Machine Learning Final Project

Matthew Renze

April 22, 2017

Overview

In this project we will attempt to classify human activity based on data collected from accelerometers on the belt, forearm, and dumbells of six test subjects.

Load the Data

First, we will load the necessary package for our analysis.

```
library(caret)
```

Next, we will set a random seed to make our results reproducible

```
set.seed(42)
```

Then we'll set our working directory

```
setwd("C:/Work/School/Machine Learning/Project")
```

Next, we'll read the training data from the training CSV file

```
training.all <- read.csv("pml-training.csv")
```

Finally, we'll read the test data from the test CSV file

```
testing <- read.csv("pml-testing.csv")
```

Transform and Clean the Data

First, we will remove the first seven columns since they contain metadata rather than sensor data. These variables will leak information that will increase the prediction accuracy of our model in unrealistic ways. So they should be excluded to ensure our predictions are being done only on sensor data.

```
training.all <- training.all[, -(1:6)]

testing <- testing[, -(1:6)]
```

Next, we will remove variables with nearly zero variance since they have little predictive power relative to the complexity they add to our model.

```r
near.zero.var <- nearZeroVar(training.all)

training.all <- training.all[, -near.zero.var]

testing <- testing[, -near.zero.var]
```

Finally, we will remove variables that contain 95% or more NA values since they provide little predictive power relative to the complexity they add to our model.

```r
mostly.na <- sapply(training.all, function(x) mean(is.na(x))) > 0.95

training.all <- training.all[, mostly.na == FALSE]

testing <- testing[, mostly.na == FALSE]
```

Split the Data

First, we will randomly sample indexes to create our training set. We'll use 75% of the values for our training set and 25% for our test set. This specific ratio was chosen as it is a best practice in the industry.

```r
index <- createDataPartition(
  y = training.all$classe,
  p = 0.75)[[1]]
```

Next, we will use the indexes to create our training set

```r
training <- training.all[index, ]
```

Finally, we will create our validation set

```r
validation <- training.all[-index, ]
```

Create a Decision Tree Classifier

We will first create a decision-tree classifier to see what type of accuracy the model will provide. We chose this algorithm to start with because it produces models that are simple and easy to interpret.

First, we will create a decision-tree model using our training set.

```r
tree.model <- train(
  form = classe ~ .,
  data = training,
  method = "rpart")
```

```
## Loading required package: rpart
```

Next, we'll use our decision-tree model to predict values from our validation set.

```
tree.predictions <- predict(
  object = tree.model,
  newdata = validation)
```

Then, we’ll create a confusion matrix to assess our model’s prediction accuracy.

```
tree.matrix <- confusionMatrix(
  data = tree.predictions,
  reference = validation$classe)

print(tree.matrix)
```

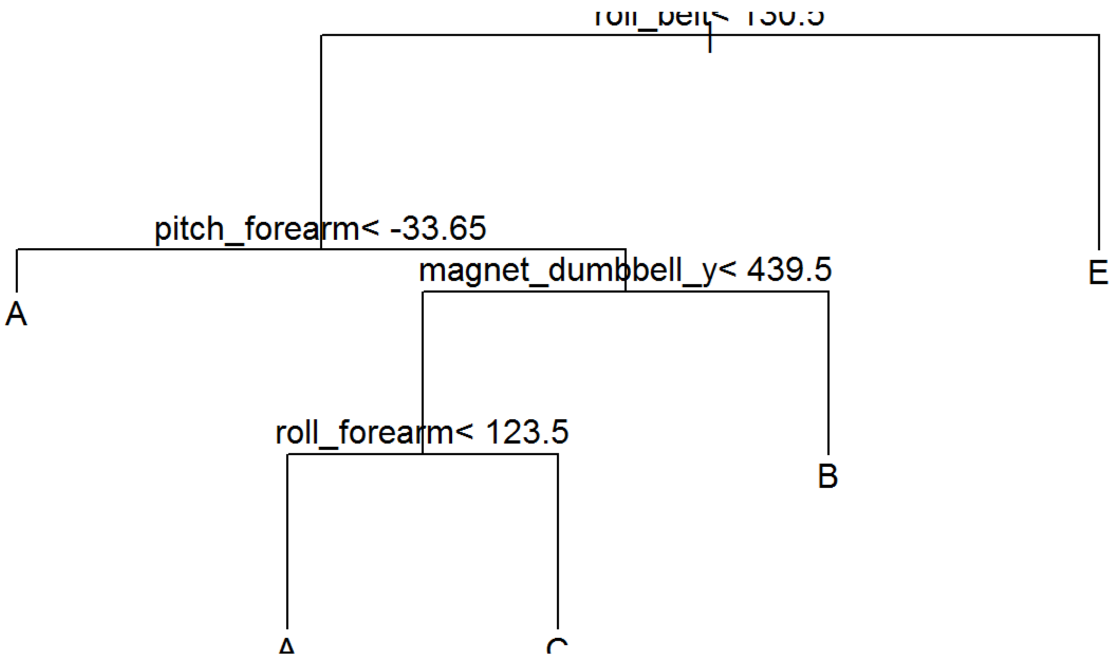
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1282  411  408  365  134
##          B   23  307   29  148  124
##          C   86  231  418  291  238
##          D    0    0    0    0    0
##          E    4    0    0    0  405
##
## Overall Statistics
##
##                Accuracy : 0.4918
##                  95% CI : (0.4778, 0.5059)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3349
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9190   0.3235  0.48889   0.0000  0.44950
## Specificity            0.6244   0.9181  0.79106   1.0000  0.99900
## Pos Pred Value         0.4931   0.4865  0.33070      NaN  0.99022
## Neg Pred Value         0.9510   0.8498  0.87995   0.8361  0.88966
## Prevalence             0.2845   0.1935  0.17435   0.1639  0.18373
## Detection Rate         0.2614   0.0626  0.08524   0.0000  0.08259
## Detection Prevalence   0.5302   0.1287  0.25775   0.0000  0.08340
## Balanced Accuracy      0.7717   0.6208  0.63997   0.5000  0.72425
```

As we can see from the confusion matrix, the accuracy of our decision tree model is only `tree.matrix$overall[1]` . This is a low accuracy so we will need to try a more complex model to see if our prediction accuracy improves.

Finally, before we attempt to create a more powerful model, let’s visualize the decision tree to see which variables were used in the model.

```
plot(tree.model$finalModel)

text(tree.model$finalModel)
```



Create a Random-forest Classifier

Now we will attempt to create a more powerful model in attempt to increase our prediction accuracy. We will use a random-forest classifier since it provides significantly more power by using a series of decision-tree classifiers as an ensemble.

We will manually set the training control to specify the use of k-fold cross-validation, with k = 5 folds. This number was chosen to reduce the number of iterations necessary to construct our model relative to the potential increase in prediction accuracy of a larger k.

First, we will create a random-forest classifier.

```
forest.model <- train(
  form = classe ~ .,
  data = training,
  method = "rf",
  trControl = trainControl(
    method = "cv",
    number = 5))
```

Next, we will use our random-forest classifier to predict values for our validation set.

```
forest.predictions <- predict(
  object = forest.model,
  newdata = validation)
```

Then, we will create a confusion matrix to assess our new model's accuracy.

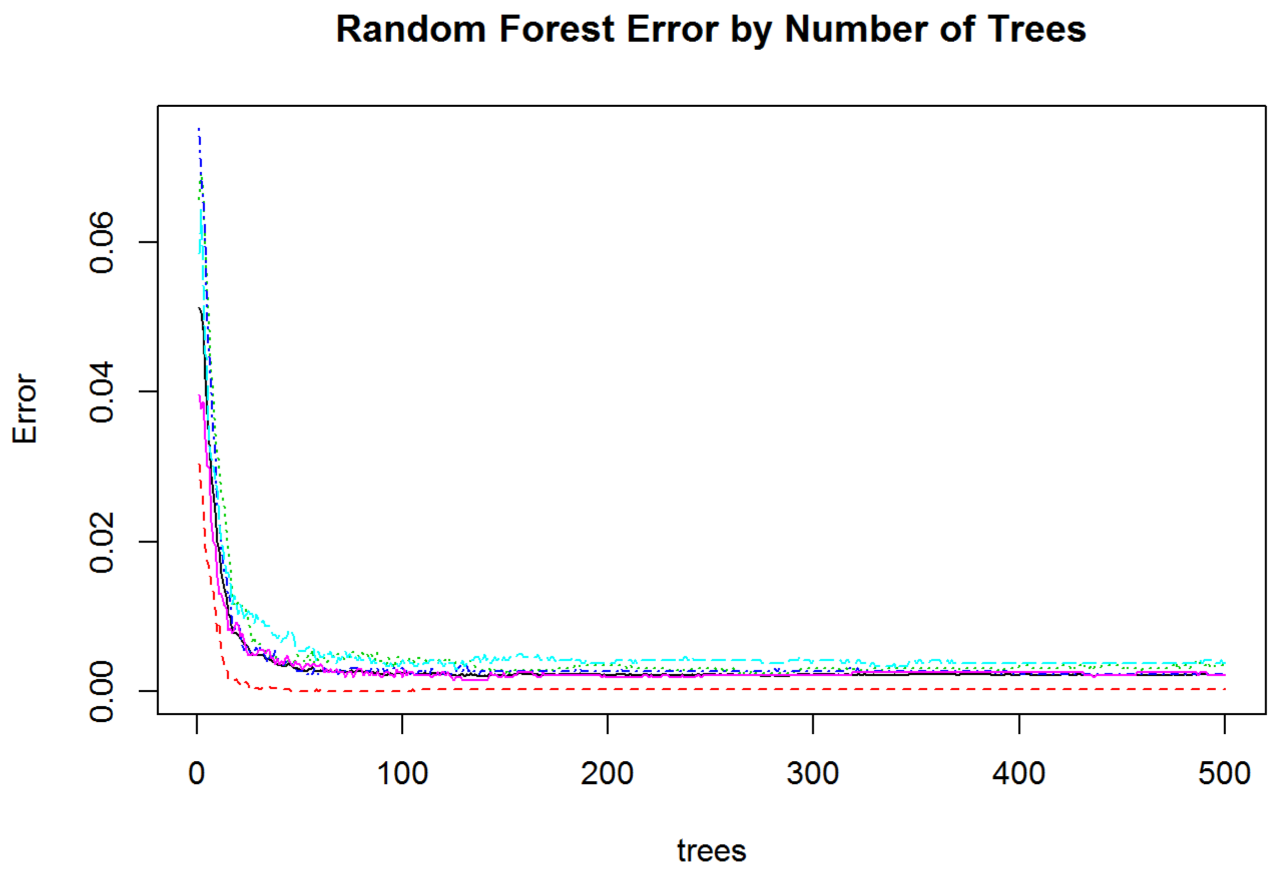
```
forest.matrix <- confusionMatrix(
  data = forest.predictions,
  reference = validation$classe)
```

As we can see from our results, the random-forest classifier has an accuracy of `forest.matrix$overall[1]` , which is significantly better than our decision-tree classifier.

This model's prediction accuracy should be sufficient, so we will use it to make our final predictions.

Before we make our final predictions, let's visualize the decrease in error rate of our random-forest classifier as a function of the number of trees in our model.

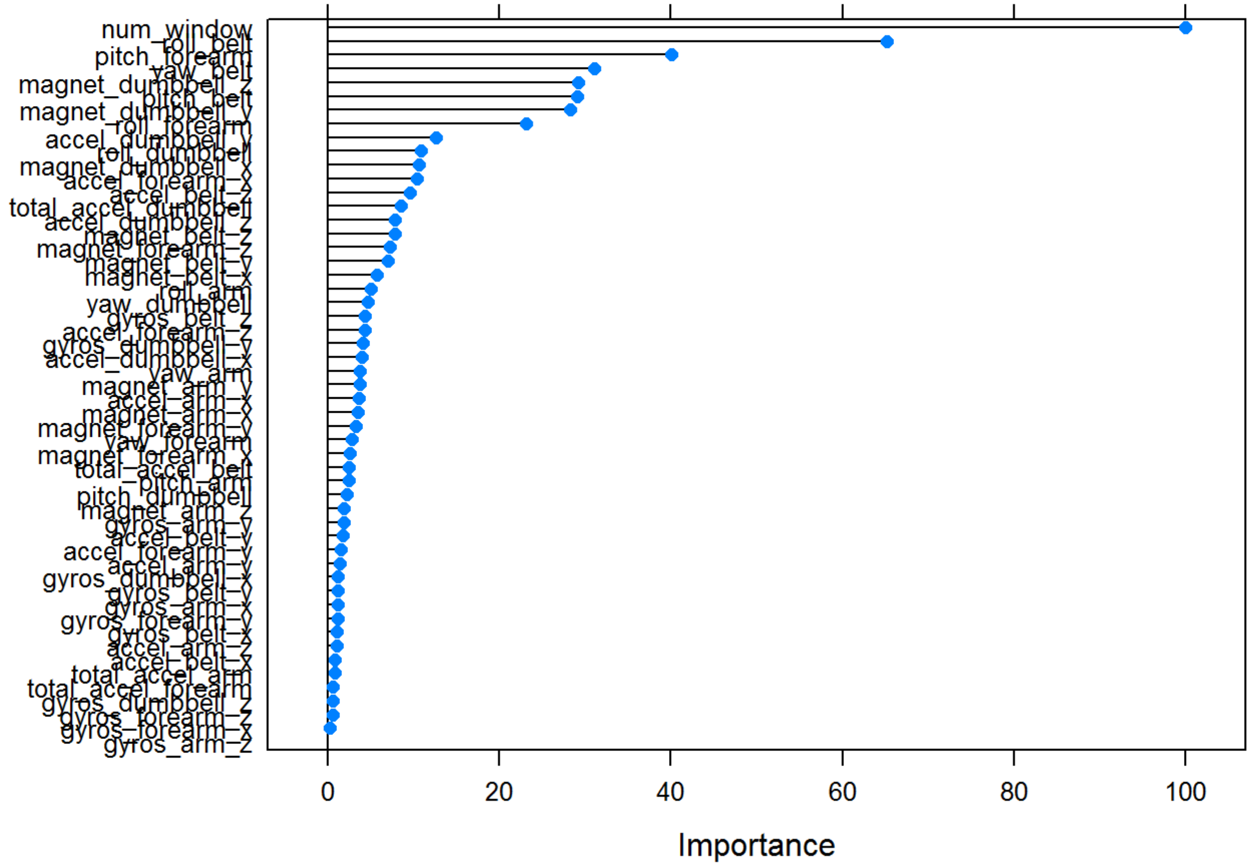
```
plot(
  x = forest.model$finalModel,
  main = "Random Forest Error by Number of Trees")
```



As we can see, the error rate goes done as we increase the number of trees in our random forest until around 50 trees. At this point, the error rate remains relatively flat.

In addition, let’s visualize the variable importance of our random-forest model to see which variables contributed most to our predictions.

```
plot(varImp(forest.model))
```



Predict Final Values

Finally, we will use our random-forest classifier to make our final predictions using the test dataset.

```
final.predictions <- predict(
  object = forest.model,
  newdata = testing)

print(final.predictions)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

These results were submitted to the Coursera website for verification. As a result of this verification process we know that our accuracy of our predictions on the test set was 100%.

While we would typically expect our in-sample error rate to be lower than our out-of-sample error rate, in this case, our out-of-sample error rate was lower than our in-sample error rate. This is likely due to the small sample size of our test data set.