set and is then used to transform both the training and testing sets. Finally, AdaBoost and RTF are also compared using a 2D toy dataset: (*Scikit-learn*'s moon dataset) with 700 points each of two dimensions distributed into two classes. This allows to visualize the predicted probability of a meshgrid of query points which will henceforth be referred as probabilistic decision boundaries. Furthermore, for each scenario, the toy dataset is pre-processed accordingly and the algorithm's performances are compared with the benchmarking results obtained with the original toy dataset. Next, the different scenarios are described.

a) Imbalanced data set: A very frequent issue with classification are imbalanced datasets. Imbalanced classification poses a challenge for machine learning algorithms as they often poorly predict the minority class. The aim of this scenario is to obtain quantitative results on the sensitivity of both algorithm to imbalanced dataset. The idea is to compare performance of both algorithms when

using the traditional dataset and a balanced version of it, where the minority class is upsampled. The upsampling is done using a gaussian mixture model (GMM) on the training data of the minority class to estimate its distribution. The optimal GMM parameters (i.e. type of covariance matrix and number of guassians) are obtained by plotting the minimum of the AIC and BIC curves over 10 fold repetitions with respect to the number of gaussians and using the elbow criterion. Once the GMM is trained on the minority class, new samples can be created matching the overall distribution to artificially create a balanced version of the training dataset. In order to illustrate the effect of the balancing in a plot, it has been decided to use the 3D-version of the Shill Bidding dataset using PCA as dimensionality reduction method. Since balancing the dataset is considered as a pre-processing method, the following experiments using the Shill Bidding dataset will balance the training set with the previously described method before training each classifier.

b) Missing data: In real-world applications, it is common to have incomplete datasets. It is thus crucial to deploy robust machine learning algorithms able to effectively deal with this issue. The aim of this scenario is to compare and contrast the sensitivity to missing data of AdaBoost and RTF. First, on the toy dataset, roughly 10% of the training points are removed before assessing the algorithm's performance on the testing set. It was decided to run two experiments, each time removing spatially close points at two different locations. The aim is to visualize the impact on the probabilistic decision boundaries at testing in comparison with the toy dataset benchmark. Then, the algorithms are tested on the Shill Bidding dataset. Initially, this dataset does not have any missing data. For this scenario, the data is manually removed by randomly selecting attributes of the training data and setting them to NaN. Since the algorithms of *Scikit-learn* are not working directly with missing values, those need to be replaced first. To evaluate the sensitivity of each algorithm, two types of replacement are considered: replacement using the median of the attribute and replacement using a prediction of the missing value with a KNN-regressor. The algorithms are tested with 0%, 10%, 20% and 30% missing data at training to obtain quantitative results, namely accuracy and F-measure.

c) Hyperparameter tuning: In this scenario, a grid search is implemented for both algorithms to show the effects on the choice of hyperparameters mentioned in section I. The idea is to show first how the number of estimators affect the algorithms using the toy dataset and then find optimal hyperparameters for the Shill Bidding dataset. Not only the effect on performance is investigated but also the impact on the computational cost is evaluated and thoroughly discussed.

d) Noisy data: In this scenario, the idea is to add gaussian noise to the Shill Bidding training set and measure if both algorithm are still able to accurately classify the sample of the testing set. The added noise for this project is proportional to the standard deviation of each attribute.

e) Outliers: The objective of this scenario is to assess the sensitivity to outliers of AdaBoost and RTF. To do so, first the label of three random points are changed on the toy data set to visualize the effect on the probabilistic decision boundaries. Then, for both algorithms on the Shill Bidding dataset, 5% and 10% of the training data is randomly mislabelled and F-measure performances are reported.

f) Small datasets: Finally, the sensitivity of Adaboost and RTF to very few samples in the training set will be evaluated on the Shill Bidding dataset with the F-measure metric. Few samples are obtained by drastically decreasing the train/test ratio.

III. Results

This section, presents the aforementioned results.

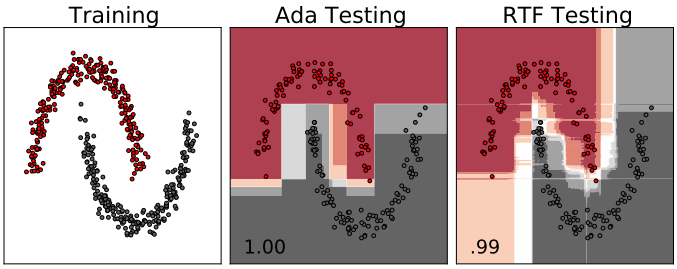

Fig. 1: Benchmark probabilistic decision boundaries and accuracy performances on the toy dataset at testing using 50 estimators for AdaBoost and 100 estimators for RTF

A. Imbalanced dataset

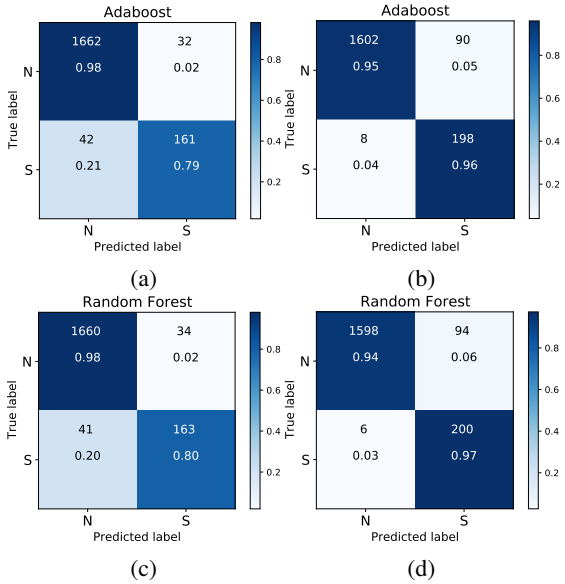

Fig. 2: Normalized confusion matrices obtained on the PCA transformed Shill Bidding dataset (a), (c) and with the balanced version of it using upsampling (b), (d)

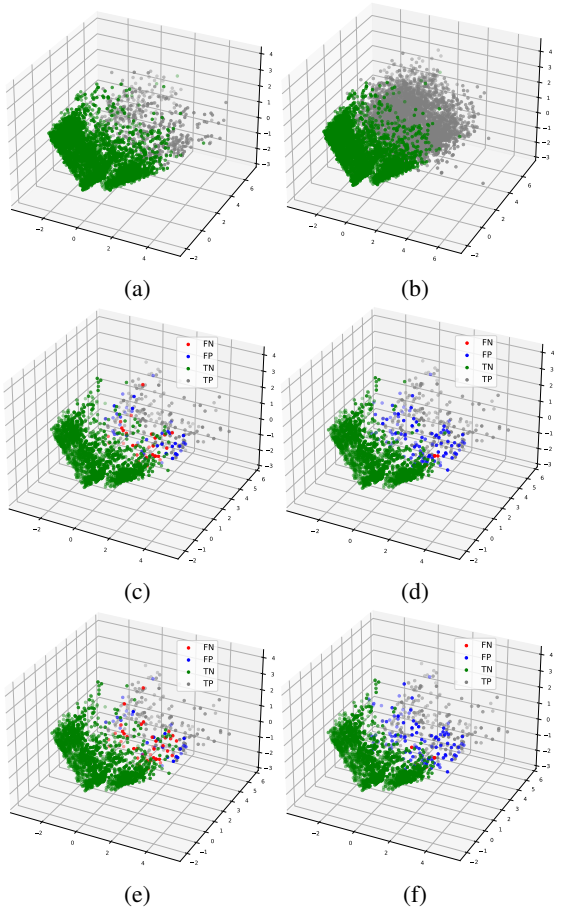

Fig. 3: Imbalanced (a) and balanced (b) PCA reduced Shill Bidding training dataset. RTF classification at testing with balanced (c) and unbalanced (d) training set. AdaBoost classification at testing with balanced (e) and unbalanced (f) training set.

B. Missing data

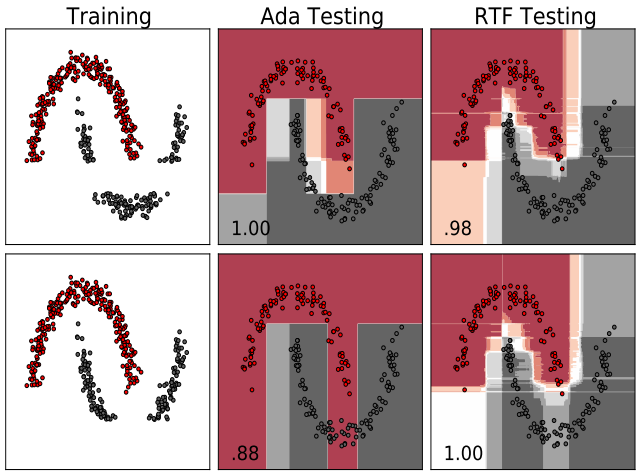

Fig. 4: Influence of missing data on the probabilistic decision boundaries and the accuracy performances on the toy dataset at testing

TABLE I: Effect of missing values on performances on the Shill Bidding dataset

Substituiton with		*Median*		*Prediction*	
Missing	Metric	***Ada***	***RTF***	***Ada***	***RTF***
0%	***Accuracy***	.990	.992	.990	.992
	F-measure	.966	.963	.966	.963
10%	***Accuracy***	.975	.980	.989	.992
	F-measure	.860	.902	.965	.962
20%	***Accuracy***	.958	.966	.988	.990
	F-measure	.823	.843	.955	.957

C. Hyperparameter tuning

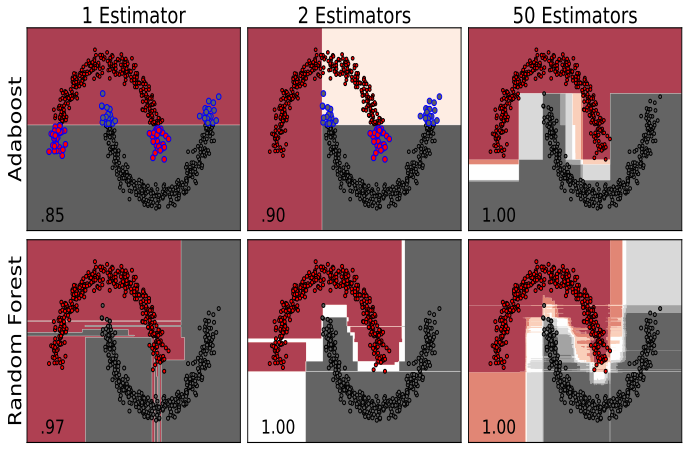

Fig. 5: Influence of number of estimators on the probabilistic boundaries and the accuracy performances on the toy dataset at training

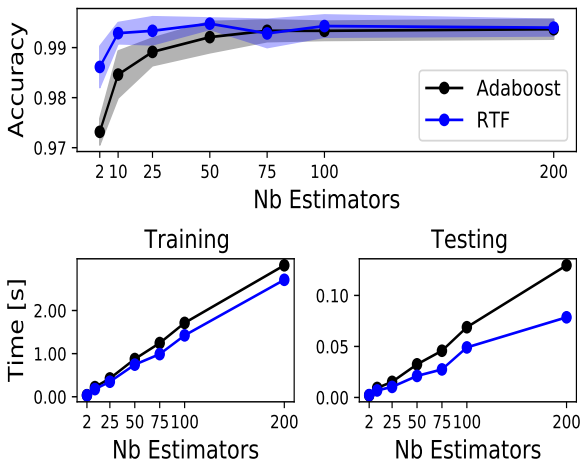

Fig. 6: Influence of the number of estimators on the accuracy (mean) at testing and the execution time (mean) at both training and testing on the Shill Bidding dataset

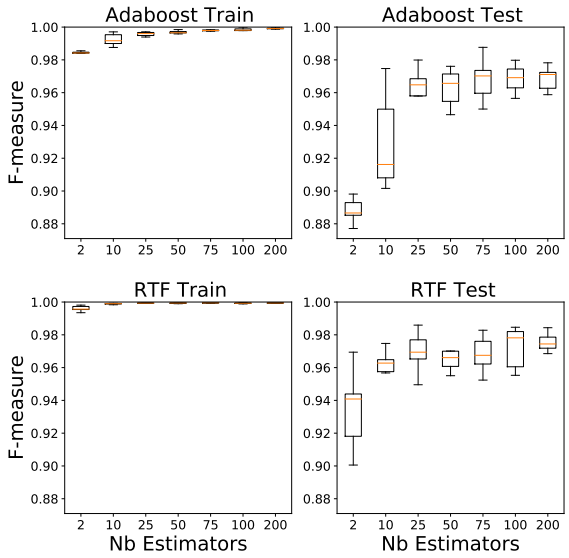

Fig. 7: Box-plot displaying the influence of the number of estimators on the F-measure on the Shill Bidding dataset

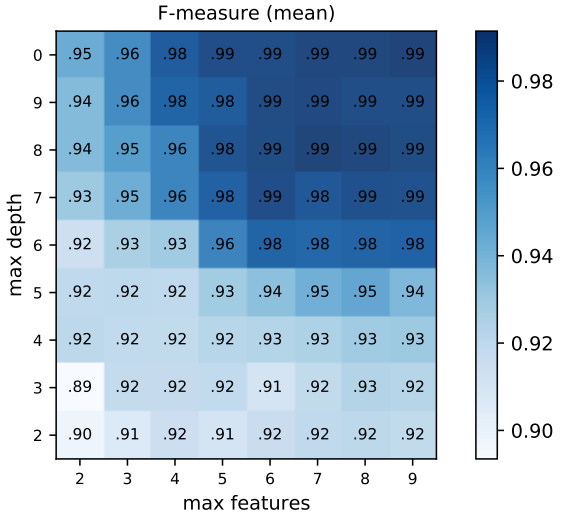

Fig. 8: Heat map of mean F-measure at testing using RTF on the Shill Bidding dataset for different combinations of max depth and max features hyperparameters

D. Noisy data

TABLE II: Effect on the F-measure at testing using the Shill Bidding dataset with added gaussian noise in the training set

	Training		Testing	
Noise	**AdaBoost**	**Rand. Forest**	**AdaBoost**	**Rand. Forest**
0	.993	.998	.962	.969
0.5σ	.975	.993	.939	.929
σ	.928	.987	.909	.919
2σ	.824	.986	.881	.873
3σ	.743	.993	.842	.837

E. Outliers

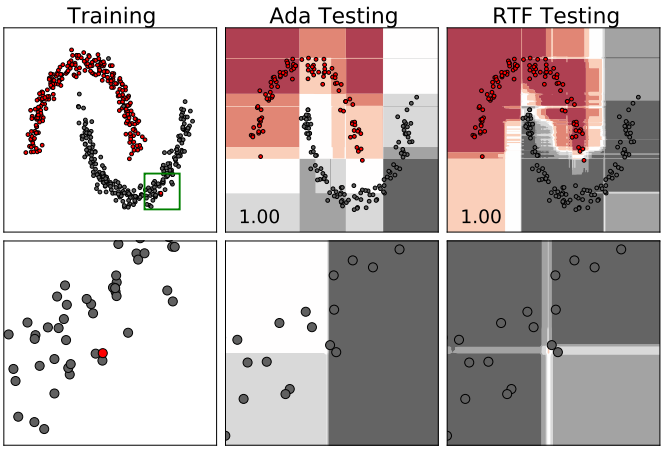

Fig. 9: Influence of three outliers on the probabilistic decision boundaries on the toy dataset with no zoom (top plots) and zoomed in an outlier (bottom plots)

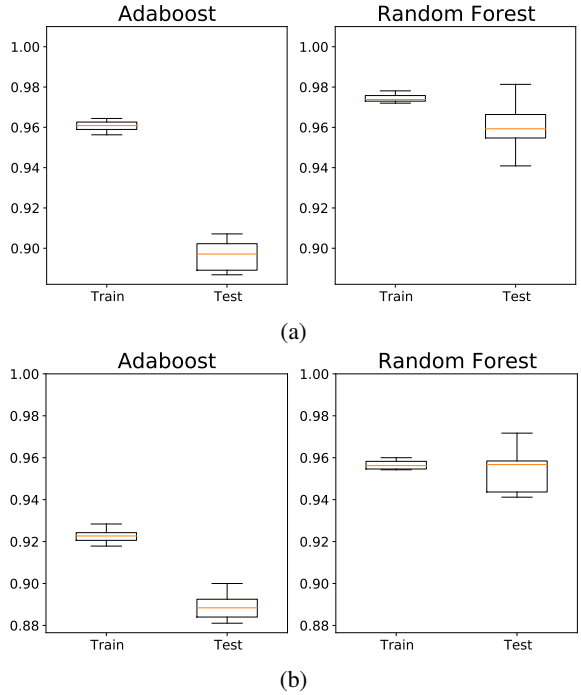

Fig. 10: Influence of mislabeling (a) 5% and (b) 10% of training datapoints of the Shill dataset on the F-measure at testing

F. Small data sets

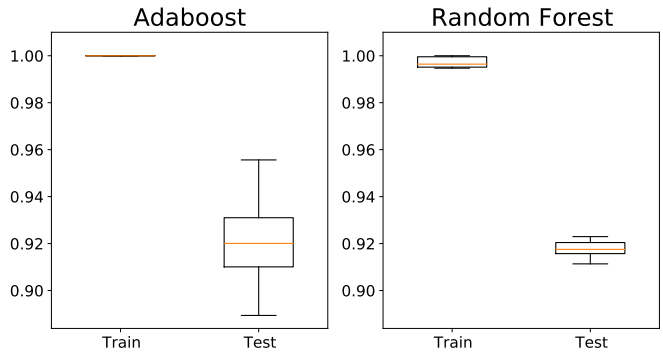

Fig. 11: Influence on the F-measure on the Shill Bidding testing set using only few training samples

IV. Discussion

In this section, the previously shown results of each scenario are thoroughly discussed and potential explanations for them investigated.

A. Imbalanced dataset

Performances on the dimensionality reduced Shill Bidding dataset can be found in figure 2 using the original imbalanced dataset as well as using the balanced version of it obtained by upsampling the minority class using GMMs. Furthermore, the overall accuracy of AdaBoost and RTF can be deduced using equation 1 in section II. It can be seen that for both algorithms, the accuracy at testing is only slightly affected by the balancing of the training data (-1.5%). However, for both AdaBoost and RTF, a shift in the number of false negatives (FN) instances towards false positives (FP) ones is noticeable when comparing the balanced and the imbalanced dataset. This can be explained by looking at figure 3 where it can be seen that the "normal" class (scattered in green) is much denser than the "Shill" class (scattered in grey). On the same figure, the previously mentioned shift can be observed by noticing that sub-figures (c) and (e) (representing the unbalanced dataset) contain roughly the same amount of red (FN) and blue (FP) points whereas sub-figures (d) and (f) (representing the balanced dataset) have almost no red points but more blue ones. The location of those shifting points also allows to understand why this happens. Due to the high imbalance between both classes, RTF and AdaBoost tend to classify datapoints that are on the boundary as "normal" whereas after balancing, both methods learn to classify them as "Shill". For this application, this behaviour is very useful as "Shill" is the most important class. Indeed, it is better to detect too many Shill bidders (FP) and check a posteriori if the prediction was correct, rather than missing some (FN). Overall, Figures 2 and 3 show that on this particular dataset, both algorithm are similarly sensitive to imbalanced of the classes. Besides, it can be concluded that in this case, the GMM balancing method helped drastically improve the results.

B. Missing data

Random Tree Forest is often described as being a robust and efficient algorithm for dealing with missing data. This can also be observed by contrasting its probabilistic decision boundaries on the benchmark toy dataset: figure 1 and the one with missing values: figure 4. When the values are missing on the side of the lower moon (see upper left graph in figure 4), RTF obtains similar results to the benchmark. This is expected as the points removed clearly belong to the lower moon class and additionally are far from the other class. Similarly, Adaboost obtains results close to its benchmark (figure 1). However, when the points are removed from the top of the bottom moon (see bottom left graph in figure 4), the classification task becomes more complicated as it's harder to predict the labels of the missing data. Indeed, it wouldn't be extreme to think that the upper moon class follows its downward trend all the way till the missing values. Through this challenging classification task big performance differences are observed between the algorithms. RTF is able not only to capture the correct trend of the dataset but also to reduce its confidence in the prediction where data is missing. This ability for handling well missing values comes from the injected randomness in each tree which yields reduced variances and better generalization. In contrast, AdaBoost has more trouble with this task and misclassifies the top of the grey class of the toy dataset resulting in a drop in accuracy of 12% in comparison with RTF's performance. Indeed, in figure 4 (bottom center graph), there is a clearly wrong vertical probabilistic decision boundary. This can be explained by the fact that contrarily to RTF, at each iteration, AdaBoost is influenced by the previous one, loosing thus the beneficial randomness of RTF. Indeed, in AdaBoost, as mentioned in I, more weight is assigned to misclassified datapoints from the previous estimator. Here, in the training set, there are no misclassifications using the "wrong" vertical decision boundary, resulting in the algorithm not learning the general trend of the data and being very confident about its wrong prediction. Furthermore, table I shows that RTF outperforms AdaBoost for both accuracy and F-measure when using the simple median replacement method. Overall, the results on both the toy and the Shill Bidding datasets corroborate the theoretical robustness of RTF with missing data. Finally, Table I also shows that when using a predictive model for dealing with missing values, both algorithm perform better than without.

C. Hyperparameter tuning

a) Number of estimators: In this work, decision stumps are used as weak learners for AdaBoost, leaving thus only one tuning hyperparameter: the number of estimators used for the final classification. This paragraph proposes an in-depth analysis of the influence of the number of estimators on both algorithms. First a visual interpretation is proposed using the toy dataset, then overfitting and computational costs are discussed on the Shill Bidding dataset.

The results in Figure 5 show the predicted probability of a meshgrid of query points using AdaBoost (top plots)

and Random Forest (bottom plots) with one, two and three estimators (from left to right plots). First, using AdaBoost, with one decision stump (top left graph), the probabilistic decision boundary is a line minimizing the classification error. In contrast to AdaBoost, using Random Forest with one decision tree (bottom left graph) results in more refined boundaries leading to a better overall accuracy. With one estimator, RTF has more decision nodes than AdaBoost thanks to its deeper decision trees, explaining the gap in performance.

To understand how the AdaBoost's prediction evolves when adding more estimators, the points misclassified by the previous estimator are displayed in distinctive color with a bigger radius as they will have more influence on the following generation of decision stumps (see Figure 5 top graphs). On the top center graph (Figure 5), the second estimator is a vertical line correctly classifying most of the previously wrongly classified samples (i.e. the left red cluster and the two right grey clusters). In RTF, adding one decision tree (bottom center graph) improves the classification as the number of decision nodes is now doubled. With two decision trees, RTF already achieves a perfect accuracy on the training set.

Finally, for both algorithms, the top and bottom right plots confirm that the classification performance improves as the number of estimators increases. Intuitively, as the number of estimators increases, so does the number of decision nodes resulting in more complex decision boundaries allowing for highly non linear classification and thus better performances. Overall, it appears that the more estimators, the better the results. However, increasing the number of estimators increases both the computational cost and the probability to overfit. This parameter must thus be chosen with care. Next, a further analysis of the F-measure performance using both testing and training sets over 10 fold repetitions is proposed to detect potential overfitting. Overfitting is a known drawback of AdaBoost when used with a large number of estimators. It occurs when the algorithm captures the specific trend of the training set instead of the general distribution of the data, leading to a poor generalization. It can be detected by looking at the variance of the F-measure across fold validation rounds. This helps assess if the model generalizes well.

Intuitively, it is expected that RTF tends to overfit less than AdaBoost. Indeed, as aforementioned, the weak learners classification decision is highly influenced by it's predecessor's mistakes. As the iterative process goes on, the estimators will thus give more and more importance on the hardest points to classify which might be outliers or points specific to the training set. This would lead to overfitting as the specificities of the training sets are considered rather than the overall trend of the data. Contrarily, in RTF, the iterative process is randomized, and should thus be more robust to overfitting.

Figure 7 presents the results obtained for both algorithms over ten-folds repetitions splitting each time randomly the dataset into training and testing sets. From the results, it is hard to assess if the algorithms overfit. Indeed, both the testing performance mean and variance improve as the number of estimators increases.

It is worth noting for both algorithms, that once the perfect score at training is achieved, increasing the number of estimators still increases the performance at testing. For instance, in Figure 7, bottom plots, when comparing RTF's performance at training with 50 and 200 estimators, the results do not change. However, the performance at testing increases when passing from 50 to 200 estimators. A possible explanation for this performance improvement is introduced in [3]. It stipulates that given more classifiers, the final classification gains in confidence resulting in the algorithm generalizing well and thus reaching better performances at testing.

Figure 7 shows that RTF reaches the maximal performance on the training set faster than AdaBoost, which is in accordance with the previous observations on the toy dataset. Once again a logical explanation is the difference in the number of decision nodes. AdaBoost uses decision stumps (equivalent to two decision nodes), whereas RTF uses deeper decision trees with more decision nodes, allowing for more complex boundaries with fewer estimators. This also explains the slightly better performances of RTF at testing, especially with few estimators, in comparison with AdaBoost.

Figure 6 (top graph) displays the accuracy at testing as a function of the number of estimators. The same trend as for the F-measure can be observed. Indeed, RTF performs better with fewer estimators, but eventually AdaBoost catches up and both get close to the maximal possible accuracy.

Finally, in figure 6, bottom graphs, the computational time for the two algorithms at both testing and training are displayed. The bottom left graph shows that the computational cost at training is similar for both algorithms, even though their computational cost differ of an $O(log(n)$ term. This is probably due to the fact that in RTF, trees are independent and can be grown in parallel which speeds up execution time. Despite the fact that both algorithms have the same theoretical computational complexity at testing, the bottom right graph shows that RTF is slightly faster than AdaBoost. However, both are efficient and much faster than at training.

b) Maximum depth and Maximum features: As explained previously, aside from the number of estimators, the maximum depth and the maximum number of features are important tuning hyperparameters of RTF. The maximum depth prevents to have deep decision tress which have a tendency to overfit the training data greatly. Additionally, as seen in section II, limiting the maximum depth results in a slight decrease of computational cost. The maximum number of features to consider when building a tree is the key parameter to add randomness to the model. Thus, decreasing the maximum number of features results in more diverse trees which tends to overfit less the training set. However, if the number of feature is too low, the model can have trouble extracting valuable information from the training data. In general, a good empirical result for the maximum number of feature is the square root of the total number of features. It can also be noted, that the less features considered to build a tree, the smaller the trees and thus, the smaller the computational cost. Figure 8 shows results of the grid search over those two parameters

in form of a heat map. It can be seen that for this specific dataset, globally the deeper the trees and the more features considered, the better the F-measure at testing. However, one can see that at some point, the performance converges and does not improve further by increasing the depth or by adding feature. One can thus argue that it would be more appropriate to choose hyperparameters that yield in good results while decreasing potential overfitting and computational cost. This means that for this specific dataset, a maximum depth of 8 and a maximum number of feature to consider for each Tree of 6 could be considered as optimal.

D. Noisy data

Random Tree Forest is known to be able to efficiently process noisy data thanks to the randomization induced when growing each decision tree [4]. In contrast with models such as AdaBoost that often overfit noise in the training set, RTF reduces the variance by applying the majority vote rule, thus producing a model that has a low bias and low variance. Results on the Shill Bidding dataset which can be found in table II, show that AdaBoost's F-measures drops heavily on the training set as more noise is added whereas RTF keeps its performance. This can be explained by the fact that AdaBoost has trouble separating classes that are intertwined such as a typical checkerboard dataset. This is mainly due to the very simple weak classifier (i.e. decision stumps) that is used in this project. It is worth noting that even with the difficulties encountered by AdaBoost on the training set, the method performs similarly to RTF on the testing set. It is thus, also able to capture a general trend in noisy data.

E. Outliers

First, the effect of an outlier on the probabilistic decision boundaries can be observed in figure 9. In the bottom graphs, the zoom on the zone with the outlier shows that both algorithms tend to overfit outliers. When comparing the overall probabilistic decision boundaries with the results obtained on the benchmark dataset 1, the more complex decision boundary of RTF is less impacted than the one of AdaBoost where the prediction confidence is significantly reduced. This can be explained by the simplicity of the weak learner AdaBoost uses. For a more complex dataset, RTF will also be less likely to overfit the training data thanks to the randomness added in each Tree and the voting scheme which results in better performances. For instance, classifying well outliers on the training set would lead to a poor classification of the testing set as outliers should have been ignored but influenced the classifications boundaries. This result can also be observed in figure 10 where AdaBoost's performance on the training as well as on the testing set is worse than RTF's one. In the case of 5% mislabelled datapoints, AdaBoost's performance on the testing set deteriorates massively while Random Forest still performs well. This can be explained by the fact that AdaBoost recursively increases the weights of the recently misclassified instances. Furthermore, those having incorrect class labels will be harder to classify (as their features correspond to the ones from the opposite class) and will thus tend to be misclassified. Therefore, AdaBoost concentrates on increasing weight on these mislabelled instances and becomes corrupt. In the scenario with 10% mislabelled datapoints, AdaBoost also shows weaknesses in classifying correctly the training data in addition to the previously mentioned errors on the testing set. Reasons for that are the same as with noisy data, were the simplicity of the weak learner does not allow to separate well intertwined datasets.

F. Small data sets

Lastly, figure 11 shows the results of few training sample on the Shill Bidding dataset. It is noticeable that on average, both method perform similarly on the testing set. However, AdaBoost's F-measure depends more on the training point than RTF which can be seen in the much bigger variance of the F-measure. This again indicates that RTF generalizes better than AdaBoost and can be explained by the added randomness to the RTF model when growing trees.

V. Conclusion

Choosing the right classification algorithm depends on multiple factors such as computational cost, robustness to noise and ability to deal with missing values. It is thus application and dataset dependant. This project focused on contrasting AdaBoost and Random Tree Forest in different scenarios to understand in-depth advantages and drawbacks of each methods. Results of the different experiments showed excellent performances of both algorithms on the binary classification task when trained on balanced or complete datasets. They also showed that RTF's randomized process gave it an edge over Adaboost with challenging data sets were data was missing over labels were misclassified. However, it is worth noting that AdaBoost has a big advantage in comparison with RTF which is the simplicity of its setup which requires much less hyperparameter tuning. Additionally, the different evaluations confirmed that the computational complexity of both methods is similar at testing as well as testing even though RTF was executed slightly faster at the latest. Finally, ideal hyperparameters for the Shill Bidding dataset for both algorithm have been discussed as well as their impact on the computational cost. Further work could address the effect of changing the type of weak learner on all the tested scenarios as well as looking more in depth at different RTF hyperparameters.

References

[1] J. Zhu, S. Rosset, H. Zou and T. Hastie, *Multi-class AdaBoost*, Statistics and its interface, vol. 2, January 2006.

[2] A. Billard, *Evaluating the performance of classifiers*, MICRO-455: Applied Machine Learning, lecture 5, slide 5, EPFL, 2020.

[3] R. Schapire, *"Explaining AdaBoost,"* in Empirical Inference, V. Vovk, Ed. Heidelberg: Springer, 2013, pp. 37-52.

[4] A. Na'eem, M. Onisimo, P. Kabir and I. Riyad, *The Impact of Simulated Spectral Noise on Random Forest and Oblique Random Forest Classification Performance*, Journal of Spectroscopy, vol. 2018, pp. 1-8, March 2018.