Machine Learning - Assignment 2

Bruno Vaz
Faculty of Sciences
University of Porto
up201705247@edu.fc.up.pt

Filipe Justiça
Faculty of Sciences
University of Porto
up201606339@edu.fc.up.pt

Tiago Coelho
Faculty of Sciences
University of Porto
up201604170@edu.fc.up.pt

Vasco Soares
Faculty of Sciences
University of Porto
up2016043640@edu.fc.up.pt

***Abstract*—This project aims to develop artificial datasets that will result in good performances of specific models from a set of methods. We decided to generate datasets for *Logistic Regression*, *Linear* and *Quadratic Discriminant Analysis*, *Decision Trees* and *Random Forests*, *linear, polynomial* and *radial basis SVMs* and, finally, *MLPs* with a *tanh* and *relu* activation functions. We were able to develop the proposed datasets and interpret the results according to the models that performed well.**

***Index Terms*—Machine Learning, Dataset Generation, Classification**

I. Introduction

In most Supervised Machine Learning Classification tasks, a dataset consisting of predictor variables and a target variable is given, and the aim is to find a function that maps the predictor variables into the target variable. The task proposed in this assignment was, however, given a method from a set of methods, to generate a dataset in which the given method performs better than all others. The essay is organized as follows. In Section II we will be explaining the objectives of this essay. In Section III we shall present the methods that were used, and in Section IV the datasets that were generated for each of the methods. Next, we will present the experimental setup used, in Section V, and the results and limitations of our experiments, in Section VI. Last, we will be discussing the results and draw conclusions (Section VII).

Along with this essay, there is also a video[1] of approximately ten minutes, in which the authors of this essay briefly expose the work that was carried, and a *Colab Notebook* with the code used to produce the datasets[2].

The generated datasets for each model were developed according to the assumptions of each method, though, in some cases, we observed that other models performed as well as the method that we developed the dataset for. In the end, we were able to generate datasets for the whole set of models and interpret why each method performed well for each dataset.

II. Objectives

The main objectives of our work are (a) to show that there is not a best general method for every dataset there is. Indeed, the choice of the best method very much depends on the data points one is given, as we will show in the following sections. The other main objective is (b) to show that, even though some of the methods used can always be approximated by other methods (e.g., a multilayer feedforward network with a non-polynomial activation function can approximate any continuous function - see [3]), the complexity needed to perform such an approximation may not be worthwhile. Thus, the use of the simpler method is preferable since it will be easier to explain the results and the training process will, typically, take less time.

[1] Video
[2] Colab Notebook

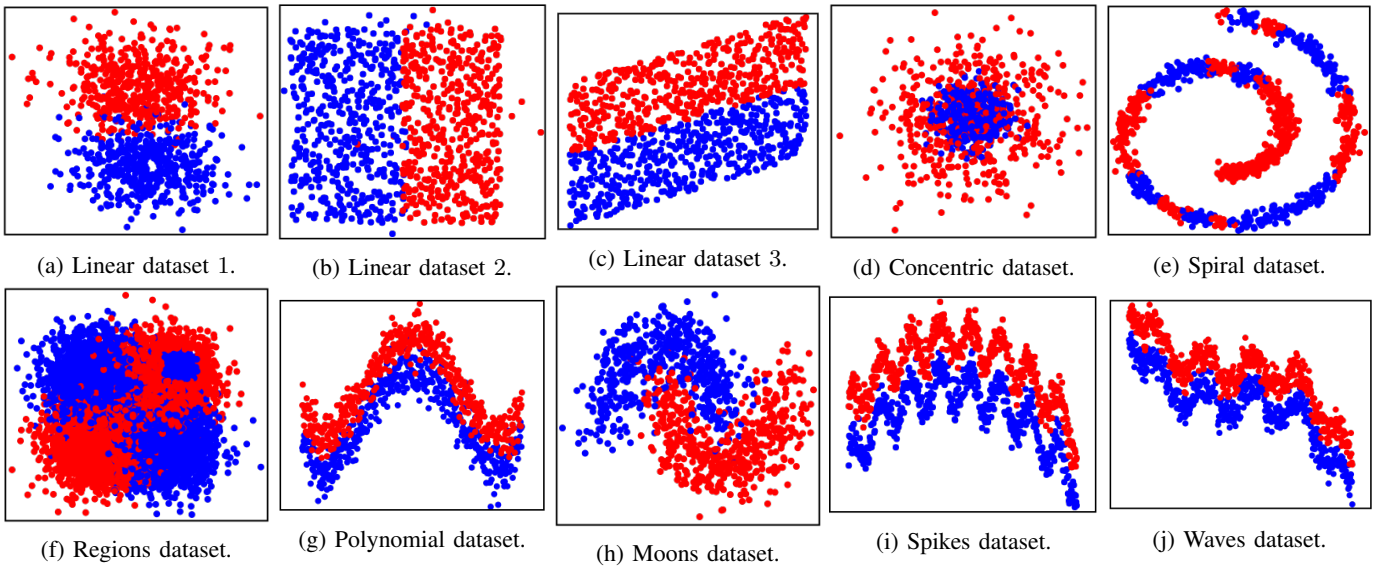
III. Methods

In order to accomplish the objectives proposed in the previous section, we have used ten different methods which are listed in the following paragraph.

The Logistic Regression (*LR*) and the Linear Discriminant Analysis (*LDA*) are both based on a linear assumption. However, the latter also assumes that each class can be modeled by a Gaussian distribution, with all the classes sharing the same covariance matrix. The Quadratic Discriminant Analysis (*QDA*) assumes that the decision boundary is quadratic and, like *LDA*, that all classes follow a Normal distribution. However, in *QDA*, the classes can have different covariance matrices. Decision Trees (*DTs*) were also used, which are a very flexible method, but prone to overfitting, as well as Random Forests (*RFs*), which are an ensemble of *DTs*. In what concerns Support Vector Machines (*SVMs*), we have used three different types of kernels: *linear*, *polynomial* and *radial basis*. Finally, we have used Multilayer Perceptrons (*MLPs*), which can be an extremely powerful method, with two types of activation functions: *ReLU* and *TanH*.

IV. Dataset Generation

All the datasets were artificially generated with the aim of serving the methods presented in the previous section, *i.e.*, for each method we generated a dataset that would allow for the method to have a good performance. For simplicity's sake, and to allow for a simpler explanation of the results, all the datasets are two-dimensional. Moreover, each dataset has two classes with an equal number of observations (each class has five-hundred observations), meaning that the classes are balanced. Figure 1 shows all the datasets that will be used throughout the paper. The names by which we will refer to each dataset will be the ones in the captions below each scatter plot. The two different colours (red and blue) represent different classes in the dataset. To conclude this section, we note that some of the datasets were generated with the ***datasets*** module from the ***scikit-learn*** library.



(a) Linear dataset 1. (b) Linear dataset 2. (c) Linear dataset 3. (d) Concentric dataset. (e) Spiral dataset.

(f) Regions dataset. (g) Polynomial dataset. (h) Moons dataset. (i) Spikes dataset. (j) Waves dataset.

Fig. 1: Visual representation of the datasets.

V. Experimental Setup

Given the datasets generated in Section IV, we will be using the methods described in Section III to fit the data and evaluate their performances. For such purposes, we will be using 5-fold cross-validation in the whole dataset, which is a useful technique for assessing the effectiveness of the models, mitigating overfitting and enabling a good choice of hyper-parameters. Moreover, and even though the *LR*, *LDA* and *QDA* methods do not have hyper-parameters, the performance of the remaining methods depends on a good choice of hyper-parameters. Hence, a grid-search will also be performed. The methods in which the grid search will be applied, and the respective hyper-parameters to tune are the following,

- *DT*: maximum depth; maximum number of features to consider when finding the best split; criterion used in the splits (*gini* or *entropy*); the minimum number of samples to perform a split
- *RF*: maximum number of features to consider when looking for the best split
- *SVMs*: kernel (*linear*, *polynomial* or *radial basis*); regularization parameter (*C*); the $coef_0$ parameter, which controls how much the model is influenced by high-degree polynomials versus low-degree polynomials (only applicable to the *polynomial* kernel, but of much importance)
- *MLPs*: size of the hidden layer; the activation function (*ReLU* or *TanH*); learning rate

Finally, we will be using the *accuracy*, the *F1 measure* and the *area under the curve (AUC)* to measure the performance of each model in each of the test folds, and average the values in order to compare all the models. Besides the three aforementioned measures, we will also consider the *complexity/number of parameters* of the models and their *training times*. Whereas the three former measures account for the performance of the models at the classification task, the latter two assess the simplicity of the models and can be important to select between the best models. For example, if a *LR* model and a *MLP* model provide similar *accuracies*, *F1 measures* and *AUCs*, one may select the *LR* model as the best between the two, since it will (most likely) take less time to fit the data and will have (most presumably) less parameters than the *MLP* model. So, when the performance measures of two models are similar, we will choose, for simplicity's sake, the less complex model.

VI. Results and Limitations

In the previous sections we have laid out the goals of our study (Section II), the methods that will be used (Section III), the generated datasets (Section IV), and the way we will conduct the experiment (Section V), so now we have the necessary conditions to obtain results and draw conclusions.

For each of the datasets we will be presenting the visual representation of the boundaries produced only by the best model, the motives being the high number of methods (which would result in an equal number of boundaries), and the limited amount of space.

A. Linear Discriminant Analysis

LDA is a method based on a linear assumption and on the fact that each class can be modeled by a *Gaussian* distribution, with all classes sharing the same covariance matrix. Hence, we generated the *Linear dataset* 1 (Figure 1a), which is both linearly separable (approximately) and each class was drawn from a *Gaussian* distribution. Figure 2 shows a plot of the decision boundary given by the *LDA* model when applied to the *Linear dataset 1*.

Visually, one can notice that the boundary produced seems to well separate the two classes, but we can measure the performance of the model by using the metrics mentioned

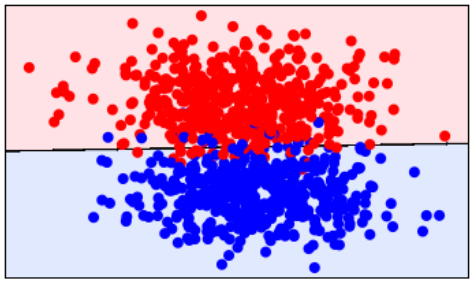


Fig. 2: Decision Boundary of *LDA* for the *Linear dataset 1*.

in Section V. Figure 3 shows two plots. The upper bar plot shows the *mean AUC*, the *mean accuracy* and the *mean F1 measure* averaged across the five test folds for each model. The models are also ordered according to their performance, so that the best performing model is on the left corner and the worst performing model is on the right corner. Moreover, regarding the *SVMs* and *MLPs*, we only show the models with the best *kernel* and *activation function*, respectively. *I.e.*, if, for example, the *SVM* with the *linear kernel* provides better results than the *SVMs* with the *radial basis* and the *polynomial kernels*, we only show the result for the *linear kernel*. The same goes for the *activation function* in the *MLPs*. Note that both the *kernel* and the *activation function* are simply hyper-parameters of the *SVMs* and *MLPs*, respectively. The lower plot shows, for each model, a boxplot constructed using the values of the accuracy averaged across all test folds[3].

Having explained what the plots represent, we can now note that the *LDA*, indeed, offers extremely good results. The plots show that the linear models (*LDA*, *LR* and the *SVM* with the *linear kernel*) give results as good or better than the remaining models. Hence, for simplicity's sake, we will rule out the non-linear models since they have hyper-parameters which need to be tuned by grid search or, in case of the *QDA* have a quadratic boundary which is not the most appropriate for the dataset at hand (besides offering slightly worse results than the linear models). Regarding the *LR*, despite offering similar results to *LDA*, it tends to be less stable since the classes were drawn from a *Gaussian* distribution - the boxplot of the *LR* model shows a slightly higher variance than the one from the *LDA* model. The disadvantage of the *SVM* model with the *linear kernel* when compared to the *LDA* model is the higher training time due to the high number of support vectors found - from Figure 2 we can see that the frontier between the two classes is densely covered by data points.

Overall, the *LDA* method can be regarded as the best one for the *Linear dataset 1*.

B. Logistic Regression

Like *LDA*, *LR* is also based on a linear assumption of the data, though it does not assume that the classes are drawn

[3]This type of plots - both the bar plot and the cross-validation accuracy distribution - will be used throughout the essay, so we will refrain from explaining them again. They will be referred to as *performance plots* when needed.

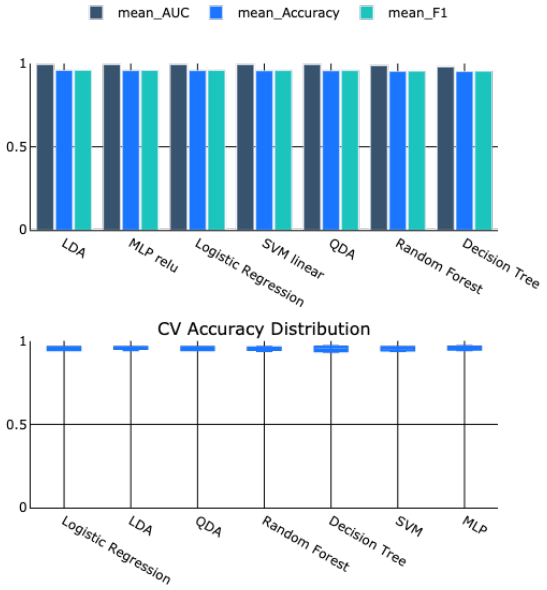


Fig. 3: Performance measures for the *Linear dataset 1*.

from a *Gaussian* distribution. Thus, although *LDA* can offer some improvements over *LR* when the *Gaussian* assumption holds, *LR* tends to offer improvements over *LDA* when the assumption does not hold. With such a thought in mind we generated the *Linear dataset 2* (Figure 1b) which is linearly separable (approximately) and, unlike the *Linear dataset 1* (Figure 1a), its data points were not drawn from a *Gaussian* distribution.

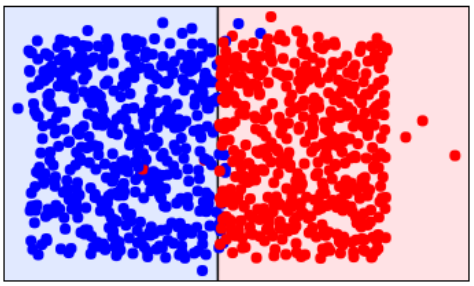


Fig. 4: Decision Boundary of *LR* for the *Linear dataset 2*.

From the previous discussion it was expected that the *LR* method would fit the *Linear dataset 2* well, which is corroborated from the boundary shown in Figure 4. Since we are dealing with a linearly separable dataset, linear methods are the ones to consider, for simplicity's sake. By analysing the performance plots in Figure 5, we can see that the linear models provide extremely good results, the *LR* being the best of the three. In fact, the *LDA* gives the worst result of the three linear models, besides being more unstable than the *LR* model, since the classes were not drawn from *Gaussian* distributions. Regarding the *linear SVM*, besides providing slightly worse results than *LR*, it takes more time to train due to the high number of *support vectors*. Hence, the *LR* can be considered the best method for the *Linear dataset* 2.

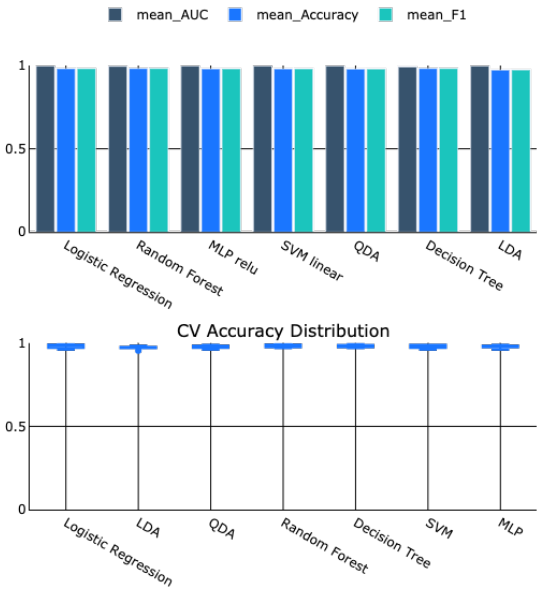


Fig. 5: Performance measures for the *Linear dataset 2*.

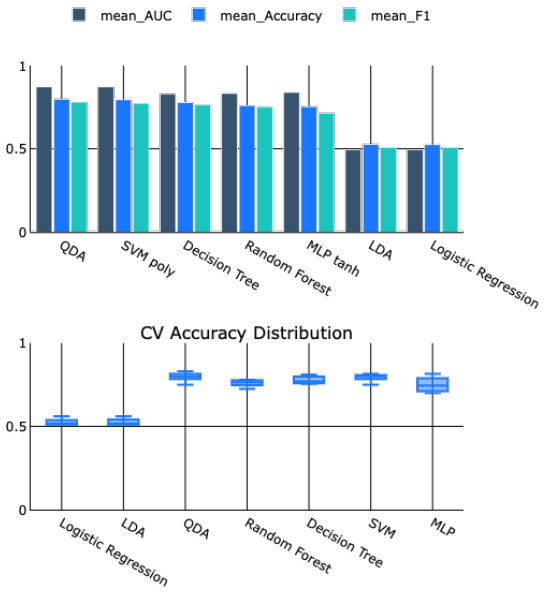


Fig. 7: Performance measures for the *Concentric dataset*.

C. Quadratic Discriminant Analysis

The *QDA* method, as the name suggests, produces a quadratic decision boundary. Like *LDA*, it assumes that the observations are drawn from a *Gaussian* distribution, but each class can have its own covariance matrix. Taking this into account, the *Concentric dataset* (Figure 1d) was artificially created. Each class is drawn from a *Gaussian* distribution and both have different covariance matrices. Figure 6 shows a scatter plot with the decision boundary given by the *QDA* model.

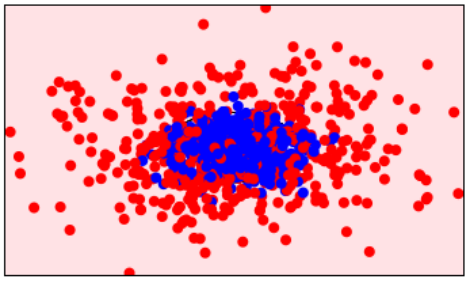


Fig. 6: Decision Boundary of *QDA* for the *Concentric dataset*.

Due to the high density of the blue points it is not easy to see the decision boundary, which encircles them, providing a good separation between the two classes. Next, we analyse the performance measures of all the models. Figure 7 shows the *performance plots*.

As it was expected from the visual representation of the dataset, linear models have a poor performance. The *MLP* model has a high number of parameters and has a slightly worse performance than the remaining models - the boxplot shows a high variance when compared with the other models. The *RF* model provides slightly worse results and has a much longer training time when compared with the remaining models since it needs to train two hundred trees. The *SVM* with the *polynomial* kernel (which is the best kernel for this particular dataset) also takes a long time to fit the data, probably due to the high number of support vectors found. We are left with two models - *QDA* and *DT* - however, due to the need of performing grid search on the *DT* to fine-tune the hyper-parameters, the best model for this particular dataset is the *QDA* due to its simplicity and good classification results.

D. Decision Trees

DTs work by, at each iteration, splitting the data according to a given feature. They can be seen as a set of rules which, given a new data point, will classify it according to the rules followed by that particular observation. Due to this functioning of the *DTs*, they divide the data into different regions. Hence, having this idea in mind, we created the *Regions dataset* (Figure 1f). Figure 8 shows the decision boundary given by the *DT* algorithm which, as expected, can well divide both classes. Even though it is not possible to see, due to the high density of points in Figure 8, the small *blue* cluster in the upper right has its own region apart from the surrounding *red* class.

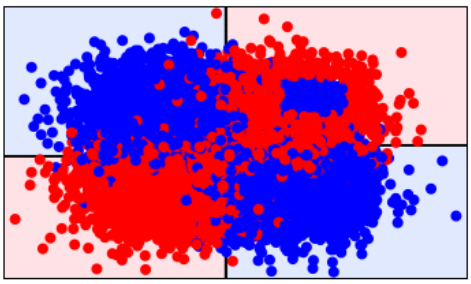


Fig. 8: Decision Boundary of *DT* for the *Regions dataset*.

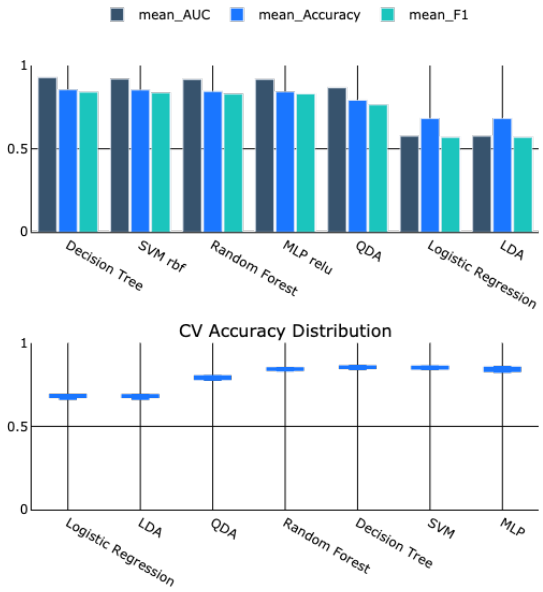


Fig. 9: Performance measures for the *Regions dataset*.

The analysis of the *performance plots* in Figure 9 show the good results obtained by the decision trees. As expected from the visual representation of the dataset, the *linear models* and the *QDA* provided worse results than the remaining models since the data is not linearly- nor quadratically-separable. The *MLP* also has slightly worse results, besides its higher variance, fitting time and number of parameters. The *RF* model provides similar results (although slightly worse) than those of the *DT* and it has the disadvantage of taking longer to fit the data since it is composed by two hundred trees, where each one needs to be trained. The *SVM* with the *radial basis kernel* takes a long time to fit the data, which is due to the high number of support vectors. Thus, we can safely conclude that the *DT* method is the better one for the *Regions dataset.*

E. Random Forests

RFs are an ensemble approach that use *decision trees*. The main idea behind this method is to learn K trees from a bootstrap sample of the original dataset, each using m randomly chosen variables.

In order to take advantage of the fact that the final model is composed by an ensemble of *decision trees*, and, in our case, with just one variable in each tree, we developed the *Spiral dataset* represented in Figure 1e. Although the dataset seems to be composed by complex regions around the formed spiral, the different regions were formed by simply applying thresholds in both axes. This way, we can explore the fact that the *RFs'* trees can find different regions from the other models and, since each one only explores one axis, some trees will find thresholds in the x-axis and others in the y-axis. Hence, the combination of the trees can find the different regions accurately.

Indeed, the *performance plots* shown in Figure 11 further corroborate the explanation given previously. Obviously, the

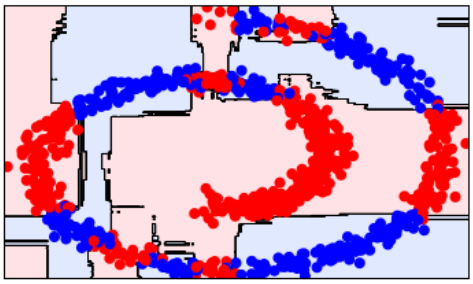


Fig. 10: Decision Boundary of *RF* for the *Spiral dataset*.

linear models and the *QDA* cannot offer good results in a non-linearly and non-quadratically separable dataset. The use of just one *DT* provides worse results than the *RF* model. The *MLP* model has a higher variance and takes more time to fit the data when compared to the *RF* model. Lastly, the *SVM* with the *radial basis kernel* provides good results, but its boxplot of the cross-validation accuracies shows slightly worse results than those of the *RF* model. Besides, the training time of the *SVM* is slightly higher due to great amount of support vectors found. Overall, the *RF* model can be regarded as the best model for the *Spiral dataset*.

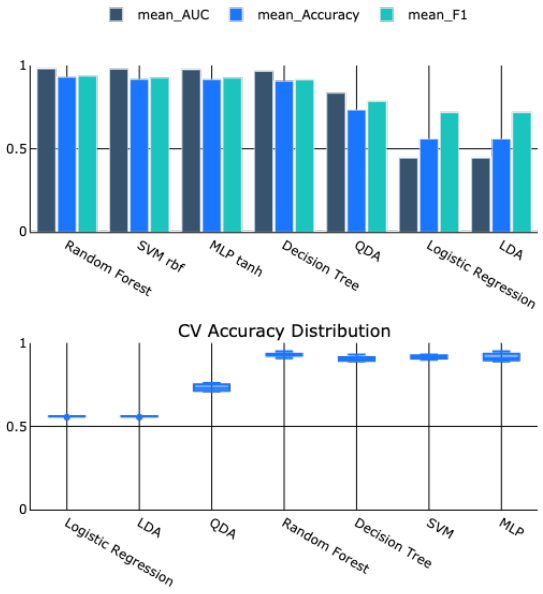


Fig. 11: Performance measures for the *Spiral dataset*.

F. Support Vector Machines

The way *SVMs* work is by projecting the data into a high dimensional space using *kernel functions*. The most common ones are, besides the *linear kernel*, the *radial basis function (RBF)*, and the *polynomial kernel*, which will be the ones used. In that new space, *SVMs* find a linear boundary, which might not be (and frequently isn't) linear in the original space, allowing for irregular boundaries in the original space.

1) Linear kernel: The *linear kernel* is used, as the name suggests, when the data is linearly separable, and training a

SVM with a *linear kernel* is faster than with any other kernel. Moreover, the smaller the number of *support vectors*, the faster the training process will be. Taking this into consideration, we have created the *Linear dataset* 3 (Figure 1c), which was built by placing a class of points to the left of the line with equation $y = x+0.01$, another to the right of the line with equation $y = x$, and a small number of points in both lines (those will be the *support vectors*). The idea is that, since the data is linearly separable and there is a small number of *support vectors*, the *SVM* with the *linear kernel* ought to yield good results. Indeed, judging by Figure 12, we can see that the boundary produced by the *linear SVM* separates the data extremely well. In fact, if we analyse the metrics and the boxplots given in Figure 13, we can see that the *linear SVM* shows the best results.

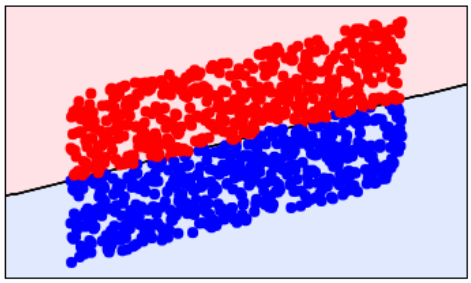


Fig. 12: Decision Boundary of the *SVM* with a *linear kernel* for the *Linear dataset* 3.

Since the data is linearly separable, there is no need to use *non-linear models* (for simplicity's sake). Thus, we are only comparing the *linear SVM*, the *LDA* and the *LR*. But as we can see from the *LR's* boxplot, the variance is quite high when compared to the ones of the *LDA* and *linear SVM*, so it can be ruled out. The *LDA* model seems extremely good, but since the *linear SVM* has few *supporting vectors*, implying a fast training time, and presents slightly better results for the three metrics used, we find the *SVM* with the *linear kernel* to be the best method for the *Linear dataset 3*.

2) Polynomial kernel: The *SVMs* with a *polynomial kernel*, as the name suggests, represent the data in a *feature space* over polynomials of the original variables, allowing for the obtainment of non-linear boundaries. With such a thought in mind, we created the *Polynomial dataset* (Figure 1g), which made use of a polynomial function to place the data points in a two-dimensional space. Thus, both classes can be well separated by a polynomial, allowing the *SVM* with a *polynomial kernel* to perform a good separation of the classes, as Figure 14 shows.

Moreover, by analysing Figure 15, we can see, from both plots, that the *SVM* with the *polynomial kernel* provides better results than the other models.

In fact, given that the classes are non-linearly separable, the *LR* and *LDA* models could not perform well in this dataset. The boundary is also not quadratic, so the *QDA* model offers poor results too. Dividing the dataset into regions doesn't (visually) seem a good option, which is further corroborated by the not-so-good results of both the *RF* and the *DT* models,

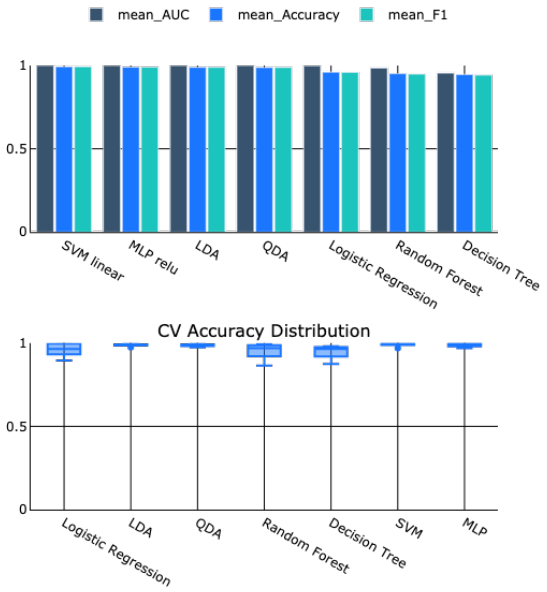


Fig. 13: Performance measures for the *Linear dataset* 3.

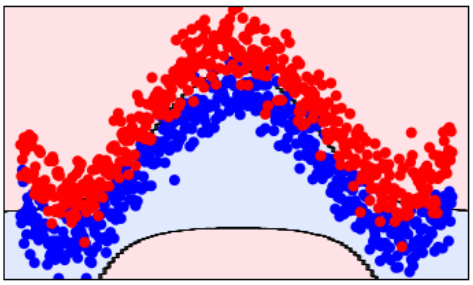


Fig. 14: Decision Boundary of the *SVM* with a *polynomial kernel* for the *Polynomial dataset*.

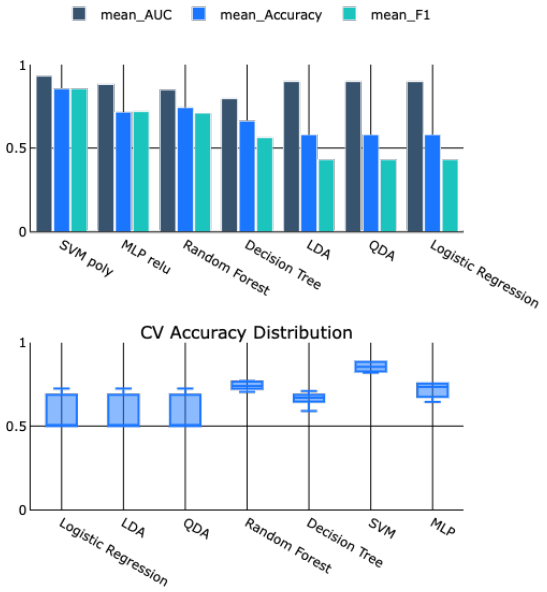


Fig. 15: Performance measures for the *Polynomial dataset*.

even though the *RF* model provides better results than the *DT* model. Regarding the *MLP* model, it also offers worse results than the *SVM* model with the *polynomial kernel*. In fact, the *MLPs* can approximate such a boundary as well as the *polynomial SVM*, but the number of parameters would be very high.

The previous discussion allows to state that the *SVMs* with a *polynomial kernel* are the better method for the *Spiral dataset*.

3) Radial Basis kernel: The *SVMs* with a *RBF* kernel work in a similar way to the ones with a *Polynomial kernel*, the only difference being the projection in the *feature space*, which makes use of the *radial basis function* instead of a *polynomial function*. The *RBF* kernel is very versatile and can be used to find irregular boundaries. Moreover, if the number of support vectors is low, the *SVMs* fit the data fast. Armed with this knowledge, we produced the *Moons dataset* (Figure 1h), which has an irregular boundary and, when fitted by a *SVM* with a *RBF* kernel, returns a small number of support vectors. Figure 16 shows the boundary produced by the *SVM* with a *RBF kernel* in the *Moons dataset*.

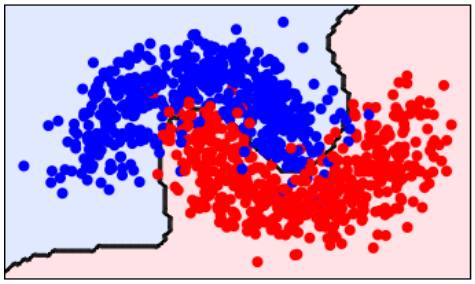


Fig. 16: Decision Boundary of the *SVM* with a *radial basis kernel* for the *Moons dataset*.

As shown in Figure. 16, the *SVM* with the *RBF kernel* can separate both classes very accurately. Moreover, by analysing Figure. 17, we can see, from both plots, that the *RBF SVM* provides better results than the other models.

As expected, the *LR* and the *LDA* models provide slightly worse results than all the others since the classes are not fully separable by means of a linear boundary. A similar argument can be used for the *QDA* model, the difference being the shape of the boundary, which is quadratic instead of linear. Once again, the *MLP* model takes longer to fit the data due to the high number of parameters, and the obtained results are not better than those of the *SVM* model. Regarding the *DT* model, although it provides good results, it has a higher variance than the *SVM* model, despite their similar training time. The *RF* model can slightly decrease the variance of the *DT* model, but at the expense of a much higher training time.

The discussion conducted previously allows to conclude that the *SVM* model with the *RBF kernel* is the most adequate for the *Moons dataset*. Besides its good performance, the training time is also very small due to the low number of support vectors found.

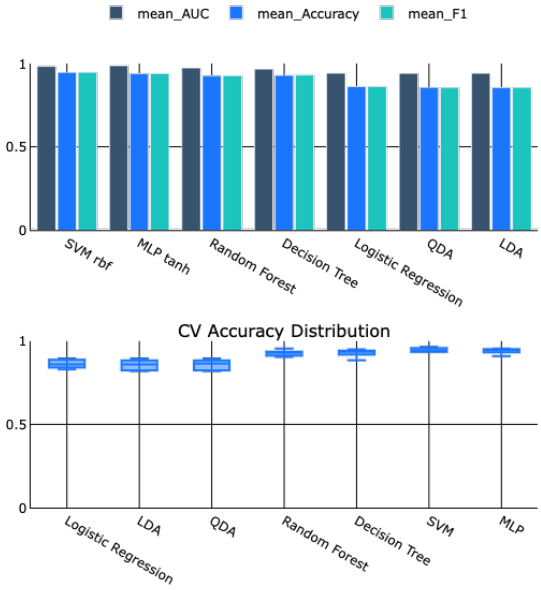


Fig. 17: Performance measures for the *Moons dataset*.

G. Multi-Layer Perceptrons

Multilayer perceptrons are *artificial neural networks* composed by multiple *perceptrons* side by side in each layer. The network composed by this method is a *feedforward network*, since values are only processed from left to right. The *MLP* is a very expressive method, that can easily adapt to irregular boundaries, something that we used in order to create the datasets for this method. Another important property of this *ANN* is the activation function. In our work, we decided to use both the *hyperbolic tangent* and the *ReLU* activation functions.

1) ReLU activation function: The *ReLU* activation function, $f(x) = max(0, x)$ is surprisingly effective at approximating non-linear boundaries, computationally simple and behaves like a linear activation function, something that leads to an easier optimization.

In order to take advantage of the properties stated before, we built the *Spikes dataset*, presented in Figure 1i. The dataset is just a combination of a sine function with a second degree polynomial, this way we could create an irregular frontier that wasn't easily separable by a linear boundary.

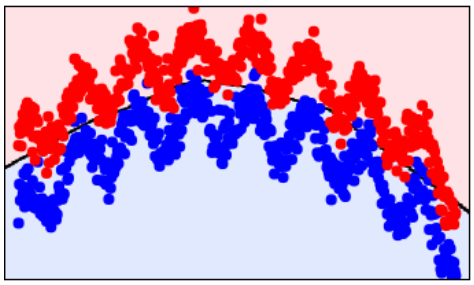


Fig. 18: Decision Boundary of the *MLP* with the *ReLU activation function* for the *Spikes dataset*.

Figure 18 shows that the *MLP* with the *ReLU* activation function is able to separate both classes, so it deals well with the irregular boundary. Besides that, in Figure 19 it is possible to see that the results obtained are significantly better than the ones obtained by the other models, which justifies the time spent at fitting the data. One thing to acknowledge is the fact that we are not tuning some important parameters like the batch size, because, if we did, the results would be even better.

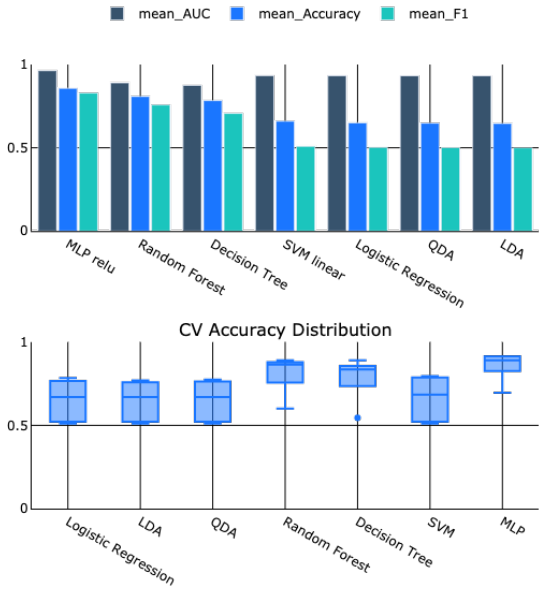


Fig. 19: Performance measures for the *Spikes dataset*.

2) TanH activation function: The *hyperbolic tangent* activation function has a similar shape to the logistic activation function, though it has the advantage that it ranges between -1 and 1, and it has more expressive gradients. The *Waves dataset* (Figure 1j) was developed with the *MLP* with the hyperbolic tangent activation function in mind. The idea is to have an irregular boundary with a quicker variation between class 0 and 1. We were able to achieve this by combining a third degree polynomial with a sine wave multiplied by a squared wave.

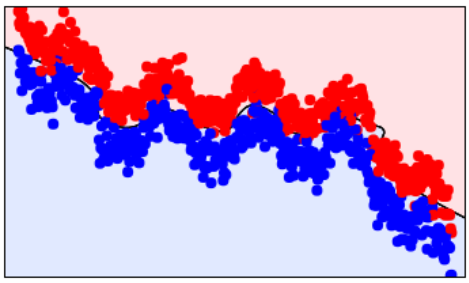


Fig. 20: Decision Boundary of the *MLP* with the *Tanh activation function* for the *Waves dataset*.

The boundary produced by the *MLP* with *tanh* activation function seems to divide very well both classes, which shows that the *tanh* is good to approximate non-linear boundaries with a high gradient.

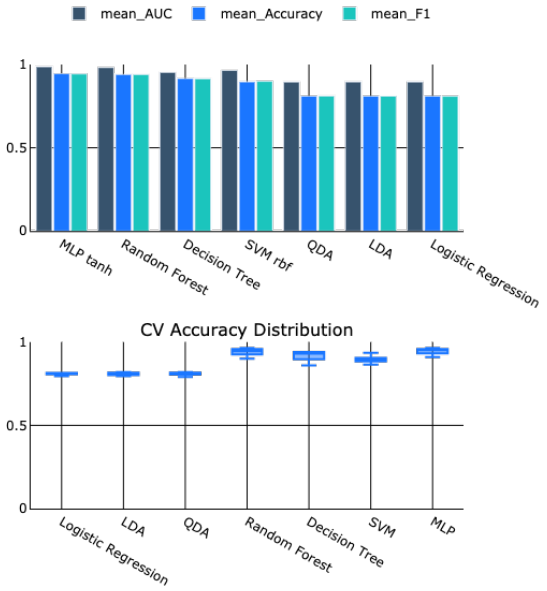


Fig. 21: Performance measures for the *Waves dataset*.

The results seen in Figure 21 show that the *MLP* was the model that obtained the best metrics, though the random forest got almost the same performance and the fit time was a lot smaller. With this, the best model could be considered to be the random forest, but considering that the *MLP* wasn't very well tuned (for example the batch size is too small) its performance could be considerably better than the one obtained by the random forest.

VII. Discussion and Conclusion

Data scientists are used to, when provided with a dataset, find the method that can better separate the classes present in the data. The work carried during the formulation of this paper allowed us to think of the opposite problem, *i.e.*, instead of thinking "Which method can better fit my dataset?", we had to think "Which dataset will better fit my method?". This enabled us to better understand how the different methods work, which can be of great help when in the need of choosing a model, given a dataset. Overall, diving into the world of two-dimensional, binary-classified data gave us a useful insight into the methods used by showing us their limitations and strengths.

References

[1] Wolpert, D.H. and Macready, W.G., 1997. No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), pp.67-82.

[2] Wolpert, D.H. and Macready, W.G., 1995. No free lunch theorems for search (Vol. 10). Technical Report SFI-TR-95-02-010, Santa Fe Institute. Vancouver.

[3] Leshno, M., Lin, V.Y., Pinkus, A. and Schocken, S., 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural networks, 6(6), pp.861-867. Vancouver.

[4] James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017.

[5] https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/