Will it rain tomorrow ?

Will it rain tomorrow is a Machine Learning project, where we aim to predict whether it will rain in Pully on the next day. To solve this question, a training dataset was given. The dataset is constituted of multiple measurements at different weather stations. The project is a supervised learning problem, more precisely a binary classification task, as we are trying to predict whether it will rain or not on the next day. The dataset is the input and the output is the response to "*Will it rain tomorrow ?*". In this project, we developed multiple methods that can solve this classification problem. All the methods do not obtain the same performances in responding to the problem. To evaluate our methods, we used the area under the receiver operating characteristics curve (AUC). A random classifier will lead to an AUC equal to 0.5 whereas the better the classifier is, the closer to 1 the AUC will be.

1 - Exploration and visualization of the data:

In this project, three datasets were provided: the training set, the test set and an example of a submission file for the competition[1]. The training dataset was used to fit our models. It is composed of 528 predictors and one output column. The predictors represent 6 different measures made at the four quartiles of the day in 22 different Swiss stations, and the output is a boolean value telling whether it will rain in Pully the next day. Each data point represents a different day. The dataset is composed of 3176 data points.

The dataset contained missing data. To avoid problems generated by the empty cells, we constructed two new data sets using two methods: dropping the rows containing missing data or imputing the missing data with the median of the predictor (drop and med). Both data sets were used with each model, and the better one was chosen by comparing the returned AUC.

To obtain a reliable AUC, we divided our data into a training set, a validation set and a test set. As often as it was possible, we use cross-validation to generate our training and validation set of data. Each sub data set was generated by randomized sampling among the training set. The AUC were always performed on the test set, which wasn't used to fit the models.

Some models work better with standardized data, such as multilayer perceptrons (MLP). We standardized the two data sets (drop and med), and divided them as described above. We then fit each model drop and med data sets, standardize or not, and pick the best one out of the four.

In order to assess the predictors accounting for most of the variance of the output, we used principal component analysis. This allows us to assess how many predictors are needed to predict a particular fraction of the variance, and which are these predictors that are more important in predicting the output.

We didn't observe particular correlation in the data that could allow us to construct functions such as feature engineering methods.

2 - Linear methods

When using a **logistic classifier** model, without regularization, we got an AUC of around 0.79 with the drop data and 0.88 with the med out data. The AUC was computed for both model on a test set. Then, we used l2 and l1 regularization to avoid hyper flexibility and overfitting of the data. We needed first to standardize the data for better Ridge and Lasso performance. We trained both datasets with both regularization on, and it turned out that l2 regularization on the drop dataset had the best performance on the test set with an AUC of 0.917. To tune our regularization, we used a logistic scaled lambda over a range of values.

---

[1] https://www.kaggle.com/c/bio322-will-it-rain-tomorrow

3 - Non-linear methods

K Nearest-Neighbors (kNN) is a non-parametric method that uses the classification of the k-nearest data points to perform classification. We tuned the hyperparameter k by looking at the AUC values using cross-validation on k between 1 and 50.
We found that when dropping the rows containing missing data, our best model had an AUC of 0.904 with K = 27. When using the filled out data, the AUC was 0.898 for k = 25. kNN is fast to fit but slow to make predictions. It works better than our classical linear models, but less than regularized linear models.

Our second non-linear model was a multilayer perceptron using the **NeuralNetworkClassifier** method of MLJFlux. There is no explicit recipe for multilayer perceptron tuning. The Relu function seems to be the most adapted to our classification problem. Then we used a loop where we could increase some parameters and save the AUC of a given model in order to plot the performances of each combination (learning curve). The final chosen parameters were n_hidden = 200, dropout = 0.1, sigma = relu, batch_size = 1000, epochs = 300 and optimizer = ADAM. This combination gave us an AUC of 0.912. But we think that most of the better AUC are caused by too flexible neural networks.

We also used random decision forests (**RandomForestClassifier**). In classification tasks, the output of a random forest is the class selected by most trees. Random decision forests address the problem of decision trees overfitting their training set[2]. However, they have a less accuracy than gradient boosted trees (see below). We did not use standardized data as it did not achieve better results. We tuned the hyperparameter n_trees (number of trees) of our model and found that our best model was with 1200 trees. Furthermore, we found that when using median-imputed data, we had a better AUC (0.927) than with the dropped data (0.90). The final chosen parameter was n_tree = 1200.

XGBoostClassifier (extreme gradient boosting) is an implementation of a gradient boosted decision tree. We tuned our model using the following hyperparameters : learning rate (eta), number of rounds (num_round), and maximal tree depth (max_depth). The number of rounds determines how many gradient steps should be taken (like epochs in neural networks). One must be careful with gradient boosting trees: with a high number of rounds, one usually overfits the training data, and with lower values one does too early stop.
When tuning the hyperparameters, we saw that we got a better AUC with the missing values filled with the median (0.9297) than with the missing values dropped (0.9176). This model worked better with our data standardized (AUC of 0.93 with standardized median imputed data). The final chosen parameters were eta = 0.1, num_round = 500, max_depth = 6. This tuned model was our best non-linear model.

Conclusion

In order to predict whether it will rain or not the next day in Pully, using the given training data set, we found that our best linear model was a logistic classifier with L2 regularization, and our best non-linear model was a tuned XGBoost classifier. It is difficult to assess how much our models are overfitting the data they have been trained on. Every model were compared using a test set, but it still possible that we didn't pick the best model we computed. Our best model have a score of 0.958 in the competition.

[2] Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1, No. 10). New York: Springer series in statistics.