Thy Nearest Neighbours

Stephen Carmody, Stefano Costantini, Monika Krzyzanowska

27 March 2015

Introduction

This technical report summarises the analysis that our team (**'Thy Nearest Neighbours'**) has carried out for the Machine Learning competition. The objective is to provide an overview of our approach, showing the techniques we have considered and the results that have led us make our final submission to the competition. The core of our analysis, implementing the training of the chosen classifiers and generating the predictions is provided in our Github repository in the file `Report_Code-ThyNearestNeighbours.R`

The report is structured as follows:

- First we describe the approach we have used to test the chosen techniques and to construct / modify the features of the dataset.
- We then discuss our results relative to the application of the shortlisted techniques. For each technique we show the results of the testing that we used to choose the method underpinning our final submission.
- We conclude by selecting our preferred option for our submission.

Initial assessment and approach

Our first step to address this classification problem was to identify a short-list of tecniques that could be used. We reviewed the existing literature and identified a longlist of possible candidates. For each of these techniques we carried out some preliminary testing to check whether they warrated further assessement.

During this initial testing phase, we decided to discard some approaches that were not delivering acceptable results. For example, we tested the **Adaboost.M1** algorithm (using the `R` package `adabag`) and found the error to be too high. Specifically, the test error on a 20% holdout sample was **0.282** and the 10-fold cross validation error was **0.276.** We decided not to pursue this approach further. We also tested a **support vector machine** but our best performance was a test error of **0.220** on a 20% holdout sample.

Classifiers shortlist

At the end of this initial phase we identified a shortlist of techniques to explore further. These are:

- k-Nearest Neighbour rule
- Random Forests

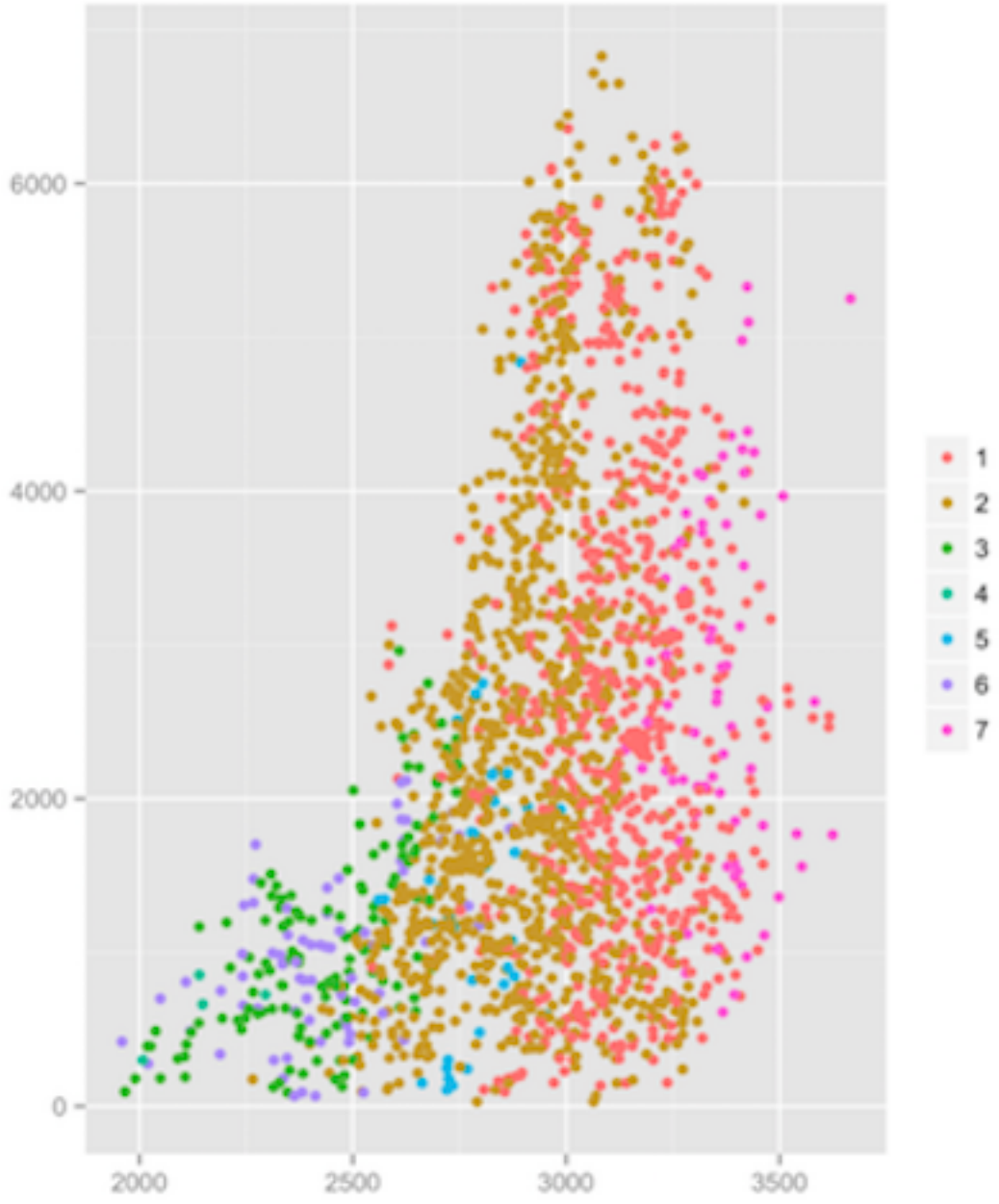
For each technique, we implemented the following testing procedure:

1. First we identified the set of parameters that we wished to change. On this basis we defined a "testing grid" to set out all possible combination of parameters.
2. We implemented the testing, parallelising the calculations over the "testing grid". We did this by means of the `foreach`, `snow` and `doSNOW` `R` packages. We trained each model on 40,000 observations and calculated the test error on a hold-out sample of 10,000 observation.
3. We summarised the results of the multiple testing in a data table and selected the specification yielding the lowest test error.

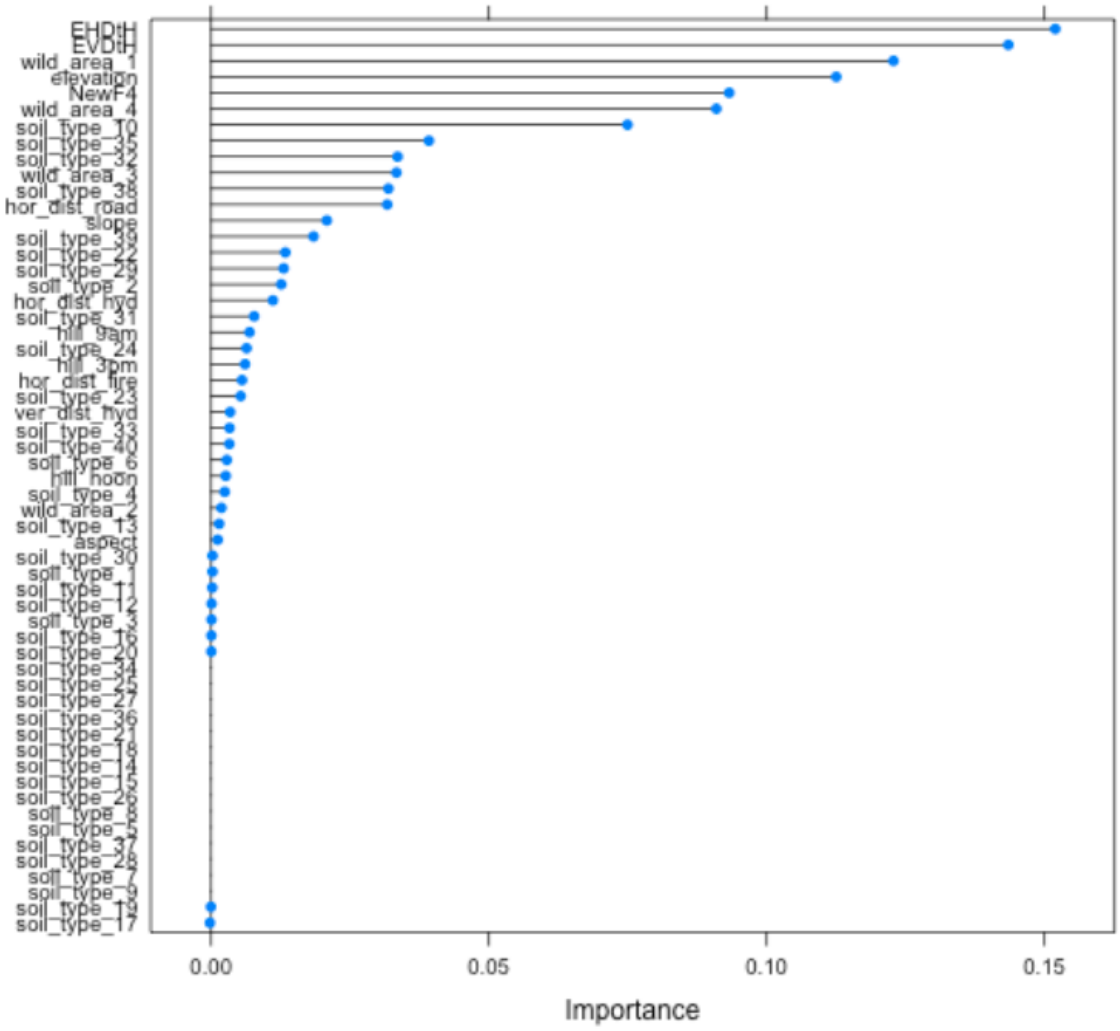
Feature construction

After carrying out an initial exploratory analysis on a range of classifiers, and having chosen the most promising ones for further testing, we focused our efforts on feature engineering, i.e creating new features from the pre-existing data, in an attempt to improve algorithm accuracy. After finding good candidates, we manually constructed these features, we trained and optimised our most promising classifiers using the modified dataset.

The approach to feature construction is heuristic in nature, with the the first step in our process starting with graphing all features against each other to find relationships that may be exploited by the classifiers. Below we can see a linear relationship between **Elevation** and **Horizontal distance to hydrology**. We can exploit this relationship to create a new feature, and using new test errors and feature analysis conclude if this new feature will benefit our model.



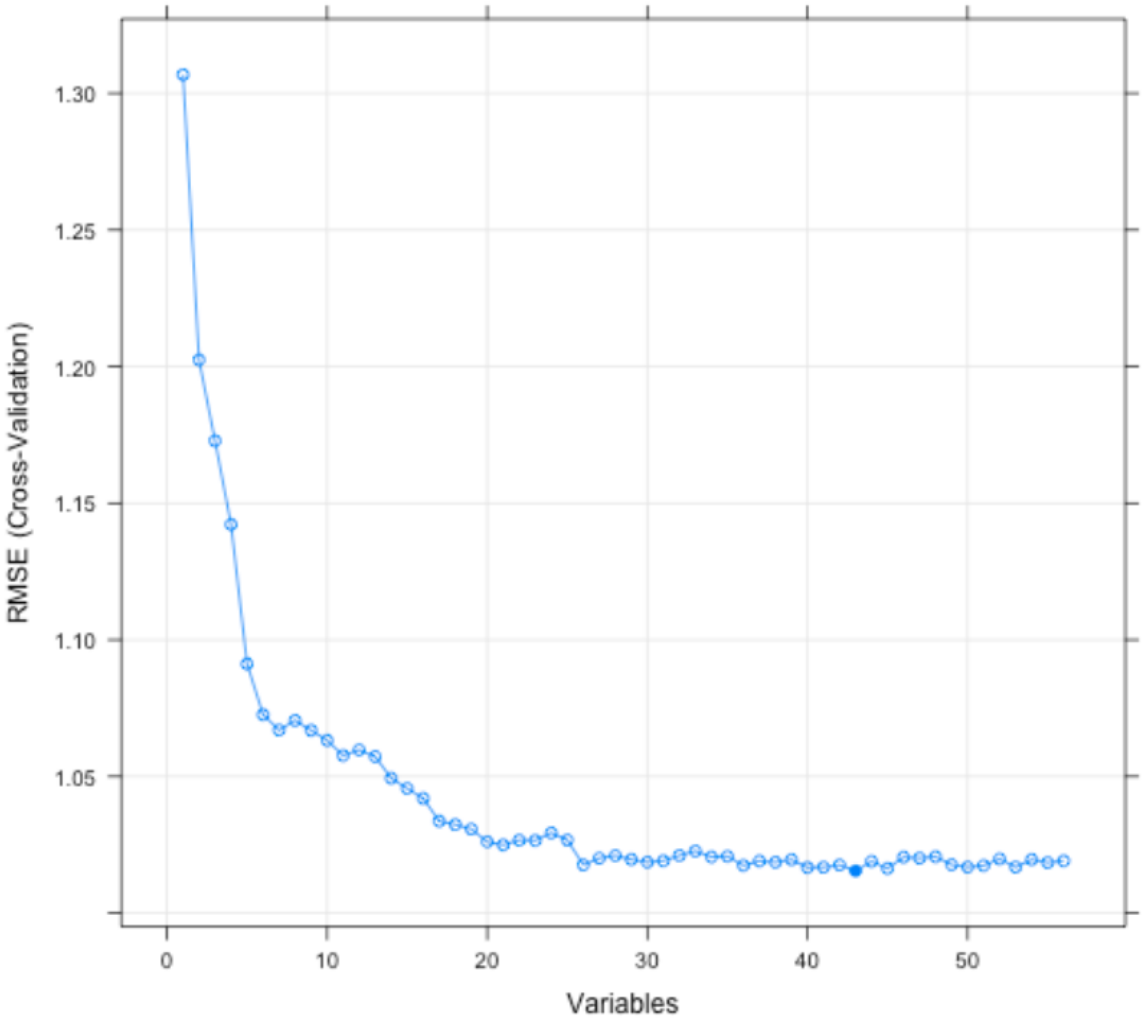
The package we used to rank feature importance is `Caret`, and more specifically the `varImp()` function. The function exams a trained model to rank the importance of features in predicting classes. As seen in the graph below, the graph lists the top features, ranked by importance, along with the percent of influence on the model.



This plot shows the relevance of new features we constructed. Three of the top 5 best predictors are manually constructed, accounting for almost 40% of the classification prediction. These are:

- EHDtH: Elevation vs. Horizontal Distance to Hydrology
- EVDtH: Elevation vs. Vertical Distance to Hydrology
- NewF4: Elevation vs. Vertical Distance to Road

Finally, after taking the rank of features importance, we can rate our classifiers effectiveness starting from a model with one variable to the full set of variables. In this way we can find a more parsimonious model the best amount of effective parameters to give a more parsimonious model. We note that this improved the results for our Random Forest classifier but had no significant effect on the KNN model.



On the basis of these results we decided to include the newly constructed features in our final dataset. We also decided to remove many of the `soil_type` features which did not seem to have a particularly strong importance and did not affect the performance of the classifier.

The code we used for the feature analysis can be found on our Github repository in the file `FeatureTesting.R`.

Classifier training

Random Forest

We have tested a Random Forest approach by considering these ranges of parameters:

- **Number of trees in each iteration:** from 50 to 1,550, with an increment of 100.
- **Number of features considered in each iteration:** from 3 to 30, with an increment of 3.

The following table provides a summary of the 5 best results:

# of trees	# of features	Test error
1350	13	0.1038

# of trees	# of features	Test error
1250	10	0.1039
850	10	0.1040
350	9	0.1041
250	12	0.1042

On this basis, for this technique we selected a Random Forest approach with these parameters:

- Number of trees: 1350
- Number of features in each iteration: 13

k-Nearest Neighbour

We have tested a k-NN approach by considering these values of the parameter k, i.e. the number of neighbour observations to consider in order to assign a class:

- $k = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 23, 27, 29, 31, 37, 41, 43, 47, 51$

The following table provides a summary of the 5 best results:

k	Test error
1	0.0977
3	0.1129
5	0.1320
7	0.1404
9	0.1510

On this basis, for this technique we selected a 1-NN approach.

Conclusions

The following table summarises the best results that we have achieved with each of the techiques we have tested.

Technique	Test Error
Random Forest	0.1038
kNN	0.0977

On the basis of the results above, we have selected 1-NN approach, applied to a modified version of the dataset with new features included.

We have used this approach to make our final submission to Kaggle which, on the 10% test set, yielded a score of 0.91820.