Michael Pilcer
Valentin Braux-Guillin

January 2022

REPORT MAP553

Kaggle Challenge - Forest Cover Type Prediction

INTRODUCTION

Our objective is to predict the forest cover type, a categorical variable, from various features related to an area. The metric to optimise is the accuracy on a test set with hidden labels. The main issue is that we only have access to a small training set to build our model. This adds a particularly interesting difficulty to the project as we can not simply rely on classical models to obtain a good performance.

1 DATA EXPLORATION

We started with some data exploration of training and testing sets. We indicate some shape and feature information about both sets in the following table. We can indeed observe that training data represents less than 3% of testing data in size.

	Train set	Test set
Rows	15k	581k
Columns	56	55
Quantitative Features	10	10
Categorical Features	44	44

The first thing we did was investigate the datasets to check if there was any missing data. The good news is that there was not. We checked by looking for *null* values in the datasets (the obvious way) but we also plotted histograms of all of the quantitative features to verify that there was no missing data that had been filled beforehand, with a constant value for example.

We ran several checks to verify consistency between training and testing data, by plotting and comparing distributions of the features. Interestingly, almost all features had a similar shape across both datasets, but not the area distribution. We found out that while training data was mostly composed of wilderness areas 1, 3 and 4, testing data was exclusively composed of wilderness areas 1, 2 and 3 (and mostly areas 1 and 3). We show both distributions in figure 1. We also checked that data was consistent conditionally to every area.

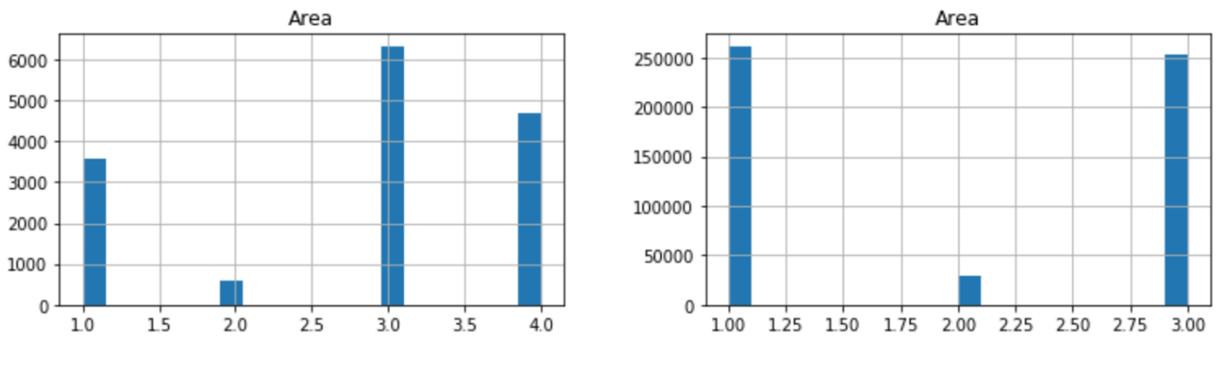


Fig. 1 – *Distribution of area in training data (left) and testing data (right).*

We gave a particular importance to this feature as wilderness area likely had a great impact on the cover type. This gave us an idea for the model, which was to build conditional classifiers depending on the area (in addition to a global classifier).

Concerning the main focus of our study, *i.e.* forest cover types, we observed that the training set was balanced. But when we looked area by area, we saw that each area had its own cover types, some like Cottonwood/Willow (cover type 4) being present in only one area. We present the distribution of cover types by wilderness areas in figure 2 below.

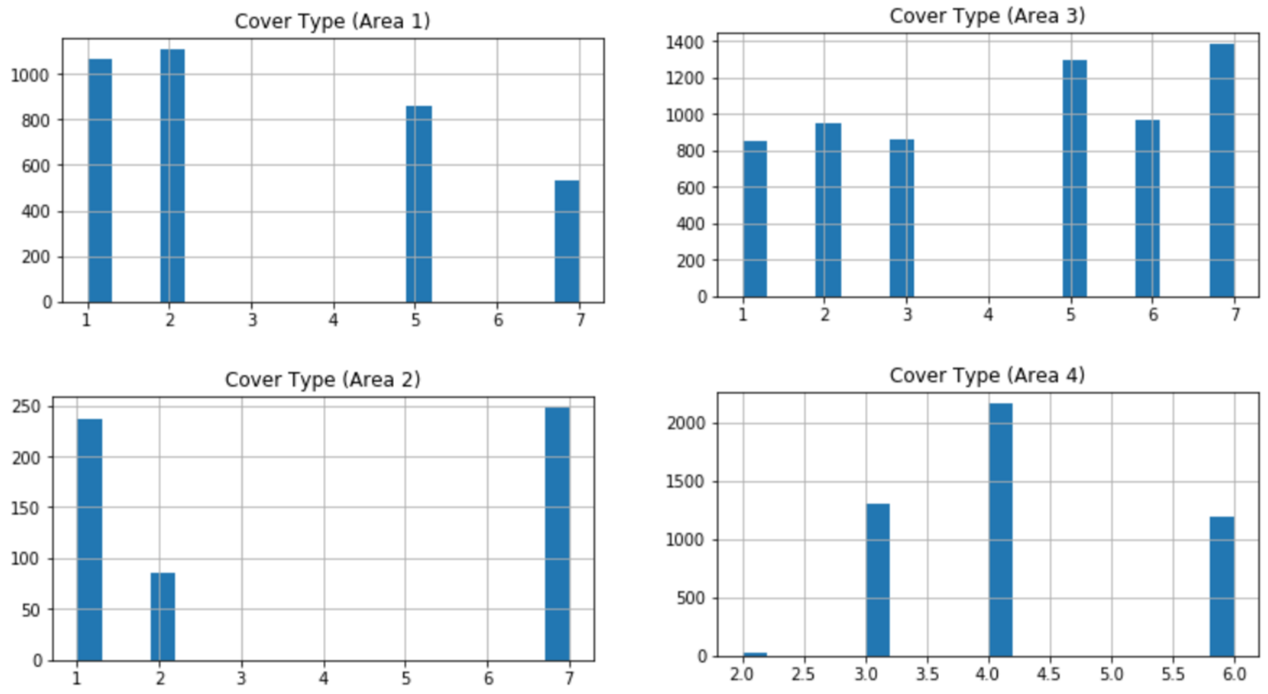


Fig. 2 – *Distribution of cover types in each wilderness area.*

Moreover, the balanced distribution of cover types in the training set may not be found in the testing set (especially with the absence of area 4). For this reason we will need to build models to infer the distribution first, and then to adapt our models with weights related to the estimated distribution to make a more accurate prediction.

Finally, the 40 columns associated with values of soil types were a problem for building models, in particular for models with random selection of a smaller number of features (with trees) that would give a biased importance to soil type. Considering the significance of ELU digits, we chose to replace soil types with climatic and geologic zones. This reduced the 40 columns to 11, as our data only contained 7 climatic zones and 4 geologic zones.

We ended up with 25 columns of features (10 quantitative features, 4 wilderness areas, 7 climatic zones and 4 geologic zones), that we complemented with feature design to improve model efficiency. We notably added combination of the distance variables, with sums, differences, euclidean distances...

2 BUILDING THE MODEL

We describe the evolution of our model as we faced new challenges or had new ideas at each step of our study. Before doing anything with the data, we split the training dataset in a train set (80%) and a validation set (20%) to be able to compare performance between different models and track validation error during training.

We first tried a simple feed forward network but we quickly realised that there was not enough data to achieve high accuracy. Indeed, we only achieved an accuracy of 0.567 on the test data. Looking at such a poor result, we decided to put away neural networks. Other classical supervised learning methods such as support vector machines or logistic regression did not seem promising.

We then focused on tree methods that could have good performances with less data. In particular, we tried gradient boosted decision trees with the famous XGBoost implementation, but also random forest with hyperparameter tuning. Even though the best overall performance we could achieve was with the XGBoost classifier, with an accuracy of 0.871 on the validation set, we quickly saw that a challenging point for the model was to differentiate between cover types 1 and 2. We present the confusion matrices associated with the XGBoost classifier in figure 3. Moreover, we later saw by testing models that cover types 1 and 2 were probably largely dominant in testing data, so we gave an strong importance on being able to differentiate well between these two labels.

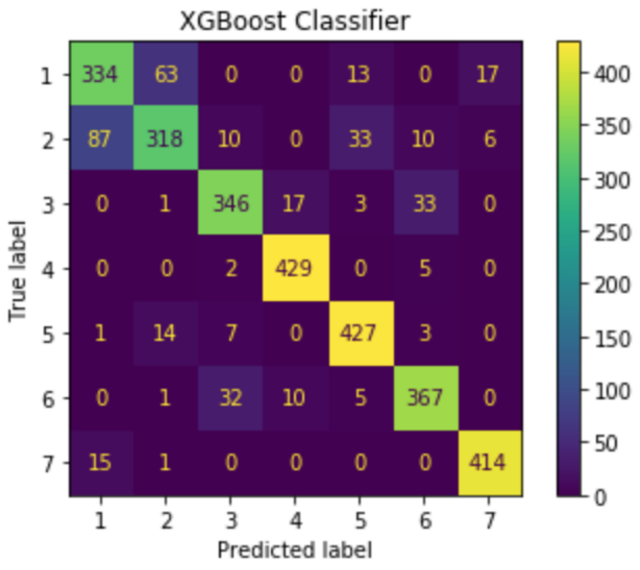


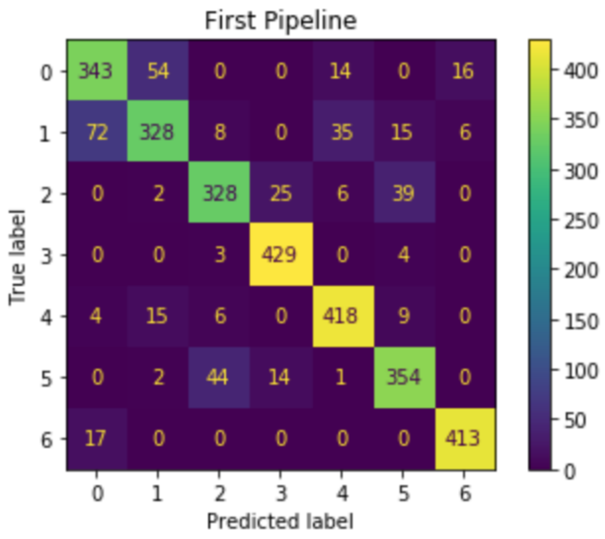
Fig. 3 – *Confusion matrix of the XGBoost classifier.*

The best single model that we found with this objective was an Extra Tree classifier with fine-tuned hyperparameters, and we kept a solid 0.861 global accuracy on the validation set. When we tested models on the testing set, we even observed that Extra Trees classifiers generalised better than XGBoost, probably because of the randomization induced during the training of the classifier.

We used a cross-validated grid-search to find the optimal hyperparameters of the Extra Trees classifier (number of estimators, maximum number of features, maximum depth) and finally achieved an accuracy of 0.773 on the testing set.

Looking at the confusion matrix, we still knew that the main problem was our classification error between cover types 1 and 2, and also between cover types 3 and 6. With our reasonable model, we estimated that cover types 1 and 2 represented about 80% of cover types in the testing data. This contributes to explaining the gap between our performance on the balanced validation set and on the strongly unbalanced testing set. We used the estimated weights of all cover types to better train the Extra Trees classifier, but the improvement was only marginal and not sufficient.

Another idea we had was to create a sequence of classifiers to allow to spend more time on the specific hard tasks. We kept Extra Trees classifiers as our base classifiers as they had shown to be well suited for the problem. Our first "pipeline" was built with one classifier between labels 1/2, 3/6, 4, 5, 7, and then two binary classifiers between labels 1 and 2 or labels 3 and 6. After training each classifier with accurately modified training data, we could predict the cover type for an input by applying the three classifiers and recombining the probabilities with the formula for conditional probability. We obtained a better accuracy of 0.865 on the validation set, and the confusion matrix really improved as shown in figure 4.



	0	1	2	3	4	5	6
0	343	54	0	0	14	0	16
1	72	328	8	0	35	15	6
2	0	2	328	25	6	39	0
3	0	0	3	429	0	4	0
4	4	15	6	0	418	9	0
5	0	2	44	14	1	354	0
6	17	0	0	0	0	0	413

Fig. 4 – *Confusion matrix of the first pipeline.*

Capitalising on what we had discovered on wilderness areas during data exploration, we decided to further improve our model by introducing conditional classifiers depending on area. We reproduced the model of our first pipeline to areas 1,2,3 (with light adaptations when all cover types were not present). Our second pipeline was thus composed of 12 classifiers (3 for the global prediction and 3 for each area). The prediction pipeline for an input was to construct the probability vector as a barycenter between the global prediction and the prediction specific to its area. We trained all the classifiers and grid-searched the optimal coefficients for the barycenters. We show the corresponding confusion matrix in figure 5.

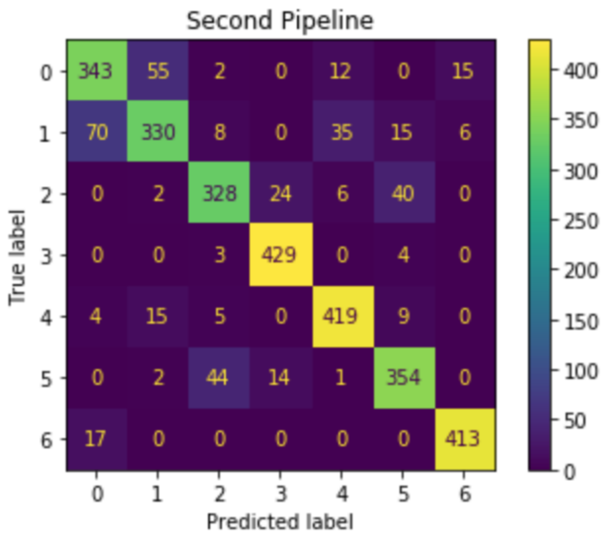


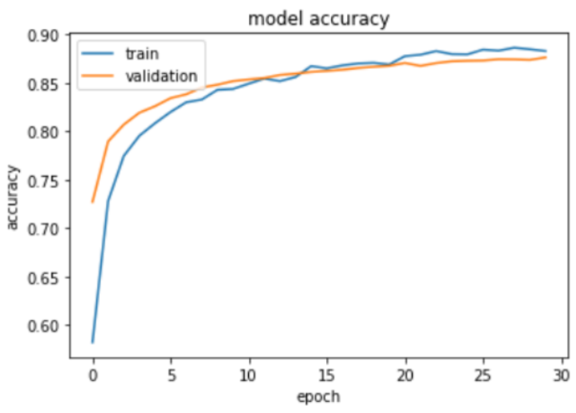
Fig. 5 – *Confusion matrix of the second pipeline.*

Still, the improvement was marginal, and we were blocked at an accuracy around 0.82 on the testing set. Our final idea was that maybe that combining probabilities ourselves with simple sums was not the best ways. We considered that this type of decision could be better taken by a feed forward neural network.

We thus trained a neural network to choose a cover type from all the probabilities given by our classifiers. For training, we had to be careful to split the train set once again. Indeed, if the neural network had been trained on data used to fit the classifiers, it would have had an over-easy jobs, probabilities being too close to 0 or 1. The network would then have a hard-time generalising on new data. We chose to split the train set in two equal parts, one to train the classifiers (from scratch) and the other one to train the neural network. Once the network was trained, we actually fitted the classifiers with all the data available to maximise our accuracy.

We found out that two hidden layers of 32 neurons in between the inputs neurons and a dense layer of 7 neurons gave good results. We chose ReLU activation functions for the hidden layers, and a classical softmax for the output since we were in a multi-output classification setting. We trained the network with the Adam optimize and a categorical cross-entropy loss for 30 epochs.

We present the accuracy and loss of the model in figure 6, on both its train set and the remaining validation set. We combined this model with feature design, which was relevant as Extra Trees do not perform basic operations between features like neural networks would. This final model performed really well, as it allowed us to reach an accuracy of 0.847 on the testing data.



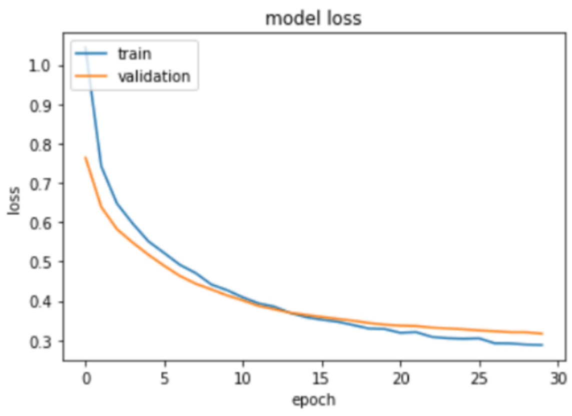


Fig. 6 – *Accuracy and loss of the network on training and validation sets.*

CONCLUSION

In a nutshell, we explored a lot of machine learning models and the practical issues associated with their application to concrete datasets. We enjoyed the Kaggle competition format, especially as we managed to rank in the top 10 submissions (at least in the public leaderboard). It may be the first of a long series of other challenges for our team!