UPPSALA
UNIVERSITET

`I(RandomForest^2)`

Random Forests Applied to Random Forests

Snillgot Österlund & Valentin Zulj

Department of Statistics
Uppsala University

November 2018

Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
– G. James, D. Witten, T. Hastie & R. Tibshirani

Abstract

The aim of this report is to use a Random forest model to classify the main type of tree in a specific area. There are 13 different variables ment to explain the tree type, where elevation and soil type seems to have the biggest impact. The Random forest model is compared with two other methods for classification, decision tree and a multinomial logit model. The results shows that the Random forest is the best model for classifying the correct type of tree in the test set, with a correct classification rate of 84.4%. It is followed by the decision tree model and the multinomial logit model, with a correct classification rate of 73.6% and 70%. The report also includes a small simulation study, where the effect of tree depth on the classification rate is examined.

Table of Contents

1 Introduction

Think of a warm summer's eve in Barcelona. Or Paris. Perhaps even Lisbon. You are strolling down the most beautiful of avenues, fingers intertwined with those of your loved one. Everything seems so effortless, and you feel as if you are gliding. Trees lining the streets. What we want to you to take away from this scenario is the trees. Trees are very important to us in many different ways. They provide us with oxygen and apples, timber and pears, and, of course, they always seem to be there to provide some much needed shade when the sun is burning your skin. Trees, however, come in many different shapes, and we would argue that perhaps the mathematical ones are just as important as the physical ones. This report will examine they way in which trees can be used on trees. We will use mathematical trees in order to classify the natural ones.

In machine learning, random forests are used to classify variables such as cancer cells, election results, and even trees. We will use the theory of machine learning in order to classify the cover type of different plots of land, picked at random from four wilderness areas throughout the United States.

Our modelling will be done as part of a competition hosted by Kaggle, and apart from the competitve aspect of the project, we believe that classifying the cover type of a certain plot of land might be very useful indeed. Developing a model that successfully manages to classify the forest cover type could save plenty of resources, since it would reduce the need of flying over certain areas in order to determine what type of forest cover them – which is, of course, of service to the environment as well. Thus, our research question is formulated as follows:

Research question: *How do we best classify the forest cover type using machine learning and random forests?*

To answer our research question, we will use the data provided by Kaggle and grow a random forest with the purpose of classifying the cover types of any given plot of forest. The random forest itself is a collection of decision trees, that are also used for classification, and thus, we will grow a single classification tree and see how it compares to an entire forest. Furthermore, we will compare the results of the two tree-based models to that of a multinomial regression model.

We grow two separate forests, one using only a fraction of the variables available, and one using all feature we have. We compare them to the individual decision tree and the multinomial model, and conclude that the reduced forest manages to extract roughly the same amount of information using two variables as the other two models using all of them. Also, we conclude that the full random forest model is most suitable in classifying the forest cover type, seeing as it achieves a classification rate of roughly 85%.

We will perform the statistical analysis presented in this paper using `R` and `Rstudio`, as well as packages called `randomForest`, `party`, and `nnet` in order to fit random forests, classification trees, and multinomial models respectively.

Finally, we will conduct a simulation study in which we generate a number of unique data sets in order to examine the effect of tree depth on the classification rates of random forest models, concluding that the classification rate does not seem to change notably when the tree depth is allowed to vary from 100 to 4000 terminal nodes.

2 Data

2.1 Original data

The data used in the project can be found on found on Kaggle, and it consists of 15 120 observations of 55 variables. The response variable is, of course, the type of tree, and consists of discrete, interger values on the interval [1, 7]. These types are: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir and Krummholz. In total, the data consists of 2160 observations of each cover type.

The data set consists of 55 variables, most of which – 40 to be precise – are dummies regarding different types of soil. As for the other variables, we present a brief summary in the bullets below:

- Elevation: Elevation – in metres – of the plot of land
- Aspect: Aspect in degrees azimuth
- Slope: Slope – in degrees – of the plot of land
- The horizontal distance to the nearest water source
- The vertical distance to the nearest source of water
- Horizontal roadways: Horizontal distance to the nearest roadway
- Hillshade (9am, noon, 3pm): Index showing degree of shade from hills at different times of day
- The horizontal distance to the nearest point where wildfire ignition is allowed

The variables themselves are all quite straight forward, althoguht it is worth mentioning that the hillshade index is measured on a scale from 0 to 255. Furthermore, there are some binary columns specifying whether the plot of land is located within one of four wilderness areas, as well as the 40 binary columns that indicate the type of soil in which the trees grow. However, two types of soil – coded as 7 and 15 – do not show up in any observation.

2.2 Data Wrangling

In order to clean the data and make it more manageable and easier to understand and visualise, we merge the binary columns mentioned above into two factor variables, one containing the 38 levels – excluding soil types 7 and 15 for which we have no observations – of soil types, and the other the four different wilderness areas from which the samples have been taken.

Seeing as this is a competition data set the issue of variable selection is not too relevant in our case, we simply want to use the data we have been given in the best way possible. The model we want to create is a model used for prediction – or classification – rather than inference, hence the thing that is most important to us is accuracy, and not interpretability. Because of this, we do not spend that much time deliberating on the variables we have, other than the exploration of data presented in section 2.3, in which we try to establish what relationships can be found between the different features, if any.

Regarding transformations, we find it hard to see what could be done to our variables. Two of them are – as stated above – factor variables, while the others are mostly indices and distances or slopes. Factor variables, of course, are quite hard to transform, and we do not believe that transforming the others would help us extract information efficiently. Hence, we keep the data as is, apart from merging the binary columns into factor variables.

Lastly, we split the data set into a couple of subsets, one used for training the machine, and the other used to test the accuray of it. We encounter a couple of issues in doing so, again related to the soil type variable. For soil types 8 and 25, we only have one observation each. Because of this, we force these two observations to be part of the training data, so that we can try to use our model on the competition data. In doing so, we end up training our model using 7562 observations, and testing it on a set of 7558 observations.

Moving on, we take a closer look at the data itself, trying to establish realtionships that can be used in order to fit the best model and reduce the computational burden in fitting them.

2.3 Exploratory Data Analysis

We begin by exploring the variable measuring the elevation of each plot of land, seeing as we believe it might be an intuitive starting point for the study of the occurence of different trees. Figure 1 makes clear that different trees grow at different elevations. If we imagine a model using only elevation as a feature, it seems very unlikely it would predict a plot situated at an altitude of less than 3000 metres to have cover type 7. The plot tells us that the elevation of a specific area might be of great importance when it comes to what type of trees make up the cover.

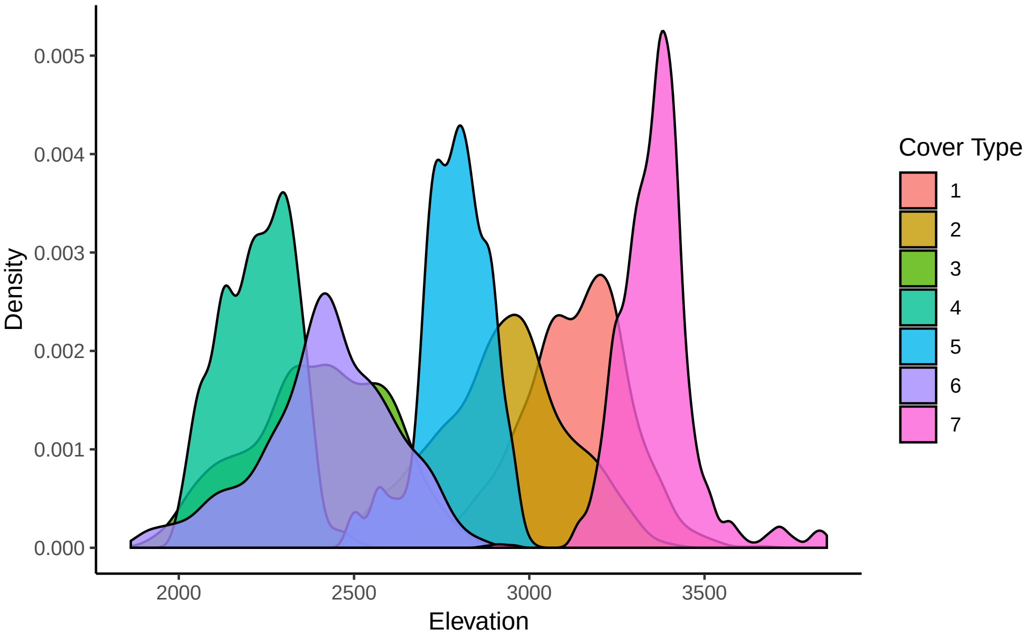

Figure 1: Density plot showing the occurence of different cover types at different elevations

From Figure 2 we can tell that no relationship seems to exist between the elevation and the afternoon hillshade index, nor does the afternoon hillshade itself seem to have any

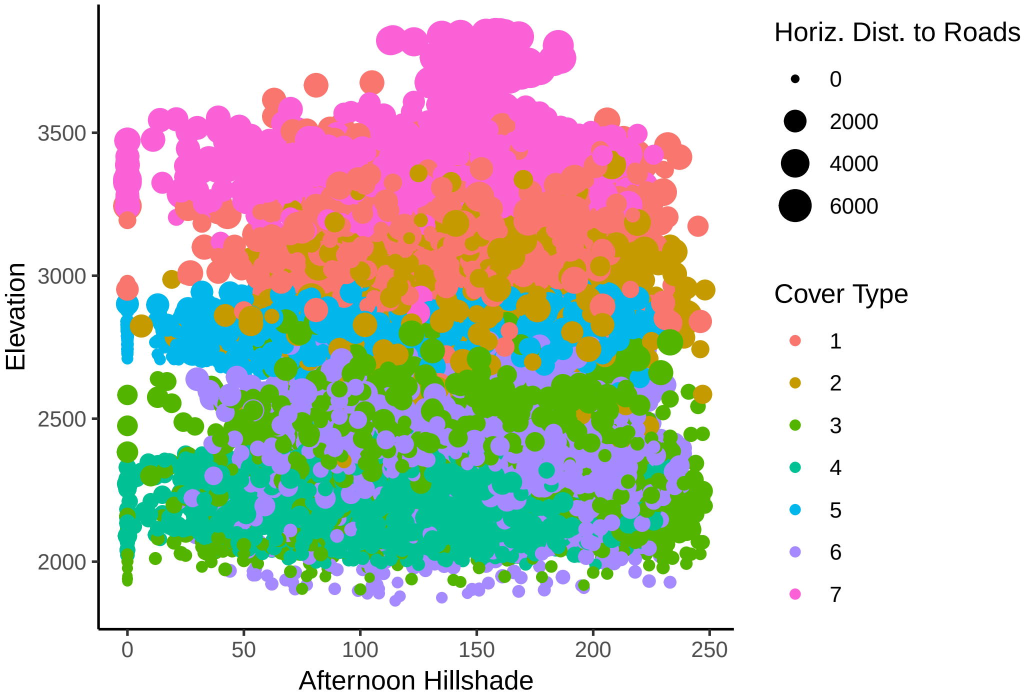

Figure 2: Scatterplot showing the relationship between afternoon hillshade and elevation, as well as the horizontal distance to roads

great effect on the type of cover found on each plot of land. This we can tell from the fact that all types of trees grow at all levels of hillshade. We can see that the points in 2 tend to get bigger when the elevation is higher. This is intuitve, since the size of the points is determined by the horizontal distance to the nearest roadway, and roadways tend to be few and far between at high altitudes.

In a sense, we find it could be of interest to have a variable measuring the levels of CO_2 – or perhaps even oxygen – in the air by each plot of land, to see whether the distance to the nearest roadway is in fact a result of the environment in which each type of tree thrives, or whether human intervention might be a factor to account for in this case.

Figure 3 shows a boxplot of the cover type and the horizontal distance to water of the corresponding plot of land. The horizontal distance to water looks to be of little importance in determining the cover type of any given plot of land. Perhaps a greater distance could be used to filter out covers of type seven, and plots in relative proximity to water might be likely to have a cover of type four. Otherwise, there seems to be no particular way of distinguising cover types using the variable itself.

In the process of visually explorating our data, we finnd that the relationship found in Figure 4 was the most interesting. The plot shows how the cover types are spread out over different elevations and soil types, and looking at it we quickly see that some sort of clusters start to take shape. For example, the top right of the graph is almost exclusively pink, meaning that cover type 7 looks most likely to occur when then plot of land is situated at a high elevation, and the soil type ranges from approximately 35 to 40.

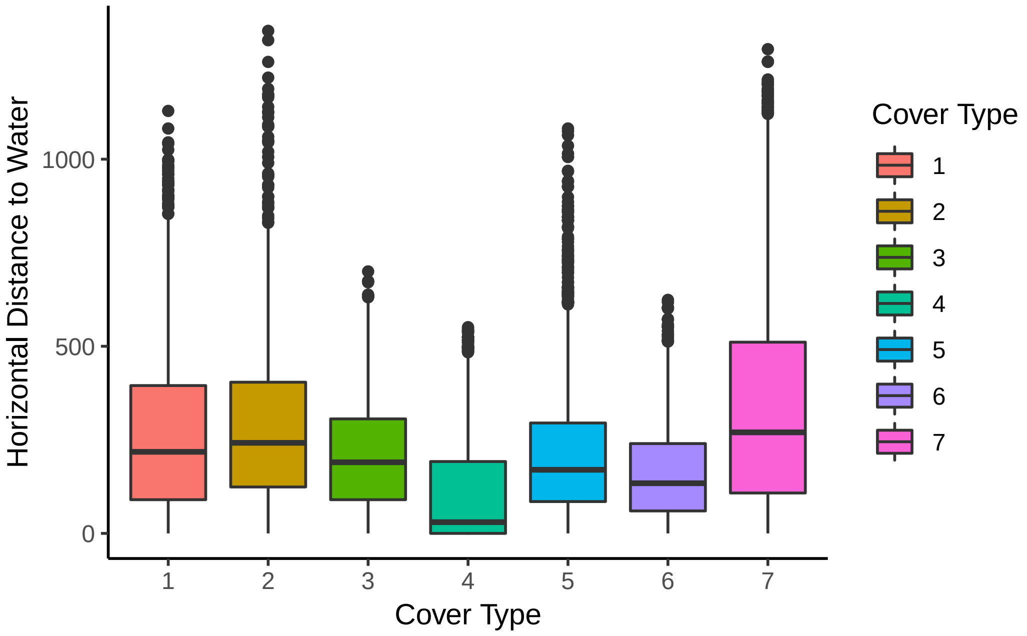

Figure 3: Boxplot showing where different cover types occur in relation to the nearest source of water

Furthermore, we can see orange occuring mostly in the middle of the graph, and the lighter shade of green in the bottom left. We also note that cover types 3 and 6 seem to intertwine rather heavily in the bottom left of the graph, which will prove out to be a limiting factor in the success rate of our models. All in all, we believe that Figure 4 indicates that elevation and soil type are very important in determining the forest cover type.

3 Methods

This section will provide descriptions of the methods used in order to carry out this study. First, we will provide a brief discussion regarding decision trees, seeing as they make up the individual components of the random forests we want to grow. After that, we will delve deep into the world of random forests, in order to provide a deeper understanding of what goes on behind the algorithms of the `randomForest` package.

3.1 Decision Trees

Decision trees are a fairly simple way of classifying observations using cut-off points in order to divide the data set into groups. The cut-off points are usually called inner nodes, and they separate the data set into two new sets that do not overlap. Repeating this process for many variables will result in several inner nodes, and at the very end we will attain terminal nodes that decide which class each observation will belong to.

Consider the elevation variable in our data set. It is quite reasonable to assume that different trees grow at different altitudes, seeing as different species require differing amounts

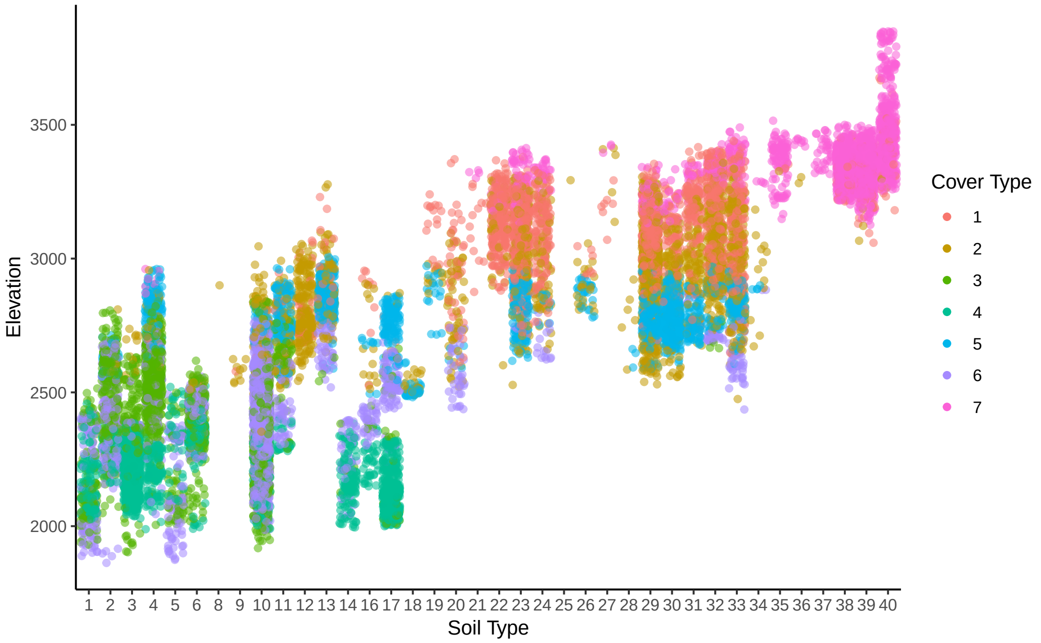

Figure 4: Scatterplot of soil type and elevtion.

of oxygen and sunlight, and perhaps even different temperatures to thrive. Hence, if we try to build a classification tree, we could assume that 4 out of the 7 types of tree tend to grow at altitudes lower than 1500 metres, and that the remaining 3 groups are usually found somewhat higher than that. Then we will have an inner node at which the tree splits into two branches, one containing the lower altitude trees and the other the trees growing higher up. For each branch, we continue searching for variables by which to grow even more branches, and eventually end up with a tree classifying every single observation in our data set. The concept of decision trees is illustrated in Figure 5, where the elevation has been used to classify the cover type.

In order to better understand the mathematics of classification trees, we need to introduce a measure of accuracy called the *Gini index*. It can be thought of as a version of the residual sum of squares, only that the Gini index is used when the response is a factor variable. The index is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right), \tag{1}$$

where $\hat{p}_{mk}$ is the proportion of training observations of class k in the m:th branch. The Gini index is a measure of variance across the K number classes, and hence we want to choose the cut-off values of each branch in a way that minimizes the Gini index. There are several other types of variance measures that can be used to fit classification trees, such as *classification error rate* and *entropy*. However, the classification error rate is too robust for growing a meaningful tree [James et al., 2013]. We find the the Gini index is more intuitive than the measure of entropy, and that the Gini index easier to understand, and hence choose to use it in our project.

Classification trees grown according to the method described above are prone to overfitting, and using them on data other than the training set is bound to yield a very high variance. Hence, we introduce the concept of *pruning*, which strives to simplify the trees and make them more generally applicable.

Pruning means that we use the training data to grow a tree of maximum depth – or the maximum amount of splits – and then work our way back to find the subtree that provides the best classification rate. Doing this, we reduce the risk of overfitting, and find the tree that gives us the greatest classification power. In short, we grow our trees accoring to the following plan:

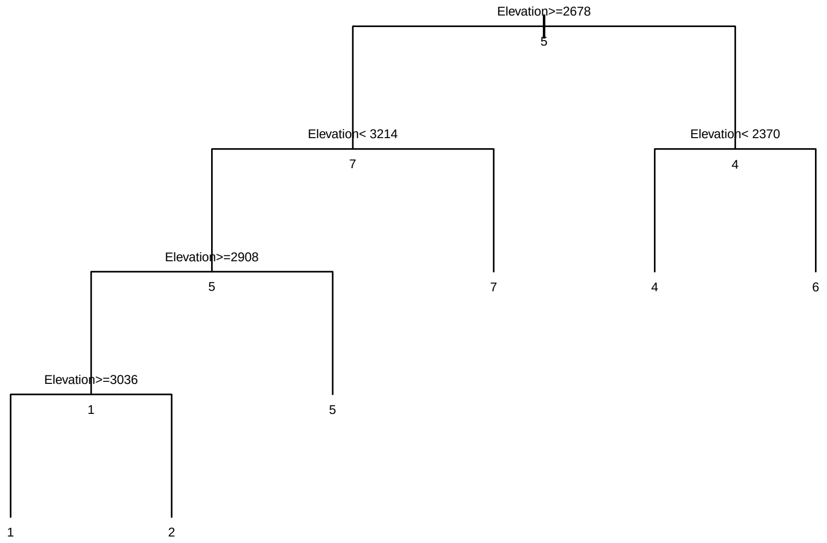

Figure 5: Classification tree for Cover Type, using elevation as a covariate

1. Grow a tree – call it T_0 – using the training data and the maximum number of splits allowed

2. Prune T_0 in order to find the subtree – call it T_1 – that produces the best classification rate

3. Use T_1 to to classify the response variable of the test set

Finally, we intend to grow a classification tree to compare its performance to that of the random forest model. In order to grow the simple decision trees, we use a package called `party` [Hothorn et al., 2018]. Random forests models will be described in Section 3.2.

3.2 Random Forests

Random forests are a way of reducing variability in the estimates of individual decision trees, as well as decorrelating the trees that are grown, using bootstrap samples drawn from the original training data in order to do so. In essence, the theory of random forests is the same as that of decision trees, the only diffrence being that a random forest consists of a multitude of individual decision trees – as can be deduced from the name. In order to understand the workings of a random forest, we need to introduce bootstrap aggregation – or bagging for short.

Bagging entails drawing bootstrap samples from the training data and fitting a decision tree to them, using each tree to classify the response of the observations that are not in the sample. This process is repeated B times, and in the end, the class of each observation is decided by the most occuring class among the sample classifications. If $B = 5$, the classifications of a hypothetical observation could amount to $\{2, 2, 3, 2, 4\}$. Hence, the bagging procedure will list the observation as an element of class 2. The use of bagging tends to be somewhat problematic, seeing as all covariates are used in every tree. Thus, the trees run the risk of being highly correlated, and having an inflated variance. Because of this, a method of decorrelating the trees has been introduced. The method in question is called random forests, and it will be used in the analysis provided by this paper.

In growing random forests, a random element is added to the bagging procedure. At each split of the tree, a sample of covariates is drawn. Considering the covariates found in the sample, the split is made using the variable that gives the greatest decrease in the Gini index. Usually, the number of covariates in each sample is kept in the vicinity of $\sqrt{p}$, where p is the number of covariates available for selection [Hastie et al., 2009]. As mentioned above, this randomization of covariates reduces the correlation between trees, meaning that pruning will not be very fruitful when using random forests. Hence, we do not spend any time controlling for pruning while growing our forests. As a result of this, the only real tuning parameter of the random forest model is the depth of the trees used.

In drawing the bootstrap samples, a part of the training data will clearly be left out each time, meaning that it can be used to validate each model. Letting each model classify the observations that are not in the sample, we obtain a classification rate that measures the usefulness of each model. This procedure is called out-of-bag (OOB) error estimation – since the observations are literally out of the bag, and it can be used instead of the usual k-fold cross validation. Hence, we can measure the success of our forest in two ways, one being the OOB error rate, and the other the test classification rate.

Finally, we will grow a random forest in order to try and classify the cover of our plots of land, using the `randomForest` package [Breiman et al., 2018]. Theoretically, the random forest should have a higher rate of success than the individual classification tree. Hence, we will compare the two of them, together with a multinomial logistic regression model, which is discussed further in Section 3.3.

3.3 Multinomial Logistic Regression

The binomial logistic regression model is often used for classifying response variables that are factors of two levels. As stated above, however, our response variable has several levels – 7 to be precise – and thus we need to use a multinomial logistic regression model in order to try and classify the cover types. Seeing as the mathematics of the multinomial models are quite complicated, we will only provide the briefest of summaries in this section.

The model that we use is often referred to as the *multinomial logit model*, and in order to define it properly, we need to take a closer look at the notation used. As mentioned, our response variable is a factor of several levels, and that number of levels will be denoted by c from now on. Furthermore,π_{ij} denotes the probability that the repsonse of the i:th observation is category j , meaning that $\sum_{j=1}^{c} = 1$. The model itself compares the different classes of response to a baseline category called c. Hence, the baseline-category *logit* is defined as

$$\log\left(\frac{\pi_{ij}}{\pi_{ic}}\right).$$

Seeing as we use a baseline class, we will only have $c-1$ logit terms.

Now, if we define $\boldsymbol{x_i} = (x_{i1}, ..., x_{ip})$ as the covariates, and $\boldsymbol{\beta_j} = (\beta_{j1}, ..., \beta_{jp})^{\top}$ as the parameters for the j:th logit, the model is given by

$$\log\left(\frac{\pi_{ij}}{\pi_{ic}}\right) = \boldsymbol{x_i}\boldsymbol{\beta_j} = \sum_{k=1}^{p} \beta_{jk} x_{ik}, \tag{2}$$

giving us the baseline category logit model [Agresti, 2015]. The parameters of the multinomial logit model are estimated is the maximum likelihood method, but the derivation of the estimators is left out of this report. However, we use the `multinom` function in the `nnet` package in order to implement it in `R` [Ripley, 2016].

That concludes the methodological section of this paper. Moving on, we will use the three methods we have described to classify the cover types of each plot of land. The results of these classifications will be presented in Section 5.

4 Implementation

In this section we will discuss the implementation of the methods described above, and provide an overview of the simulation study we plan to perform in order to test the `randomForest` package out. When it comes to the single classification tree and the multinomial model, our comments will be rather shallow, but for the random forest we will explain the choices we made in applying the method to our data.

4.1 Classification Tree

We choose not to spend too much time on the making of a classification tree, since we want to focus on the implementation of random forests. We use the `ctree` function to fit a tree using all the covariates in our training data, and use it to classify the cover types of the test set. Hence, we will not provide any description of the package, and we will proceed by using only the default settings of the tree growing function.

Seeing as the tree is grown only to provide some sort of relation to the random forest, we find it reasonable to leave it at that.

4.2 Random Forest

As stated in section 3.2, the only real possibility of tuning the random forest model is modifying the depth, as well as the number of trees used in the forest, at least at the level

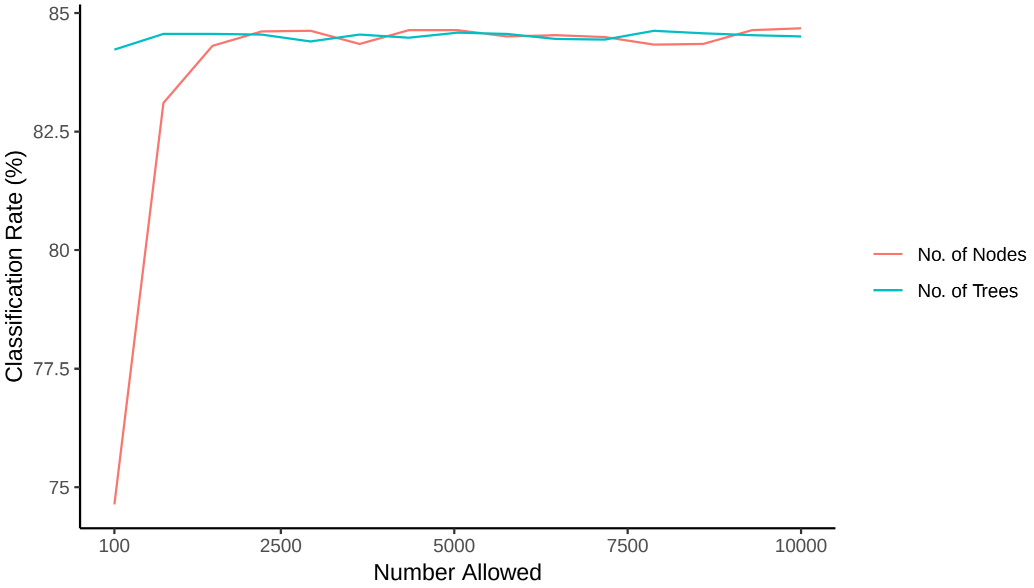

Figure 6: Effects of terminal nodes and number of trees on the classification rate

of beginners. In order to find a final model to use for classification, we run a couple of loops that that fit models and make predictions using different restrictions to the number of terminal nodes and the number of trees allowed. The loops themselves are fairly simple, and they will only be included in the `.R` appendix file of this report. Having evaluated the model for different values of the maximum amount of terminal nodes allowed, we conclude that the classification rate seems to converge to a value close to 85% when `maxnodes` takes on values that exceed 2500. Hence, 2500 will be the value we use in our final model.

Knowing that, we set `maxnodes` to 2500 and run the model allowing for different amounts of trees to be used in the forest. As can be seen in Figure 6, the classification rate does not seem to be affected by the number of trees, given the number of terminal nodes allowed. Hence, we use the default setting when it comes to `ntree`, which grows a forest of 500 trees. The `randomForest` function does not seem to contain a setting that controls the direct depth of the tree. Thus, we use `maxnodes` as a proxy.

For each of the separations, we let the function consider $\sqrt{p}$ covariates, and proceed using the one that gives the best classification rate. This decision is based mainly with reference to the literature, with both referenced books using the same principle. The number of covariates to consider is determined by the `mtry` option of the package, and when growing a forest of classification trees it is set to $\sqrt{p}$ by default. Hence, we do no consider making any changes to it.

In addition, we have the option to vary the size of the samples drawn in order to fit the forests, all through the option named `sampsize`. The option has different default settings depending on whether samples are made with replacement or not. Generally, a large sample will decrease the level of randomness, making the model more likely to overfit, while a smaller sample will be more random but lead to a model with greater variability. We choose to sample with replacement, using a sample size that corresponds to the number of observations found in the set of training data. This method is called bootstrapping, and is usually thought to set up a basis for balanced modelling. When sampling with replacement – which is done by default – the `sampsize` option is automatically set to `nrow(x)`, which gives the number of observations in the training set. Because of this, we let both `replace` and `sampsize` be set by default.

4.3 Multinomial Logit Model

As in Section 4.1, the logit model is only fitted in order to compare it with the random forest. Hence, we only run it using default options, spending most of our time on tweaking the random forest model. The package we use to estimate the model, however, is fairly straightforward, and using the baseline multinomial logit model in its purest form does not allow for a great degree of tweaking to be performed. Thus, as is the case above, the model will be estimated by default in order to be used as a point of reference.

4.4 Simulation Study

The simulation study can be described in two parts, the first part centers around generating new data and the second part is about the simulation. The idea is to use the new data to examine how different settings in the random forest model affect the performance of the model.

4.4.1 Data Generation

Our simulation study will use a data generating process – DGP – made up of several regression models, some linear in character, and others multinomial. We will add two stochastic elements to our DGP, one in generating the new cover types, and the other when generating the new elevations. Our idea regarding data generation can be reduced to two main steps, given in the list below:

1. Fit a regression model to the j:th column of the original data set, using the rest of the columns as regressors. If the column making up the response is a factor variable, use a multinomial logit model. Otherwise, use an ordinary linear regression model.

2. Use the fitted model to predict new values for each column, generating a data set of completely new observations.

Now, the process described above is very deterministic, and repeating it several times will yield the same predictions. Hence, we add randomness when generating the cover types and elevations of each observation.

For the cover types, we use the class probabilites given by the multinomial logit model. Using them, we sample a cover type for each observation, giving us a random element that is still connected to the actual data itself. For the elevation, we consult the Gauss-Markov theorem and use the fact that $\varepsilon_i \sim N(0, \sigma)$, where ε denotes the error terms of a regression model, and σ is a constant greater than 0. When estimating the elevations, we add a random term to each observation, drawn from $N(0, \hat{\sigma})$, where $\hat{\sigma}$ is the estimated standard error of the residuals given by the regression model – in our case, it has been estimated to 99.

Carrying out the proces described above yields a data set that is completely new, and stochastic in some elements. However, the new data set is still clearly connected to the old data set, and hence, we use it to perform our simulation study.

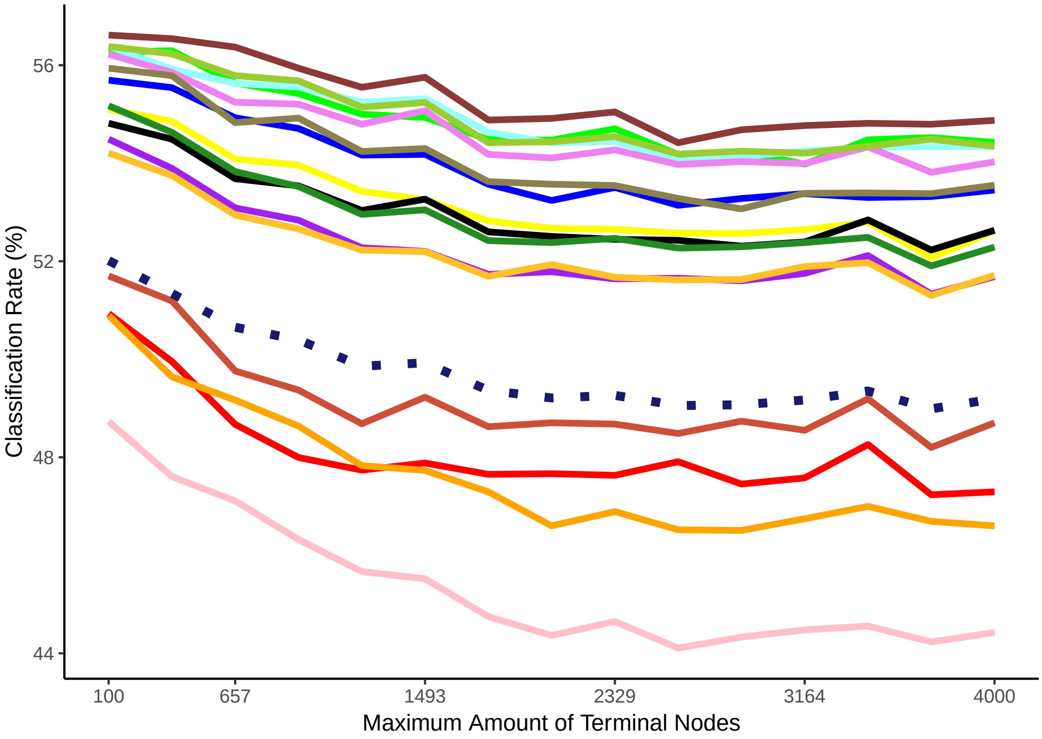

Figure 7: The classiification rate of each model for different tree depths. The dashed line gives the mean classification rate of all data sets generated

4.4.2 Simulation

Our simulation study aims to examine what effect the depth of each tree has on the classification rate of a random forest. As in Section 4.2, we use the `maxnodes` setting as a proxy for tree depth. We train a set of 15 random forest models using the entire data set provided by Kaggle, with each model restricted by a given amount of maximum terminal nodes. The values of `maxnodes` are taken from the interval [100, 4000], and then rounded into integers using the `round` function. The range of the maximum terminal nodes allowed is chosen with respect to the discussion given in Section 4.2.

Having grown the 15 forests, we use the DGP described in Section 4.4.1 to generate 30 uniquie data sets, and classify the cover types of every data set using each of the 15 forests. For the sake of aesthetics, we will present a figure showing the classification rates for 15 of the data sets, as well as the mean classification rate of each model for different values of `maxnodes`. The results of the simulation study are shown in Figure 7.

In figure 7 the results from the classification of 15 of the new data sets are shown. Each line corresponds to one data set, and the dotted line represents the mean of all 30 data sets. Surprisingly, and directly opposite to the discussion found in Section 4.2, we see that the classification rate tends to decrease when the maximum number of allowed nodes increase. We are unsure about what causes this, but it might be because a case of overfitting the model to the original data. The classification rate seems to stabilize at around 2000 nodes. Perhaps this can be seen as something similar to what was found in Section **??**, where our classification rates seemed to converge after about 2500 nodes.

Adding to that, the contradicting results might stem from allowing randomness in the data genarating process. We would expect the classification rate to decrease because of the increased variability of the data, but the fact that the classification rate decreases as the number of terminal nodes increase is, to us, a very strange phenomenon. Looking back at Figure 4, however, we see that som trees grow at very similar elevations. Hence, seeing as the variance of the random term added to the elevation in our DGP is so high, this might mean that some observations move over the cut-off points of the decision trees, deceiving the model in the process.

5 Results

5.1 Classification Tree

We were quite surprised to see see the relatively high classification rate of the simple decision tree. The classification tree managed to correctly classify roughly 73.6% of the test observations, using only the default settings of the `ctree` function. We present the confusion matrix of the model in Table 1.

	Actual Cover Types							
Predicted	1	2	3	4	5	6	7	**Error Rate**
1	675	242	4	0	49	9	117	0.384
2	198	597	29	0	180	50	10	0.439
3	0	2	786	106	24	168	0	0.276
4	0	0	61	972	0	39	0	0.093
5	6	96	41	0	900	26	0	0.158
6	0	21	287	73	30	643	0	0.390
7	115	5	4	0	0	0	993	0.111

Table 1: Confusion table of single decision tree

As is shown in the table, cover types 1 and 2 seem to be rather similar, and the model finds it difficult to distinguish between them. Perhaps this is not very surprising, seeing as the plots in Section 2 suggest that the two types of trees seem to grow at similar altidtudes, and also tend to keep the same horizontal distance to water. Furthermore, classes 3 and 6 look to be hard to predict. Looking at the table, the predictions of these two classes in particular show greater variability than the rest.

Moving on, we will fit random forest models to the same data in order to see if we can generate better classification rates.

5.2 Random Forests

Looking at Figure 4, we can see something that resembles clusters of different cover types. In the top right corner, for example, the dominating cover isof type 7. Also, trees of type 3 seem to be concentrated at the lower left portion of the figure. In other words, it looks as if elevation and soil type capture a significant share of the variation in cover types. Hence, we will begin by growing a random forest that uses only soil and elevation as features. The bivariate random forest model manages to classify about 70% of the test observations correctly, and the confusion matrix of the model is given in Table 2.

	Actual Cover Types							
Predicted	1	2	3	4	5	6	7	**Error Rate**
1	677	202	3	0	56	5	121	0.364
2	265	514	16	0	243	45	13	0.531
3	0	5	679	153	37	200	0	0.368
4	0	0	125	917	0	46	0	0.157
5	2	86	41	0	925	37	0	0.152
6	0	11	325	62	48	660	0	0.403
7	91	5	0	0	3	0	944	0.095

Table 2: Confusion table of random forest using only elevation and soil type as covariates

The table shows that, again, classes 1 and 2 are difficult to tell apart. Furthermore, the troubles in classifying cover types 3 and 6 seem to remain the same as when the individual classification tree was used. In fact, the lone decision tree proved out to better at classifying the cover types of the test data. The random forest, however, managed to produce a similar rate of classification using the information stored in only two covariates, while the individual tree used all features available. Hence, comparing the two models is somewhat unfair, and we decide to grow a random forest using all 12 covariates.

The confusion matrix of the full random forest is shown in Table 3. As can be seen, the main source of miss-classification remains the same as before. However, the error rates of each individual class have decreased markedly, and the diagonal elements of the confusion matrix are more pronounced than in the matrices of any of the previous models. All in all, the model manages to produce a success rate of 84.4%, which is a great improvement from the bivariate case.

	Actual Cover Types							
Predicted	1	2	3	4	5	6	7	**Error Rate**
1	782	172	1	0	30	5	74	0.265
2	207	729	24	0	97	29	10	0.335
3	0	4	836	55	17	162	0	0.222
4	0	0	24	1052	0	12	0	0.033
5	3	29	14	0	1033	12	0	0.053
6	0	9	105	25	10	957	0	0.135
7	42	1	0	0	1	0	999	0.042

Table 3: Confusion table of random forest using all covariates

In Section 2.3, Figure 4 suggested that elevation and soil type might be the most important variables when it comes to classifying the cover types in our data set. Using the Gini index defined in Section 3.1, we can compute how much each variable contributed to decreasing the variability on average. We present such a measure in Figure 8. Evidently, elevation and soil type are the two variables that provide the model with most of it predictive power.

All in all, it looks as if the random forest is a suitable way of classifying the cover types of the different plots of land. We managed to find a model able of calssifying about 85% of the test data cover types correctly without having to add too much complexity to the model itself. We think it is interesting to find the power of even the simplest form of random forest models, and so we feel encouraged and well convinced that machine learning in general – and random forest in particular – are indeed are very important tool in the world of data analysis.

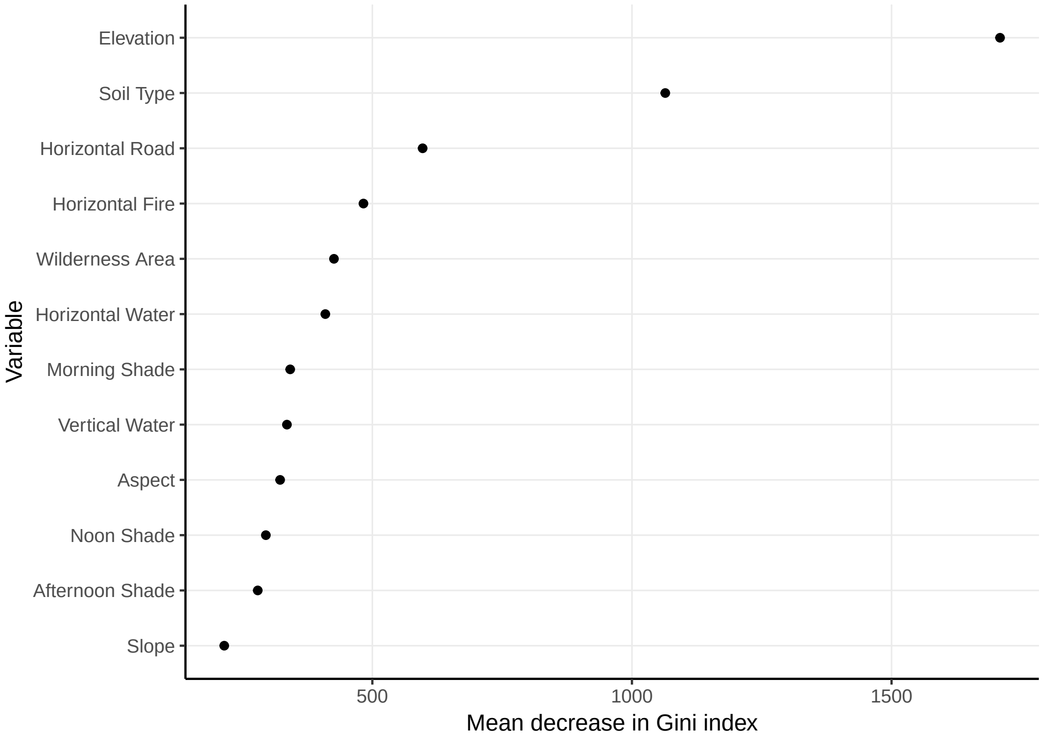

Figure 8: Plot showing the mean decrease in Gini index generated by each feature. Elevation and soil type look to be by far the most important

Finally, we want to compare the results of our random forest model to those of a model that might seem more intuitive at first glance, at least to those who are not statisticians by trade. Regression models often seem to be the threshold at which most people choose to stop learning about statistics. Hence, we want to explore the possibilities of classifying the cover types using models that most people can relate to at least by their names, and so we decide to use the multinomial logit regression model as a means of setting a benchmark for the random forest to try to beat.

5.3 Logit Model

The multinomial logit model at its very simplest manages to classify 70% of the test data cover types correctly, hence performing at a rate similar to that of the bivariate random forest. Looking at the confusion matrix shown in Table 4, we see that the error sources seem to be the same as before, namely cover types 1, 2, 3, and 6.

However, the fact we use all covariates available in the logit model, as opposed to only two in the bivariate forest, points to the power of the random forest as a classifying tool. Not only because it performs better, but also because it is not much harder to understand in any sense.

	Actual Cover Types							
Predicted	1	2	3	4	5	6	7	**Error Rate**
1	722	154	0	2	38	7	98	0.293
2	285	540	20	4	217	45	6	0.517
3	0	7	610	133	40	297	1	0.439
4	0	0	67	982	0	65	0	0.118
5	30	123	38	0	816	50	2	0.229
6	0	12	252	91	48	667	0	0.377
7	109	6	4	0	0	0	972	0.109

Table 4: Confusion table of multinomial logit model

6 Conclusion

As stated in the introduction of this report, we set out to examine how machine learning – random forests in particular – can be used to classify the forest cover type of any given plot of land. In doing so, we tried using three different methods, fitting four models. We have summarized the results of our study in Table 5, which shows that the random forest grown using all features available is by far the most successful in classifying the forest cover types.

	Model			
	Tree	**Reduced RF**	**Full RF**	**Logit**
Success Rate (%)	73.6	70	84.4	70

Table 5: Classification rates of our estimated models

Furthermore, we conducted a minor simulation study in order to determine what effect tree depth has on the performance of a random forest model. The results turn out to be quite surprising, pointing to the fact that increasing the number of terminal nodes will initially decrease the classification rate, which stands in stark contrast to the results presented in our methodological section, where the classification rate begins by increasing instead. It is hard to tell what causes this discrapancy, but perhaps it might be a case of overfitting or just the effect of introducing randomness to the data.

As any, our models do have limitations. Firstly, we would like to bring forward the way in which we went about tuning the random forest models – or choosing values of `maxnodes` and `ntree`. As discussed in Section 4.2, we determine the value of `maxnodes` first, and then use it as a given when finding a suitable value of `ntree`. Going through the literature, there seems to be a consensus stating that the number of trees used does not seem to have a great effect on random forest models when a certain level has been reached [James et al., 2013]. Hence, we chose to focus on tree depth first. It could be of interest, of course, to see what model would be deemed most suitable if the options were tried the other way around.

Secondly, the models we use are very simple, and we have not dug very far into the teory of random forests in order to find a model performing almost without fault. Instead, we have tried to understand how the basics of the models actually work, and thought of ways to fit good models that are relatively intuitive and easy to understand. While searching the web, however, we find extensions to the models that could be used to improve the rates of classification. The one that we find most interesting is adding the random forest as a kernel when using some other form of classification tool – like the support vector machine for example [Davies and Ghahramani, 2014].

Also, we believe that the DGP of our simulation study could be improved by using individual random forests rather than multinomial logit models when generating the new cover types. Such a proces would be very computationally intesive, and we chose to use logit models in the end. Now, the use of random forests could yield different results, but generally we believe that the main idea would be the same – namely that the classification rate converges to a certain level as the tree depth continues to increase.

Finally, what has been most striking to us is the sheer power of the random forest as a way of classifying data. Having only scratched the proverbial surface, we manage to grow a forest classifying more than 80% of the observations correctly, and so, we imagine getting well into the nineties would not be impossible if more advanced forests were to be grown. Perhaps most importantly, we find it very encouraging that such a high classification rate can be achieved using such an intuitive theoretical framework, and we have indeed grown very fond of machine learning techiques.

References

[Agresti, 2015] Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models.* John Wiley & Sons, Hoboken, New Jersey.

[Breiman et al., 2018] Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression.* R package version 4.6-14.

[Davies and Ghahramani, 2014] Davies, A. and Ghahramani, Z. (2014). The Random Forest Kernel and other kernels for big data from random partitions. *ArXiv e-prints.*

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, second edition.

[Hothorn et al., 2018] Hothorn, T., Hornik, K., Strobl, C., and Zeileis, A. (2018). *party: A Laboratory for Recursive Partytioning.* R package version 1.3-1.

[James et al., 2013] James, G., Witten, D., Hasite, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R.* Springer, New York.

[Ripley, 2016] Ripley, B. (2016). *nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models.* R package version 7.3-12.