Machine Learning Project 1

Brent Kuenzi

December 23, 2015

Synopsis

Using data generated from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who 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.

Data and Dependencies

Lets start by loading the data and required packages

```
library(caret); library(ggplot2); library(dplyr); library(corrplot)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
registerDoParallel(cores=2)

train_set <- read.csv(file="pml-training.csv")
test_set <- read.csv(file="pml-testing.csv")
```

Preprocessing

First we will split the **training** set into *test* and *train* set for accuracy predictions. Then will will perform some basic preprocessing. For ease of use, we will perform the following preprocessing steps:

- Remove columns with all NA values
- Remove the first column (index variable)
- Remove user_name, timestamp and window variables which are not useful for prediction

```
set.seed(1234); library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
inTrain <- createDataPartition(y=train_set$classe, p=0.7, list=FALSE)
cv_train <- train_set[inTrain,]
cv_test <- train_set[-inTrain,]
```

There are lots of columns with mostly NA values and these should be removed as they will not be useful for prediction

```
test_NA <- sapply(cv_test, function(x) {sum(is.na(x))})
table(test_NA)
```

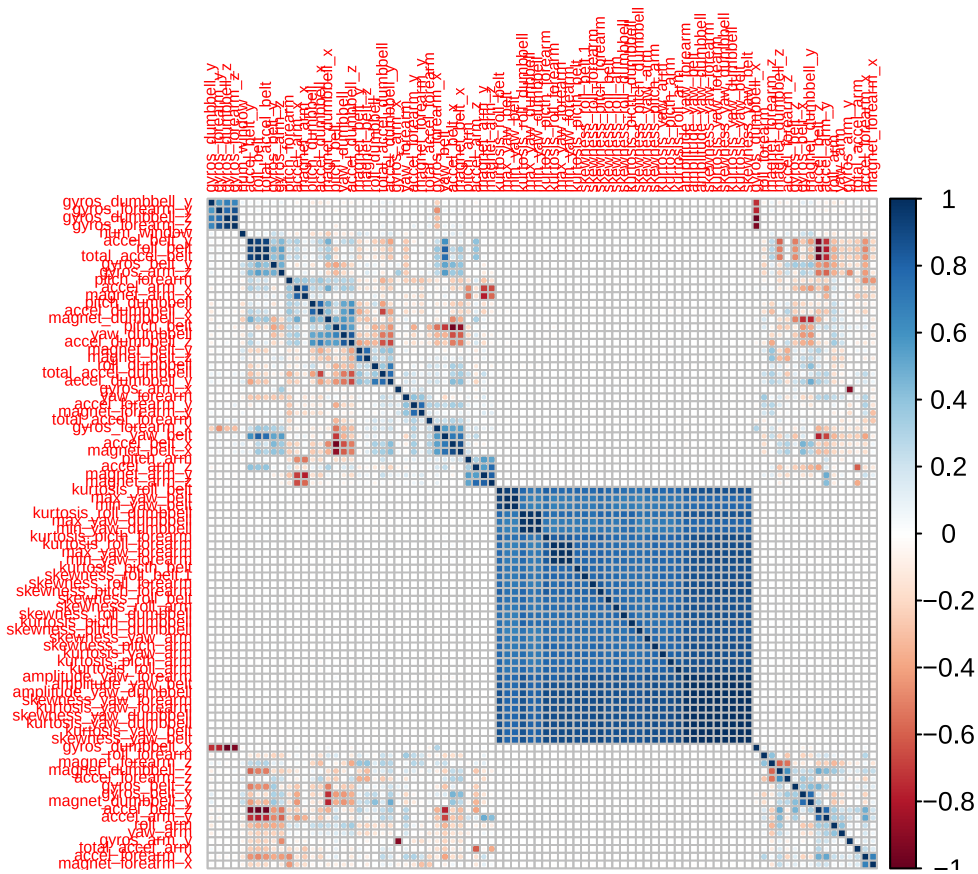
```
## test_NA
##    0 5763
##   93   67
```

```
# remove columns of all NA values
train_col <- cv_train[, colSums(is.na(cv_train)) != 13453]
# remove unnecessary timestamp and factor variables
train <- train_col[,-(1:6)]
train[,1:86] <- as.data.frame(sapply(train[,1:86], function(x) {as.numeric(x)}))
# Same with test set
test <- cv_test[, colnames(cv_test) %in% colnames(train)]
test[,1:86] <- as.data.frame(sapply(test[,1:86], function(x) {as.numeric(x)}))
```

Data Exploration

A number of variables within the training set seem to be highly correlated. This should allow modeling to create a rather robust prediction.

```
corrs <- cor(subset(train, select=-c(classe)))
corrplot(corrs, order="hclust", tl.cex=0.5)
```



Model Building and Evaluation

We will now create a model to predict the *classe* variable using a random forest model

```
MyTrainControl=trainControl(
  method = "cv",
  number=5,
  returnResamp = "all",
   classProbs = TRUE
)
model <- train(classe~., method="parRF", trControl = MyTrainControl, data=train)
```

```
## Loading required package: e1071
```

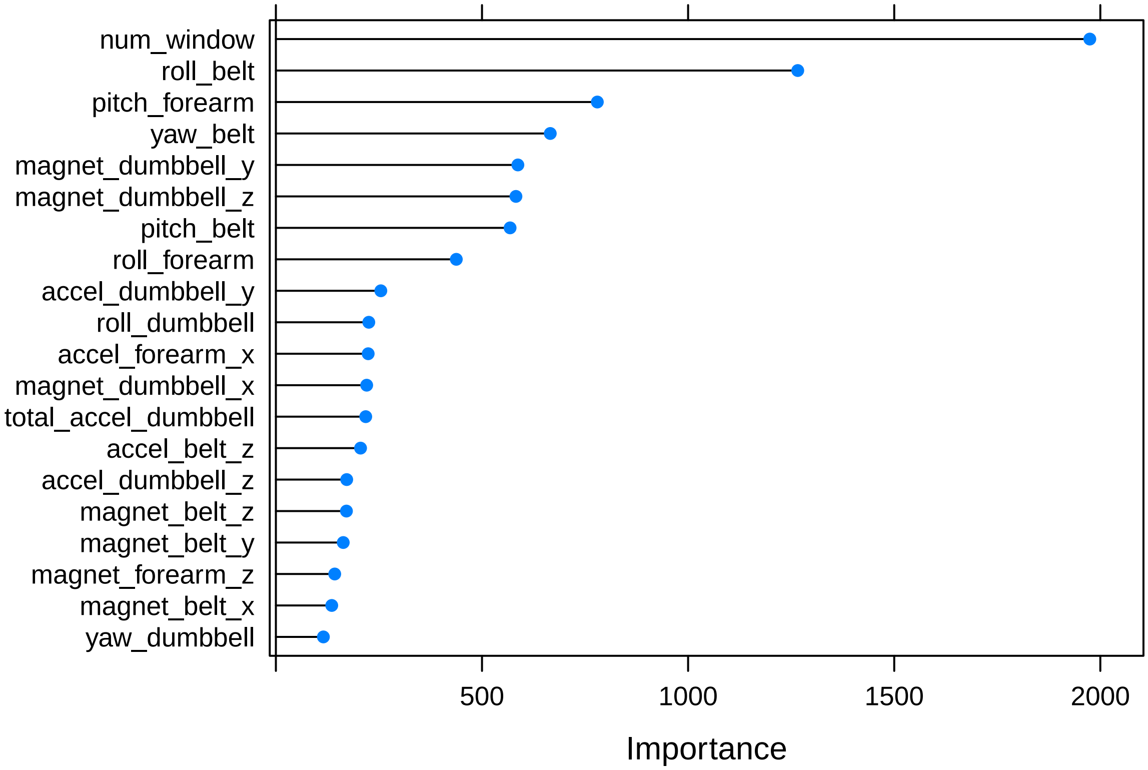
```
modFit <- predict(model,newdata=test)
confusionMatrix(modFit,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1135    3    0    0
```

```
##          C    0    1 1023    1    0
##          D    0    0    0  963    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9971   0.9990   0.9972
## Specificity            0.9993   0.9994   0.9996   0.9994   1.0000
## Pos Pred Value         0.9982   0.9974   0.9980   0.9969   1.0000
## Neg Pred Value         1.0000   0.9992   0.9994   0.9998   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1738   0.1636   0.1833
## Detection Prevalence   0.2850   0.1934   0.1742   0.1641   0.1833
## Balanced Accuracy      0.9996   0.9979   0.9983   0.9992   0.9986
```

The result of the model shows that the accuracy is 99.8 %. The sensitivity and the specificity are higher than 99% for each class. We estimate our out of sample error to be 0.2%. Taking a look at the variables that were most important in prediction (below), we see that *num_window*, *roll_belt*, and *pitch_forearm* were the most important variables for this prediction model.

```
plot(varImp(model, scale=FALSE),top=20)
```



We can now apply this model to the test cases.