Forest-3A Report

Forest cover type prediction

IAN BLAIR
MAX REHMAN LINDER

JANUARY 2024

MAP553 FOUNDATIONS OF MACHINE LEARNING

MAX.REHMAN-LINDER@POLYTECHNIQUE.EDU | IAN.BLAIR@POLYTECHNIQUE.EDU

1 Introduction and Data Exploration

The task consists of predicting cover-type of trees, based on various features of the environment. The types of cover is labeled 1-7. Each data-point also consists of 54 data-points that describes the environment in which the trees grow.

1.1 Variable Identification

Features	
Feature Name	Feature Data
Elevation	Elevation in meters (integer)
Aspect	Angel towards north in the horizontal plane (degrees)
Slope	Slope of terrain (degrees)
Horizontal Distance To Hydrology	Horisontal distance to body of water (meters)
Vertical Distance To Hydrology	Vertical distance to body of water (meters)
Hillshade 9am	Hillshade index at 9am (0 to 255 index)
Hillshade Noon	Hillshade index at noon (0 to 255 index)
Hillshade 3pm	Hillshade index at 3pm (0 to 255 index)
Horizontal Distance To Fire Points	Distance to nearest wildfire ignition points (meters)
Wilderness Area	Wilderness area designation (4 binary columns)
Soil Type	Soil Type designation (40 binary columns)
Cover Type	Forest Cover Type designation (integer 1 to 7)

We note that that the first 9 features are represented by integers that corresponds to its environment. Whereas the last 44 columns are binary and corresponds to two features, and they are therefore going to be more sparse. In other words, the density of information in those columns is intuitively going to be less dense that in the first columns.

One common strategy to reduce dimension is by using PCA. However, this method does not work well for very parse data and much information is unavoidably lost. Additionally, the number of entries in the training set is almost 3 orders of magnitude larger than the number of dimensions of training data.

We also note that there are15120 entries and of which there are 2160 of each cover-type.

1.2 Univariate analysis

Before the data is fed into the model of choice, to have a look at the variables themselves see if they provide any useful information. One method of doing so is by Mean-Decrease-Analysis, which creates a random forest based on the training data. The analysis is then done by predicting the values on the same data, but shuffling the values for one feature. The MDA value for that feature is the decrease in accuracy when it is shuffled. The idea is that if the precision decreases a lot if a feature is shuffled, the it is important. If it remains unaffected, that feature doesn't anything but noise.

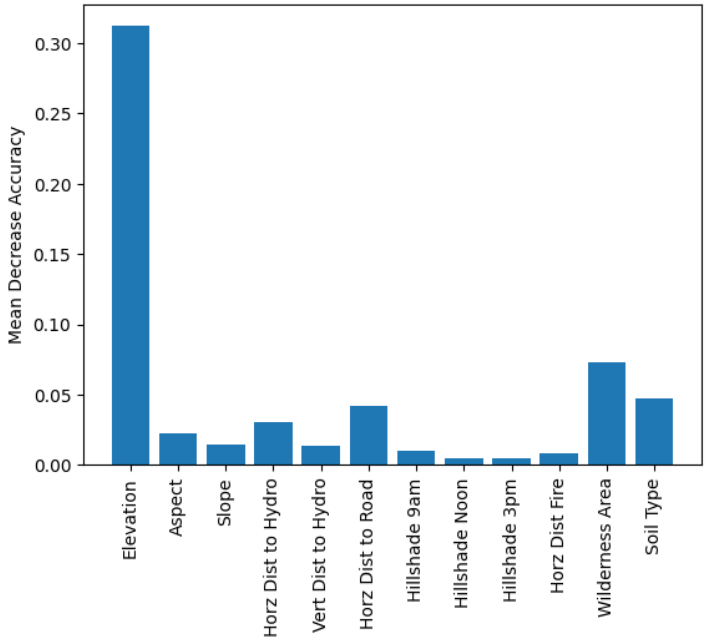
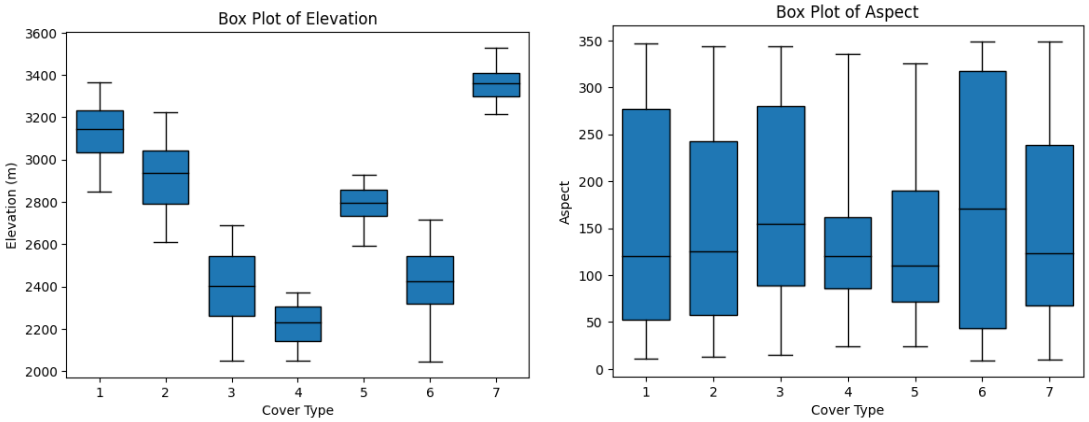


Figure 1: *MDA-analysis of features from a random forest of 300. The number oof features chosen to be evaluated at each tree was 7 ($\approx \sqrt{54}$ which is dimension of data).*

We note that elevation in Figure 1 by far is the most important feature. Comparing Elevation to the feature Aspect in Figure 2, we see that the data for elevation relatively separated between cover-types. This translates into elevation being a good predictor for classification.



(a) (b)

Figure 2: *Box plot of elevation (a) and and aspect (b). The box contains 50 % of the data, 25th to 75th percentile. The line contains 90% of the data.*

1.3 Covariate analysis

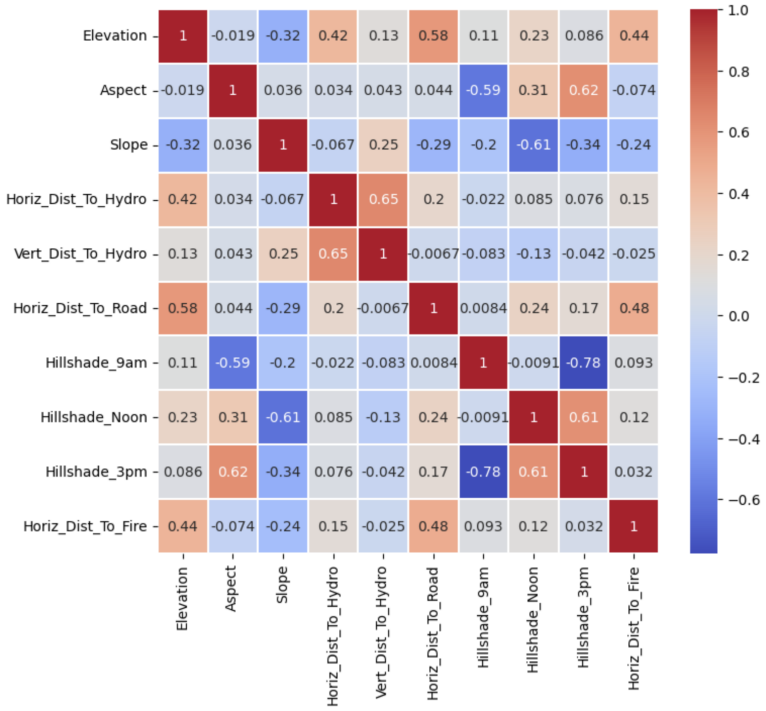


Figure 3: *Correlation Heatmap for 10 Features*

Notes: Using the above correlation heatmap, we that that only 5 pairs of features have an absolute value of covariance above 0.5. That being the amount of sun the trees receive combined with the aspect (slope), which is intuitive. The conclusion is that there is little to no need to exclude any data since all features have a positive MDA-value as can be seen in Figure 1 and they are not highly correlated, as can be seen in Figure 3

1.4 Scaling

While not necessary, scaling may be useful for helping decision tree accuracy. Specifically, the continuous features (elevation, distance to water, etc.) were scaled using sklearn.preprocessing's StandardScaler(). This proved to be slightly beneficial for the RandomForest Classifier, so it was kept in place for the other classifiers which made it easier to compare the models.

1.5 Feature Combination/Addition

As detailed later on, we discovered that some features were more important than others, and, using methods (e.g correlation heatmap, jointplots) we found online, analyzed the relationship between certain features (e.g elevation and distance to water), and realized that there was some correlation between the two. Thus, we made a couple new features that combined some features (total distance to water from horizontal and vertical distance to water), and added some features (elevation minus vertical hydrology). As seen in the feature importance graph, these became the most useful features later on.

2 Modeling

For models, the main method uses was various kinds of decision trees, and in particular random forrests of various kinds since they are easy to implement and yields reliably good results. While SVM's was experimented with, they did not yield as good of a result for us, and was therefore left out.

These trees have a few parameters in common that was optimized using cross-validation. In short, RandomizedSearchCV tries various hyper-parameters in a grid in order to find as good ones as possible. Increasing depth, reducing minimum split/leaf and concidering more features for each split can increase the complexity of the model but with the risk of overfitting.

2.1 Random Forests Classifier

Parameters to consider:

- 'num estimators': The number of trees in the forest
- 'max depth': The maximum depth of each tree
- 'min samples split': The minimum number of samples required to split an internal node
- 'min samples leaf': The minimum number of samples required to be at a leaf node
- "max features": The number of features to consider when looking for the best split

Model selection: RandomizedSearchCV to tune hyperparameters, using cross-validation

Best score on training set: 0.8406712734452121
Accuracy on validation set: 0.8587174348697395
Accuracy on testing set: 0.72687

Notes: We spent a long time on Random Forests, incorporated feature scaling and feature addition / combination, and attempted to tune hyperparameters manually, but could not get the testing accuracy as high as we would have liked.

2.2 AdaBoost Classifier

Parameters to consider:

- 'num estimators': The number of trees in the forest
- 'learning rate': Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier

Model selection: RandomizedSearchCV to tune hyperparameters, using cross-validation

Best score on training set: around 0.8
Accuracy on validation set: around 0.8
Accuracy on testing set: 0.72075

Notes: We used an AdaBoosted decision tree, which unfortunately gave us problems in terms of time it took for RandomizedSearchCV to find ideal hyperparameters, which led us away from this classifier.

2.3 Extra Trees Classifier (Classifier used)

Parameters to consider:

- 'num estimators': Specifies the kernel type to be used in the algorithm
- 'criterion': The function to measure the quality of a split
- 'max depth': The maximum depth of the tree

- 'min samples leaf': The minimum number of samples required to be at a leaf node

Model selection: RandomizedSearchCV to tune hyperparameters, using cross-validation

Best score on training set: 0.869101678183613
Accuracy on validation set: 0.8865731462925852
Accuracy on testing set: 0.78708

Notes: The ExtraTrees Classifier proved to be immediately successful, and efficient in terms of time to tune the hyperparameters. After just two iterations we got our best testing score, and decided to use it as our classifier.

3 Model Analysis and sanity-check

3.1 Example Tree

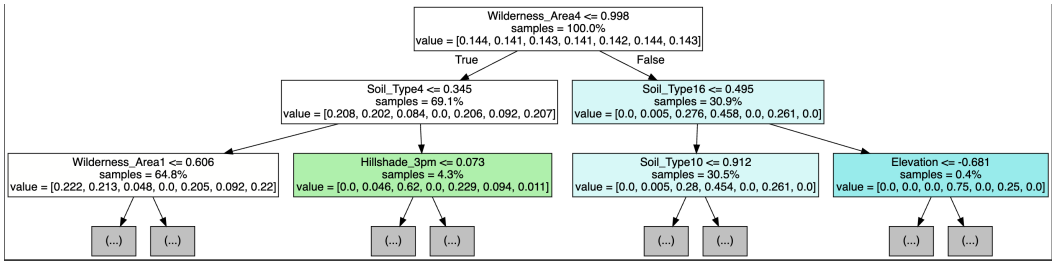


Figure 4: *example of a tree in RF classifier*

Notes: This is an example of a decision tree that was created in the Random Forest classifier. It allowed us to visualize how a classification might be made, and provided as another sanity check.

3.2 Histogram of predictions

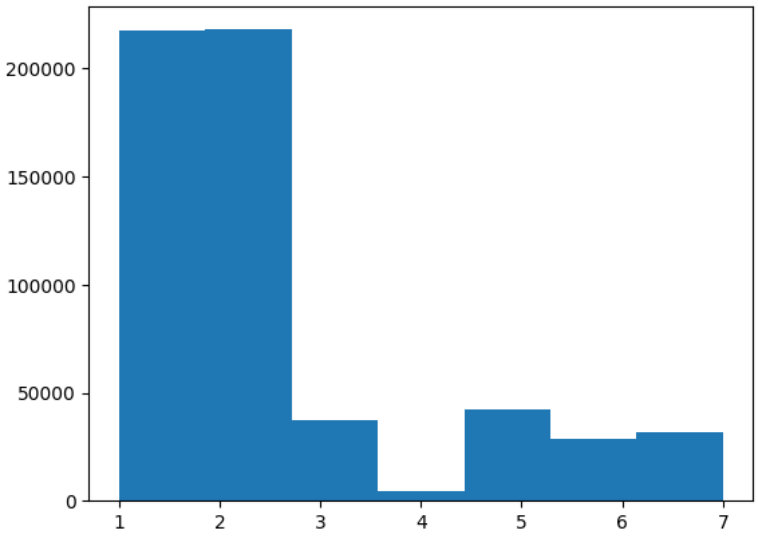


Figure 5: *Cover Histogram*

Notes: Although not analytically very useful, this provided a good sanity check to ensure that the predictions were made normally. When incorporating feature scaling, this histogram helped us identify an error we had where we were not scaling the testing data as well, leading to irregular predictions.
We can also see in Figure 5 that when the random forest is constructed, it is putting much weight to type 1 and 2, the reason for which we'll see in the next subsection.

3.3 Confusion Matrix

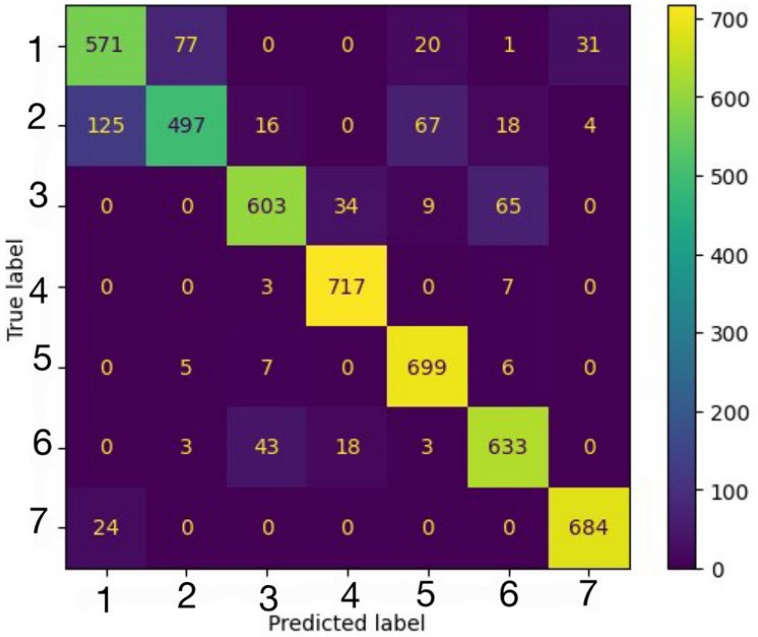


Figure 6: *Confusion Matrix*

Notes: The confusion matrix shows the occurrences of every prediction as well as if it was correct or not. As can be seen, type 1 & 2 are often confused with each other and are most commonly incorrectly

predicted. This explains figure 5. Since they are most often are wrong, the tree is constructed in such way that is puts weight on getting those right. We can also see in Figure 5 that label 4 is put very little weight onto. Looking at the confusion matrix, only 10 out of 727 predictions was wrong for this prediction which indicates that it is easily predicted.
Referring back to Figure 2 that show the boxplots of elevation, we see that type 1 & 2 have overlapping interval of elevation whereas type 4 is on the extreme end that plot. This might help indicate why 1 & 2 are mixed up whereas 4 is more easily predicted.

3.4 Top 10 Feature Importance

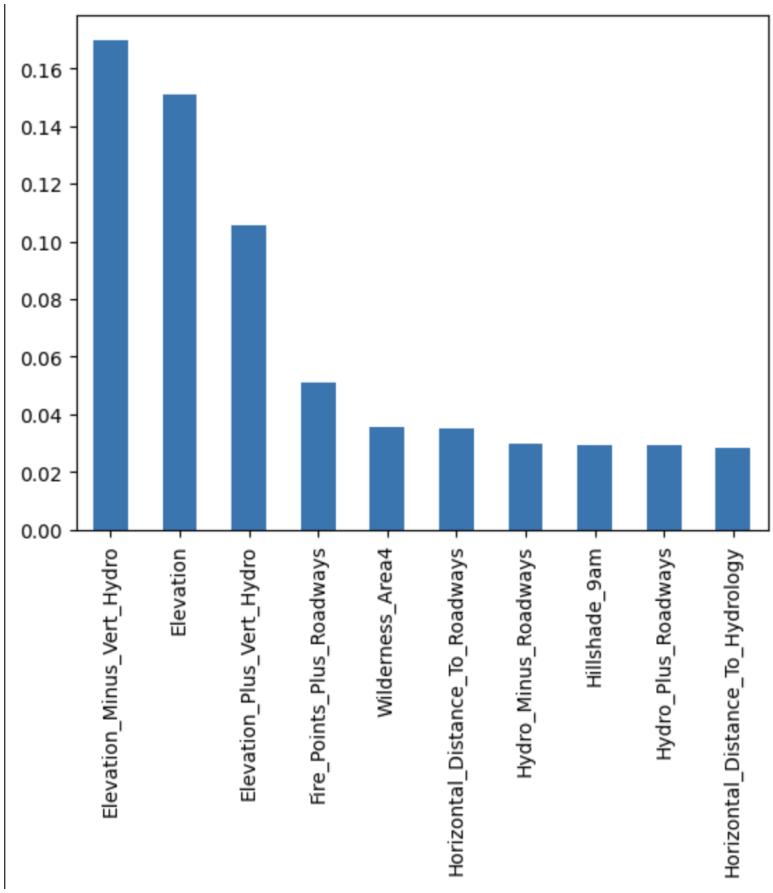


Figure 7: *Feature Importance*

Notes: The feature importance graph was insightful in helping to combine features, as we noticed that at first, elevation was by far the most important feature. We took the covariance of elevation and various other features, and decided to combine some of them into our new feature list. The graph was also informative in showing how some features were rarely used. This led us to dropping some features, though those features were later added back as we did not see any performance increase.