MACHINE LEARNING REPORT

INTRODUCTION

The goal of the project is to predict the probability of precipitation in Pully for the next day based on the data from the current day. We dispose of 528 predictors from 22 swiss weather stations. Each station gives us 6 measures at four different times of the day (that we can assume evenly spaced during the day). We have for these predictors the measures of 3176 days (3176 rows).

We will start by exploring our data, trying to have intuitions about them and engineering the features and then we will train and evaluate linear and non-linear models.

DATA EXPLORATION AND FEATURE ENGINEERING

I started by exploring our dataset, which had a lot of missing values, especially in the delta pressure column, but the highest number of missing data in a column was 321, which is approximately 10% of the data in the column. This is a relatively small number, so I filled the missing data with the median value of the column.Using PCA to visualize the data in 2D, I noticed that the two classes (precipitation or no precipitations) are not linearly separable and thus non-linear methods would be best-suited for the task.

To evaluate the importance of each feature and rank them, I used a random forest classifier on the filled dataset and extracted the feature importance. I then used recursive feature elimination with cross validation (RFECV) on the model to only keep the model with the optimal set of features. I was left with 111 features, the best-ranking ones being sunshine in the 3rd recording of the day and wind direction on the 3rd recording of the day. I then used these features to construct a data set that I'll call D1.

I included 88 new features in the dataset (three new features per station), making sure they were well correlated with the response variable and not linearly dependent with the variables they were constructed with. They were included in the original filled dataset to create a 591-features dataset that I'll call D2.

The same process of feature selection was applied to D2 to form D3, a dataset including 390 features, deemed the best combination by the RFECV algorithm.

DATASET COMPARAISON AND FIRST MODELS

Of course, the elimination of features improved the computation efficiency of the models, but the initial assumption would be that this increase in efficiency would be at the cost of predictive accuracy. I thus compared the accuracy of two methods on the three datasets: logistic classifier and gradient boosting. The area under curve on the cross-validation for each benchmark model can be found in the table below

	D1	D2	D3
Logistic classifier	0.901	0.915	0.891
Gradient boosting	0.934	0.935	0.938

Figure 1: Table of the area under curve measurement on the cross validation of the different models tested

For the logistic classifier, we find that the best results are achieved with D2, which can be surprising as it is the smallest out of the three, but it might be explained by the fact that there is almost no noisy feature in this dataset. In the case gradient boosting, we can see a gradual improvement of the benchmark model's accuracy moving from D1 to D3. This is an encouraging result as we see that the removal of features improves accuracy and performance for this task. Not surprisingly, gradient boosting out-performed the logistic classifier on all datasets.

I then tested a random forest classifier on D3 but soon enough reached a plateau in accuracy and decided to abandon the method because it did not outperform gradient boosting, which is not surprising.

NEURAL NETWORKS

As seen just before, I started by testing linear classification and gradient boosting, getting decent results. After I reached a plateau in performance with gradient boosting, I moved on to building neural network classifier. Considering the last point, I sticked with D3 for the remaining of the project to train these networks. There are many heuristics when it comes to neural network's architecture. As guidelines for my exploration, I tried not to have more hidden neurons than inputs dimension, and didn't go deeper than 2 hidden layers, because any decision boundary can theoretically be approximated with a single hidden layer. I mainly tested two types of architecture, a bottle-neck type where I reduced the number of neurons from the first to the second hidden layer and a constant type where I kept the number of hidden neurons constant for the one or two layers. I sticked to relu as an activation function because I judged not necessary the exploration of other methods, as architecture is the main factor affecting performance.

Neural networks being flexible methods, they are prone to overfitting, which happened very often when I increased epochs or had a lot of hidden neurons. As little as 30 epochs where often found to be optimal when I tuned model and evaluated them with cross-validation. I tried l1 and l2 regularization, but they only lowered the AUC for cross validation, which might be because I already eliminated a lot of features from the dataset and thus already introduced an element of regularization.

I obtained a cross-validation AUC of 0.928 at best but with high variability between sets. The best results were obtained with two layers, the first of 128 neurons and the second of 64 neurons.

A neural network with no hidden layers being a linear method, I compared it to the results obtained with the linear classifier. Using this network on the D3 dataset outperformed the logistic classifier trained on D2. This might be due to the usage of an optimizer

CONCLUSION

The careful crafting and selecting of the information available to us enabled me to train and evaluate increasingly precise machines. Benchmark results were achieved with Linear classification, Random tree forest, gradient boosting, deep and short neural networks. The best results have been produced using gradient boosting and neural networks, although neural networks proved to be a lot harder to tune and prone to easily overfit the data.