Practical Machine Learning - Prediction Assignment

James Portman

March 4, 2016

Background

By 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. Six participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The data is from accelerometers on the belt, forearm, arm, and dumbell of the participants. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Goal

Predict the manner in which the participants did their exercises. This paper describes the prediction model built, how cross validation was used, and what the expected sample error was. The prediction model will be used to predict 20 different test cases.
The training data used was copied locally from:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
The test data used was copied locally from: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Contents


```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

1) Preparation of Datasets

1.1 Load data and replace #DIV/0! with an NA value.

```
trainURL <- "/Users/admin/Documents/training.csv"
testURL <- "/Users/admin/Documents/test.csv"
training <- read.csv(trainURL, na.strings=c("#DIV/0!"), row.names = 1)
testing <- read.csv(testURL, na.strings=c("#DIV/0!"), row.names = 1)

dim(training)
```

```
## [1] 19622   159
```

There are 19,622 observations across 159 factors.

Distribution of the five measured stances A,B,C,D,E is:

1.2 Cleanse data

```
# Remove variables that are irrelevant : user_name, raw_timestamp_part_1, raw_time
stamp_part_,2 cvtd_timestamp, new_window, and num_window
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]

# Exclude near zero variance features
nzvColumns <- nearZeroVar(training)
training <- training[, -nzvColumns]

# Exclude features with NA
training <- training[, which(as.numeric(colSums(is.na(training)))==0)]

dim(training)
```

```
## [1] 19622    52
```

There are 52 factors after cleansing the data.

1.3 Partition Training data into Training and Validation sets

```
# Partition into training and validating sets
trainSet <- training[sample(nrow(training), 14000), ] # Pick 1400 random observati
ons for our Training set.
validationSet <- training[sample(nrow(training), 5000), ] # Pick 5000 random obser
vations for our Validation set.
```

2) Training a Prediction Model

```
lda_model <- train(classe ~ ., data=trainSet, method="lda")
```

```
## Loading required package: MASS
```

```
lda_Accuracy <- confusionMatrix(trainSet$classe,predict(lda_model,trainSet))$overa
ll[1]
lda_Accuracy
```

```
##  Accuracy
## 0.6920714
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
gbm_model <- train(classe ~ ., data=trainSet, method="gbm")
gbm_Accuracy <- confusionMatrix(trainSet$classe,predict(gbm_model,trainSet))$overa
ll[1]
gbm_Accuracy
```

```
##  Accuracy
## 0.9731429
```

rf_model <- train(classe ~ ., data=trainSet, method="rf")

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf_Accuracy <- confusionMatrix(trainSet$classe,predict(rf_model,trainSet))$overall
[1]
rf_Accuracy
```

```
## Accuracy
##        1
```

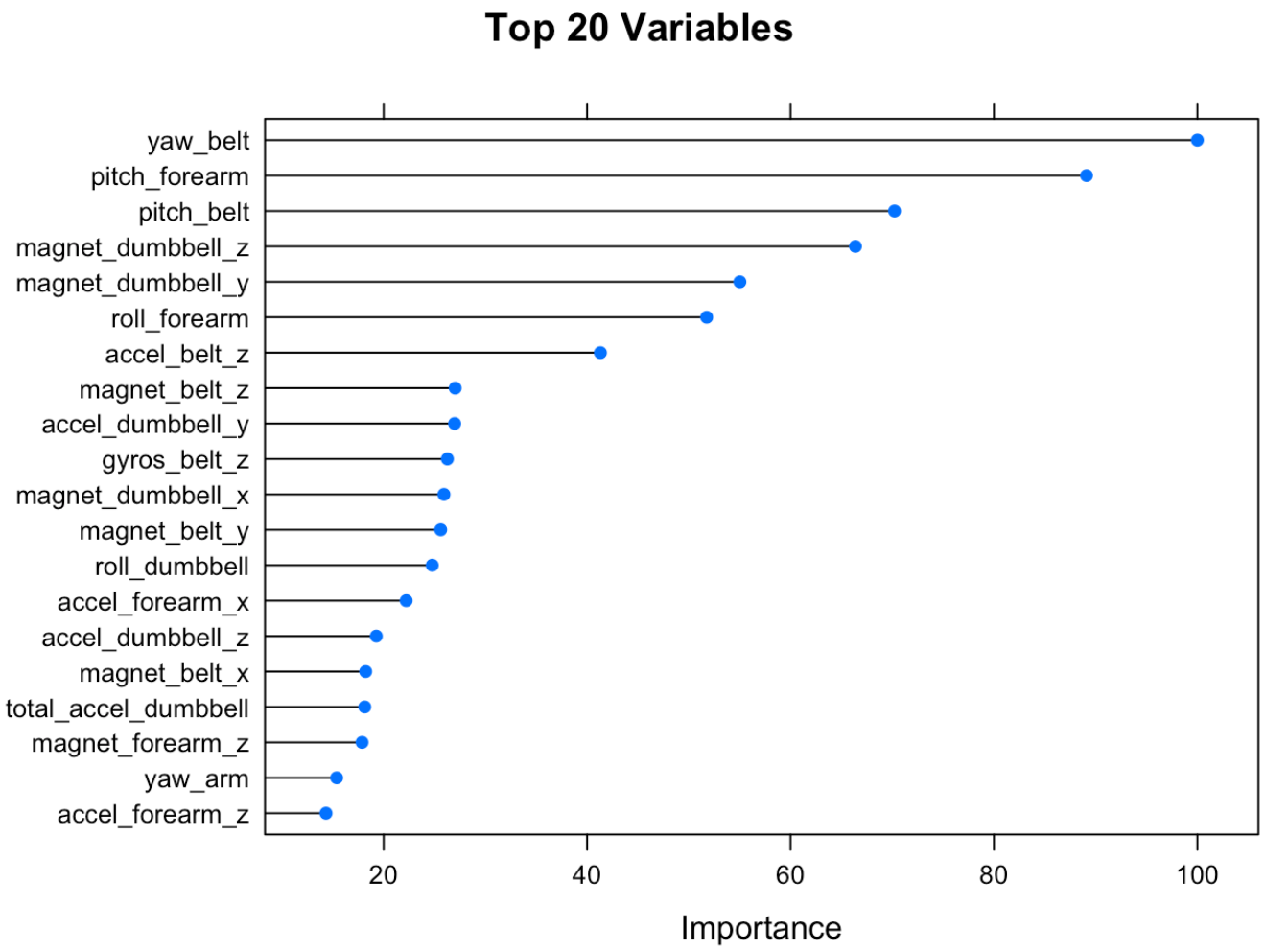
```
# rf_Accuracy is 1.
```

3) Model Evaluation

Base on the Accuracy results above, the Random Forest model was more accurate that the Gradient boosting model or the Linear Discriminant Analysis model.

Let’s review the top 20 variables.

```
rfObject <- varImp(rf_model)
plot(rfObject, main = "Top 20 Variables", top = 20)
```



4) Prediction on Validation data

Let's see what our out-of-sample accuracy is by seeing how our model performs on our validation set that we held out from our training set.

```
pValidation <- predict(rf_model, validationSet)
validation_rf_Accuracy <- confusionMatrix(pValidation, validationSet$classe)$overa
ll[1]
validation_rf_Accuracy
```

```
## Accuracy
##   0.9984
```

5) Prediction on Test data

The prediction of our algorithm for the test set is:

```
pTest <- predict(rf_model, testing)
pTest
```

```
##  [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) Conclusions

The Random Forest method worked with an accuracy on the Test set with an out of sample error of 99.8%.

Randoms forests are good at modelling large number of factors when the relationships between factors is not known. In this project, a Random forest model was able to handle unscaled variables and categorical variables.

7) Coursera Submission

```
# Save the output to files according to instructions and post to the Coursera Subm
ission page.
answers <- as.vector(pTest)
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
            col.names = FALSE)
    }
}

pml_write_files(answers)
```