Introduction

This project aims to predict if it rains in Pully on day 2, given a collection of weather related data (predictors) from day 1. For this we are provided with a training dataset, containing predictors and the information if it rained on the following day and a test dataset which only contains predictor values. The machines trained and fitted on the training set are then used to predict the result of the test dataset generating the output that is then uploaded to Kaggle.

Setting the environment

In order to have access to quick customizability we decided to copy the `Manifest.toml` file from the course into our folder in case we needed a specific package. However in the course of the miniproject we did not encounter such a situation. Input files, namely the `testdata.csv` and `trainingdata.csv` file are stored in `./res/`. Once the code block to fit a certain machine has been executed it is stored in `./machines/` following the naming `mach_i.jlso` for the ith machine. Prediction from the different machines are stored in `./out/` while the naming of the predictions follows `output_i.csv` for the ith machine. Further correlation plots are stored in `./figures/`.

Data pre-processing

The test and training data need to be treated. Firstly missing values are complemented by using `FillImputer()`, which replaces missing values by the mean of the corresponding predictor. Next all predictors that have standard deviation zero are sorted out, since they don't contain any useful information. Finally each predictor is normalized in order to avoid things like different units (for example m/s vs km/h) having an impact. Additionally correlation plots were created, comparing certain predictors.

Machines

The approach chosen is to manually change hyperparameters and upload the resulting outputs to the Kaggle website for evaluation since utilizing tuning packages adds an additional layer of complexity. For each approach a new machine is defined. Intuitively machine i uses model i. To enable reproducibility before defining and fitting the machine the random seed is specified (`Random.seed!(10)`) controling any potential random processes used in the functions.

model #	type	layers	nodes per layer	optimizer	sigma	lambda	epochs
1	LC	-	-	-	-	-	-
2	NNC	3	64	ADAMW	relu	0	1000
3	NNC	3	64	ADAM	relu	0	1000
4	NNC	3	64	ADAMW	relu	0	2000
5	NNC	6	32	ADAMW	relu	0	1000
6	NNC	3	128	ADAMW	relu	0	1000
7	NNC	4	128	ADAMW	relu	0	1000
8	NNC	3	128	ADAMW	sigmoid	0	1000
9	NNC	3	64	ADAMW	tanh	0	1000
10	NNC	3	64	ADAMW	softmax	0	1000
11	NNC	3	64	ADAMW	softmax	0.2	1000

Table 1: Table showing characteristics of the different models employed. LC: Logistic Classifier; NNC: NeuralNetworkClassifier

Results

Linear model & correlation plots

Our linear approach with model 1 left us with a good result early on. After some minor changes we hit a score of 0.92440 (see output.csv upload on kaggle). We initially wanted to use the correlation plots to check which parameters influenced the prediction the most or if there was parameters that either didn't help in predicting or went hand in hand with another parameter, however because we had no issue with the time it took to execute the code blocks we decided to not further investigate into the predictor significance but instead use the interity of predictors that passed the cleaning process.

Multilayer perceptrons

Using the more sophisticated approach of using mulitlayer perceptrons it was decided not to implement features since it is believed that a neural network with enough freedom (in the sense of enough nodes and layers provided) will do a better job of learning features in the course of fitting.

Layers and nodes: Different attempts have been made to evaluate if better results are obtained when using less layers with more nodes per layer or more layers with less nodes per layer. For example machine 5 used 6 layers with 32 nodes each, which resulted in a score of 0.94543. In contrast machine 6 used 3 layers of 128 nodes each and scored 0.96159, which is the highest score we obtained (interestingly even higher than using the same model but with 4 layers instead of 3). It can thus be stated that the amount of nodes per layer seems to have a more significant impact on the score than the number of layers itself.

Optimizer: With machine 4 a short attempt was made to see if `ADAM` is a better option as an optimizer than `ADAMW`. However due to obtaining a worse score with `ADAM` it was decided to continue with `ADAMW` for future experimentations.

Sigma: For the activation function of the neurons relu, tanh, softmax and the sigmoid function were applied. Model 10 using the softmax function and scoring 0.95201 achieved a slightly worse result than model 2 with 0.95326 (the models being identical to each other with exception of the activation function). Unfortunately other activation functions could not be evaluated.

Lambda: The attempt to utilize a regularization to "encourage" parameters to tend to 0 was very unsuccessful scoring only 0.62843 (machine 11). This idea was thus aborted. However a score between 0.95 and 0.96 seems unlikely to overfit.

Discussion

Since certain changes were made to the way the data is pre-processed and implementing the ancorage of the random seed at a late stage in the project not all submits to the Kaggle website will be possible to reproduce, since they are based on old (and likely falsely pre-processed) data. While the predictions from model 2 should be able to be reproduced, the predicitons from machine 6 will likely not be able to be reproduced.

Conclusion

With machine 2 based on a non-linear method we are able to predict precipitation probability with an accuracy of 0.95326. Machine 2 uses 3 layers of 64 nodes, the `ADAMW` optimizer, the relu activation function, no lambda, and 1000 epochs. Further machine 6 scored the highscore of 0.96159. However unfortunately this result is not reproducible due to a later implementation of `Random.seed!` and different data pre-processing With principle component analysis of the predictors and more varied hyperparameter tuning, by K-fold cross validation, this result could be improved.