Machine Learning - Assignment 2

Emanuel Tomé
MDS
DCC-FCUP
Porto, Portugal
up200702634@edu.fc.up.pt

Klismam Pereira
MDS
DCC-FCUP
Porto, Portugal
up201900035@edu.fc.up.pt

Luís Reis
MBCB
DCC-FCUP
Porto, Portugal
up201205115@edu.fc.up.pt

Vânia Guimarães
MDS
DCC-FCUP
Porto, Portugal
up200505287@edu.fc.up.pt

I. Introduction

The No Free Lunch Theorem (NFLT) for Machine Learning states that all optimization algorithms perform equally well when their performance is averaged across all possible problems, which implies that there is not a best optimization algorithm. Due to the close relationship between optimization and machine learning, it also implies that there is not a best machine learning algorithm [1].

II. Objectives

In this project we aim to evaluate the classification task performance of seven machine learning methods in 8 datasets with different characteristics. Attempts are made to explain the performance of methods regarding to each dataset, and to indicate which method would be a better choice for the problem.

III. Methods and hyper parameters

The performance of seven methods was evaluated in regards to the classification task considering each dataset described in the section 4 (Dataset Generation). The datasets were divided in train/validation and test sets in the proportion $80/20$. Grid search with 10-fold cross validation procedures were performed in order to select appropriate hyper parameters for the methods Decision Tree, Random Forest, Support Vector Machines, and Multi-layer Perceptrons. The methods and their respective searched hyper parameters space are:

- Logistic Regression
  - no search performed
- Linear Discriminant Analysis (LDA)
  - no search performed
- Quadratic Discriminant Analysis (QDA)
  - no search performed
- Decision Tree (DT)
  - complexity parameter α: [0.001, 0.01, 0.1]
- Random Forest (RF)
  - complexity parameter α: [0.001, 0.01, 0.1]
- Support Vector Machines (SVC)
  - kernel:
    - linear
    - polynomial
    - radial basis function
  - regularization parameter **C**: [0.1, 0.5, 1]
- Multi-layer Perceptrons (MLP)
  - activation function:
    - tanh: $f(x) = \tanh(x)$
    - relu: $f(x) = max(0, x)$
  - number of nodes: [10, 25, 50]
  - constant learning rate: [0.001, 0.01, 0.1]

Hyper parameters that are absent in the search space are set to the default values for each method as defined in the implementations of the package Sci-kit Learn [2].

This work was submitted on the framework of the course Machine Learning. The video presentation can be found in this link: https://web.microsoftstream.com/video/ac093695-06ef-43f2-875f-ea359fde4762

IV. Dataset generation

A. Linear separable dataset

The dataset was artificially generated by the function make_classification available in Scikit-Learn. The dataset has two features, two classes and 700 observations. The variables are normally distributed and the standard deviation is different for each class. The dataset is adequate for several types of boundaries, but it also can be linearly separable. The challenge is to understand the divergences in results between linear and non-linear algorithms.

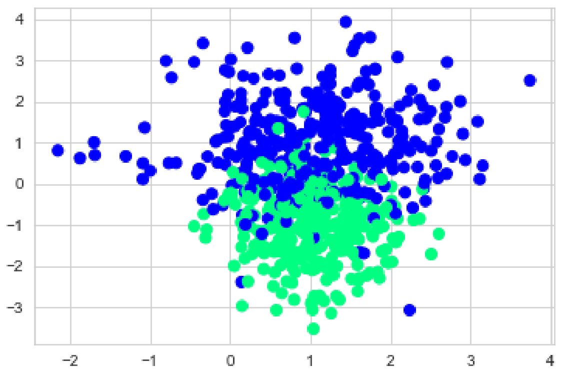

Figure 1: Scatter plot of linear separable dataset.

B. Small dataset

The dataset was generated by the function random.multivariate_normal available in Scikit-Learn. There are 4 classes that contain only 30 observations. The predictor variables X are bivariate normally distributed. The covariance matrix is the same for each class. The point of interest will be to understand the behavior of the algorithms with few observations in each class.

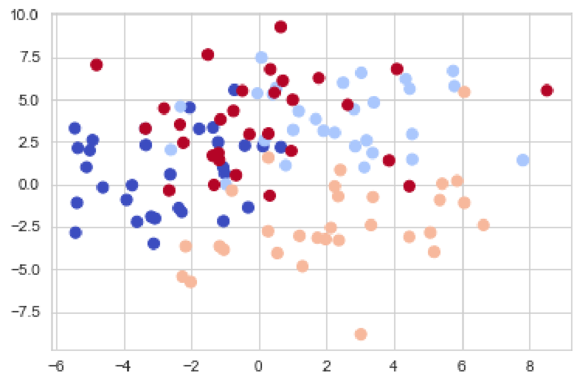

Figure 2: Scatter plot of small dataset.

C. 10 Features Dataset

Artificially generated, the dataset has 10 predictor variables X associated to five outcomes, represented each one by 100 samples. All subsets of X are themselves multivariate normal. The scatter plot displays the points as a function of the first two variables. The covariance matrix among the predictor variables across each class is different.

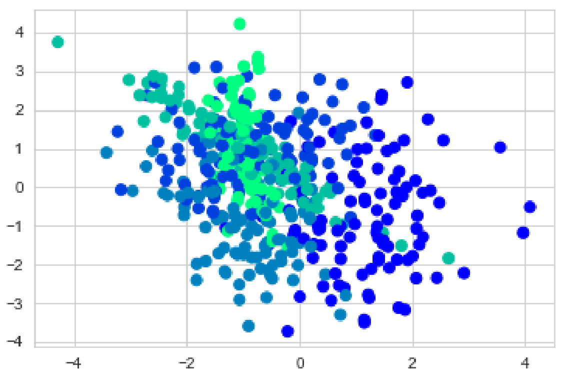

Figure 3: Scatter plot of dataset with 5 classes.

D. Square boundaries dataset

This dataset has 1500 observations, two features, and two classes. Points were generated through the method random.rand from the package Numpy. The boundaries between classes were defined using Python conditional statements.

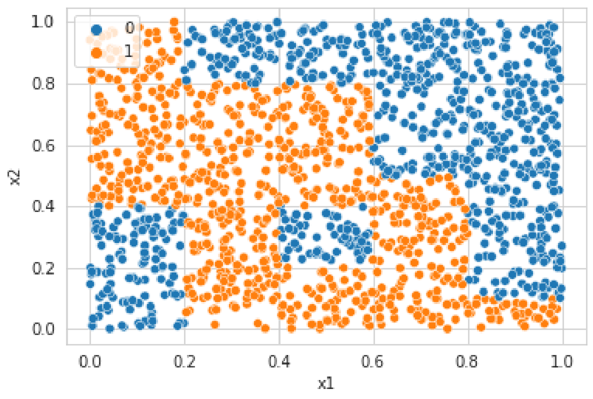

Figure 4: Scatter plot of dataset with classes divided by horizontal and vertical boundaries.

E. Square and diagonal boundaries dataset

This dataset has 1500 observations, two features, and two classes. It is based on the dataset shown in Figure 4, but with the addition of diagonal linear boundaries. The result is shown below, where the classes present shapes which resemble diagonal stripes, triangles and rectangles.

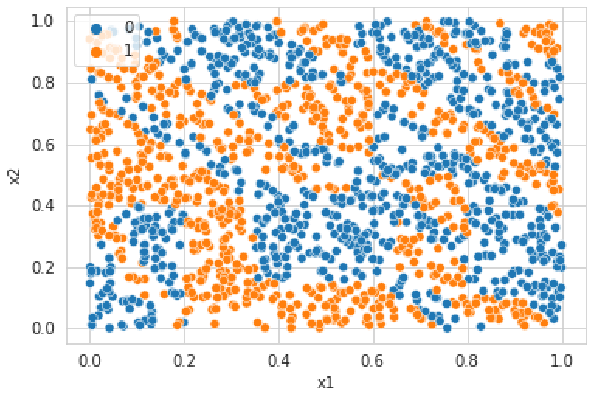

Figure 5: Scatter plot of dataset with classes divided by horizontal, vertical, and diagonal boundaries.

F. Circles dataset

The circles dataset was artificially generated using the function make_circles from scikit-learn with the parameters noise, factor and random_state set to 0.5, 0.2 and 0, respectively. The generated dataset has 1000 observations, two features and two classes. The data has a Gaussian distribution and spherical decision boundary [2].

G. Moons dataset

The moons dataset has 1000 samples and was artificially generated using the function make_moons of the scikit-learn with the parameters noise and random_state set to 0.3 and

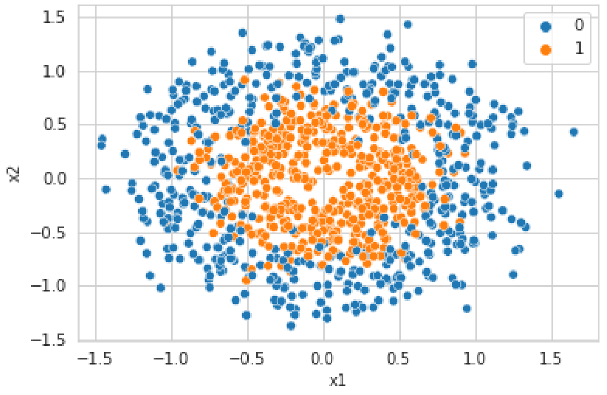

Figure 6: Scatter plot of the circles dataset.

0, respectively. The parameter noise set to 0.3 means that Gaussian noise with a standard deviation of 0.3 is added. Similar to the circle dataset, this dataset has also a non-linear boundary between the classes as one can see in Figure 7, even though with different characteristics, since the classes are interleaved in two half circles [2].

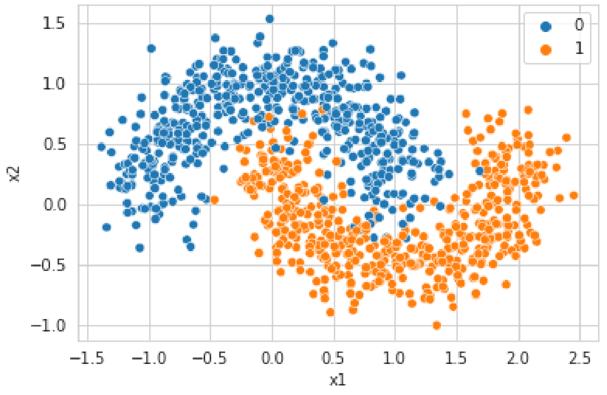

Figure 7: Scatter plot of the moons dataset.

H. Dense and 15 features dataset

This dataset has 1000 samples, artificially generated using the function make_classifications from scikit-learn. 1000 points were divided over 6 classes, each with 5 clusters. Critically, the class separation hyper parameter is set to 0.1.

V. Experimental setup

Models were fit to each dataset using the hyper parameters chosen by grid search with 10-fold cross validation procedure. Accuracy, f1, and area under the curve (AUC) were obtained for each fold as well as the mean fit time across folds. For datasets with more than two classes the f1 score is calculated as the average score between all classes, and AUC is calculated using one-versus-rest and macro averaging configuration.

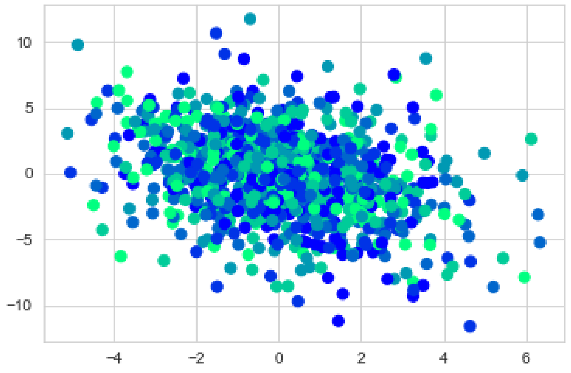

Figure 8: Scatter plot of the 2 feature representation of 15 feature dense dataset

Tables were created to present the mean validation/train set scores for each method, as well as the test set scores.

The validation scores were then compared through the Friedman statistic. This is done in order to evaluate if there are statistically significant differences between them, such that the null and alternative hypothesis can be written as H_0: there is no difference between the scores of all compared models, H_1: at least two models yield statistically different scores.

Next, the post-hoc Nemenyi statistic is computed in order to verify differences in performance between pairs of models, and the results are shown through heatmaps (Appendix VIII-C). Similarly, one can write the null and alternative hypothesis as H_0: there is no difference between scores of compared models, H_1: models yield statistically different scores.

$\alpha = 0.05$ for both statistical tests.

Finally, based on the test set scores, mean fit time, method complexity, and in statistical evidence obtained from the validation set scores, we attempt to explain the performance of models given the datasets and to indicate which method would be more appropriate for them.

VI. Results and limitations

A. Linear separable dataset

The Tables I and II show the performance of ten algorithms in a linear separable dataset. The first table shows the mean of results obtained in 10 folds by cross-validation. The second table displays the results obtained in the test set. For all models the performance of Accuracy, F1 and AUC were pretty good, almost all values are above 85%, in both situations. Only according to these three metrics we can not infer the method which had the best performance. The situation occurs because when the balanced classes of a dataset are well separated by a linear boundary, the others methods able to find more flexible boundaries can easily adjust to the data, even with a linear boundary if necessary, and find also a good separation. Although the advantages of non-linear models, Logistic Regression is the preferred model because of its simplicity, ease of implementation, low resources and time spent, and above all for its interpretability. The researchers are often interested in why each class occurs, what are the variables that most influence the class, quantify it, and understand the probability an element belongs to the class, etc. Logistic Regression provides useful insights, showing a linear relationship between independent variables and log odds. The method not only gives a measure of how relevant an independent variable is, but also tells us about the direction of the relationship. However, it requires large sample sizes to be more reliable. If the model produced by logistic regression is based on a smaller number of actual observations, the estimated parameters become unstable. The same situation occurs when the classes are very well separated, and the existence of collinearity inflates the variances of the parameter estimates, and consequently incorrect inferences about relationships between explanatory and response variables. The dataset fits the requirements of Logistic Regression algorithm, the results are very good without the worry of changing hyper-parameters, the gains with interpretation are enormous and the resources applied relatively low. If we compared the training time, logistic regression is not among the best times. However, the theoretical running time complexity of the Logistic Regression is relatively small, $O(Nd)$, where N is the number of elements and d the number of dimensions. For example, to SVM with linear kernel, with low training time in this example, has a time complexity $O(N^3)$ at worst scenario, which means for a large dataset the training time for SVM is worse than Logistic Regression.

Table I: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the linear separable dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean AUC
Log. Regression	0.0415	0.8571	0.8587	0.9285
LDA	0.0085	0.8554	0.8613	0.9284
QDA	0.0016	0.8607	0.8667	0.9329
Decision Tree	0.0029	0.8536	0.8599	0.9069
Random Forest	0.4302	0.8536	0.858	0.908
SVC linear	0.0243	0.8589	0.8624	0.9285
SVC poly	0.0305	0.7964	0.8235	0.9196
SVC rbf	0.0292	0.8732	0.8786	0.9283
MLP tanh	0.237	0.8661	0.8663	0.9306
MLP relu	0.6098	0.8661	0.8703	0.9298

Table II: Accuracy, f1-score and AUC for the test set in the linear separable dataset.

	Accuracy	f1score	AUC
Log. Regression	0.8500	0.8591	0.9408
LDA	0.8571	0.8667	0.9406
QDA	0.8500	0.8609	0.9435
Decision Tree	0.8500	0.8627	0.8999
Random Forest	0.8571	0.8684	0.8972
SVC linear	0.8500	0.8591	0.941
SVC poly	0.7786	0.8187	0.9198
SVC rbf	0.8571	0.8684	0.9226
MLP tanh	0.8571	0.863	0.9363
MLP relu	0.8571	0.8667	0.9339

B. Small dataset

Nowadays the amount of data available for studies grows exponentially. Despite that, there are situations where it is difficult to collect more data, like when rare diseases are evaluated. The dataset represents this kind of problem that we should know how to deal. The more powerful machine learning algorithms are often referred to as nonlinear algorithms. Although, in general, these are more flexible, they need more data to achieve a good performance, because they have high-variance and easily overfit. To control the variance linear models are more suitable because, in spite of having high bias, with few training observations reducing the variance is crucial. The results of Tables III and IV translate this idea.

The results obtained across folds were, sometimes, significantly higher than the results achieved on test set, showing that the algorithms have difficulty in generalizing the data. In test set, Linear Discriminant Analysis was better in the three evaluation metrics. Applying the algorithms in other test sets, the results had slight changes, but LDA was among the best performances. Despite the Multi-Layer Perceptron models good results, the resources and time was more than 15 times of those spent by the LDA. This occurs because the methods have more operations to execute, the estimated time complexity of a MLP is $O(NdML)$, where N is the observations, d predictors, M hidden units and L training epochs, and for LDA is only $O(Nd^2)$. Besides, with those models we lost interpretability. That situation doesn't happen with LDA which finds a linear combination of features that separates different classes. The probability of each individual belonging to the different class is computed and then the individual is affected to the group with the highest probability score. By this method we can infer several conclusions, measure the effect of the features in the characterization of the classes and the individuals.

Linear Discriminant Analysis is a powerful method for a small dataset, but under a set of conditions. The method assumes that the predictor variables X are drawn from a multivariate Gaussian distribution, and equal covariance among the predictor variables across each class. For that reason, the decision boundary is linear. As we can see the method is very restrictive, but if all the conditions are met it produces robust and interpretable classification results. By statistical tests, we can not declare that the model had the best performance among the others, but taking into account its advantages and prediction results, it isa good option.

Table III: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the small dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.0772	0.6567	0.6105	0.8792
LDA	0.0424	0.6789	0.6453	0.8881
QDA	0.0015	0.6456	0.5758	0.8783
Decision Tree	0.0018	0.6156	0.5197	0.7225
Random Forest	0.3146	0.6667	0.5726	0.856
SVC linear	0.0085	0.6356	0.5815	0.8671
SVC poly	0.0065	0.6467	0.5474	0.863
SVC rbf	0.0062	0.6878	0.6564	0.8548
MLP tanh	0.6662	0.6456	0.5969	0.8908
MLP relu	0.6630	0.6456	0.6103	0.8823

Table IV: Accuracy, f1-score and AUC for the test set in the small dataset.

	Accuracy	f1score	AUC
Log. Regression	0.4583	0.4389	0.8308
LDA	0.5417	0.5249	0.8354
QDA	0.4583	0.4389	0.8176
Decision Tree	0.4167	0.4017	0.7054
Random Forest	0.4583	0.4335	0.7845
SVC linear	0.4583	0.4389	0.8111
SVC poly	0.5000	0.5061	0.7723
SVC rbf	0.5000	0.4889	0.8163
MLP tanh	0.4583	0.4271	0.8333
MLP relu	0.4583	0.4271	0.8297

C. 10 Features Dataset

So far, we used datasets with only 2 predictors variables X, so we tried to see the behavior of the algorithms in a dataset with 10 features, 5 classes and 500 samples. The evaluation of the performances of the models are displayed in Tables V and VI. All the models, except MLP with relu activation function obtained better results in test set than in 10 folds cross-validation. The hyper-parameters of complex models were able to avoid overfit and the values of Accuracy, F1 and AUC were relatively good for all of them. The model that stands out is Quadratic Discriminant Analysis, it provides the best results for all evaluations.

In statistical analysis (see Appendix VIII-C) comparing the Accuracy and F1, Quadratic Discriminat Analysis is better than Logistic Regression, Linear Discriminant Analysis, Decision Trees, Support Vector Classifier with linear and poly kernel for a significance level of 0.05. For AUC, it is statistically better than all models mentioned above except Support Vector Classifier with linear kernel. Besides the good results produced by the model, the training time spent was the lowest one, reinforcing its advantages. QDA is the best model since the data respect its assumptions and requirements. The algorithm is adequate for binary or multi-class classification problem. This method is flexible in the sense that despite its assumption that the predictor variables have a normal distribution, the covariance matrix can be different for each class. For that reason the decision boundary is a quadratic function. We have to be careful with the number of parameters to be estimated which can be large since there are separate covariance matrix for every class. If we have many classes and not so many sample points, this can be a problem. However, if it is able to produce robust estimated parameters, the performance of the model can beat the more complex algorithms, as we can see in the example.

D. Square boundaries dataset

We now present the results obtained for the square boundaries dataset (Tables VII and VIII). This dataset was generated having in mind the way how Decision Trees work, therefore it was expected that tree based methods would show superior performance. Indeed, this was verified if one takes in consideration both the metrics obtained (Tables VII and VIII) and the boundaries presented in the Appendix VIII-A.

Table V: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the 5 classes and 10 features dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.0861	0.7425	0.7364	0.9381
LDA	0.0124	0.7275	0.7219	0.9344
QDA	0.0051	0.9575	0.9575	0.9982
Decision Tree	0.0054	0.78	0.7778	0.8625
Random Forest	0.4164	0.8375	0.8373	0.9674
SVC linear	0.0519	0.7675	0.7635	0.955
SVC poly	0.0498	0.7475	0.7422	0.9431
SVC rbf	0.0565	0.82	0.8211	0.967
MLP tanh	3.4872	0.8175	0.8164	0.9668
MLP relu	0.2410	0.8175	0.8164	0.9596

Table VI: Accuracy, f1-score and AUC for the test set in the 5 classes and 10 features dataset.

	Accuracy	f1score	AUC
Log. Regression	0.79	0.7905	0.9672
LDA	0.7700	0.7637	0.9594
QDA	0.9500	0.9514	0.9988
Decision Tree	0.7100	0.7157	0.8209
Random Forest	0.8500	0.8455	0.9741
SVC linear	0.8600	0.8578	0.9786
SVC poly	0.7200	0.7208	0.9575
SVC rbf	0.8600	0.8589	0.9818
MLP tanh	0.8600	0.8579	0.9718
MLP relu	0.7600	0.7603	0.9594

It is possible to verify through the decision boundaries that both Decision Trees and Random Forests methods can easily overfit training data (Figures 12 and 13), which is an useful property to tackle this toy dataset. In comparison, the decision boundaries generated by the remaining methods cannot reconstruct the underlying structure of the classes.

The Friedman test used to compare accuracy, f1, and AUC scores between methods yielded the respective p-values of $5.28E-14$, $2.71E-14$, and $2.94E-14$, thus we reject the null hypothesis that the performance is equal across methods. The method that yield the highest mean scores on the validation set (Table VII) is Random Forests, followed by Decision Trees, and Multi-layer Perceptron ($f(x) = max(0, x)$). All the scores from these methods are higher or equal to 0.9. Among these methods, the mean fit time is lowest for the Decision Tree method ($0.0083s$), followed by Random Forest ($0.8008s$), and Multi-layer Perceptron ($1.9219s$).

Table VII: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the square boundaries dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.1865	0.7517	0.7549	0.7771
LDA	0.1782	0.7492	0.7518	0.7771
QDA	0.0038	0.8083	0.8204	0.9023
Decision Tree	0.0083	0.9908	0.991	0.9915
Random Forest	0.8008	0.995	0.9951	0.9998
SVC linear	0.6429	0.7683	0.7828	0.7758
SVC poly	1.5297	0.7783	0.8041	0.7759
SVC rbf	0.4321	0.895	0.9019	0.9487
MLP tanh	2.9575	0.8808	0.8838	0.9638
MLP relu	1.9219	0.9	0.9037	0.9769

The post-hoc Nemenyi tests used to compare pairs of methods shows that (Figures 48, 49, and 50) Decision Trees and Random Forests performances are overall significantly different from the remaining methods, with the exception of Support Vector Machine (radial basis function kernel) and both Multi-layer Perceptron models. Also, there is no statistically significant difference in performance across scores between Decision Tree and Random Forest.

Finally, the test set scores table (Table VIII) shows that the Decision Tree model was able to classify all cases perfectly. The next best scores are yielded by Random Forest and Multi-layer Perceptron ($f(x) = max(0, x)$).

Table VIII: Accuracy, f1-score and AUC for the test set in the square boundaries dataset.

	Accuracy	f1score	AUC
Log. Regression	0.7433	0.7298	0.809
LDA	0.7433	0.7298	0.8086
QDA	0.7867	0.7935	0.8941
Decision Tree	1	1	1
Random Forest	0.9967	0.9966	0.9999
SVC linear	0.77	0.7692	0.8063
SVC poly	0.7567	0.7781	0.8025
SVC rbf	0.8967	0.8997	0.9504
MLP tanh	0.8833	0.8875	0.9624
MLP relu	0.94	0.9423	0.9943

It is important to point out that this is a toy dataset, where classes were separated using decision rules which can be easily reconstructed by Decision Trees and Random Forests, and that such structure might not be representative of real problems.

Of the best performing methods, the validation set scores were statistically equal between Decision Tree, Random Forest, Support Vector Machine (radial basis function kernel), and both Multi-layer Perceptrons. The tree based methods were able to closely reconstruct the classes decision boundaries. The fastest mean fit time belongs to the Decision Trees method, which also presents the highest test set scores. The computational complexity of Decision Trees is $O(pN \log_2(N))$, and of Random Forests is $O(kpN \log_2(N))$, where n is the number of cases, p is the number of features, and k is the number of trees, which denotes Decision Trees is the less complex method of the two. Finally, Random Forests models lack interpretability, while splits and leaves of a single Decision Tree can be more intuitive and easier to understand. Therefore, we indicate that the Decision Trees method is the best choice for the classification task on the dataset with square boundaries.

E. Square and diagonal boundaries dataset

With this dataset we intended to make the classification task evaluated in the previous section harder (square boundaries dataset). The validation and test sets results are shown in Tables IX and X. The classification boundaries produced by the methods can be seen in the Appendix VIII-A.

As in the previous problem, Decision Trees and Random Forests methods can leverage their characteristics to produce decision boundaries that try to reconstruct the underlying division of classes, while the boundaries approximated by the remaining methods are linear or have round shapes that do not resemble the original data. That said, even the tree based methods had difficulty in reconstructing the diagonal stripes introduced (Figures 22 and 23). Also, it seems the decision boundary created by the Random Forests method is following points and corners more closely than the one created by the Decision Trees. The Random Forests method should be able to reduce variance when compared to a single Decision Tree. That said, the dataset only has two features, which should create several correlated trees in the bagging step of the Random Forests method, thus it cannot reduce variability. This might be what is causing the severely overfit boundary, other than the characteristics of the dataset.

The Friedman test used to compare accuracy, f1, and AUC scores between methods yielded the respective p-values of $3.45E-14$, $2.80E-13$, and $1.98E-13$, thus we reject the null hypothesis that the performance is equal across methods. The method that yield the highest mean scores on the validation set (Table IX) is Random Forest, followed by Decision Tree, and Multi-layer Perceptron ($f(x) = \tanh x$). Both Random Forests and Decision Trees scores are higher than or equal to 0.84, while scores from the Multi-layer Perceptron are higher than 0.69. Among these methods, the mean fit time is lowest for the Decision Tree method ($0.0178s$), followed by Random Forest ($0.8674s$), and Multi-layer Perceptron ($2.2836s$).

Table IX: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the square and diagonal boundaries dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.3321	0.5983	0.5803	0.6166
LDA	0.0744	0.5967	0.5793	0.6162
QDA	0.0026	0.59	0.5771	0.6148
Decision Tree	0.0178	0.8467	0.8432	0.8566
Random Forest	0.8674	0.8658	0.8626	0.943
SVC linear	1.2018	0.5817	0.5724	0.6145
SVC poly	2.4035	0.5675	0.6181	0.609
SVC rbf	1.4704	0.7125	0.6979	0.7462
MLP tanh	2.2836	0.7275	0.6928	0.7855
MLP relu	17.4289	0.705	0.6738	0.7527

Similarly to the previous dataset studied, the post-hoc Nemenyi tests used to compare pairs of methods show that (Figures 51, 52, and 53) Decision Trees and Random Forests performances are overall significantly different from the remaining methods, with the exception of Support Vector Machine (radial basis function kernel) and both Multi-layer Perceptron models. Also, there is no statistically significant difference in performance across scores between Decision Trees and Random Forests.

The test set scores table (Table X) shows that the Random Forests model yielded the best scores, with accuracy, f1, and AUC equal to 0.8967, 0.883, and 0.9492 respectively. Next comes Decision Trees, with scores 0.8267, 0.8116, and 0.8344, and Multi-layer Perceptron ($f(x) = \tanh x$) follows with scores 0.7267, 0.6917, and 0.7481.

Much like in the previous dataset, considering only the best performing methods, the validation set scores were statistically equal between Decision Trees, Random Forests, Support Vector Machines (radial basis function kernel), and both Multi-layer Perceptrons. Even though they had more difficulty in doing so, the tree based methods were able to reconstruct the

Table X: Accuracy, f1-score and AUC for the test set in the square and diagonal boundaries dataset.

	Accuracy	f1score	AUC
Log. Regression	0.6067	0.5597	0.6242
LDA	0.6033	0.5576	0.6243
QDA	0.6033	0.5641	0.6186
Decision Tree	0.8267	0.8116	0.8344
Random Forest	0.8967	0.883	0.9492
SVC linear	0.59	0.5495	0.6233
SVC poly	0.5733	0.6049	0.6151
SVC rbf	0.72	0.6769	0.7571
MLP tanh	0.7267	0.6917	0.7712
MLP relu	0.7267	0.6667	0.7481

decision boundaries to some extent. The fastest mean fit time belongs to the Decision Trees method. That put, the highest test set scores belong to the Random Forests method, with differences from the second highest scoring model (Decision Trees) of 0.07 for accuracy, 0.07 for f1, and 0.11 for AUC. We have already described the computational complexity of both tree based methods, being Decision Trees the less complex method of the two.

Both Decision Tree and Random Forest had overall satisfactory results, and to choose which would be a better method for the classification task of this dataset is no trivial. Nonetheless, considering that this is a small dataset where time complexity is not an issue, we conclude that Random Forests is a better choice given its test set performance.

F. *Circles dataset*

The Table XI presents the averaged values of the fit time and metrics Accuracy, f1-macro and AUC obtained in the circles dataset for all models that have been considered. Since the boundary between the classes is spherical, in other words, non-linear, poor performance metrics were expected for algorithms such as Logistic Regression, LDA and SVC with the linear kernel since these algorithms are only capable to separate classes separated by a linear boundary. On the other hand, highly expressive algorithms such as Decision Trees, Random Forest, SVC with the rbf kernel or the MLP were expected to capture this highly non-linear boundary. For QDA high scores were also obtained together with the lowest mean fit time. Note that the classes were generated from a Gaussian distribution, being for that reason the ideal conditions for QDA. The scores obtained for the test set are presented in Table XII, which are higher for the same algorithms referred before. Considering the obtained boundaries (see Appendix VIII-A), the boundaries obtained for QDA (see Figure 29) and SVC with the rbf kernel (see Figure 32) are the ones that seems to be closer to the ground truth and for that reason, the ones with better metrics. Considering the results obtained for the Nemenyi statistic (see Appendix VIII-C), one can state there is no differences between the obtained scores for QDA, DT, RF, SVC with the rbf kernel and MLP. Therefore, taking this in consideration, for this dataset in particular the QDA algorithm is recommended since it is the simplest one, has lower time complexity and can be used for data interpretation.

Table XI: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the circles dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.7225	0.4038	0.1404	0.4204
LDA	0.0461	0.405	0.1429	0.4204
QDA	0.0013	0.8838	0.8768	**0.9497**
Decision Tree	0.0061	0.8588	0.8528	0.8652
Random Forest	0.3843	0.8613	0.8543	**0.9123**
SVC linear	0.2529	0.51	0	0.5073
SVC poly	0.2778	0.5163	0.0679	0.4299
SVC rbf	0.2007	0.8838	0.8777	**0.9515**
MLP tanh	4.9058	0.89	0.8855	**0.9493**
MLP relu	3.6456	0.8825	0.8784	**0.9478**

Table XII: Accuracy, f1-score and AUC for the test set in the circles dataset.

	Accuracy	f1score	AUC
Log. Regression	0.825	0.8325	0.9292
LDA	0.37	0	0.4859
QDA	0.9	0.901	0.9714
Decision Tree	0.905	0.9091	0.8996
Random Forest	0.9	0.9029	0.9456
SVC linear	0.46	0	0.4789
SVC poly	0.46	0	0.6063
SVC rbf	0.915	0.9171	0.9724
MLP tanh	0.89	0.8932	0.9707
MLP relu	0.9	0.9038	0.9714

G. *Moons dataset*

Similar to the other datasets, we start by presenting in Tables XIII and XIV the averaged fit time and scores in the folds of the 10-fold cross validation and the scores obtained in the test set, respectively. This dataset has a highly non-linear boundary, so better scores were expected from expressive algorithms. Indeed, accuracies and f1-scores higher than 90% where obtained for RF, SVC with the rbf kernel, and MLPs both for the mean values obtained in the 10-fold cross-validation and the ones obtained in the test set. Taking in consideration the post-hoc Nemenyi statistics presented in Appendix VIII-C, one can not state that those models have different scores.

The relative high score values yielded by other models that are better used in datasets with classes separable by a linear boundary, such as Logistic Regression or LDA, may be misleading. Indeed, this happens because a linear boundary is able to in fact classify most of the datapoints, especially the ones further way from the boundary. If one looks to the obtained boundaries which are depicted in Appendix VIII-A, easily concludes that the decision boundaries for those algorithms are far way for the true boundary. The boundary obtained for RF may seem to be overfiting to the data since there are some islands, while the one obtained for DT may seem to be too simple when compared with the true boundary. The ones obtained for SVC with the rbf kernel or MLP with the tanh activation function seem closer to the true boundary.

Taking in consideration all that have been said, the more appropriate models for this dataset seems to be SVC with the rbf kernel and MLP with the tanh activation function. However,the fiting time for SVC is considerably lower. That said, for a dataset with a considerable number of points the MLP may be preferred since its time complexity will be smaller (for MLP the complexity is $O(NpML)$ - where N, p,

M and L are the number of observations, predictors, hidden units and training epochs, respectively - and $O(N^3)$ for SVC). In other words, the complexity for SVC depends on N^3, while for MLP it depends on N.

Table XIII: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the moons dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean cv AUC
Log. Regression	0.8211	0.8513	0.8476	0.9346
LDA	0.0597	0.8425	0.8402	0.9342
QDA	0.0014	0.8438	0.8413	0.934
Decision Tree	0.0078	0.8938	0.8888	0.9201
Random Forest	0.8681	0.9125	0.9129	0.9544
SVC linear	0.1451	0.85	0.8466	0.9347
SVC poly	0.186	0.8763	0.8795	0.9331
SVC rbf	0.1114	0.9088	0.9098	0.963
MLP tanh	0.3122	0.9088	0.9079	0.9654
MLP relu	0.43	0.9113	0.9103	0.9649

Table XIV: Accuracy, f1-score and AUC for the test set in the moons dataset.

	Accuracy	f1score	AUC
Log. Regression	0.825	0.8325	0.9292
LDA	0.82	0.8286	0.929
QDA	0.82	0.8286	0.9294
Decision Tree	0.885	0.8844	0.9468
Random Forest	0.895	0.9023	0.9641
SVC linear	0.825	0.8325	0.9294
SVC poly	0.875	0.8869	0.9217
SVC rbf	0.92	0.9252	0.9704
MLP tanh	0.915	0.9202	0.9745
MLP relu	0.925	0.9302	0.9706

H. Dense and 15 features dataset

Table XV presents the averaged values of the fit time and scores Accuracy, f1-macro and AUC obtained in the dense and 15 features dataset for all models that have been considered. The complexity of this dataset, with 15 features and 5 classes, with a small class separation makes a simple 2D representation challenging and uninformative. The best model to separate this dataset is the MLP. MLPs, particularly using the ReLu activation function, have the best results, followed by Random Forests, QDA and SVM-rbf, although without statistically significant differences between the three scores used (see Nemenyi heatmaps for this dataset, Figures 60, 61, and 62).

Of note, the accuracy of all methods decreases from training to test, with the MLP maintaining the best accuracy. Overall, the accuracy of these models is low, some even failing to reach the accuracy of a random classifier. The methods with the longer train times are the MLPs, although in this dataset they also achieve the best scores. MLPs have an important space in large complex datasets with multiple features. The RELU activation is currently the default function with deep learning neural networks, mainly due to 3 reasons, the computational simplicity of a max() function vs exponential calculation, the ability to output a true zero value, with accelerated learning producing a simpler model, and the linear behavior that avoids the problem of vanishing gradients, a major issue with other activation functions [3]. Still, and due to turning all negative values to 0, the ReLu activation function doesn't map the negative values appropriately, and better results could be obtained with a Leaky ReLu activation function [4]. Considering the results obtained for the Nemenyi statistic (see Appendix VIII-C), the differences seen in the global results tables are reduced to statistically significant differences between MLPs against LR, LDA, Decision Trees, SVC Poly and SVC linear and MLP TanH. Therefore, taking this in consideration, for this dataset in particular, the MLP has the best results, but also takes the longest to be trained, hence the choice would depend on the value given by the user to extra training time or having a marginally better model.

Table XV: Mean fit time, mean accuracy, mean f1-score and mean AUC for 10-fold cross validation for the dense and 15 featured dataset.

	mean fit time (s)	mean cv accuracy	mean cv f1	mean AUC
Log. Regression	0.04	0.1775	0.1735	0.5036
LDA	0.0024	0.1775	0.1741	0.5033
QDA	0.0056	0.2675	0.2609	0.6161
Decision Tree	0.02	0.2263	0.2213	0.5354
Random Forest	0.5688	0.285	0.2825	0.6259
SVC linear	1.3448	0.1775	0.1749	0.4804
SVC poly	0.4088	0.1763	0.0871	0.4923
SVC rbf	0.4885	0.29	0.2794	0.6401
MLP TanH	4.7741	0.2275	0.2237	0.5633
MLP relu	4.7048	0.335	0.3318	0.6952

Table XVI: Accuracy, f1-score and AUC for the test set in the dense and 15 featured dataset.

	Accuracy	f1score	AUC
Log. Regression	0.145	0.1355	0.4647
LDA	0.145	0.1354	0.4661
QDA	0.225	0.2258	0.5678
Decision Tree	0.175	0.1702	0.5066
Random Forest	0.305	0.3056	0.6331
SVC linear	0.145	0.1411	0.5592
SVC poly	0.17	0.1246	0.477
SVC rbf	0.295	0.2846	0.6631
MLP tanh	0.205	0.2042	0.5598
MLP relu	0.33	0.327	0.6777

VII. Discussion and Conclusions

In this work we evaluated the performance of seven algorithms in eight datasets with distinct characteristics in order to evaluate their suitability in different problems. We tried to generate datasets with different characteristics covering as many types of boundaries as possible. Regarding the training of the models, we used a grid search procedure with 10-fold cross-validation in order to determine the best hyper-parameters for each model in each dataset. However, the hyper-parameter search space was limited for practical reasons, so different results may have been obtained if more values were considered. For instance, SVC poly may have had better performance in some of the datasets if we allowed the tuning of the degree of the polynomial in the grid search.

Although for each dataset an algorithm was generally recommended, one should note that recommendation may have been different if, for instance, the dataset had a different dimension, both in number of samples or number of features, or different degree of separability or random noise. For instance, for datasets with a high number of samples, SVC or MLPs may be prohibitive due to its time complexity, particularly when compared to LDA or Decision Trees. On the other hand, since the boundaries defined by SVC are only affected by the points near the margin (the support vectors), they are effective in datasets with a greater number of features. In practice, only small difference are found between SVC, DT, RF, MLPs and QDA, in such a dataset. The best algorithm may also vary in function of the application. For instance, in a situation where concept drift may occur and fast retraining of the model may be needed, a algorithm like QDA or DT with lower time complexity may be preferred, even if has a lower performance, whereas with a very large dataset a neural network may be preferred.

It is also important to point out that the toy datasets used may not be representative of real problems. For instance, in the datasets "square boundaries" and "square and diagonal boundaries" the classes were split using rules which are easily reconstructed by an algorithm such as Decision Trees or Random Forests. Several real datasets, such as credit and banking, sports, housing, among others, were also tested, and rarely a method showed a significantly better performance. It should be emphasized that several experiments and datasets were tested by the authors. The ones presented were the ones where more interesting results were achieved.

Finally, we conclude by asserting that no single method was capable of producing superior results in all datasets, while in most of them more than a single algorithm had similar performances. In a sense, these results agree with the NFLT, even though just a small set of examples was used, and the experiment was subject to other restrictions such as the limited hyper-parameter space searched and number of methods evaluated. For future works, more example datasets of diverse real life problems might be experimented on.

References

[1] Jason Brownlee. *No Free Lunch Theorem for Machine Learning*. Oct. 2020. URL: https://machinelearningmastery.com/no-free-lunch-theorem-for-machine-learning/.

[2] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[3] Jason Brownlee. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Aug. 2020. URL: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[4] Sagar Sharma and Simone Sharma. “Activation functions in neural networks”. In: *Towards Data Science* 6.12 (2017), pp. 310–316.

VIII. Appendixes

A. Plot of the decision boundaries

1) Square boundaries dataset

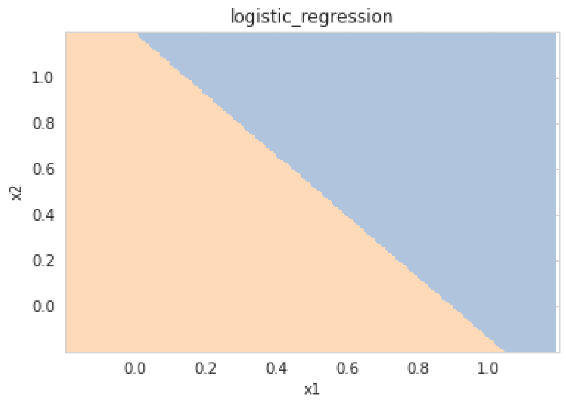

Figure 9: Decision boundary for Logistic Regression in the square boundaries dataset.

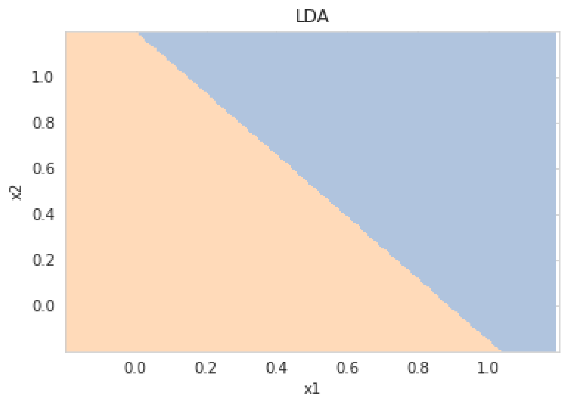

Figure 10: Decision boundary for LDA in the square boundaries dataset.

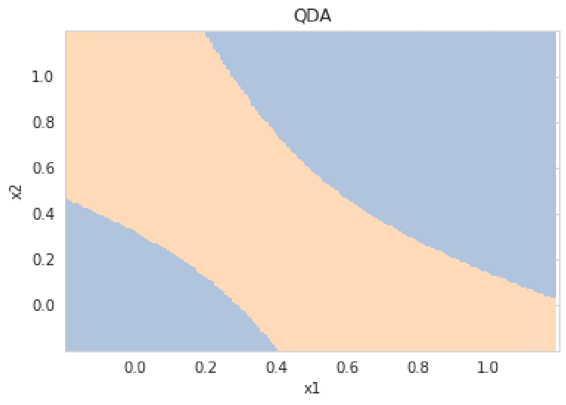

Figure 11: Decision boundary for QDA in the square boundaries dataset.

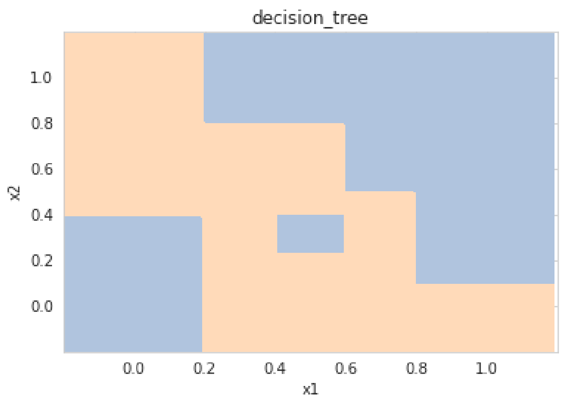

Figure 12: Decision boundary for DT in the square boundaries dataset.

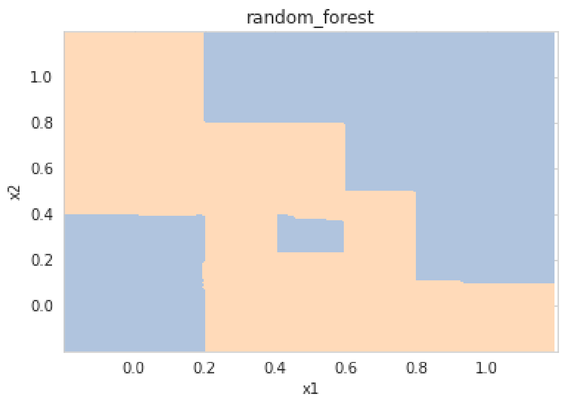

Figure 13: Decision boundary for RF in the square boundaries dataset.

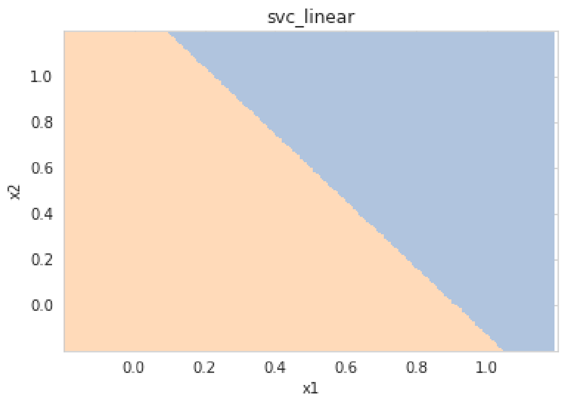

Figure 14: Decision boundary for SVC linear in the square boundaries dataset.

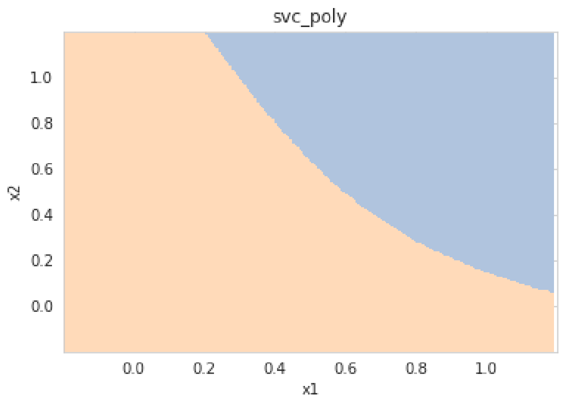

Figure 15: Decision boundary for SVC poly in the square boundaries dataset.

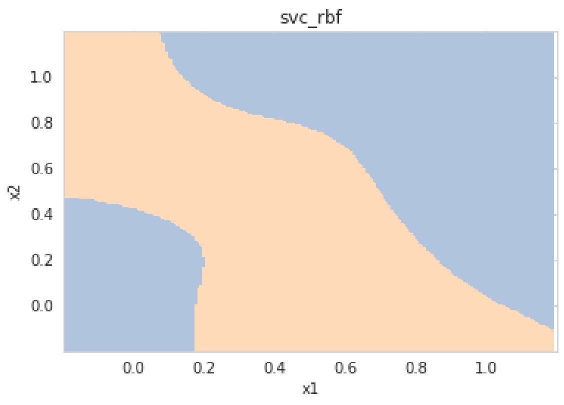

Figure 16: Decision boundary for SVC rbf in the square boundaries dataset.

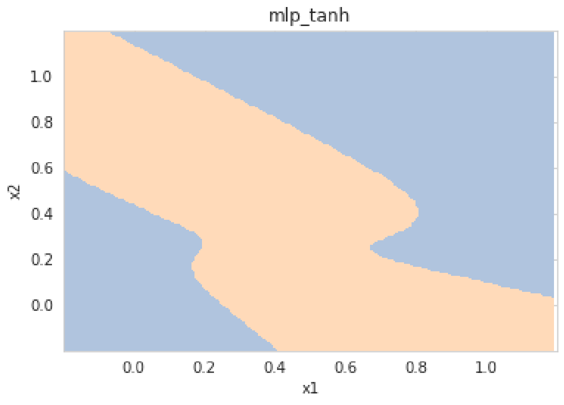

Figure 17: Decision boundary for MLP tanh in the square boundaries dataset.

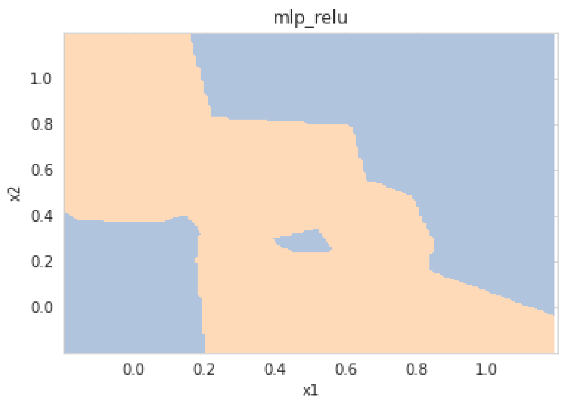

Figure 18: Decision boundary for MLP relu in the square boundaries dataset.

2) Square and diagonal boundaries dataset

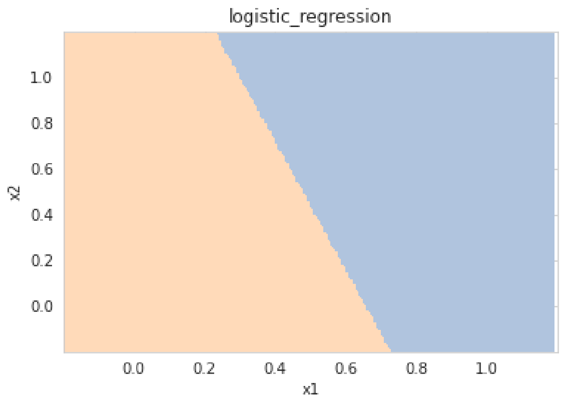

Figure 19: Decision boundary for Logistic Regression in the Square and diagonal boundaries dataset.

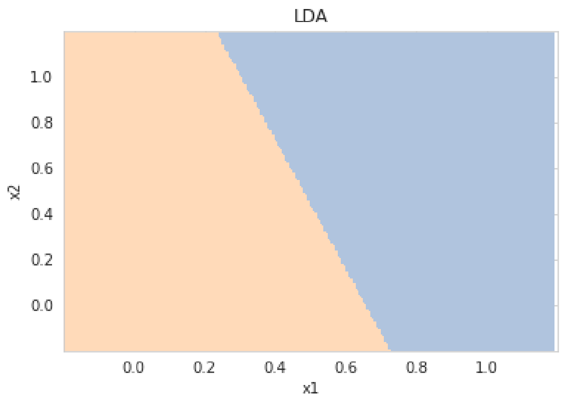

Figure 20: Decision boundary for LDA in the square and diagonal boundaries dataset.

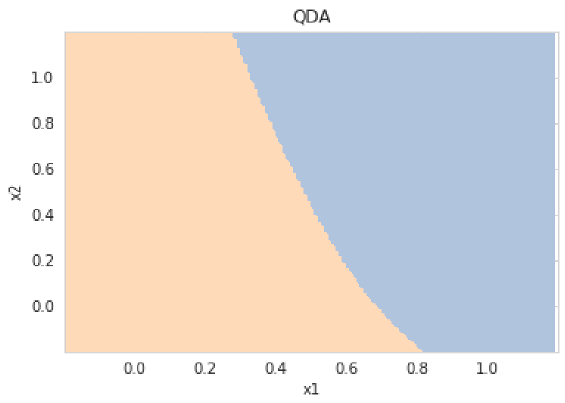

Figure 21: Decision boundary for QDA in the square and diagonal boundaries dataset.

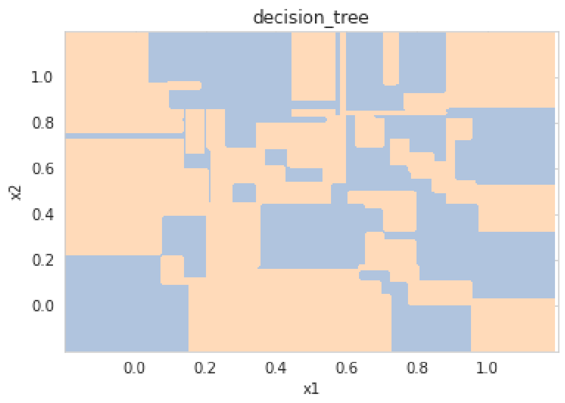

Figure 22: Decision boundary for DT in the square and diagonal boundaries dataset.

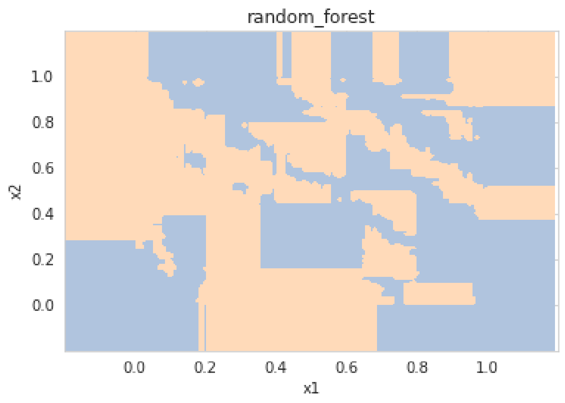

Figure 23: Decision boundary for RF in the square and diagonal boundaries dataset.

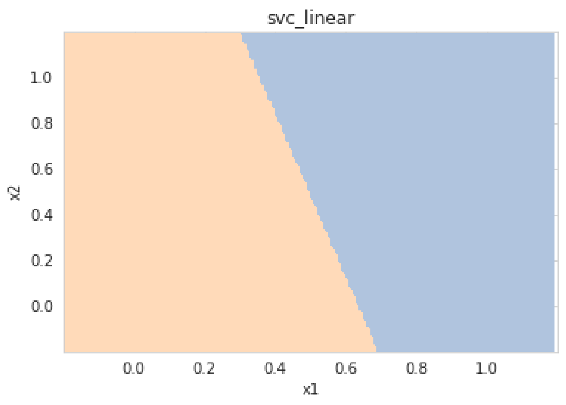

Figure 24: Decision boundary for SVC linear in the square and diagonal boundaries dataset.

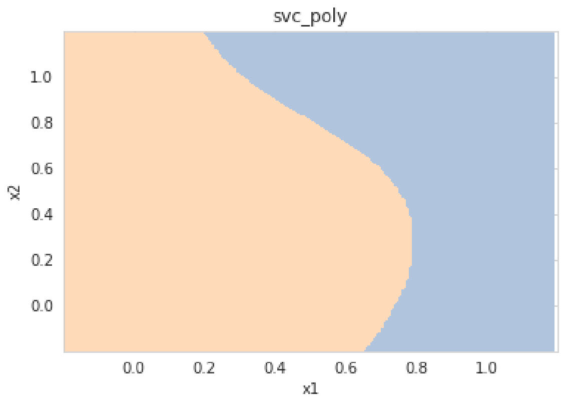

Figure 25: Decision boundary for SVC poly in the square and diagonal boundaries dataset.

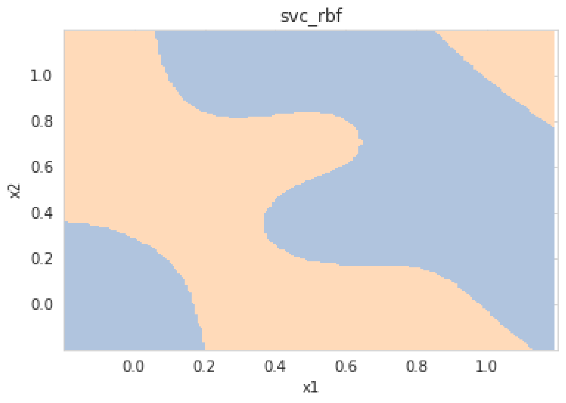

Figure 26: Decision boundary for SVC rbf in the square and diagonal boundaries dataset.

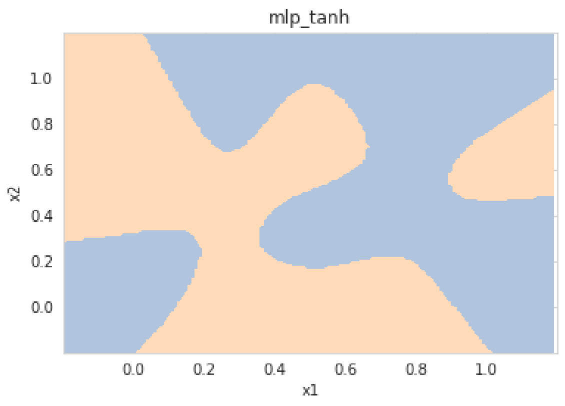

Figure 27: Decision boundary for MLP tanh in the square and diagonal boundaries dataset.

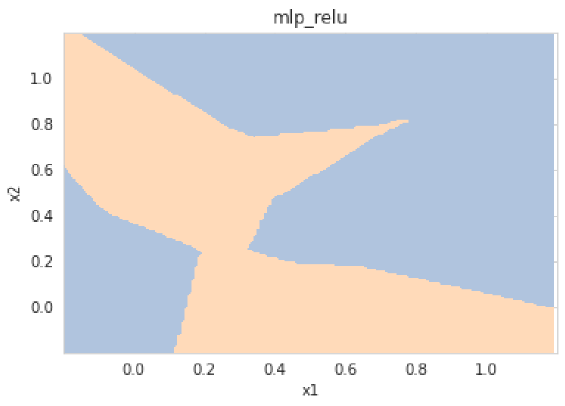

Figure 28: Decision boundary for MLP relu in the square and diagonal boundaries dataset.

3) Circles dataset

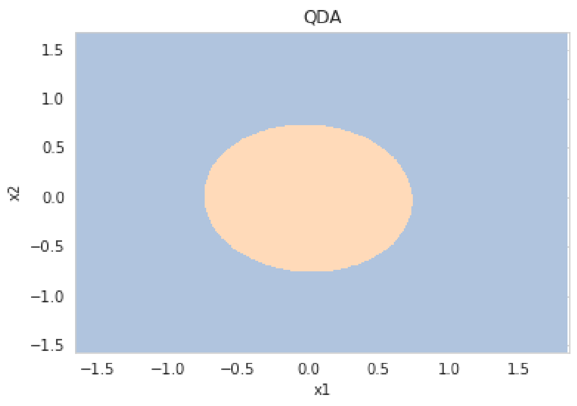

Figure 29: Decision boundary for QDA in the circles dataset.

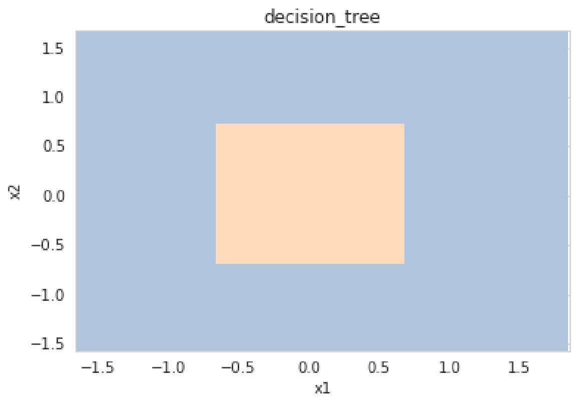

Figure 30: Decision boundary for DT in the circles dataset.

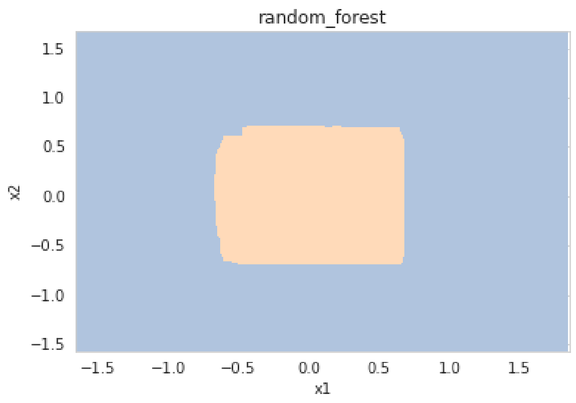

Figure 31: Decision boundary for RF in the circles dataset.

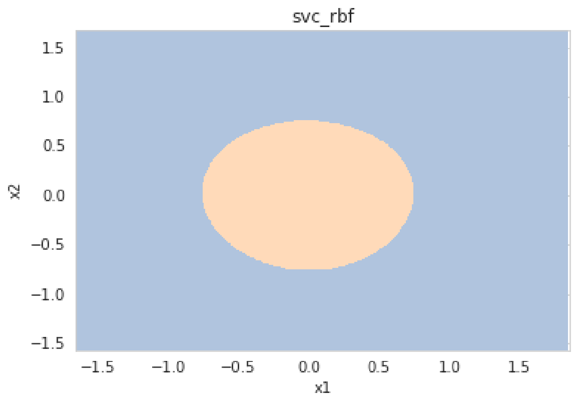

Figure 32: Decision boundary for SVC rbf in the circles dataset.

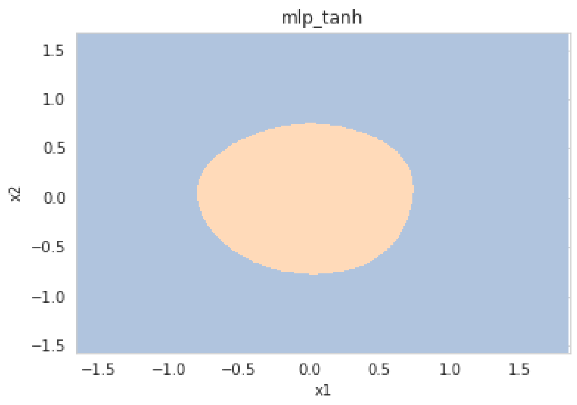

Figure 33: Decision boundary for MLP tanh in the circles dataset.

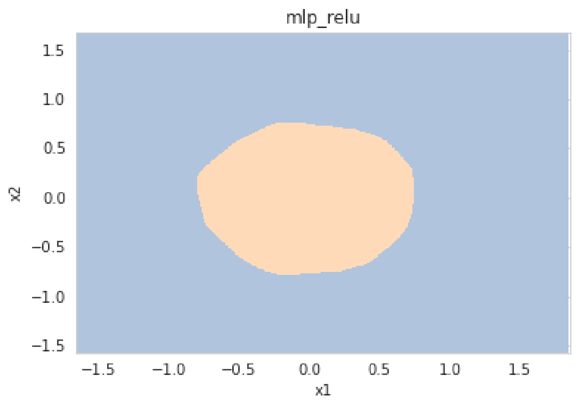

Figure 34: Decision boundary for MLP relu in the circles dataset.

B. Moons dataset

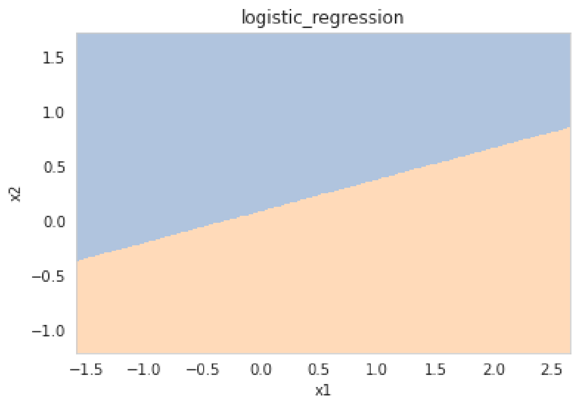

Figure 35: Decision boundary for Logistic Regression in the moons dataset.

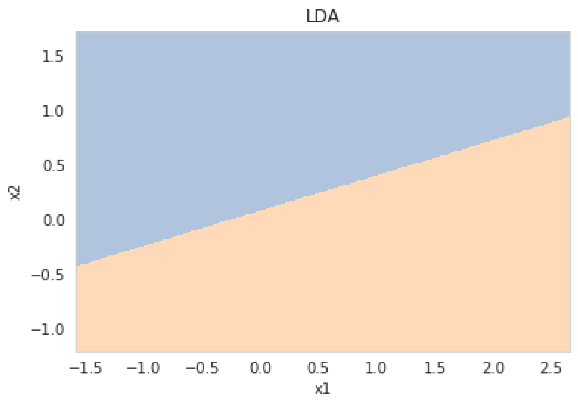

Figure 36: Decision boundary for LDA in the moons dataset.

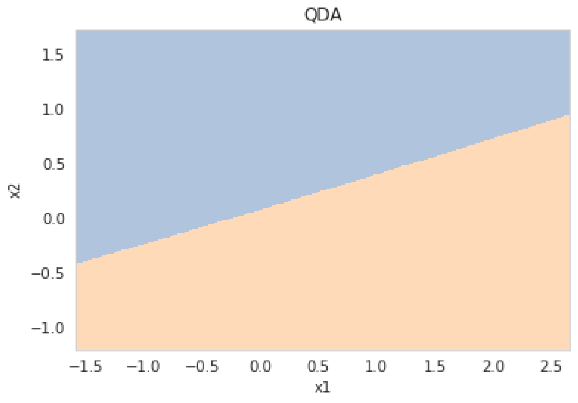

Figure 37: Decision boundary for QDA in the moons dataset.

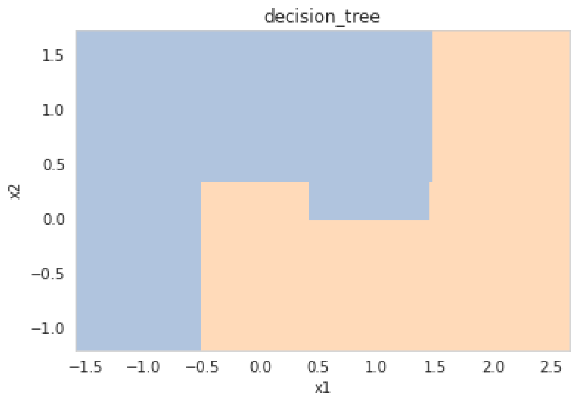

Figure 38: Decision boundary for DT in the moons dataset.

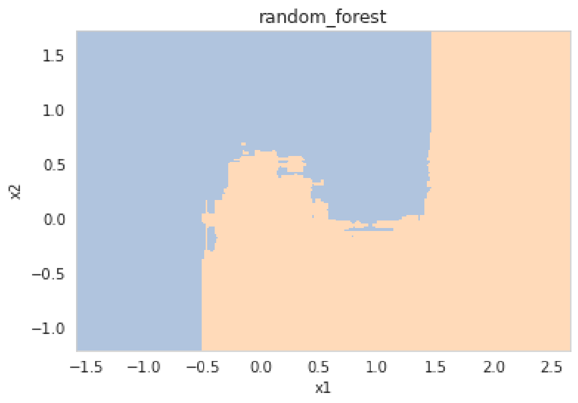

Figure 39: Decision boundary for RF in the moons dataset.

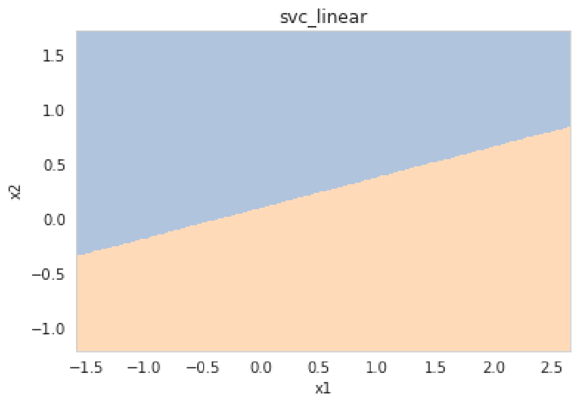

Figure 40: Decision boundary for SVC linear in the moons dataset.

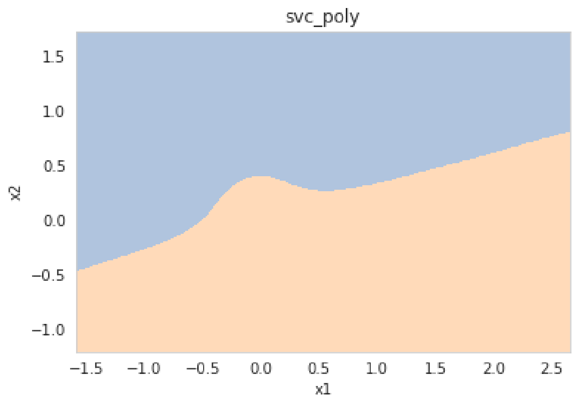

Figure 41: Decision boundary for SVC poly in the moons dataset.

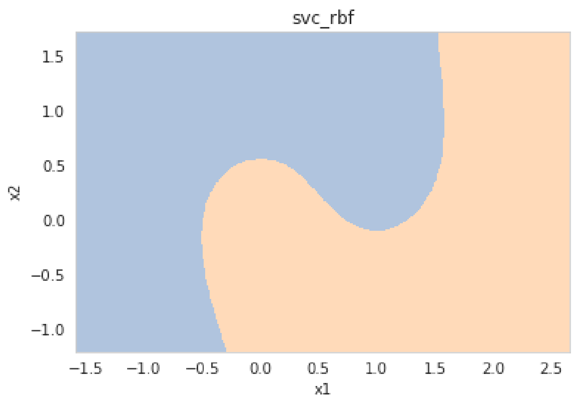

Figure 42: Decision boundary for SVC rbf in the moons dataset.

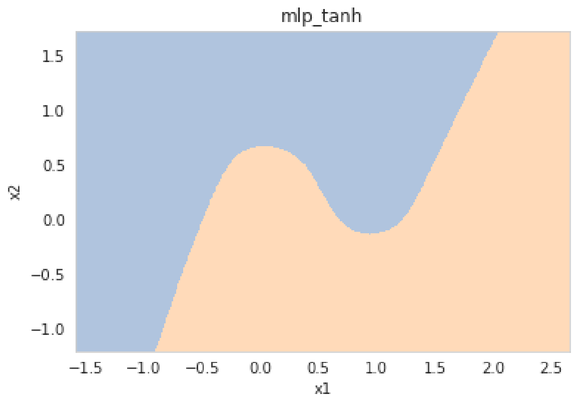

Figure 43: Decision boundary for MLP tanh in the moons dataset.

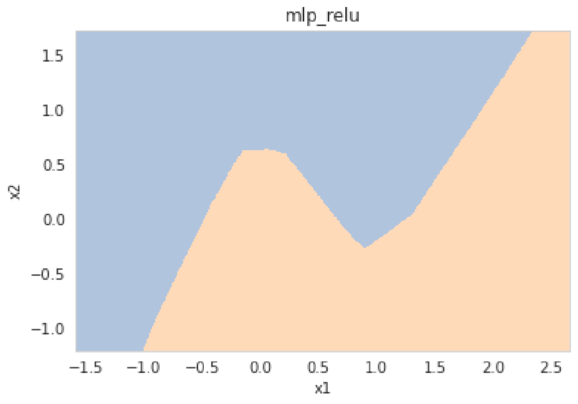

Figure 44: Decision boundary for MLP relu in the moons dataset.

C. Post-hoc Nemenyi statistic

1) 10 Features Dataset

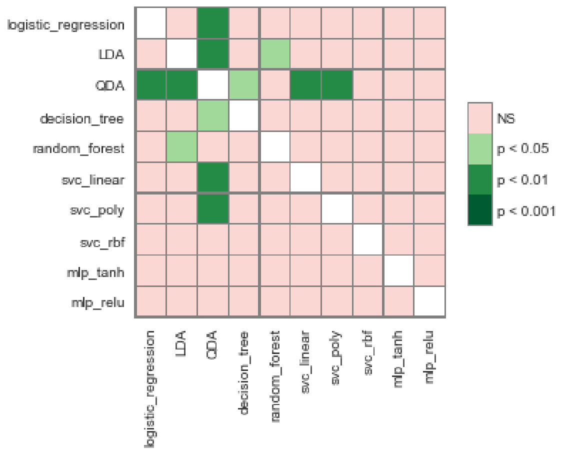

Figure 45: Post-hoc Nemenyi statistic using accuracy for the 10 features dataset.

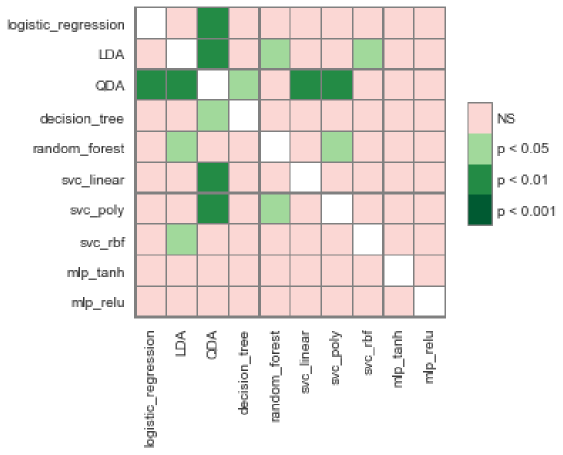

Figure 46: Post-hoc Nemenyi statistic using f1-score for the 10 features dataset.

2) Square boundaries dataset

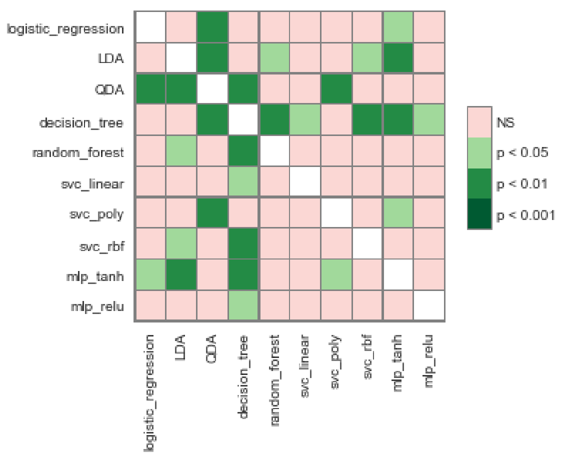

Figure 47: Post-hoc Nemenyi statistic using AUC for the 10 features dataset.

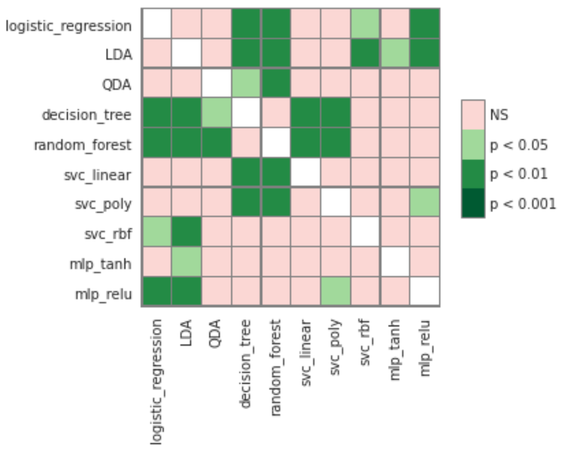

Figure 48: Post-hoc Nemenyi statistic using accuracy for the square boundaries dataset.

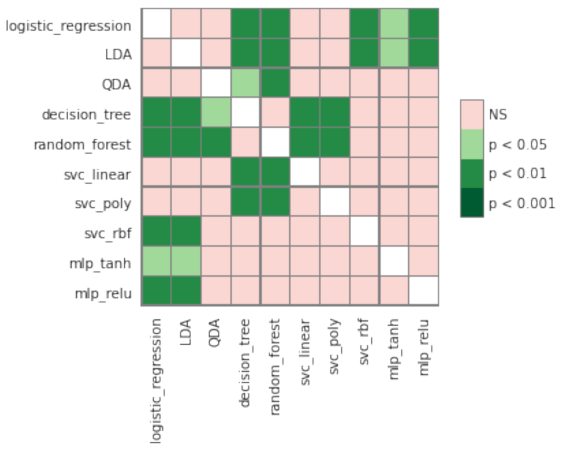

Figure 49: Post-hoc Nemenyi statistic using f1-score for the square boundaries dataset.

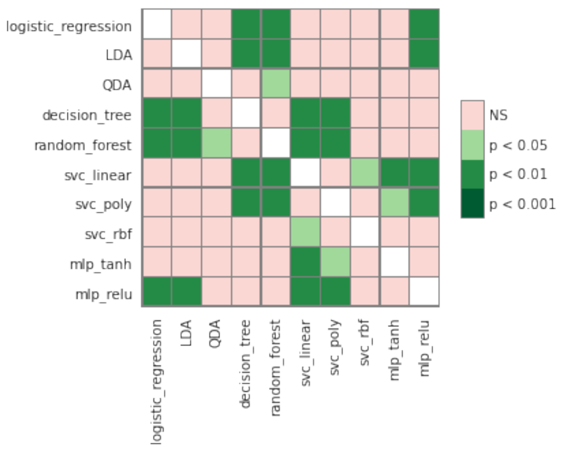

Figure 50: Post-hoc Nemenyi statistic using AUC for the square boundaries dataset.

3) Square and diagonal boundaries dataset

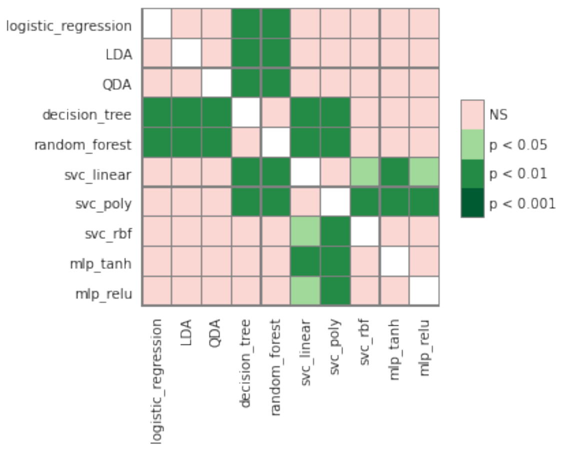

Figure 51: Post-hoc Nemenyi statistic using accuracy for the square and diagonal boundaries dataset.

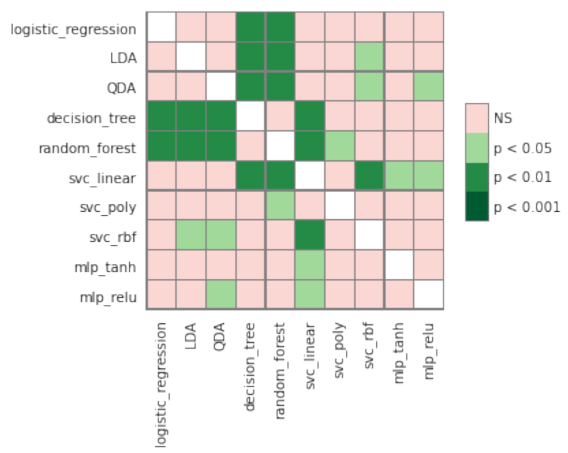

Figure 52: Post-hoc Nemenyi statistic using f1-score for the square and diagonal boundaries dataset.

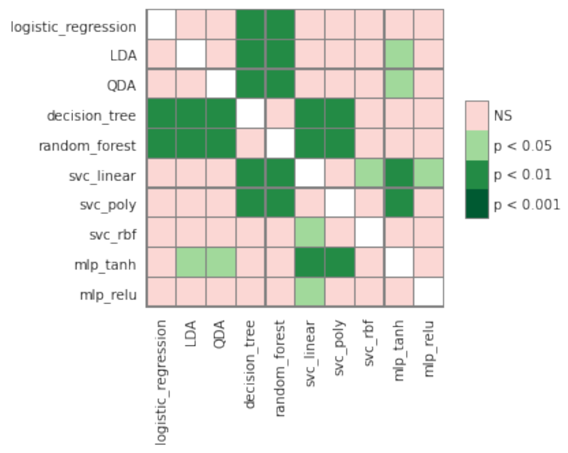

Figure 53: Post-hoc Nemenyi statistic using AUC for the square and diagonal boundaries dataset.

4) *Circles Dataset*

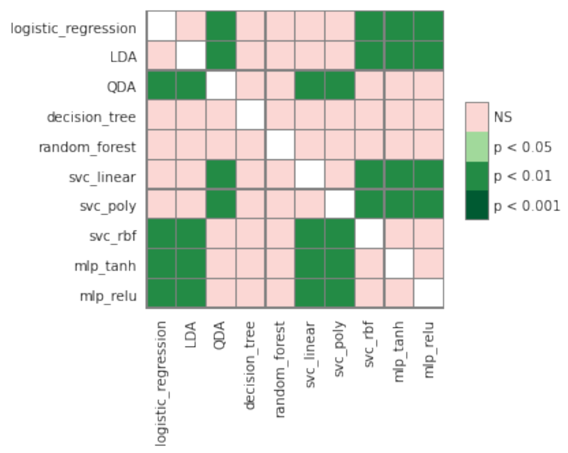

Figure 54: Post-hoc Nemenyi statistic using accuracy for the circles dataset.

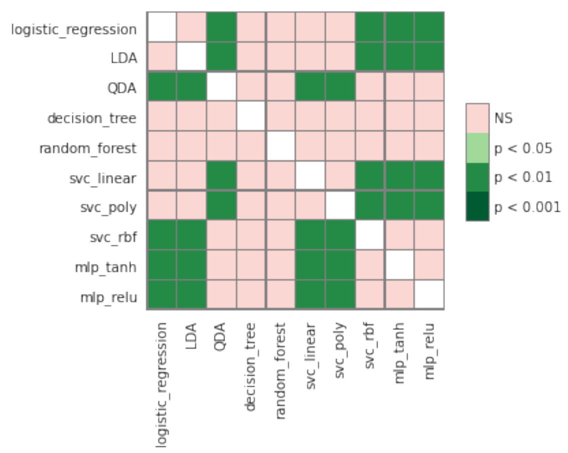

Figure 55: Post-hoc Nemenyi statistic using f1-score for the circles dataset.

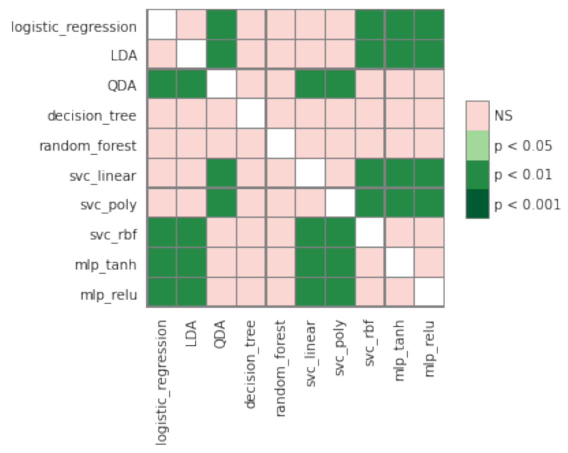

Figure 56: Post-hoc Nemenyi statistic using AUC for the circles dataset.

5) *Moons Dataset*

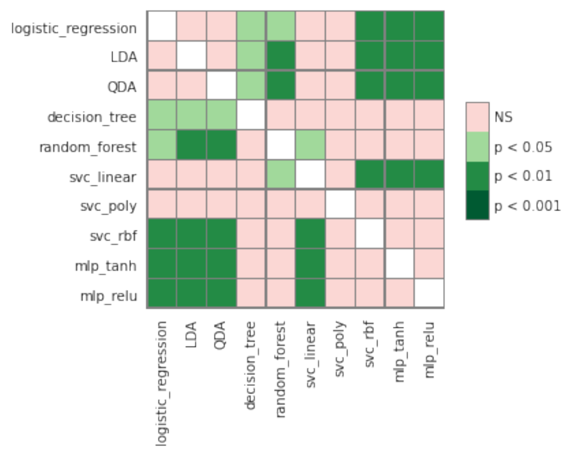

Figure 57: Post-hoc Nemenyi statistic using accuracy for the moons dataset.

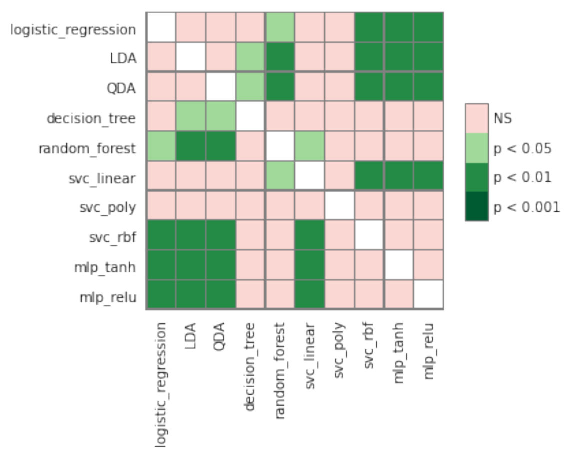

Figure 58: Post-hoc Nemenyi statistic using f1-score for the moons dataset.

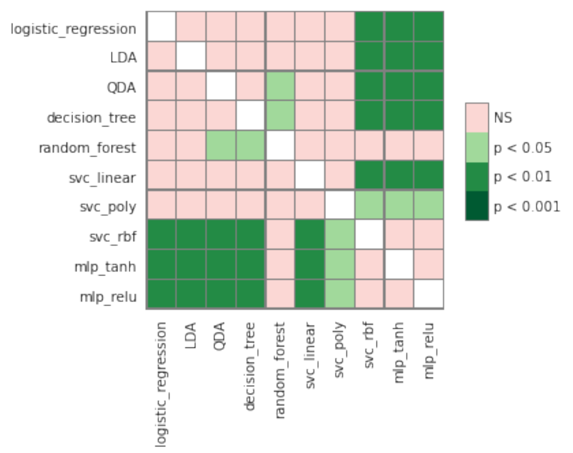

Figure 59: Post-hoc Nemenyi statistic using AUC for the moons dataset.

6) Dense and 15 features dataset

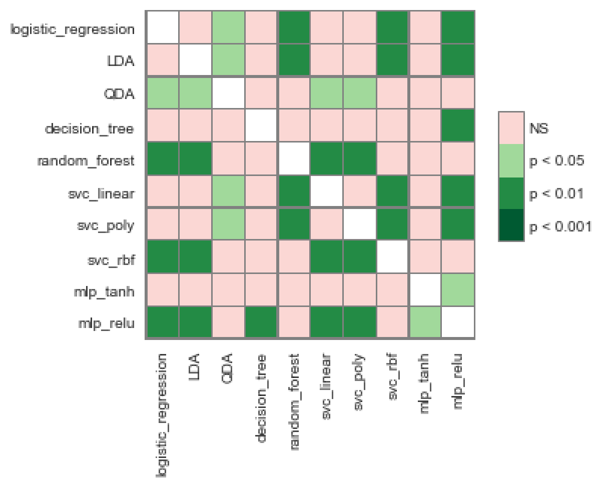

Figure 60: Post-hoc Nemenyi statistic using accuracy for the dense and 15 features dataset.

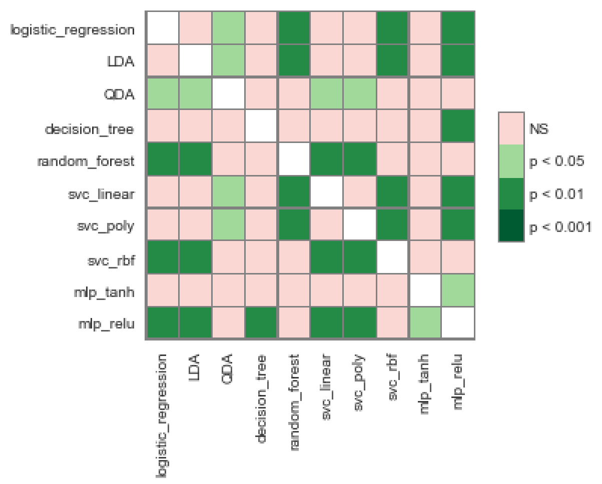

Figure 61: Post-hoc Nemenyi statistic using f1-score for the dense and 15 features dataset.

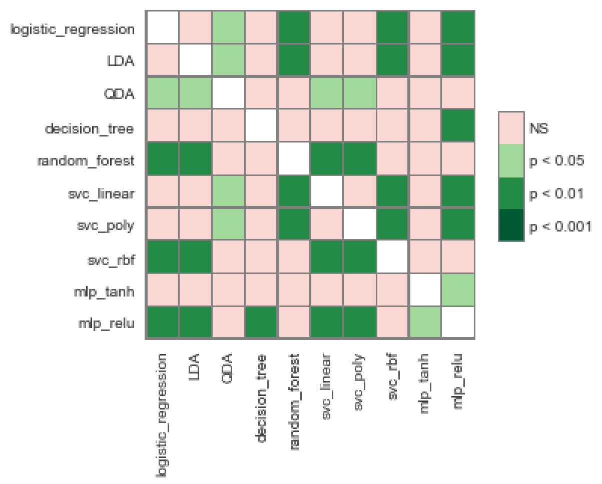

Figure 62: Post-hoc Nemenyi statistic using AUC for the dense and 15 features dataset.