Classifier comparison for student's test results

María Camila Vásquez Correa
Mathematics department
Universidad EAFIT
Medellín, Colombia
mvasqu49@eafit.edu.co

***Abstract*—In this article is presented a comparison of several classifiers in order to determine, based in the results of a test and other features, whether or not the student in question took a preparation course for said test. The classifiers include: Logistic regression, decision tree, support vector machine and neural networks in configuration multi layer perceptron. For each one of them, the probable learning guarantee is evaluated and the results of the classification are tested in terms of the ROC curve.**

***Index Terms*—Classification, test results, ROC, sklearn.**

I. Introduction

A classifier is a function that assigns a class label to an instance. For example, given information about a credit-card applicant, a classifier could decide whether the person is a good risk and so should receive a credit card [1]. There is much interest in the field of pattern recognition on trainable pattern classifiers, as seen, for example, in the growth in the area of neural networks. Parallel to this development, in the field of machine learning there have been many theoretical advances on distribution- free learning of Boolean functions. This learning frame- work is known as the probably approximately correct (PAC) model [2].

II. Literature review

Several authors in several fields have addressed the PAC learning in different architectures, algorithms and tasks in machine learning. For example, [2] proposes, and is a headstart for boolean classifiers, a PAC learning guarantee for the min-max classifiers, that takes into account a binary classification problem with binary valued features. Even when this is a useful result, in real life the data is way more complex. This is the case of the tasks presented in [3], that are tasks with low budget (data, memory, time) which need to have a bound or a guarantee for what can be learned from the given data.

Authors like [4] and [5] have proposed learning guarantees based in a PAC-bayes learning, for linear classifiers. Although the method for defining the guarantee is different, is really useful for more complex classification tasks with more complex features. Other constrains for the PAC framework are based in the discriminant of the functions used, as in [6]. And many other tasks can be bounded by this guarantee, like the collaborative learning ([7]), the learning from the crowd ([8]) and the averaging classifiers ([9]).

In most cases, the learning task we have ahead is a known label task, and we can determine some guarantee for it ([10]), and we can even determine the complexity of the sample in the task ([11]). However, this may not always be the case for the learning. In this case, we may not be able to guarantee a learning, since the task is unknown.

For this exercise, we will precise the learning guarantees for a task with known labels in different learners and watch how this theory is applied in real problems.

III. Methodology

A. Data

The data used for this project is a table containing in each row the information about a certain student that took a test, this features include: gender, ethnicity, parental level of education, the lunch they had that day, whether or not they finished a preparation course and their scores in each category: math, reading an writing. The task our classifiers will try to learn is how to predict if a student took a preparation course given the rest of the features present in the dataset.

B. Probable approximated correct learning

As regarding of PAC learning, we will give some definitions that will guide our trip trough learning [12]:

- **Instance space:** Let X denote the set of possible instances or examples that may be seen by the learning algorithm. This will be our instance space. In our case, are all the examples of students.
- **Concept class:** A *concept* c over X is a boolean function $c : X \to \{0, 1\}$. A concept class C over X is a collection of concepts c over X. Our concept class is, in this case, all the ways the classifier learns how to predict if the student took the test or not.
- **Data generation:** Let D be a fixed probability distribution over X. The training data is obtained by drawing an example $x \in X$ according to the distribution D. If c is the target concept, the example is labelled as $c(x)$. The learning algorithm observes $(x, c(x))$.

Finally, with all this concepts we can define PAC learning: Let C_n be a concept class over X_n and let $C = \cup_{n\geq 1}$ and $X = \cup_{n\geq 1}$. We will say that C is PAC learnable if there exists an algorithm L that satisfies the following: for all $n \in \mathbb{N}$, for every $c \in C_N$, for every D over X_n and $\varepsilon, \delta > 0$, if L is given

access to the data, and inputs ε, δ and $size(c)$, L outputs $h \in C_n$ that with probability at least $1-\delta$ satisfies $err(h) \leq \varepsilon$.

However, this definition can be rewritten in terms of the VC-dimension, that is defined over the instance space by the size of the largest finite subset of X that is shattered by an hypothesis class H.

C. Algorithms

For this exercise, the algorithms to be used and their respective VC dimension are listed bellow (with an exception in the Multi Layer Perceptron):

- Logistic Regression: This is a linear classifier and so the VC dimension associated with it is $d+1$, where d is the number of features present in each example (In our case, $d = 7$)
- Decision tree: The VC-dimension for this classifier depends in the depth of the tree, that we will name l. And so, the VC-dimension is 2^l.
- Support Vector machine: This classifier is split in three parts, depending on the kernel used:
  - Linear Kernel: As a linear classifier, its VC dimension is again $d+1$.
  - Polynomial Kernel: The VC dimension is the equal to the enumerative combinatorics of the number of monomials for a polynomial of degree k and d variables, thus $\binom{d+k}{k}$.
  - Radial Basis function Kernel: This VC dimension is infinite.
- Neural network in configuration Multi Layer perceptron: This classifier is used in several configurations of parameters: Number of hidden layers and Number of nodes in each layer in {1,2,3}, and learning rate in {0.2,0.5,0.9}.

And so, we define the optimal sample size as:

$$m \geq \frac{1}{\varepsilon}\left(VC(H) + \ln\left(\frac{1}{\delta}\right)\right)$$

In our case, we will fix the confidence 1-δ in a value of 95%, so we can find the error that we can expect for the number of samples we are dealing with (currently, 600 for the training set).

D. Data preprocessing

The data contains several categorical features that are mapped into numeric features using a one hot encoder. This leaves us with binary features instead of categorical ones. The numeric features related to the scores in the test are scaled into the [0,1] interval.

The data is then splitted into 3 subsets: Training, validation and tests. And we proceed to do the same with and embbeded dataset (using the BH-tsne algorithm) to try and learn the data in lower dimensions, as shown in Figure 1

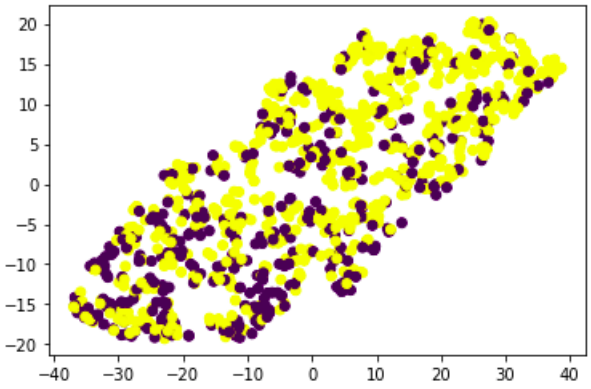

Fig. 1. Embbeded dataset

IV. Results

A. Logistic regression

The maximum error for the high dimensional dataset applying the equation we showed above, is 0.03. However, as evidenced in the cross validation procedure in Figure 2 and the performance in the test set shown in Figure 3 the generalization of the model is far from that. Some factors that can contribute to this results can be that the data is not linear separable, and so the algorithm is far from finding a good hypothesis for the data.

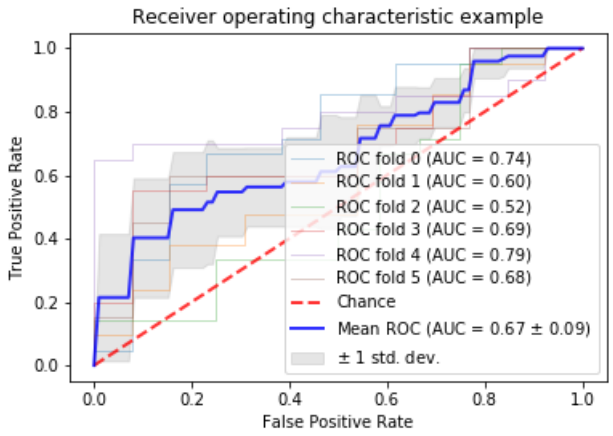

Fig. 2. K-fold cross validation for logistic regression using the validation set in high dimensions.

For the embbeded dataset, the maximum error guaranteed for a confidence of 95% is of 0.009. However, seeing the results of the Figure 4, that shows the performance in the validation test and in Figure 5 that shows the final performance of the algorithm in the test set we evidence that the error is far from this value. The reasons behind this can be multiple, like the splitting done for the tests, the fact that the data in 2 dimensions is also no linearly separable, and so the real generator of the data is not in the hypothesis class build by the algorithm.

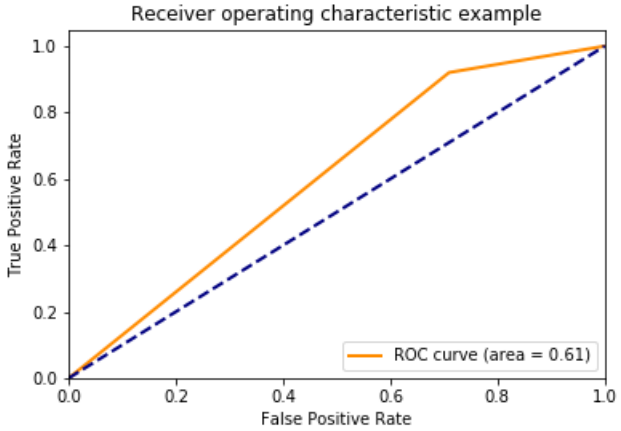

Fig. 3. Results in the test set for logistic regression in high dimensions.

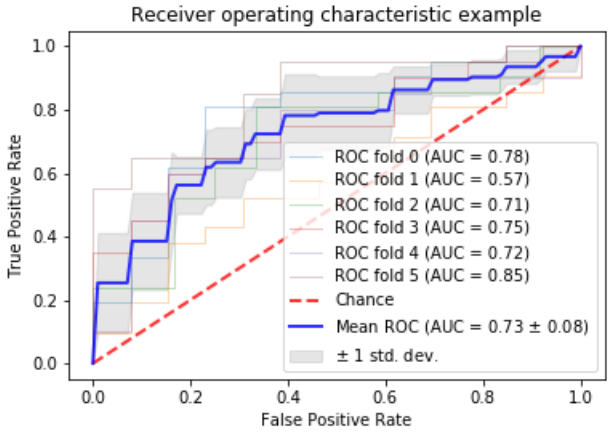

Fig. 4. K-fold cross validation for logistic regression using the validation set in low dimensions.

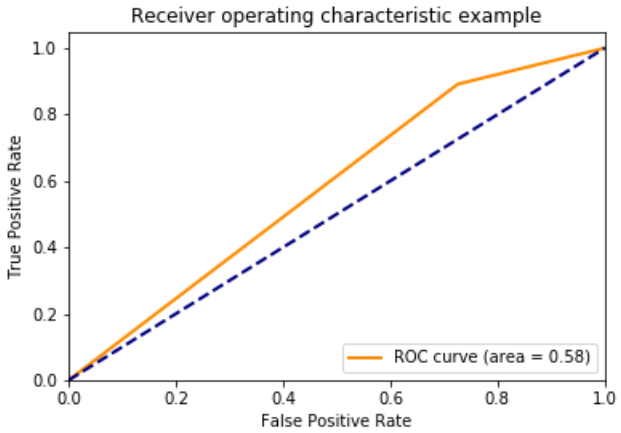

Fig. 5. Results in the test set for logistic regression in low dimensions.

B. Decision tree

In this case, the election of the depth of the three was fundamental for the guarantee. However, there is much more to that. A very deep tree can easily over fit the dataset, making the generalization even worse. For the original dataset, the error guaranteed was around 0.05, but again, our results are far beyond that, as shown by the validation scheme in Figure 6 and the test performance in Figure 7. A simple conclusion to draw is that the real function that explains the data is not present in the hypothesis class generated by the decision tree. However, other factors can be implied, like the implementation that approximates the building of the tree.

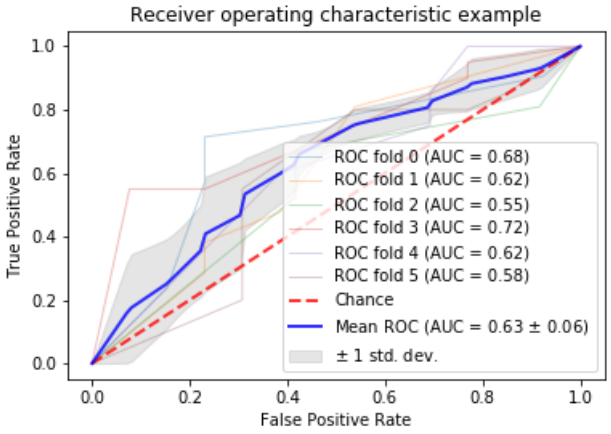

Fig. 6. K-fold cross validation for decision tree using the validation set in high dimensions.

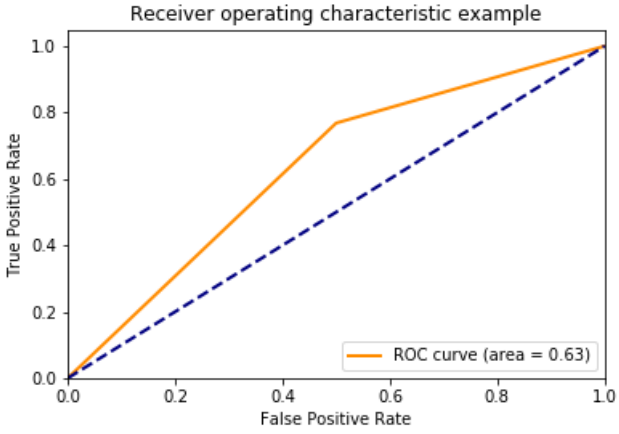

Fig. 7. Results in the test set for decision tree in high dimensions.

For the embbeded dataset, in this case, the depth was preserved, even when the data had so low features in this case. So we are expecting some over fitting here. The error guaranteed by the tree is the same, due to the fact that the VC dimension do not change according to the features in the dataset. Indeed, as shown in Figure 8, the validation scheme and the test performance in Figure 9 are worse than for the higher dimension case, possibly because the overfitting problem we have.

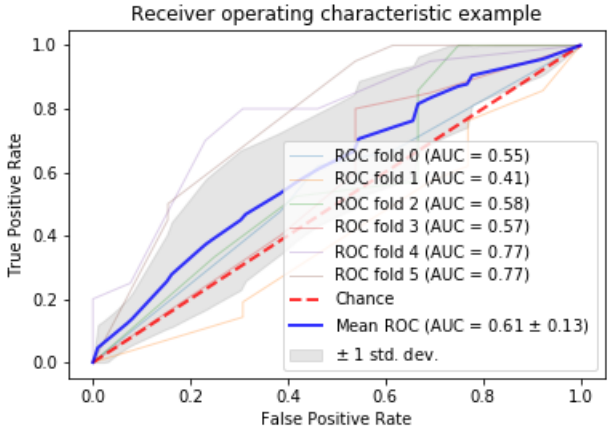

Fig. 8. K-fold cross validation for decision tree using the validation set in low dimensions.

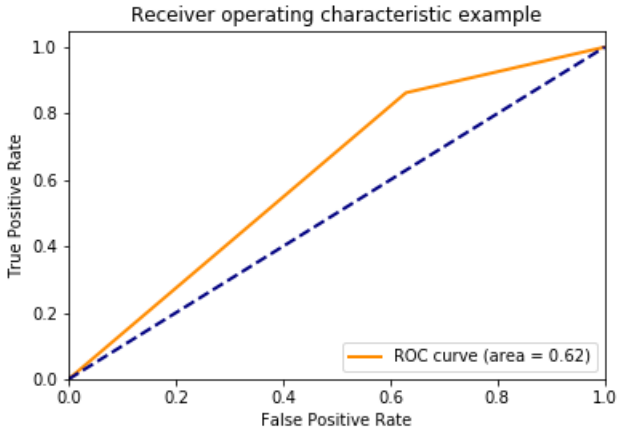

Fig. 9. Results in the test set for decision tree in low dimensions.

C. Support vector machine

1) Linear Kernel: In this case, the results are quite similar to the ones shown by the logistic regression, given their linear separator nature. The expected error was the same, and again, and as expected by the previous results, the algorithm does not reach the error guaranteed in theory, as shown in Figures 10, the cross validation scheme, and 11 the performance in the test set. The difference in this case is that the algorithm does not provide much difference in the validation and the performance is a little lower that with the logistic regression.

For the embbeded dataset we face a similar problem. As seen before, the data is no linearly separable and, in fact, the results are worse than for the embedded dataset in the logistic regression, probably because of the implementation or the support vectors generated being different than the hyperplane drawn by the logistic regression. However, they do not vary a lot for the support vector machine in linear kernel with high dimensions, this do not provide more information for the algorithm. The results are shown in Figures 12 and 13.

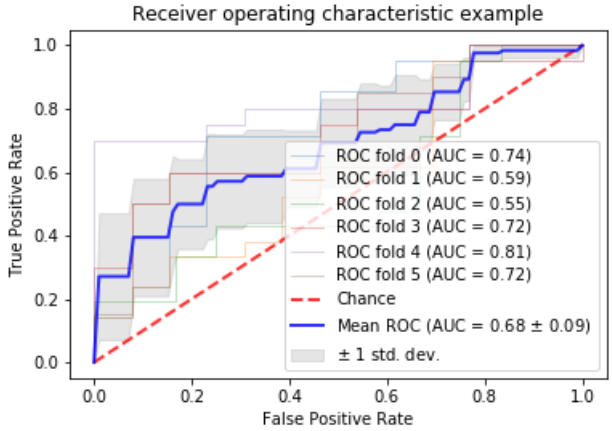

Fig. 10. K-fold cross validation for Support Vector machine with linear kernel using the validation set in high dimensions.

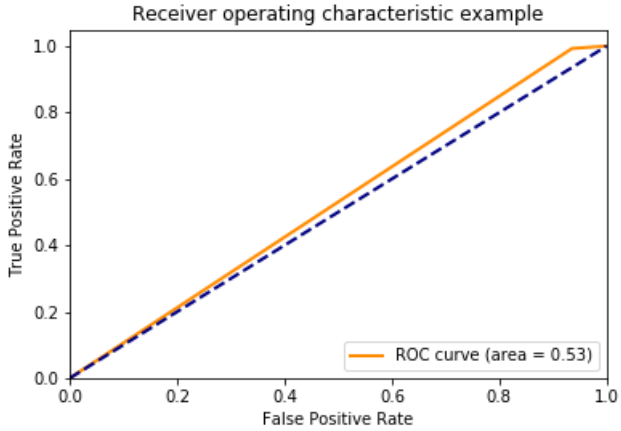

Fig. 11. Results in the test set for Support Vector machine with linear kernel in high dimensions.

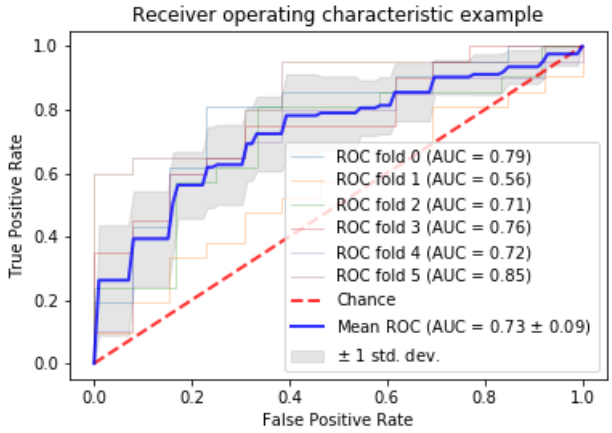

Fig. 12. K-fold cross validation for Support Vector machine with linear kernel using the validation set in low dimensions.

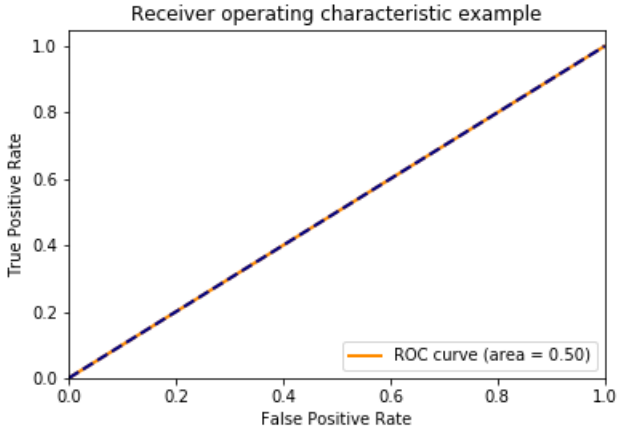

Fig. 13. Results in the test set for Support Vector machine with linear kernel in low dimensions.

2) Polynomial Kernel: This is a much more complex machine, as shown by its VC dimension of 1330 for the complete dataset and 10 for the embbeded dataset. As such, the data required to have an error that makes sense is way more than the data that we have available in this particular problem. For the complete dataset, the guaranteed error was around 2.2, which is insanely big. However, that is the maximum error, and we evidence that our error is lower than that in the validation scheme presented in Figure 14 and the performance in the test set shown in Figure 15.

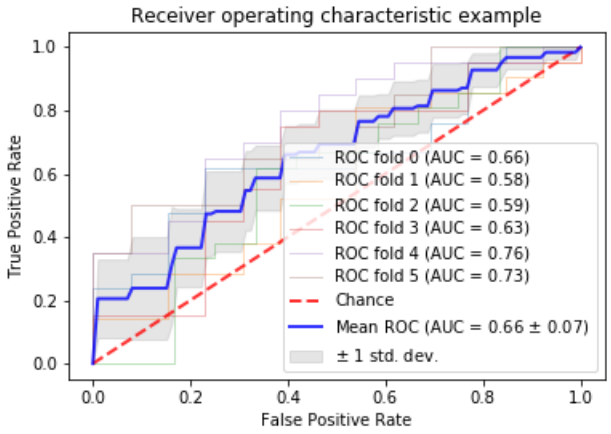

Fig. 14. K-fold cross validation for Support Vector machine with polynomial kernel using the validation set in high dimensions.

In the embbeded dataset we evidence a similar situation. The machine is not that complex given that the default polynomial degree is 3, however, the guaranteed error for its dimensions is around 0.02, and this error is not achieved in the validation and test set (Figures 16 and 17) with the provided training data.

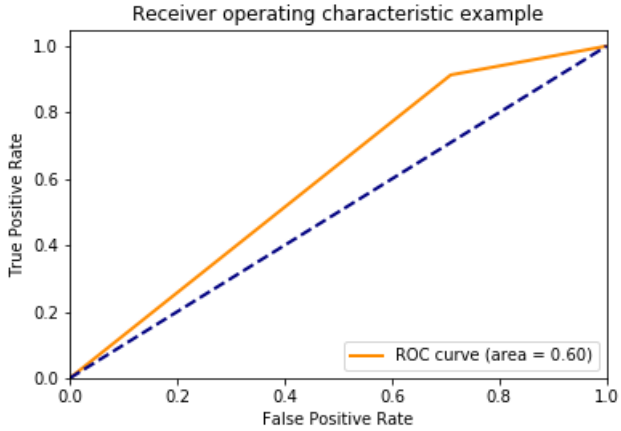

Fig. 15. Results in the test set for Support Vector machine with polynomial kernel in high dimensions.

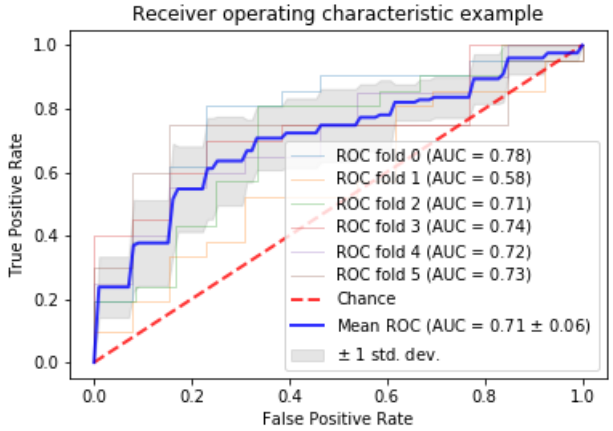

Fig. 16. K-fold cross validation for Support Vector machine with polynomial kernel using the validation set in low dimensions.

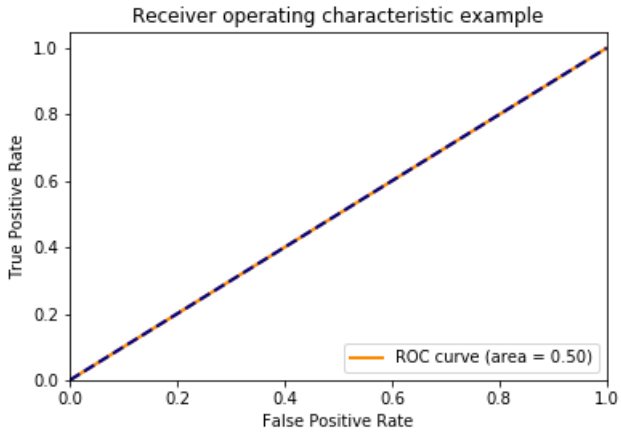

Fig. 17. Results in the test set for Support Vector machine with polynomial kernel in low dimensions.

3) Radial basis function Kernel: As evidenced by its infinite VC dimension, this machine is really complex and should be trained with the most amount of data as possible. We were not able to determine a guarantee for the error. However, we did validate and test the algorithm for both datasets. In the complete one, the results for the validation scheme (in Figure 18) shows not very good results, and include a high variation among them. In the test set, the performance is really bad, having 0.5 area under the ROC curve. However, this performance matches the one achieved by this machine using different kernels.

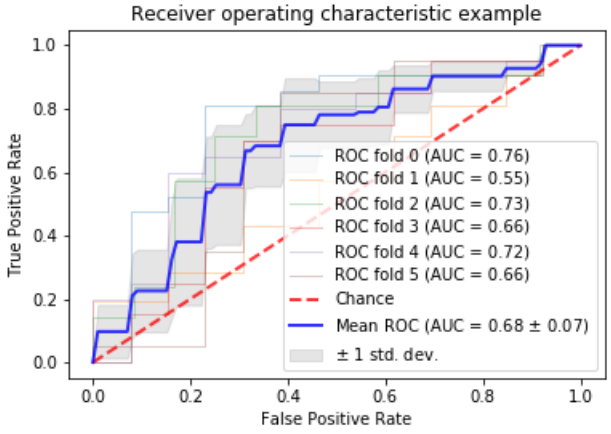

Fig. 18. K-fold cross validation for Support Vector machine with rbf kernel using the validation set in high dimensions.

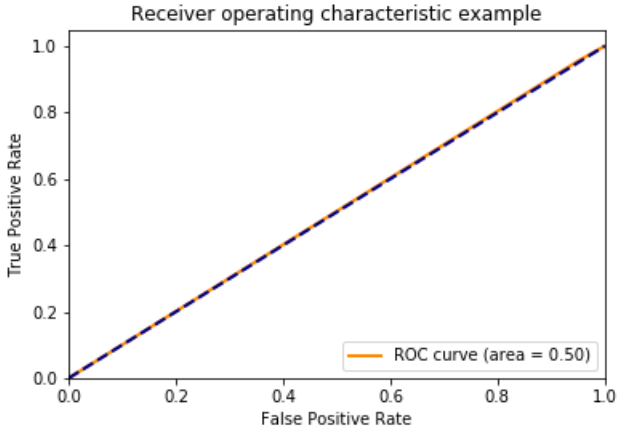

Fig. 19. Results in the test set for Support Vector machine with rbf kernel in high dimensions.

In Figures 20 and 21 are shown the results achieved for the embbeded dataset, that seem to be better than for the complete dataset. This is another example that can be used to determine if the features present in the complete dataset are informative enough to learn.

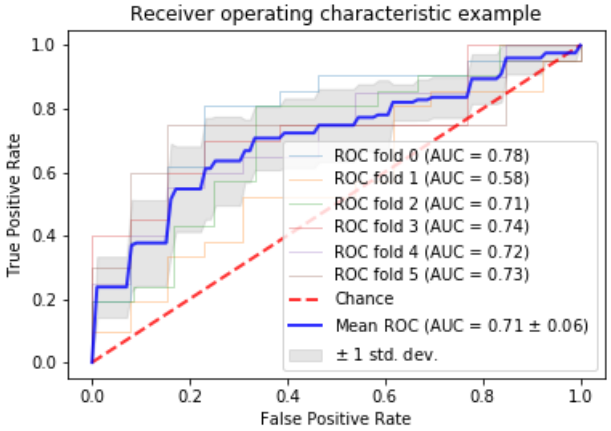

Fig. 20. K-fold cross validation for Support Vector machine with rbf kernel using the validation set in low dimensions.

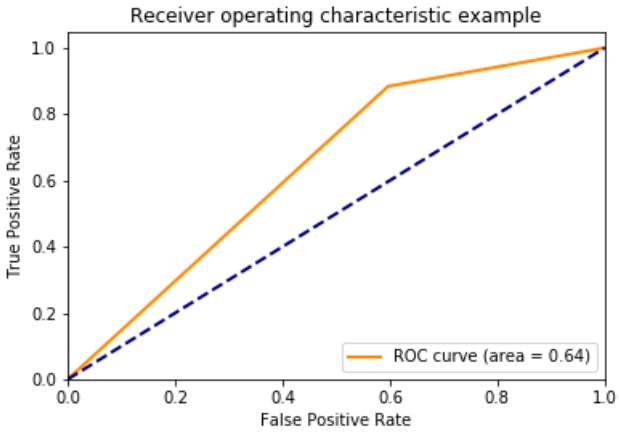

Fig. 21. Results in the test set for Support Vector machine with rbf kernel in low dimensions.

D. Neural networks

For this particular machine, due to its complexity and the number of parameters required to train it (this parameters include the learning rate, the number of layers, the parameters for the activation function of the nodes, the number of nodes, the optimization algorithm among others), we were not able to determine a guarantee for the error. Instead of explicitly looking at the figures, let us look at the area under the ROC curve for each combination of parameters in the validation scheme and in the test set. Tables I to VI show these results.

Learning rate	1	2	3
0.2	0.53109	0.592338	0.578175
0.5	0.50000	0.500000	0.500000
0.9	0.50000	0.500000	0.500000

TABLE I
MEAN AREA UNDER THE ROC CURVE FOR THE K-FOLD WITH 1 HIDDEN LAYER.

Learning rate	1	2	3
0.2	0.5	0.699275	0.672043
0.5	0.5	0.500000	0.500000
0.9	0.5	0.500000	0.500000

TABLE II
AREA UNDER THE ROC CURVE FOR THE TEST SET WITH 1 HIDDEN LAYER.

Learning rate	1	2	3
0.2	0.526603	0.561279	0.548901
0.5	0.500000	0.500000	0.500000
0.9	0.500000	0.500000	0.500000

TABLE III
MEAN AREA UNDER THE ROC CURVE FOR THE K-FOLD WITH 2 HIDDEN LAYERS.

Learning rate	1	2	3
0.2	0.5	0.5	0.5
0.5	0.5	0.5	0.5
0.9	0.5	0.5	0.5

TABLE IV
AREA UNDER THE ROC CURVE FOR THE TEST SET WITH 2 HIDDEN LAYERS.

Learning rate	1	2	3
0.2	0.5	0.5	0.509615
0.5	0.5	0.5	0.500000
0.9	0.5	0.5	0.500000

TABLE V
MEAN AREA UNDER THE ROC CURVE FOR THE K-FOLD WITH 3 HIDDEN LAYERS.

Learning rate	1	2	3
0.2	0.5	0.5	0.5
0.5	0.5	0.5	0.5
0.9	0.5	0.5	0.5

TABLE VI
AREA UNDER THE ROC CURVE FOR THE TEST SET WITH 3 HIDDEN LAYERS.

The results do not vary a lot from combination to combination, with one exception being 1 hidden layer with 1 node and learning rate 0.2, that shows in TableII and I the best performance (although it is not considered a good performance overall). Seeing that this machine, that we may consider the most complex of them all (maybe paired with the support vector machine with rbf Kernel), does not learn a lot from the data, we can say that, clearly, it need more data to be able to generalize well in the test and validation sets. Also, it may depend in some hyper parameters (that remained unchanged during all the experiments from the defaults provided by scikit-learn) like the type of optimization, the regularization and others, that the algorithm does not behave well in this particular dataset.

E. Autoencoder

Finally, we want to see the performance of the best model (Decision tree) in a dataset encoded by an Autoencoder with 1 hidden layer that had 10 nodes in it, and a Sigmoid activation, that was trained with the same dataset and its respective training loss is shown in Figure 22. The cross validation score for this model is near the training score, around 0.04 on mean squared error.

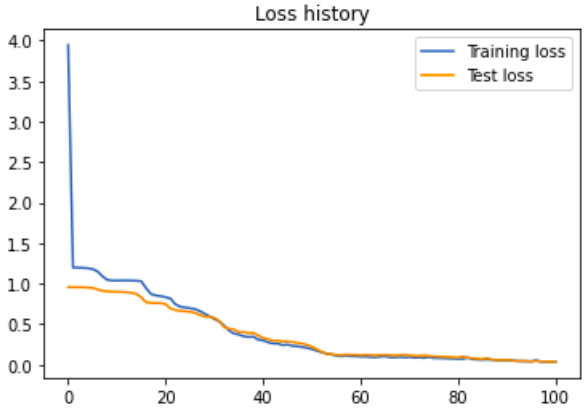

Fig. 22. Loss history for the autoencoder

With the autoencoder trained and the encoded dataset extracted, we proceed to train our decision tree in this new and smaller dataset. The validation process for this architecture is shown in Figure 23 and the test results are shown in Figure 24. Even when the autoencoder reached a significantly low loss in train as well as in test and validation, the encoding of the dataset did not bring more information to the best classification model we identified in the previous steps.

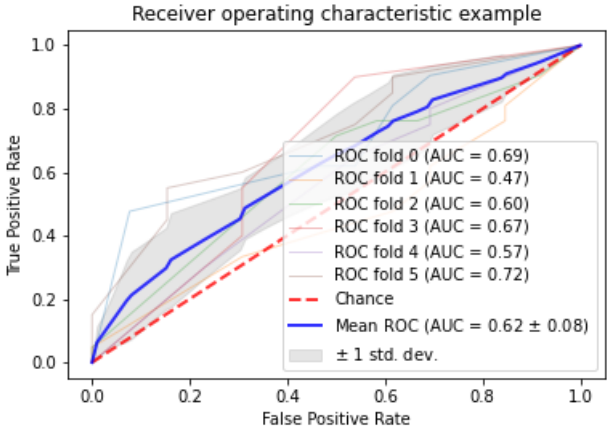

Fig. 23. K-fold cross validation for the decision tree in the encoded dataset

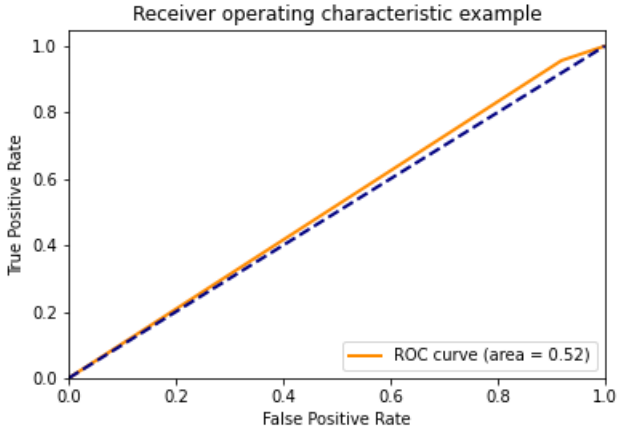

Fig. 24. Results in test for the decision tree in the encoded dataset

V. Conclusions

We tested some of the most famous classification algorithms in a dataset and determined (in most cases) their probable approximated correct learning guarantee given the sample size we had (after partitioning the dataset in training, validation and test to show the generalization capability of the algorithms). After this work, some conclusions can be drawn: The PAC learning guarantee is only feasible if the function that is consistent with every data point (on the population) exists in the hypothesis class generated by the machine learning algorithm. Although this does not mean that the machine does not learn due to the fact that this function is not generated, it can be a factor in the failure of the learning process.

The VC dimension of the machines is an approximated number in some cases, found by iterative or approximated algorithms, and does not take into account the implementation of the algorithms, for example, the optimization procedure and its parameters. This one can also be a reason for the failure in most of the machines in the learning process (one of the machines had a guaranteed error really high, due to its complexity and the low amount of data).

The PAC learning guarantee is a great guide when dealing with these particular machines (the ones with a guarantee that can be determined) cause it shows how the learning process can be finished and can be an indicator of some other failures (considering a linear separation of non linearly-separable data, for example).

Finally, the most complex machines require a lot of data, but also hyper parameter tunning and choosing the correct machine, that can generate hypothesis classes that somehow match the real function that generates the dataset.

References

[1] R. Greiner, A. J. Grove, and D. Roth, "Learning cost-sensitive active classifiers," *Artificial Intelligence*, vol. 139, no. 2, pp. 137 – 174, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370202002096

[2] P. F. Yang and P. Maragos, "Min-max classifiers: Learnability, design and application," *Pattern Recognition*, vol. 28, no. 6, pp. 879–899, 1995.

[3] A. Kapoor and R. Greiner, "Learning and classifying under hard budgets," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3720 LNAI, pp. 170–181, 2005.

[4] J. Honorio and T. Jaakkola, "Tight bounds for the expected risk of linear classifiers and PAC-bayes finite-sample guarantees," *Journal of Machine Learning Research*, vol. 33, pp. 384–392, 2014.

[5] L. Begin, P. Germain, F. Laviolette, and J. F. Roy, "PAC-Bayesian theory for transductive learning," *Journal of Machine Learning Research*, vol. 33, pp. 105–113, 2014.

[6] P. W. Goldberg, "Some discriminant-based PAC algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 283–306, 2006.

[7] A. Blum, N. Haghtalab, A. D. Procaccia, and M. Qiao, "Collaborative PAC learning," *Advances in Neural Information Processing Systems*, vol. 2017-December, no. Nips, pp. 2393–2402, 2017.

[8] P. Awasthi, A. Blum, N. Haghtalab, and Y. Mansour, "Efficient PAC Learning from the Crowd," *arXiv*, no. 4, pp. 1–22, 2017, http://arxiv.org/abs/1703.07432.

[9] J. Langford and M. Seeger, "Bounds for averaging classifiers," 2001.

[10] S. Ben-David and S. Ben-David, "Learning a classifier when the labeling is known," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6925 LNAI, pp. 440–451, 2011.

[11] S. Hanneke, "The optimal sample complexity of PAC learning," *Journal of Machine Learning Research*, vol. 17, pp. 1–15, 2016.

[12] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[13] A. Blum, N. Haghtalab, A. D. Procaccia, and M. Qiao, "Collaborative pac learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2392–2401. [Online]. Available: http://papers.nips.cc/paper/6833-collaborative-pac-learning.pdf

[14] O. L. Quintero, *Machine Intelligence for Human Decision Making*. Universidad EAFIT, November 2018.

[15] S. Hanneke, "The optimal sample complexity of pac learning," *J. Mach. Learn. Res.*, vol. 17, no. 1, p. 1319–1333, Jan. 2016.