Report : will it rain tomorrow at Pully ?

1. Introduction

The task that was given aims at coding a machine that computes the probability of rain the next day in Pully.This prediction should be done based on the data collected over a few days from several meteo stations in Switzerland.

Looking at the training set and in particular "precipitation_nextday", the output of the training data, it was concluded that classifiers were better fitted for the problem. Missing data points were found in the training set, which highlighted the necessity of cleaning the data beforehand. This would involve selecting relevant columns, performing feature engineering - thus optimizing the training set.

2. Cleaning of data, visualization and exploration

In order to clean the data, two different methods were tested. First, any missing values were removed from the training set. This resulted in a reduction of the set by half. Therefore another mehod was sought.

The second option was to replace missing data by the mean of the inputs present in the same column and was performed using the machine FillImputer(). This allowed to keep the size of the orginal data set.

Standardization of the inputs is important for classifier models to perform well. The machine Standardizer() was thus implemented on our set. A problem occured during this process, the standadization of some data resulted in NaNs throughout the whole column and caused errors in our machine.This error was due to a standard deviation that was too small in particular columns. Therefore columns such as "sunshine" in the trainnig set were exempted from the process. For the test set, more columns were taken out for the same reason.

At the begining of the exploration, some regressors were implemented to predict the rain and provided acceptable results (around 0.945 on kaggle). However, the classification models were best suited for the problem.

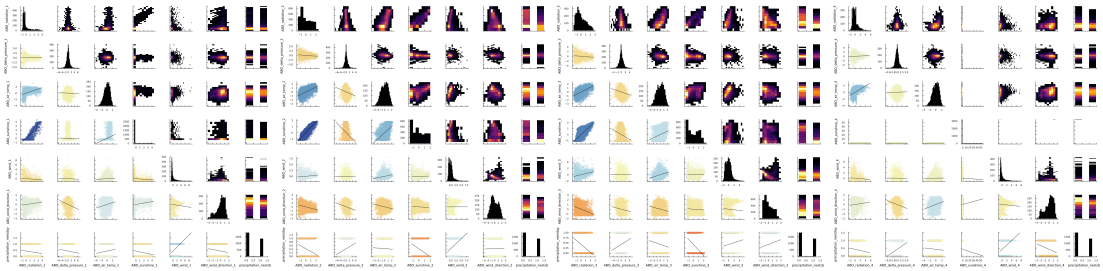

Figure 1: correlation plot after standardization for 1 (ABO) station, in each part of the day (1,2,3,4)

Looking at the data with correlation plots , some parameters collected seem to agree with the output "precipitation_nextday". No feature plotted against this parameter seems significantly non relevant and could be removed from the data set. Indeed, lighter colors can be seen on the left bar (corresponding to no rain or 0 probability) of the righthand histograms as opposed to the right one.This agrees with the last plot that presents a higher bar for 0 probability of rain than for the second one.

As seen on plot 4, the parameter "sunshine" shows - for some stations on particular days - values with almost no standard deviation which explains why these columns were exempted from standardization.

3. Classifier models

In general, classifier models were constructed using self tuning machines to be able to find the parameters that predicted best the rain. Nonetheless, the test set does not provide outputs, which could allow to test the machines on unseen data. Thus, using self-tunnig machines could result in an overfit on the training set. To avoid such problem, cross validation was used with 5 to 15 folds.

The models coded on julia are : Logistic classifier, KNN classifier, RandomForest classifier and NeuralNetwork classifier.

4. Linear model : Logistic classifier

The best linear model - Logistic classifier - was chosen as it is a commonly used linear method which gives rather accurate results.

Cross validation with 5 folds was used in the self tuning machine to avoid overfitting the data. Gradient descent is also included in the tuning. The tuned parameter is "lambda", the penalty of the logistic model. This lambda allows to reduce the flexibility of the model which also represses the overfitting. Finally, a goal was established for the tuning grid, to be more precise. The best type of regularisation found by tuning was "l2" and the optimal value of lambda was 153.437.

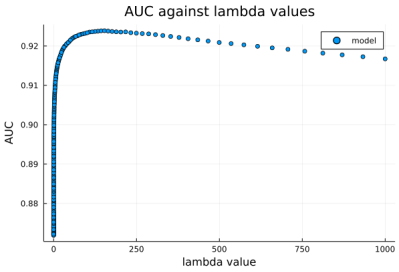

Figure 2: model tuning for the logistic classifier

5. Non-linear model : neural network classifier

NeuralNetwork classifier was chosen as the best non-linear model. Indeed, it is a model that can be tuned in several different ways wich allows to best construct the model to particular tasks.

Initially, single-layer neural network was tested and later on, multi-layer was experimented with 2 and 3 layers. With single-layer, dropout, nhidden, batchsize and epochs were tuned at the same time and afterwards optimised by tuning each parameter at time for time effciency. Since 2 layers seemed optimal, a relatively high batch size and a low epochs were tuned in order to avoid overfitting. The structure of the NeuralNetwork is composed of 130 nhidden in each layer to allow a better prediction of the output. To obtain such optimal number, several values were tested. Batchsize - the number of training samples - and Epochs - the number of cycles through the whole trainnig data set - were tuned (respectively giving 90 and 10) to better optimize the output of the NeuralNetwork. Finally, a high cross validation (15) was used to prevent overfitting.

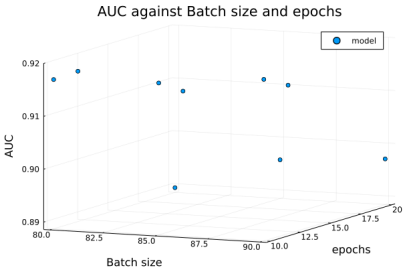

Figure 3: model tuning for the Multilayer Neural network

6. Conclusion

Feature engineering was performed on our input data – replacing missing data by the mean and standardizing. Then, multiple classifier models were tested with tuning of the parameters and cross validation. Out of these multiple models, Logistic classifier – as a linear model – and NeuralNetwork classifier – as a non linear model – were chosen as they best fitted the data. To find the bounds for the parameters that produced the optimal AUC, few values were tested at a time to limit the computation time of the code.

Finally the best results for the test were found for the Logistic classifier was 0.95418. The non-linear method, NeuralNetwork classifier, gave the best result : 0.96068.