Machine Learning Assignment

Fabiana Glenn

September 16, 2016

1. Executive Summary

Predict how well individuals perform physical exercise from a dataset of 19622 observations and 160 variables. Given the high number of variables, we used trees and random forest for prediction and achieved an accuracy of 99%.

2. Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal of this project is to predict the manner in which they did the exercise (the variable “classe”).

3. Data Processing and Environment Setting

Downloading the file and loading the information:

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "M
Ltrain.csv", method="curl")
training <- read.csv("MLtrain.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "ML
test.csv", method="curl")
validation <- read.csv("MLtest.csv")
```

Let’s understand the structure of the data provided:

```
str(training) # not showing results because because of size
summary(training) # not showing results because because of size
```

There are 19622 observations and 160 variables. There are many variables with 19216 NAs (that is, 97% of observations within that variable) and plenty of other “empty observations”. In order to create a model that works, we need to make a serious “cleaning” of these types of observations.

Let’s also load the necessary packages for our analysis and set a seed to make sure the results are replicable:

```
library(caret)
library(rpart)
library(rattle)
library(randomForest)
library(ggplot2)
library(klaR)
set.seed(78911)
```

There are a number of variables that do not contribute to the model, such as timestamps and the such. They will be deleted:

```
training <- training[,-c(1:6)]
validation <- validation[,-c(1:6)]
```

As mentioned earlier, there is an enormous number of NA values - 19216 out of a total of 19622 lines. In order to simplify the analysis, we can remove these columns.

```
clean <- apply(!is.na(training), 2, sum)>19621
training2 <- training[,clean]
```

Another problem is that the model will probably not work if there are too many empty values - we need to locate and eliminate them. We can do that with the “nearZeroVar” function, that highlights arrays with zero covariates:

```
emptycol <- nearZeroVar(training2)
if (length(emptycol)>0) {
    training2 <- training2[,-emptycol]
}
```

3. Data slicing

Since the testing dataset is quite big with 19622 observations, we can divide it in a 70% training/30% testing dataset.

```
inTrain <- createDataPartition(y=training2$classe, p=0.7, list=FALSE)
train <- training2[inTrain,]
test <- training2[-inTrain,]
tc <- trainControl(method="repeatedcv", number=10, repeats=3)
train$classe <- as.factor(train$classe)
```

Please note that it was also provided a 20 observation dataset. We will use that as an extra layer of validation of the model later on.

4. Building the Model, Cross-validation and why Random Forest was chosen

We are going to work with trees and random forest models because even with all the cleaning, the dataset still has 54 variables. Let’s build an initial random forest with 120 trees in order to have a basis to understand which variables have more importance for the model:

```
trainrf <- randomForest(classe~., data=training2, importance=TRUE, ntree=120, trControl
= tc)
trainrf2 <- train(classe~., data=training2, ntree=120)
```

With random forest, we can use a k-fold cross validation that will lead us to an unbiased estimate of the out-of-sample error. We are going to use the repeated cross validation method with 10 folds and 3 repetitions.

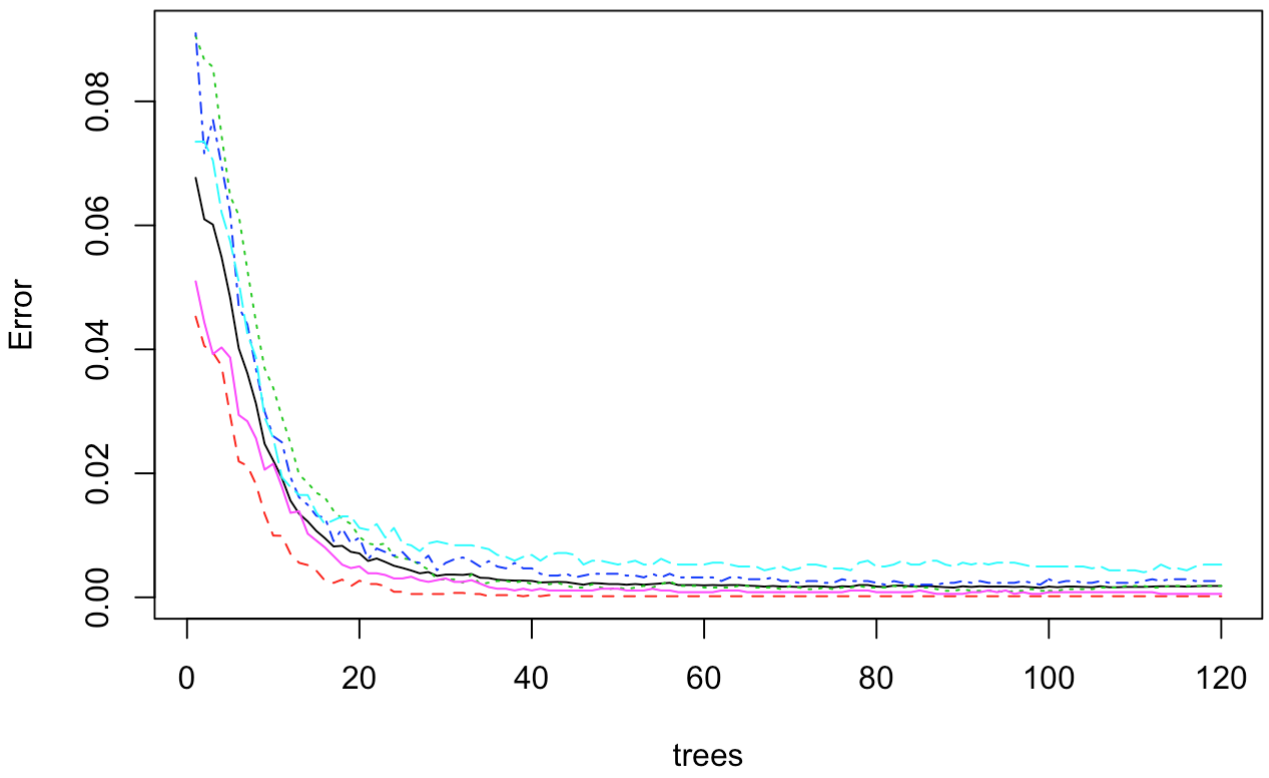
```
trainrf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training2, importance = TRUE,      ntree =
 120, trControl = tc)
##                Type of random forest: classification
##                      Number of trees: 120
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.18%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5579    1    0    0    0 0.0001792115
## B    6 3790    1    0    0 0.0018435607
## C    0    8 3413    1    0 0.0026300409
## D    0    0   15 3199    2 0.0052860697
## E    0    0    0    2 3605 0.0005544774
```

With k-fold cross validation with 10 folds and 3 repetitions, we achieved 99.82% of accuracy and 0.18% out-of-sample error, so there is a strong indication that this model might be overfitted.

This graph shows the error in relation to the amount of trees:

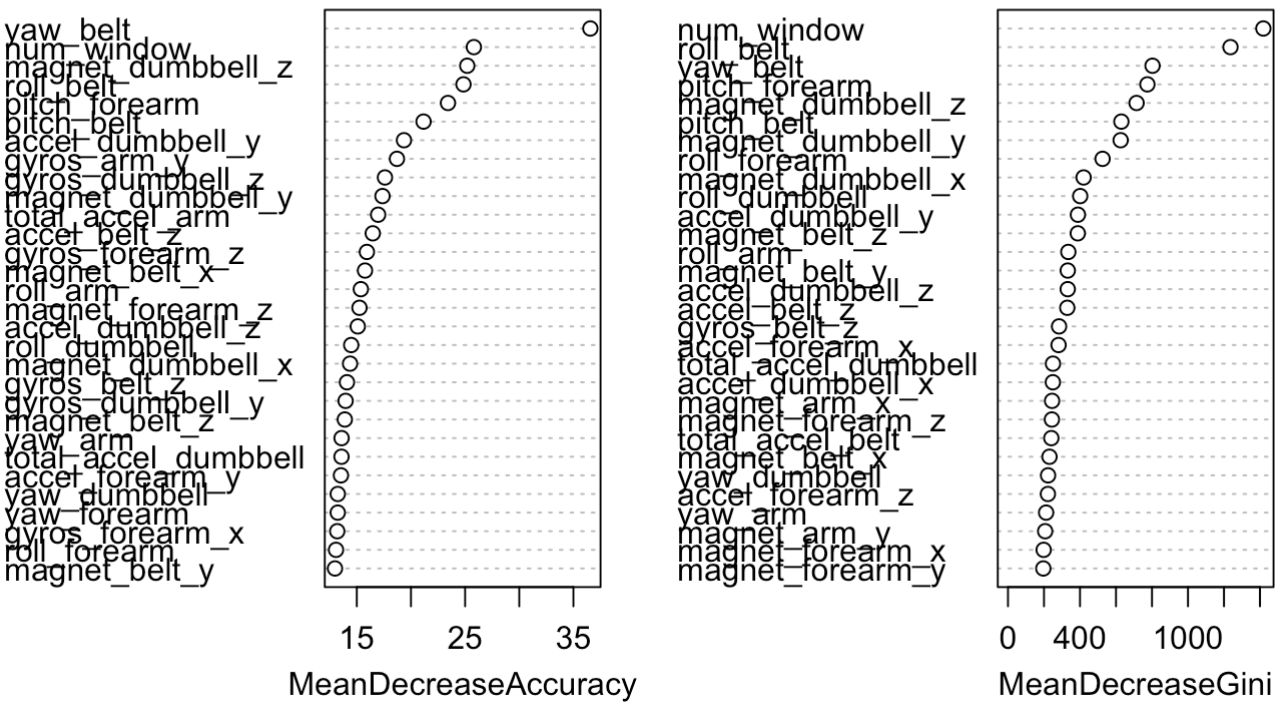
```
plot(trainrf, main = "Number of trees vs. Estimated Error Rate")
```



The solid black line shows overall out-of-bag error and the other lines shows each class.

The random forest model can also estimate variable importance, which is helpful to understand which variables are really importance in such a complex dataset.

```
varImpPlot(trainrf, main = "Importance of Variables in Model")
```



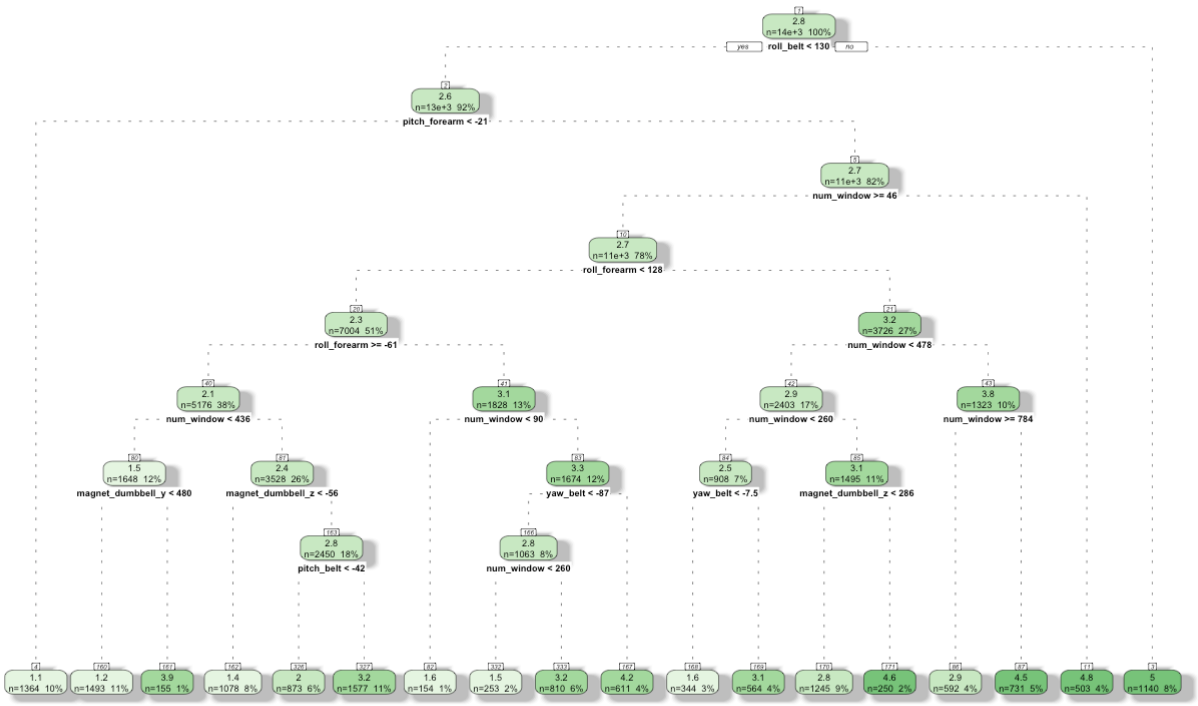
Here we can see the most important variables in the model.

MeanDecreaseAccuracy: yaw_belt, num_window, roll_belt, pitch_belt, pitch_forearm, magnet_dumbbell_z, magnet_dumbbell_y, gyros_forearm_z, gyros_dumbbell_z. This graph shows how much accuracy the model would lose if any of the variables were removed.

MeanDecreaseGini: num_window, roll_belt, pitch_forearm, pitch_belt, magnet_dumbbell_z, magnet_dumbbell_y, roll_forearm. This graph shows how much node purity (Gini index) the model would lose if the variables were removed.

Now we can make a simpler, leaner model with the most important variables. First, let's make a tree to understand how these variables behave:

```
tree <- rpart(classe~+yaw_belt+num_window+roll_belt+pitch_belt+pitch_forearm+magnet_dumb
bell_z+magnet_dumbbell_y+gyros_forearm_z+gyros_dumbbell_z+roll_forearm, data=train, meth
od="anova")
fancyRpartPlot(tree)
```



Rattle 2016-Sep-26 15:31:08 fabiana

Now, let's make a definitive random forest only with the important variables:

```
rf <- train(classe~yaw_belt+num_window+roll_belt+pitch_belt+pitch_forearm+magnet_dumbbel
l_z+magnet_dumbbell_y+gyros_forearm_z+gyros_dumbbell_z+roll_forearm, data=train, method="r
```

5. Expected Out of Sample Error

```
rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.15%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B    4 2652    2    0    0 0.0022573363
## C    0    4 2391    1    0 0.0020868114
## D    0    0    2 2250    0 0.0008880995
## E    0    3    0    4 2518 0.0027722772
```

Interestingly enough, with this simpler model we have a lower out of sample error of 0.15% compared to 0.18% with all other variables.

Let's test how the model does with the test dataset:

```
pred1 <- predict(rf, test)
confusionMatrix(rf)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.2  0.1  0.0  0.0  0.0
##          B  0.0 19.2  0.0  0.0  0.1
##          C  0.0  0.1 17.4  0.0  0.0
##          D  0.0  0.0  0.0 16.4  0.1
##          E  0.0  0.0  0.0  0.0 18.4
##
##  Accuracy (average) : 0.9961
```

```
table(pred1, test$classe)
```

```
##
## pred1    A    B    C    D    E
##     A 1674    1    0    0    0
##     B    0 1137    3    0    2
##     C    0    1 1023    0    0
##     D    0    0    0  964    0
##     E    0    0    0    0 1080
```

Our algorithm predicts with 99% of accuracy the classes in the test datataset.

Now let's make a prediction for the validation numbers that needs to be submitted together with this assignment:

```
pred2 <- predict(rf, validation)
pred2
```

```
##  [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
```

6. Conclusion

The goal of this assignment was to predict how well 6 participants would perform physical activity. Ultimately, a random forest model was used for prediction because initially there were more than 19600 observations and 150 variables. We reached the accuracy of more than 99% and an error of 0.15%. I believe that in "real life", finding such a high accuracy will not be possible, but it was a great opportunity to learn more about machine learning.