Report: Will it rain tomorrow?

Helena Binková and Aline Brunner

Lausanne, December 23, 2021

Abstract

The aim of this report, written within the framework of the *Miniproject BIO-322*, is to present the thought-process and decisions followed to solve a supervised machine learning classification problem. In the end, we could reach some very good results thanks to neural networks.

Contents

1 Introduction

The aim of the project is to predict the probability of rain on the next day in Pully, given the meteorogical measurements in different stations in Switzerland. This project takes the form of a kaggle competition. The data is available here (with access only): https://www.kaggle.com/c/bio322-will-it-rain-tomorrow/overview. The submission should be a CSV file containing the probability of raining for each row of the test data.

2 Pre-processing and Visualization

We first pre-processed the data. We had to deal with missing values; after some trials we could conclude that filling them (instead of dropping the rows with missing data) gives the best results.

We also had to standardize the data for some machines; to do this we created a machine on the (filled) training data and applied it to the training and test sets to be consistent.

After a few trials and having considered the results given by the tuning of λ on logistic regression, we deduced that there are many predictors that have no importance. We then made a lasso regularization graph (see Fig. 1), and decided to drop out all predictors with a $\log(\lambda)$ smaller than -8, which means that they aren't relevant enough. This corresponds to dropping out 130 predictors.

To visualize the training data, we plotted the standardized according to the first two components of *principal component analysis* (see Fig. 2). No obvious conclusion can be directly drawn from the graph because of the high density of predictors.

After selecting the 5 most important predictors using lasso regularization as just mentioned, we made a correlation plot (see Fig. 3). This showed us a linear correlation between these predictors.

3 Machines

3.1 Introduction and generalities

As said above, we are facing a classification task, so we have to use machines that are able to perform classification. We are presenting below the different types of machines we tested.

The MLJ machines have two prediction modes, which we are going to use: *predict* to obtain the values from the fitted probability density function and *predict_mode* to obtain a binary class output (*true/false*). The first mode is used to submit the data on Kaggle and the second to compare the predictions to the training labels.

3.2 Linear Method

3.2.1 Multiple Logistic Regression

This linear method is an efficient method for classification. It can be regularized with Ridge regression (L2 regularization) or Lasso (L1 regularization). The parameter λ, used in both methods, can be tuned inside the logistic classifier. We used 10-fold cross-validation.

3.3 Non-Linear Methods

3.3.1 K-Nearest-Neighbor Classification

K-Nearest-Neighbor Classification is a non-parametric classification method. We tuned K, the number of neighbouring points, using 20-fold cross-validation. KNN-classification being less effective for high dimensional data, we chose to reduce the amount of predictors and selected the most relevant ones given by the L1 regularization (see 2. Pre-processing and Visualization). We also normalized the data on the same scale (between 0 and 1). These techniques allowed us to increase the AUC measurement from 0.90166 to 0.91503.

3.3.2 Random Forest Classification

Random Forest classification is a method constructed on a *decision tree algorithm*, where a high number of trees increases the performance. We therefore tuned the hyper-parameter *n_trees* using 20-fold cross-validation. The additional tuning of the maximum number of splits (*max_depth*) and of the minimum number of samples per split (*min_samples_leaf*) allowed us to reduce the flexibility of the method, and thus avoid the risk of over-fitting. We tried to pre-process the data by reducing the predictors number. However, the results were worse in terms of AUC measurements.

3.3.3 Neural Network Classification

Neural networks are good at finding patterns and correlations in the dataset. Here we use MLJFlux to implement neural networks. There are many ways to build a neural network. We can build the different layers manually

and indicate how many neurons we want in each layer (this can be done with a builder called *MLP*, or by using *Chain()*). We can also build a neural network containing a single layer with the builder called *Short*. The other tunable parameters are α (indicating if l2 is used ($\alpha = 0.0$) or l1 ($\alpha = 1.0$) or a mix of both (between 0 and 1)), λ (how strong to regularize the data), the number of epochs and the batch size (corresponding to the number of training samples that will be propagated through the network per epochs). For both builders, we can tune the number of neurons per layer and visually compare the results between the number of hidden layers.

4 Results

4.1 Summary table of different machines results

Type of machine	**Results of tuned hyperparameters**	**Error rate**	**AUC**
Multiple Logistic Regression penalty = l1	*lambda* = 4.6416	0.12342	0.92499
Multiple Logistic Regression penalty = l2	*lambda* = 125.9921	0.12185	0.92589
K-Nearest-Neighbor Classification	$K = 25$	0.15270	0.91503
Random Forest Classification	*n_trees* = 1950 *max_depth* = 100 *min_samples_split* = 10	0.00503	0.92420
Neural Network Classification builder = Short, regularized data	*epochs* = 26 *n_hidden* = 25 *batch_size* = 80	0.11209	0.92707
Neural Network Classification builder = MLP, regularized data	*epochs* = 35 *hidden* = (100,40) *batch_size* = 32 (not tuned on this run)	0.0	0.92350

5 Conclusion

The last step was to choose the two best models for the Kaggle competition. Our first criterion was to avoid overfitting both the training data and the available test data. The retained models were a multiple logistic classifier, a neural network with a single hidden layer (builder *Short*) and a neural network with two hidden layers (builder *MLP*).

The obtained mean cross-validation AUC measurements were not necessarily following the same trends as the kaggle results. We were then faced with a difficult choice: which measure was it better to rely on to choose the best model? Because of the multiple measures combined to obtain it, we decided to mainly rely on the CV-AUCs. However, we did not completely discard the Kaggle results for model comparisons, as the differences between them were significantly higher than the differences in CV-AUCS.

To further answer this question, we plotted boxplots of the AUC per-fold to estimate the best performing model. We had to balance between the best mean or the lowest variance : this led us to the famous bias-variance trade-off problem (see Fig. 5). We noticed furthermore that trying out different CV-folds influenced the variance for a same tuning.

To increase our chances to perform well, we decided to take one reliable model with low variance (even though the mean was slightly lower), and a second model that has a higher mean, but simultaneously a larger variance. This second model is riskier, as we could obtain very low as well as very high results, all due to this high variance.

The last retained models are a neural network with a single hidden layer, trained on cross-validation 20-folds and a two-hidden layers neural network trained on 10-folds CV.

6 Figures

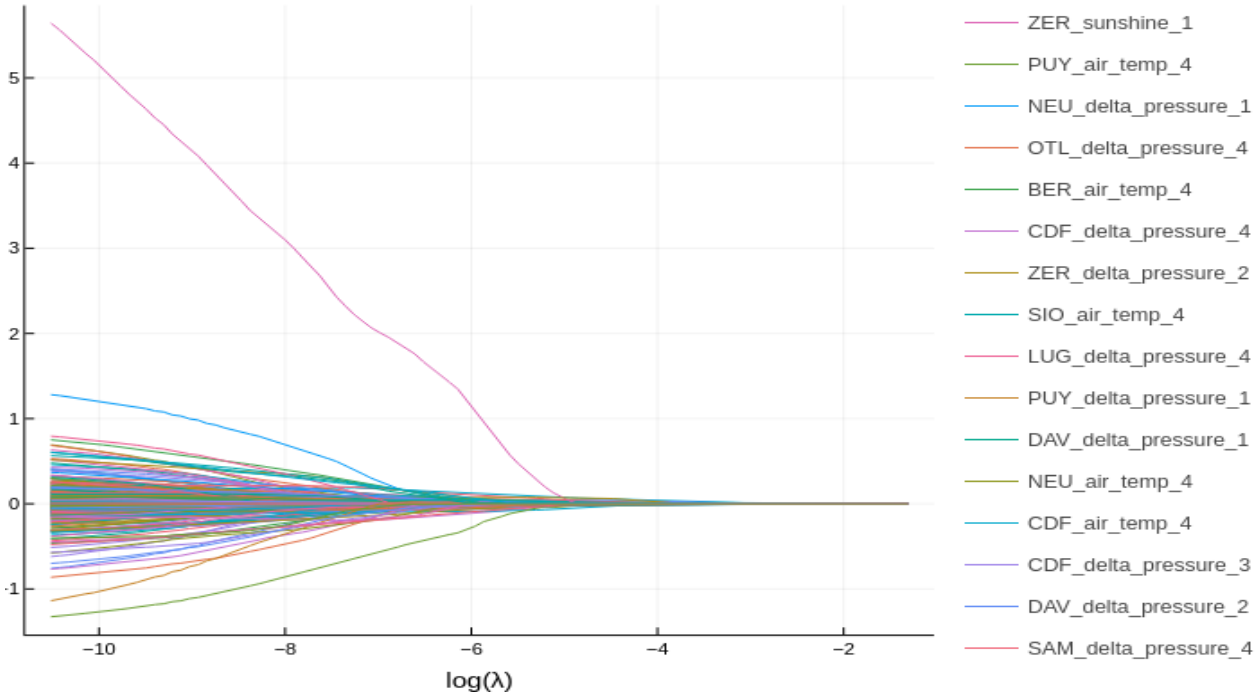


Figure 1: Lasso path. The predictors written on the right are ordered from the most impportant to the less important. We chose to show only the first 18 ones for more readability.

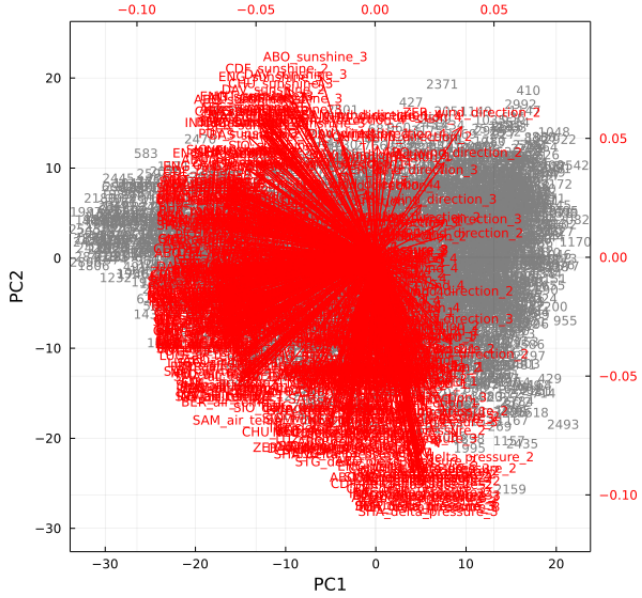


Figure 2: PCA

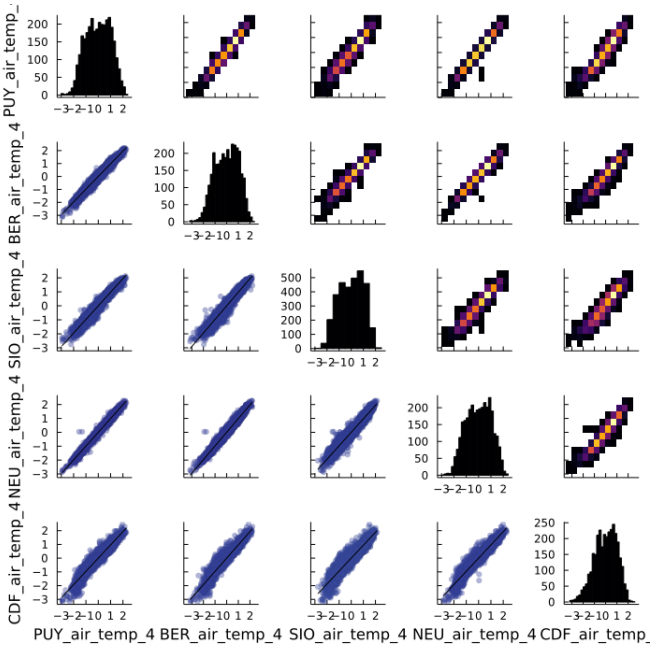
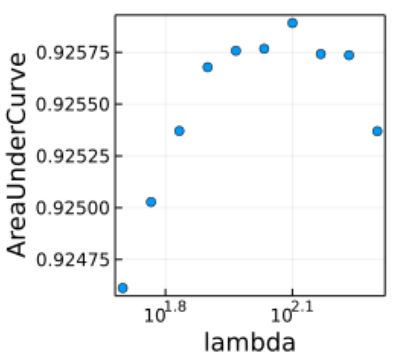
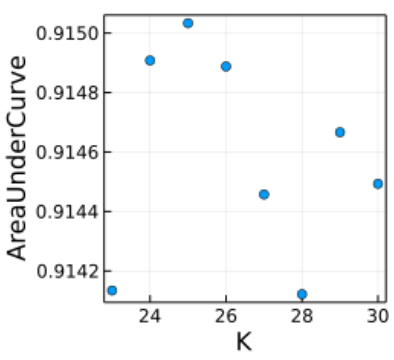


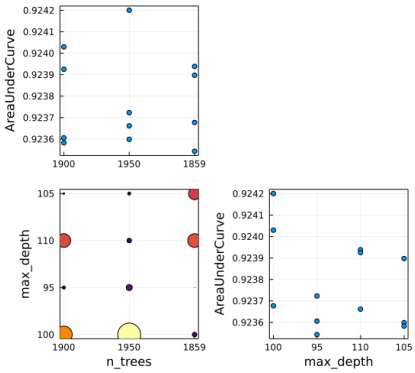
Figure 3: Correlation plot



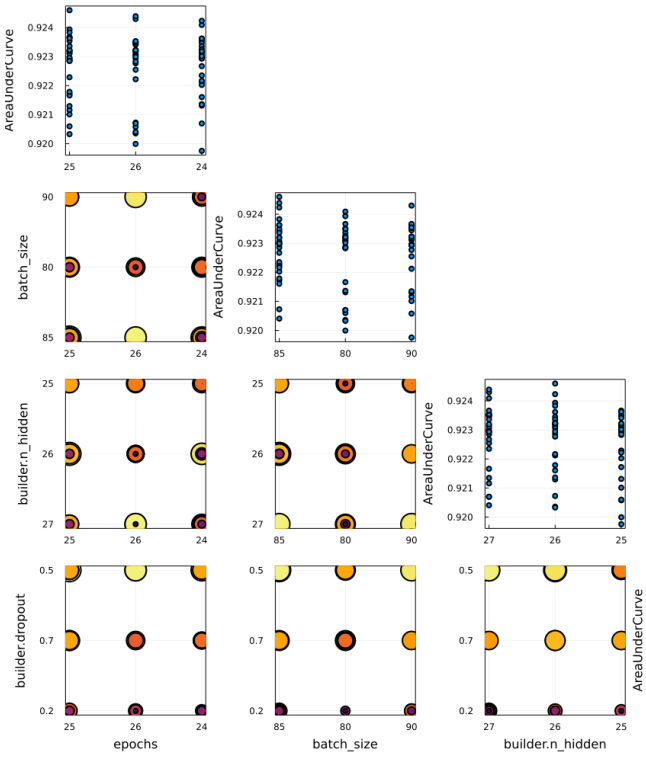
(a) Multiple Logistic Regression



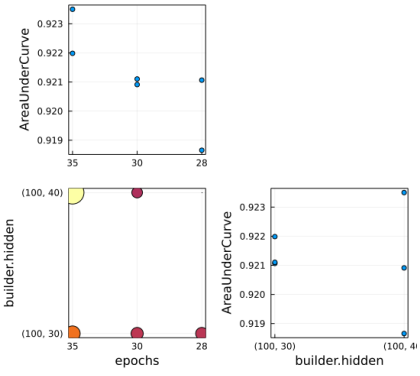
(b) KNN Classification



(c) Random Forest Classification



(d) Short Neural Network



(e) MLP Neural Network

Figure 4: Machine tuning plots

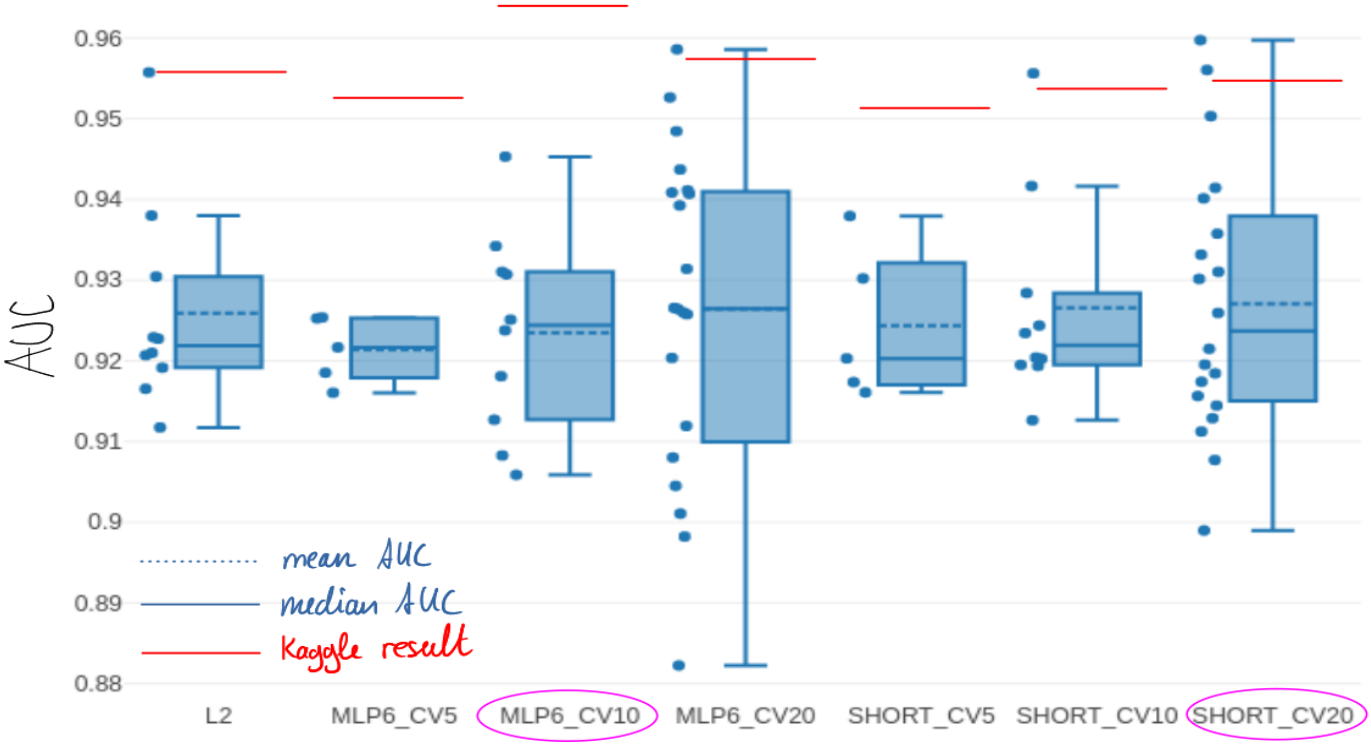


Figure 5: Training AUC boxplots