Forest Type Prediction in Kaggler

Luis Argerich

Saturday, August 02, 2014

Forest Type Prediction in Kaggler

Task Description

This competition is about predicting the forest cover type based on several geographical features such as Elevation, Distance to hydrology, slope, etc. There are a total of 7 (seven) different forest cover types so it's a multiclass classification problem.

Approach & Techniques

I used R for this project. The dataset is particularly tidy so it didn't need much pre-processing. I removed variables that had no variation at all in the training set. These are the pre-processing steps.

```r
library(randomForest)
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(C50)
library(ggplot2)

# Read datasets
train<-read.csv("train.csv")
test<-read.csv("test.csv")

# Pull cover_type and id
cover_type<-as.factor(train$Cover_Type)
id<-test$Id

# Remove cover_type from training set (this is what we want to predict)
train$Cover_Type<-NULL

# Join both datasets
combi <- rbind(train, test)

# Remove Unused variables
combi$Soil_Type7<-NULL
combi$Soil_Type15<-NULL

# Remove Ids
combi$Id<-NULL

# Recreate training and test datasets
```

```r
train <- combi[1:15120,]
test <- combi[15121:581012,]
```

Implementation

After exploring different algorithms I settled on a randomForest for this project, randomForests are easy to use and tune. I used tuneRF to find the best parameters and then used this:

```r
clf <- randomForest(train, as.factor(cover_type), ntree=500, mtry=8, importance=TRUE)
```

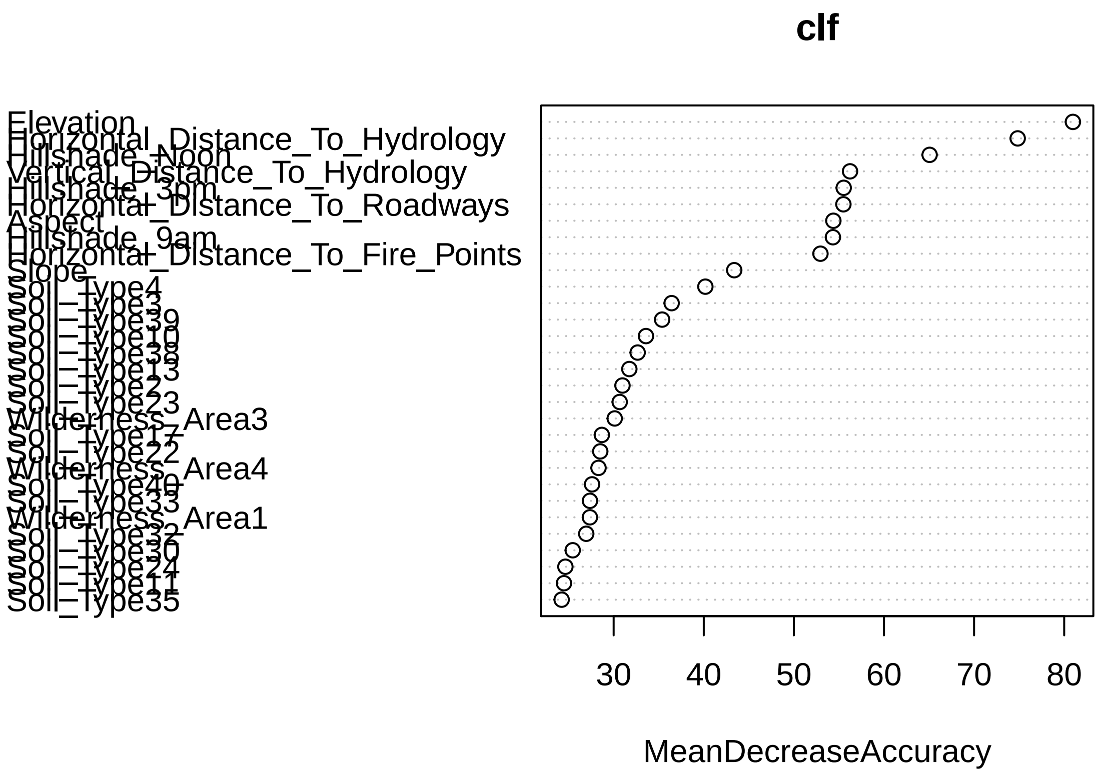
The returned object can be used to see how the algorithm fared displaying a confusion matrix.

```r
clf
```

```
##
## Call:
##  randomForest(x = train, y = as.factor(cover_type), ntree = 500,      mtry = 8, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 15.45%
## Confusion matrix:
##      1    2    3    4    5    6    7 class.error
## 1 1610  334    2    0   55   10  149     0.25463
## 2  363 1483   53    0  177   65   19     0.31343
## 3    0    5 1624  131   23  377    0     0.24815
## 4    0    0   32 2090    0   38    0     0.03241
## 5    1   63   38    0 2032   26    0     0.05926
## 6    1    8  196   69   14 1872    0     0.13333
## 7   81    4    0    0    2    0 2073     0.04028
```

Random Forests can also be used to explore the importance of features for prediction

```r
varImpPlot(clf,type=1)
```



It seems the Elevation feature is the most important one, and that makes sense because forest depends a lot on the terrain altitude. One problem with this is that an algorithm might use only the elevation to predict but a RandomForest will pick features randomly so it's forced to use different sets of attributes in different trees.

I created a prediction from the model and a submission with this code:

```r
result=predict(clf, test)
submit <- data.frame(Id = id, Cover_Type = result)
write.csv(submit, file = "basic_random_forest.csv", row.names = FALSE)
```

This got to around 0.75 score which is about the middle of the leaderboard.

Did it Work?

Yes this worked but in the confusion matrix it can be seen that the algorithm struggles with cover types 1 and 2. Sometimes a cover type 1 is classified as 2 and sometimes a 2 is classified as 1.

I created a plot to explore based on my two best predictors: Elevation and Horizontal Distance to Fire Points.

```r
qplot(train$Elevation,train$Horizontal_Distance_To_Fire_Points,color=cover_type)
```

The plot shows clearly that the Elevation is the best predictor and that there's a lot of mixing between forest cover type 1 and 2.

Improvements

I tried several different algorithms and feature engineering tricks without much success. The only small improvement I found was to use an autoencoder to create 8 features from all the sparse features in the dataset (soil types and wilderness areas). I took this from a post in the Kaggle forum and you can download the code from there, I won't include it in this report because it's quite large.

Using the Autoencoder I went down to 18 features and the score went up to 0.77 which was good enough to be in the 61st position at the time this report was written.

Further Work

I believe that the next effort should focus in how to untangle forest cover types 1 and 2. But I'm not sure how to do that yet.

Thanks for reading!