Accelerometers to Predict How Well We Work Out

Vasant Soni

October 6, 2016

Executive Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health and to find patterns in their behavior. 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.

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

The approach that we will employ to achieve our goal is the following: 1. Preprocess the initial training and test sets to select relevant variables to be used in the model 2. Split training set into a subset training and test set 3. Build model on subset training set 4. Evaluate the model on the test set 5. Assess model’s strength in predicting the outcome and estimate out of sample error rate

```
library(dplyr)
library(plyr)
library(caret)
library(data.table)
library(xgboost)
library(ggplot2)
library(Matrix)
library(reshape)
library(randomForest)

pml_train = fread('./pml-training.csv') %>% as.data.frame()
pml_test = fread('./pml-testing.csv') %>% as.data.frame()
pml_train$classe = as.factor(pml_train$classe)

pml_classe = pml_train$classe
```

Cleaning the Data

First, we will examine which variables produce a near zero variance and remove these variables since they do not provide any information in explaining the variance inherent in the classe variable. Additionally, we will remove any variable whose class is a character or contains NA values or are time related variables.

```
x = nearZeroVar(pml_train, saveMetrics = TRUE)
pml_trainset = pml_train[,!x$nzv]
pml_testset = pml_test[,!x$nzv]

z = sapply(pml_trainset, is.numeric)

pml_trainset = pml_trainset[,z]
pml_testset = pml_testset[,z]

y =  colSums(is.na(pml_trainset)) == 0

pml_trainset = pml_trainset[,y]
pml_testset = pml_testset[,y]

pml_trainset = pml_trainset[,-c(1,2)]
pml_testset = pml_testset[,-c(1,2)]

dim(pml_trainset)
```

```
## [1] 19622    53
```

Our training set has been reduced from 160 variables to 53 variables.

Preprocess the Dataset and Creating the Model

We will partition the pml_trainingset into a training set and a testing set for establishing the model. 70% of the records will be selected for the training set and 30% of the records will be attributed to the testing set.

```
set.seed(3291)
r = createDataPartition(pml_classe, p=.7, list=FALSE)
pml_trainset$classe = pml_classe

Model_Train = pml_trainset[r,]
Model_Test = pml_trainset[-r,]

dim(Model_Train)
```

```
## [1] 13737    54
```

We will not use Principal Component Analysis to reduce the number of variables and reduce the noise created by high correlated variables. The predictors are not highly correlated and a reduction in the number of predictors will lead to higher in and out of samle error rate.

With a training and testing set established, we will apply the random forests model to train our data. No additional preprocessing is needed to apply the random forest model.

```
modFit = randomForest(classe~., data=Model_Train, ntree= 300 )
modFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Model_Train, ntree = 300)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.31%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    5 2649    4    0    0 0.003386005
## C    0    9 2387    0    0 0.003756260
## D    0    0   19 2232    1 0.008880995
## E    0    0    0    5 2520 0.001980198
```

From the intial look at the model, there is a low in sample error rate and the model on the training data contains strong predictive power. It is still unclear whether the model has been overfitted to the training data.

We will use the model modFit to predict results for the testing set of the partitioned training set and run a confusion matrix to determine whether there is a high out of sample error rate. If the Kappa statistic, which is a value between 0 and 1, is closer to 1 then we can conculde there is a low out of sample error rate.

```
Test_Predictions = predict(modFit, Model_Test)
confusionMatrix(Test_Predictions,Model_Test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1135    5    0    0
##          C    0    2 1021    7    0
##          D    0    0    0  956    2
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9951   0.9917   0.9982
## Specificity            0.9995   0.9989   0.9981   0.9996   0.9998
## Pos Pred Value         0.9988   0.9956   0.9913   0.9979   0.9991
## Neg Pred Value         1.0000   0.9992   0.9990   0.9984   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1735   0.1624   0.1835
## Detection Prevalence   0.2848   0.1937   0.1750   0.1628   0.1837
## Balanced Accuracy      0.9998   0.9977   0.9966   0.9956   0.9990
```

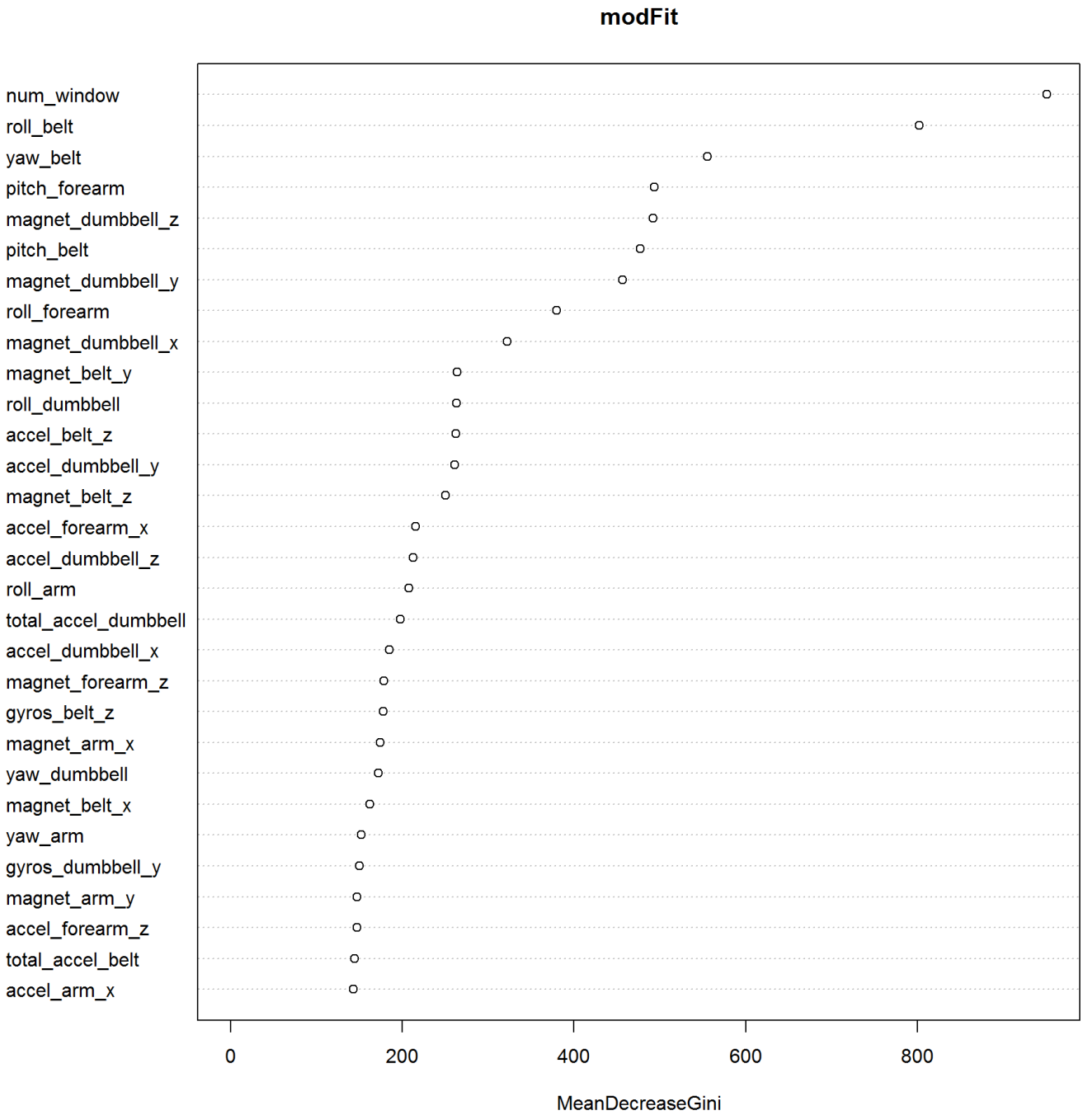
After running the model on the testing set, we get a Kappa statistic of .9961, which implies that there is a low out of sample error rate. Additionally, the sensitivity and specifcity of the predictions suggest that the predictive power of the model is very strong and is sufficiently explaining the variance inherent in the classe variable.

```
Final_Predictions = predict(modFit, pml_testset)
pml_testset$classe_predict = Final_Predictions
write.csv(pml_testset,file = paste0("./test_results_", format(Sys.time(),'%d_%m__%H_%M'), '.csv'),
 row.names =FALSE)
```

Appendix Figure: Variable Important Plot

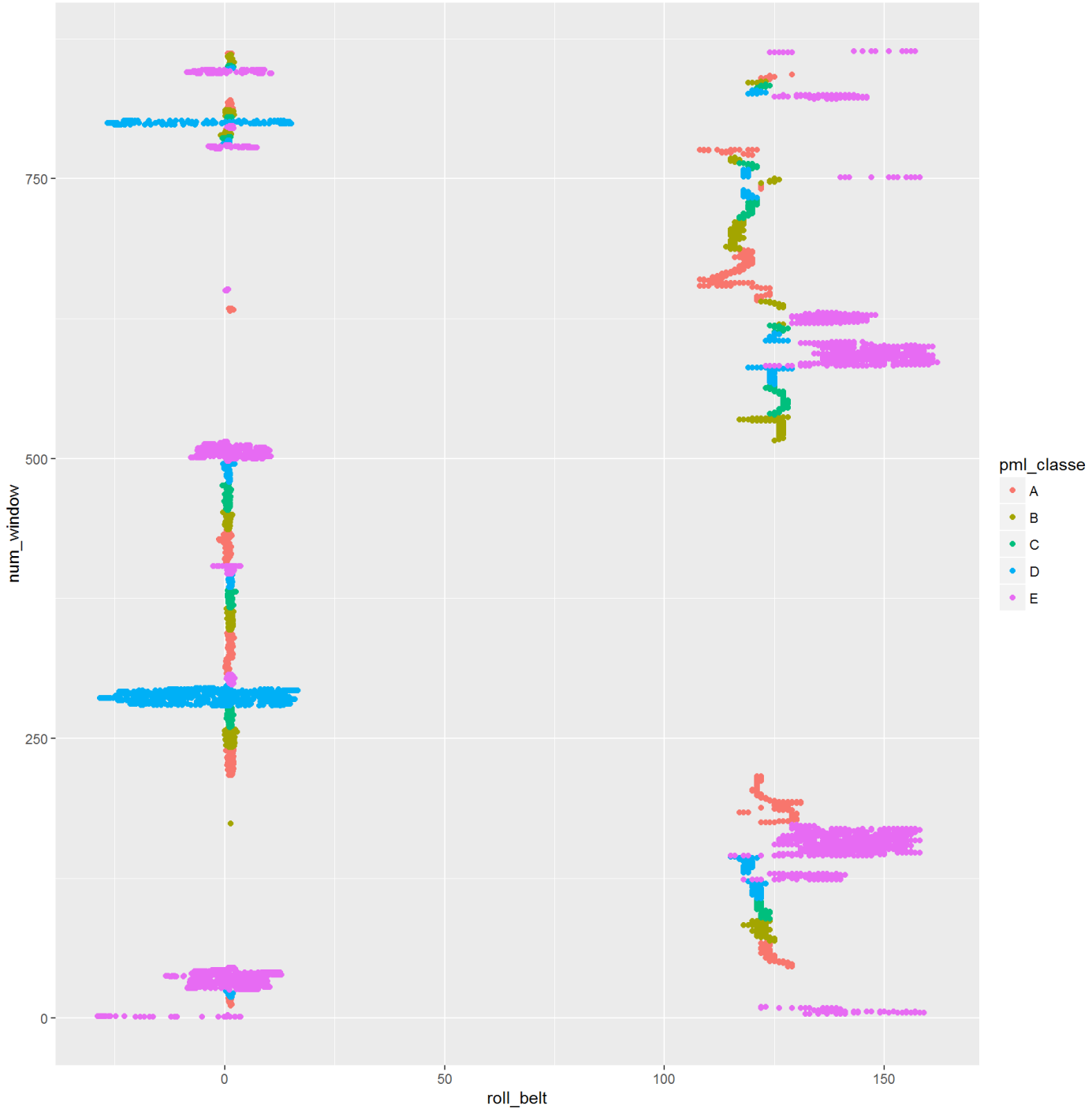
The below plot shows the predictive power of each variable and how the predictive power of each additional variable decreases rapidly.

```
varImpPlot(modFit)
```



Appendix Figure: Scatterplot

```
qplot(roll_belt, num_window, colour = pml_classe, data=pml_trainset)
```


num_window
750
500
250
0
0
50
100
150
roll_belt
pml_classe
A
B
C
D
E