# Comparison of Machine Learning models using artificial datasets Machine Learning - Assignment 2

Ana Raquel Maceiras
*Master in BBC*
*Faculdade de Ciências*
*Universidade do Porto*
Porto, Portugal
up200604342@edu.fc.up.pt

Hélder Vieira
*Master in DS*
*Faculdade de Ciências*
*Universidade do Porto*
Porto, Portugal
up201503395@edu.fc.up.pt

Miguel Tavares
*Master in DS*
*Faculdade de Ciências*
*Universidade do Porto*
Porto, Portugal
up200902937@edu.fc.up.pt

Rui Vieira
*Master in BBC*
*Faculdade de Ciências*
*Universidade do Porto*
Porto, Portugal
up201403035@edu.fc.up.pt

***Abstract*—There are many methods in supervised machine learning for finding the best approximation to the unknown function that defines the data. A given method may outperform others in some specific scenarios. In this assignment we will design several datasets aiming to create a performance discrepancy where a certain method or a subset of methods from the pool produce significantly better results than the others. When possible hyperparameter tuning is used in order to select the optimal hyperparameter values. The experimental setup along with the results and their respective discussion will be further presented in this report.**

***Index Terms*—machine learning, model selection, artificial data sets**

## I. Introduction

Data availability has been exponentially increasing worldwide, compelling institutions and corporations to implement effective data management protocols as to exploit them and propel novel breakthroughs and the improvement of existing technologies or services. In such purpose's view, machine learning is a fast-growing field of study that provides means for the development of systematic methodologies to data handling and data-driven decision making, ever more prevalent *e.g.* in the biomedical and customer services sectors.

A diverse set of machine learning algorithms is at the public disposal, each posing advantages and disadvantages depending on the task and available data at hand. Therefore, understanding the basic principles behind them and their pitfalls is paramount to make a grounded selection of a suitable model for a given problem.

Sample datasets act as proxies to reality. There is the assumption of being a reliable representation of a given phenomenon, governed by underlying periodic patterns, so that predictions can be inferred from them, in a process called learning. Provided that real world datasets are coupled with some degree of randomness or noise [1], outliers, sparse and missing data, each prediction is associated to an error, which is fractioned into irreducible error, bias, and variance [2]; an error not only introduced by the dataset itself (noise, outliers, size), but also the principles behind each model (complexity and parameter numerosity) [3]. To produce good results on a task, an estimator must be able to make generalized predictions of reality from a sample as, that is, a compromise between bias and variance must be had. By common practice, estimator performance assessment is done *via* error monitoring given by defined loss functions, such as the residual sum of squares (RSS), cross-entropy and the 0-1 loss function, or *via* scoring metrics: Accuracy, F1-score, Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) score. Thusly, the suitableness of a candidate model for a given dataset is reflected by these metrics as well as training times, computational requirements, and model complexity.

In view of acquiring an intuition on what models are the most adequate for a given binary classification, machine learning problem, ten classifiers: Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Decision Tree, Random Forest, Support Vector Machines with linear, polynomial, and radial basis kernels, Multi-Layer Perceptron with ReLu and hyperbolic tangent activation functions, were trained and tested on several bidimensional artificial datasets of different topologies, sizes and noise degree, designed to create a performance discrepancy where a certain method or a subset of methods from the pool produce significantly better results than the others. When possible, hyperparameter tuning is used in order to select the optimal hyperparameter values. The experimental setup along

Link for the group's presentation: https://tinyurl.com/5kfk23z2

with the results and their respective discussion will be further presented in this report.

## II. Objective

The main objective of this work is to design datasets, where a certain method or subset of methods performs better than the others in a given pool of classification methods, and trying to justify this performance gap, assessing this difference using several evaluation metrics and comparison criteria.

## III. Methods and hyper parameters

Table I summarizes the methods that constitute the pool and the respective hyperparameters that were optimized.

TABLE I
MODELS USED AND CORRESPONDENT HYPERPARAMETERS SUBJECT TO TUNING

| Methods | Predefined Parameters | Tuned Parameters |
|---|---|---|
| Logistic Regression | NA | None |
| Linear Discriminant Analysis (LDA) | NA | None |
| Quadratic Discriminant Analysis (QDA) | NA | None |
| Random Forest | NA | criterion<br>min_samples_split<br>min_samples_leaf<br>min_impurity_decrease |
| Decision Trees | NA | min_samples_leaf<br>min_samples_split<br>max_depth |
| Support Vector Machines | linear<br>rbf<br>poly | C |
| Multi-Layer Perceptron | tanh<br>ReLu | hidden_layer_sizes<br>solver<br>alpha<br>learning_rate<br>batch_size |

NA - Not Applicable

## IV. Dataset generation

Datasets used in this project were artificially generated and specifically designed to create specific scenarios where, keeping in mind the models' strengths and limitations, some method(s) would perform considerably better than the rest. Dataset generation was achieved mainly using functions from the scikit-learn library (sklearn.datasets package [4]). Dataset ***c*** was obtained from scikit learn documentation [5]

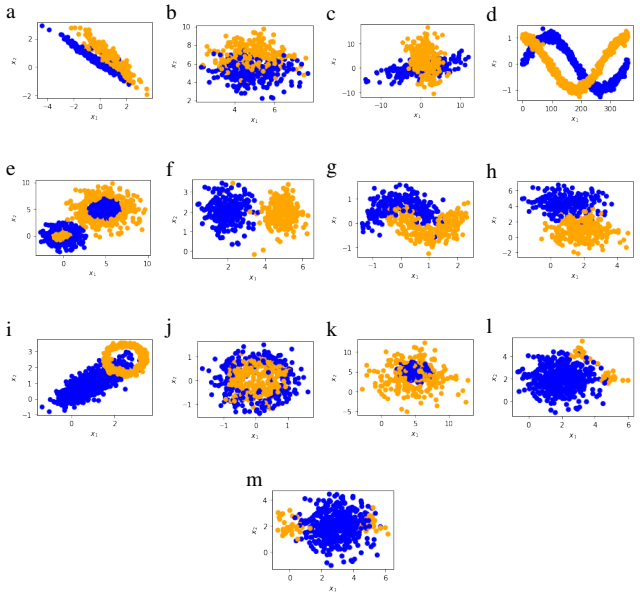


Fig. 1. Datasets used in this study

## V. Experimental setup

With the several datasets generated for this purpose, to each one of them, all the methods enumerated in table I will be applied. In this process, a grid search (with cross validation included) was used to optimize the models that require hyperparameter tuning, with F1 score as selection metric.

As soon as the hyperparameters were chosen, train-test split was applied (holdout 20%) and with the model fitted, accuracy, F1-score and ROC-AUC (receiver operating characteristic curve - area under the curve) were computed and analyzed. Additionally, a plot of the decision boundary was obtained to allow a better assessment of the model performance. Lastly, all the training times were compared.

## VI. Results and limitations

In this section will be presented some of the results of each method, where it has better and worst behavior. For each case, the decision boundary will be plotted and the result metrics as *accuracy*, *AUC* and *F1* will be shown, as well as the time required for training the model (training time), in graphical format for better comparison.

### A. Logistic Regression

Logistic regression is an extension of the linear regression for dichotomous classification problem, and it works under the assumption that there is a linear decision boundary separating two classes. It can be considered a simple model and, consequently, has the advantage of not requiring hyperparameter tuning and having low training time. Moreover, compared to LDA, logitic regression does not require data to follow a specific distribution. Hence, for dataset ***a*** (Fig. 2 left), where the two classes have a linear separation, logistic regression is the

best model as it founds a better decision boundary and higher accuracy, F1-score and ROC-AUC, than LDA and QDA (the other two models that do not require hyperparameter tuning). Furthermore, the other models that obtain values as good as logistic regression do require hyperparameter tuning and/or have higher training times (SVM with linear kernel, and MLP with hyperbolic tangent activation function). Nevertheless, as observed for dataset ***j***, logistic regression does not perform well when the two classes do not have a linear separation (Fig. 2 right).

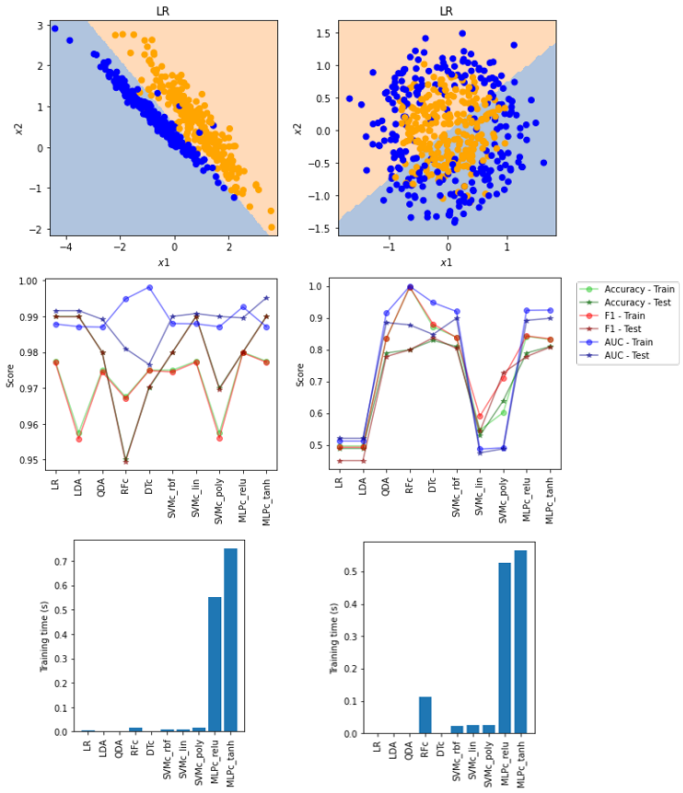


Fig. 2. Logistic Regression performance for datasets ***a*** (left) and ***j*** (right). Logistic regression decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

### *B. Linear Discriminant Analysis*

LDA is another model the takes the assumption that there is a linear decision boundary separating the classes. Moreover, it also does not require hyperparameter tuning and has low training times associated. However, in opposition to linear regression, LDA takes a second assumption that classes follow a normal distribution: LDA predicts a Gaussian density function associated to each class and the linear boundary is found on the intersection. As a consequence, this model is more affected to the presence of outliers. Taking this into account, LDA will perform best in dataset such as dataset ***b***, where there is a linear boundary separating the classes, obtaining high accuracy, F1-score and ROC-AUC, without requiring hyperparameter tuning (Fig. 3 left). Once again, as in the case of logistic regression, LDA will not be able to correctly classify datasets that do not present linear boundaries, as it is the case of dataset ***k*** (Fig. 3 right).

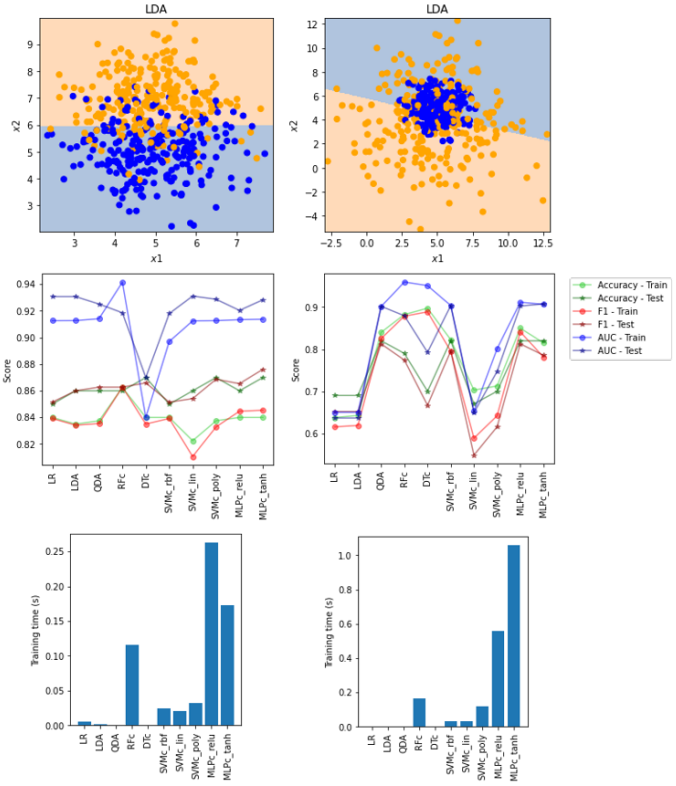


Fig. 3. LDA performance for datasets ***b*** (left) and ***k*** (right). LDA decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

### *C. Quadratic Discriminant Analysis*

Quadratic discriminant analysis assumes, as LDA, that the classes follow a normal distribution. However, while LDA assumes equality of covariances for all classes in all features, this equality is not assumed in the case of QDA, which makes QDA a more flexible model with quadratic decision boundaries. Thus, in datasets where these two characteristics are not met (linear boundary and equality of covariances) QDA will perform better than LDA. It is the case of ***c***, as it can be observed in Fig. 4 left. Nevertheless, in more complex datasets, as it is the case of dataset ***e***, QDA will not be able to predict a decision boundary that fits the data (Fig. 4 right).

### *D. Random Forest*

Random Forests is a powerful algorithm which creates many uncorrelated trees and, consequently, uncorrelated predictions as well. For that, it uses Bootstrap Aggregation (Bagging), that is, each tree is built on randomly sample data from the original training data with replacement that results in different trees. Since it does not assume any particular geometry and each of the trees will have their errors covered by the other trees, an above average performance is expected from it. Thus,

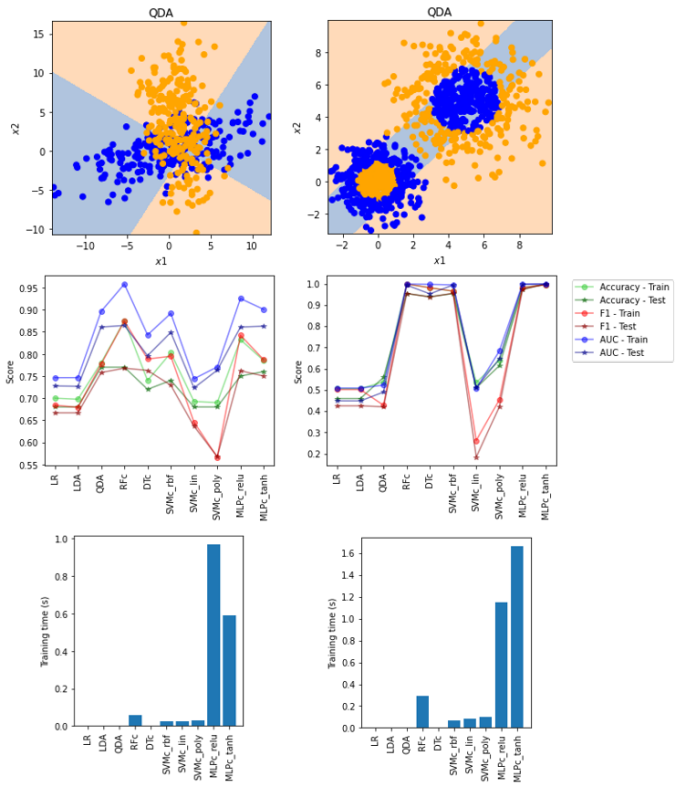


Fig. 4. QDA performance for datasets ***c*** (left) and ***e*** (right). QDA decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

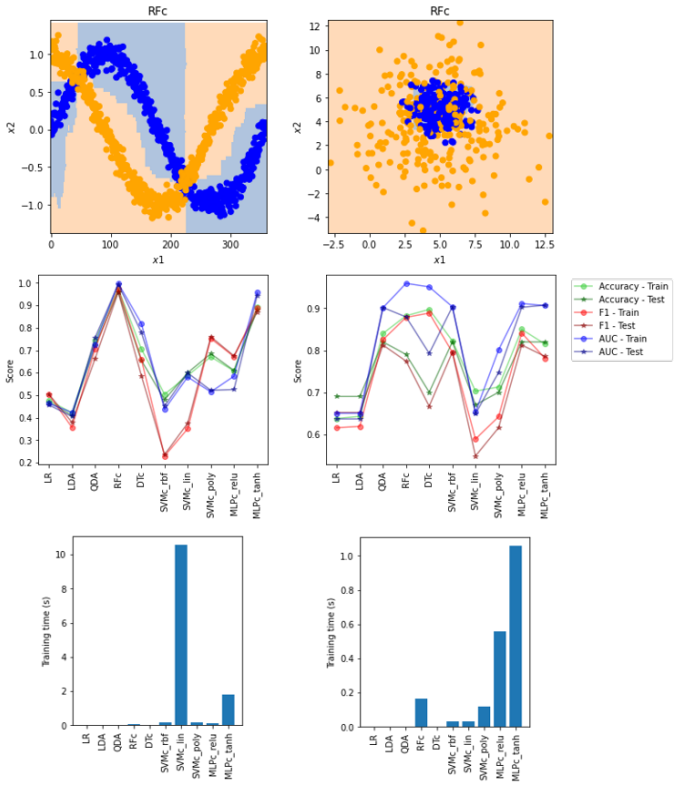


Fig. 5. Random Forest performance for datasets ***d*** (left) and ***k*** (right). Random Forest decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

Random Forests are very adaptable and capable of modelling very intricate pattern as it is the case of dataset ***d*** (Fig. 5 left). Indeed, for this dataset, Random Forest is the model presenting better results, and the only model competing is MLP with hyperbolic tangent activation function, which present a much higher training time. The main disadvantages of this model are the requirement of hyperparameter tuning, the tendency to overfit and slightly higher training time in comparison with the previous described models. A case of overfitting can be observed for the dataset ***k*** (Fig. 5 right), where this model performs better in train than test data.

### *E. Decision Trees*

Decision Trees subdivide the input space into regions, where the function is constant and we try to minimize the entropy. Unlike Random Forests, Decision Trees are much more dependent on feature importance, which can lead to errors in some situations. Moreover, Decision Trees are even more prone to overfit than Random Forest, being hyperparameter tuning of special importance to prevent it. However, in most of simple cases, its reduced training time is a plus to be taken into account in comparison with Random Forests and other complex models. It is the case of dataset ***e*** (Fig. 6 left) where Decision Trees is the model that presents smaller training time while having scores as good as Random Forest, SVM with radial basis kernel, and MLP (with either hyperbolic tangent or rectified linear unit activation functions). On the other hand, for dataset ***l***, Decision Tree classifier occurs in overfitting, specially visible for the F1-score (Fig. 6 right).

### *F. Support Vector Machines*

Applicable both in supervised regression and classification problems, Support Vector Machines (SVM) are a flexible class of algorithms with robust mathematical foundations, providing a wide number of different solutions. A non-linear mapping, known as the "kernel" trick, is applied to transform the input data into a higher dimensional space wherein a hyperplane optimally separating the classes from one another is searched for. The optimal solution for the posing optimization problem converges to the hyperplane with the largest margin in between classes, *i.e.*, the *maximum marginal hyperplane* (MMH). SVM do so by relying on the use of margins and prioritization of the information carried by a select few examples, during training. Such examples, the support vectors, are then employed for prediction. By dispensing with the remaining examples, support vectors provide good generalization power whilst rendering SVMs less prone to data overfit and robust against outliers. Performance is nonetheless dependent on the softness parameter C [6].

*1) Kernel: linear:* The decision boundary delineated by the linear kernel is that maximizes the distance from the closest

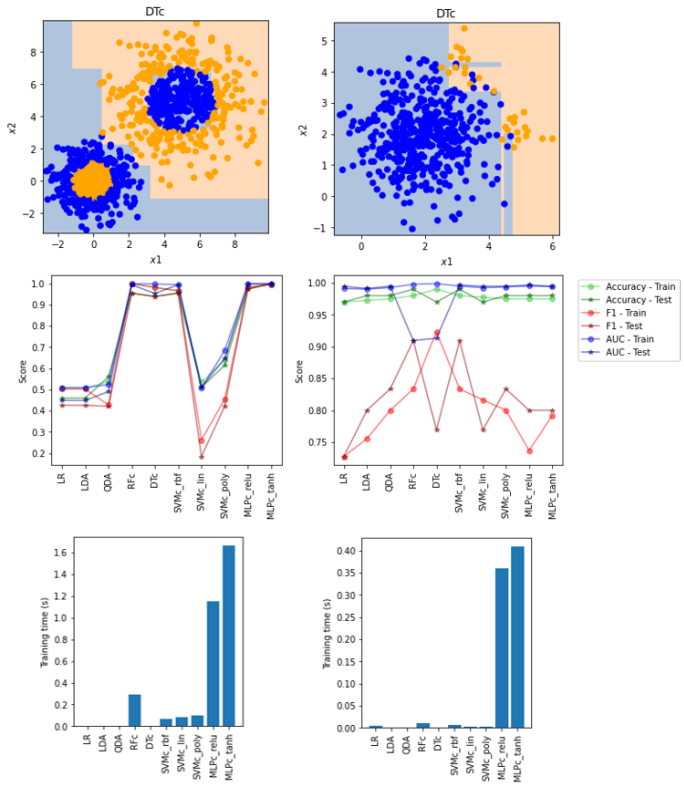


Fig. 6. Decision Tree performance for datasets ***e*** (left) and ***l*** (right). Decision Tree decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

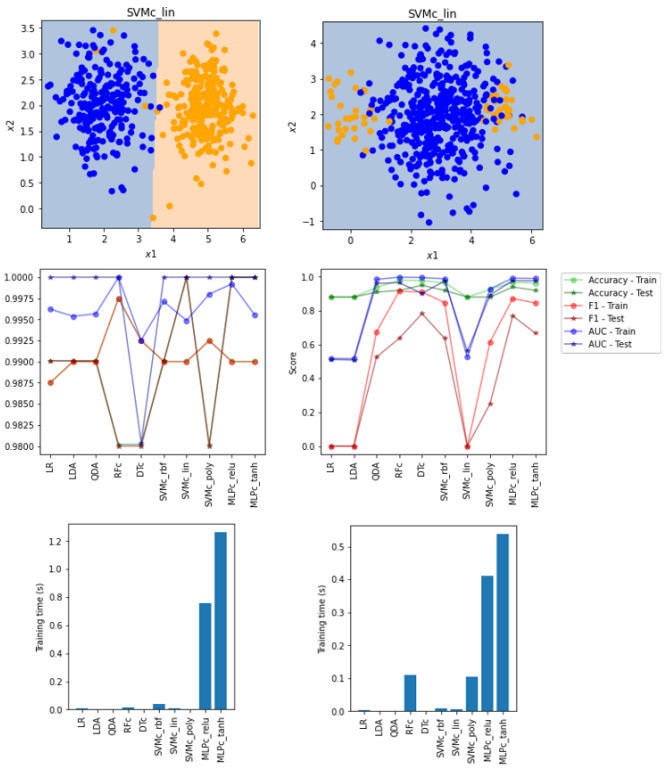


Fig. 7. SVM with Linear Kernel performance for datasets ***f*** (left) and ***m*** (right). SVM with Linear Kernel decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

datapoint(s) of each class to the bidimentional plane or line (dependent on C). Though the principle foundations differ, should linear separation be observable in a given dataset, similarly to other decision boundary models (LR and LDA), good performance scores are to be expected. Moreover, in datasets with outliers, SVM is expected to have a better performance than simpler models like logistic regression and LDA. This is evident in Fig. 2 left and Fig. 7 left, where it outperforms the remaining estimators, chiefly by its speed and test scores. The linear kernel SVM falls short however to LR and LDA, due to slower training phases and model complexity. Furthermore, it under performs when presented with the remaining datasets, which are characterized by substantial intersection among classes, as illustrated in Fig. 7 right. In this very example, regards the whole dataset altogether as a single cluster, not drawing any decision boundary w.r.t to the orange class whatsoever. Of peculiar interest, it should be noted that although its accuracy shows only a short decline from all other estimator accuracies, the f1-score declined to zero – as for disregarding a complete class of examples.

*2) Kernel: radial basis:* The effectiveness of kernel functions stands out when classes are no longer linearly separable. As a supporting claim, radial basis function SVM is on par with the competing expressive estimators, DecisionTrees, RandomForest and Multi-Layer Perceptrons, should it be challenged with sufficiently discriminated, tough non linearly, classes (Fig. 2 right; Fig. 4 right). Owing to its accuracy and f1-score, it outperforms the entirety of the estimator pool in (Fig. 8 left) and dataset ***l***. It is argued further that opting for the radial basis kernel SVM in detriment of both MLP-tanh and -relu would be most effective for less computational power-demanding and faster applications to class intersection of higher degree problems (Fig. 11 left). Notwithstanding the robust generalization power of the support vectors achieved in previously stated problems, it's worth noticing the disruptive impairment on SVM-rbf prediction accomplished by the Sin(x) vs Cos(x) illustration (Fig. 8 right). Periodic class intersection has exposed a shortcoming of SVM with radial kernel, for it fails to successfully delineate an adequate decision boundary on the original input space.

*3) Kernel: polynomial:* Lastly, the polynomial kernel offers some more flexibility when compared with its linear surrogate. However, gains from this flexibility are only exhibited only in a select few datasets.

Thus far, a tailored example wherein the polynomial kernel is most suitable was not determined. Nevertheless, its best performance is observed on the Sin(x) vs Cos(x) of size 500 problem. This is argued to be given to the polynomial natural of sine and cosine functions. Prediction power however is decreased for smaller and larger dataset versions of this same problem (data not shown).

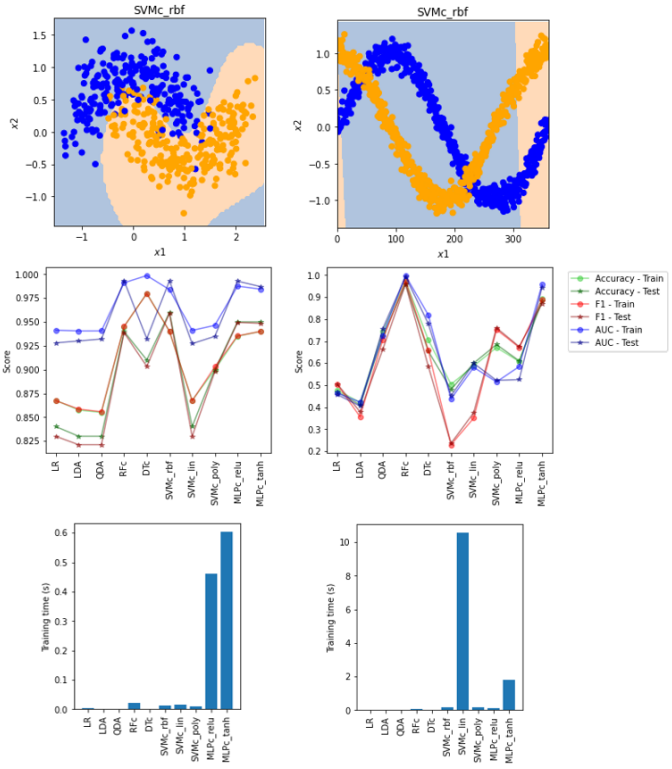


Fig. 8. SVM with Radial Kernel performance for datasets ***g*** (left) and ***d*** (right). SVM with Radial Kernel decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

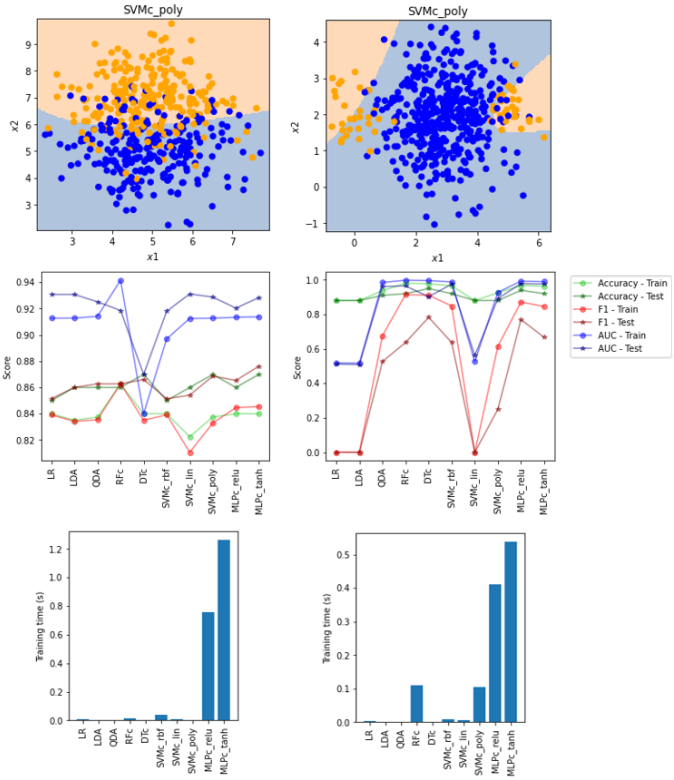


Fig. 9. SVM with Polynomial Kernel performance for datasets ***h*** (left) and ***b*** (right). SVM with Polynomial Kernel decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

### *G. Multilayer Perceptron*

The Multilayer Perceptron Neural Network is a powerful algorithm that consists in creating a network of neurons, with one or more hidden layers, to solve complex problems. Each neuron will have an associated weight, that can be compared as a coefficient of a linear regression.

Each MLP has an activation function, which will map the summed weighted inputs. Normally, the activation functions is a non-linear function in order to combine the inputs into more complex ways and provide a more robust and flexible model.

To train the network, the most commonly used algorithm is the stochastic gradient descent. With this, an input is provided to the network, and through the neurons, an output will be produced. Then it is compared with the expected value and an error is determined. Using backpropagation, the weights of each neurons are updated. This process is repeated for all datasets which consist in one cycle, or epoch. The network can be trained for several epochs.

The hyperparameter optimization carried through by grid-searchCV has led both feedforward networks to adopt 4-layered topologies, assembled by 50 units each layer, save for the output layer (For further information, please find the attached .ipynb file). Having in consideration solely the connecting weights between the input layer - say 500 units - and subsequent hidden layer, the algorithm is encumbered with at least $50^{500}$ parameters. Thus, a staggering discrepancy in training times of MLPc-relu and MLPc-tanh when confronted with other simpler models was expected and observed throughout the datasets in exposition, due to the higher level of complexity of such models. In turn, their expressiveness was proven advantageous in a few intricate situations of intersected clusters, as indicated next.

*1) Activation: hyperbolic tangent function:* The hyperbolic tangent activation function is very similar to the sigmoid function (map all inputs between 0 and 1). However, with *tanh* the range is between -1 and 1, mapping true negative inputs as negatives and the zeros will be mapped near zero.

As far as scores were concerned, the MLP-tanh performed best as exampled in Fig. 10 left, where blob clusters were slightly meshed, contributing however with just a short incremental improvement of *ca* 1.0% across the accuracy and f1-scores.

*2) Activation: rectified linear unit function:* The rectified linear unit activation function (*ReLU*) has the range from 0 to infinity. Therefore, each negative input will automatically be mapped as zero, which affects the ability to proper fit the data.

The MLPc-relu showed the better performance scores where discerning a class from another was objectively hard, showcased in dataset **i** Fig. 11 left, to the cost of higher computation

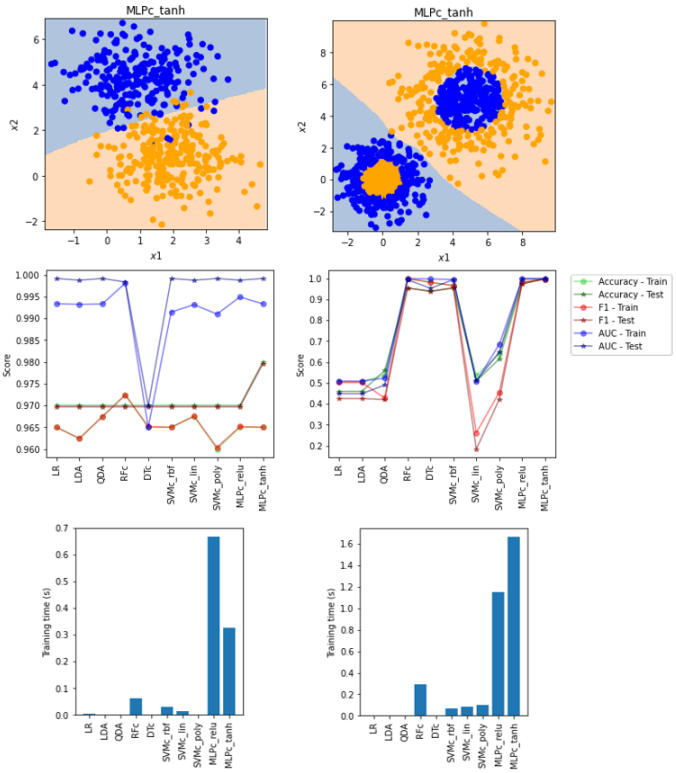


Fig. 10. MLP with hyperbolic tangent (tanh) activation function performance for datasets ***h*** (left) and ***e*** (right). MLP_tanh decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

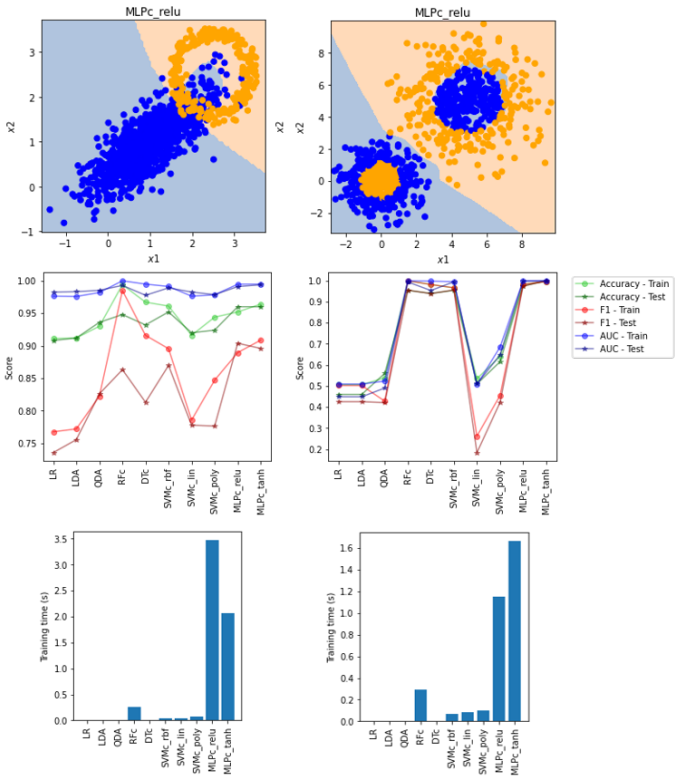


Fig. 11. MLP with rectified linear unit (relu) activation function performance for datasets ***i*** (left) and ***e*** (right). MLP_relu decision boundaries for both datasets are presented (top), as well as evaluation metrics (middle) and training time for all models, for both datasets (middle and bottom, respectively).

requirements and training times. An argument could be made that its MPLc counterpart would be preferable should speed be an issue, given that the usage of the hyperbolic tangent function offers similar results faster.

## VII. Discussion and Conclusions

For each Machine Learning model proposed, we have created and presented a scenario in which the approximation produced was significantly better than the majority of the methods, highlighting the models strengths and weaknesses. We were able to explore the strengths and weaknesses of each predictive method. Taking into account how each method works we were able to find reasons and justify the superiority of each model in the simulated scenarios. To each type of dataset is possible to take advantage of this knowledge in order to produce a more robust model in less time.

## References


[1] C. Bishop,Pattern Recognition and Machine Learning, Springer, 2006.

[2] T. Hastie, R. Tibshirani, J.Friedman, The Elements of Statistical Learning, Second Edition, New York, NY, USA: Springer New York Inc, 2008.

[3] J. VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, 1st ed., O'Reilly Media, Inc., 2016, ISBN:1491912057

[4] Scikit-learn, "Dataset loading utilities — scikit-learn documentation." `https://scikit-learn.org/stable/datasets.html` (accessed Jun. 06, 2021).

[5] Scikit-learn, "Linear and Quadratic Discriminant Analysis with covariance ellipsoid." `https://scikit-learn.org/dev/auto_examples/classification/plot_lda_qda.html#sphx-glr-auto-examples-classification-plot-lda-qda-py` (accessed Jun. 13, 2021).

[6] J. Han, et al. Data Mining: Concepts and Techniques, Third Edition. 3rd ed. Morgan Kaufmann Publishers, 2012.