

Practical Machine Learning Project

Dorothea L. Ugi

January 28, 2020

Practical Machine Learning Course Project

Build model to predict how well weight lifting exercises are being done

Background

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, your 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. More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

The data for this project come from this source: <http://groupware.les.inf.puc-rio.br/har>. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading Data

```
trainRaw <-  
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-  
training.csv")  
testRaw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-  
testing.csv")  
dim(trainRaw); dim(testRaw)  
  
## [1] 19622 160  
  
## [1] 20 160
```

Cleaning Data

Remove columns which have a lot of missing data or Near Zero Variance variables

```
NZV <- nearZeroVar(trainRaw, saveMetrics=TRUE)
head(NZV, 20)

##                                     freqRatio percentUnique zeroVar    nzv
## X                           1.000000 100.00000000 FALSE FALSE
## user_name                   1.100679  0.03057792 FALSE FALSE
## raw_timestamp_part_1        1.000000  4.26562022 FALSE FALSE
## raw_timestamp_part_2        1.000000  85.53154622 FALSE FALSE
## cvtd_timestamp              1.000668  0.10192641 FALSE FALSE
## new_window                  47.330049 0.01019264 FALSE TRUE
## num_window                  1.000000  4.37264295 FALSE FALSE
## roll_belt                   1.101904  6.77810621 FALSE FALSE
## pitch_belt                  1.036082  9.37722964 FALSE FALSE
## yaw_belt                    1.058480  9.97349913 FALSE FALSE
## total_accel_belt            1.063160  0.14779329 FALSE FALSE
## kurtosis_roll_belt          1921.600000 2.02323922 FALSE TRUE
## kurtosis_picth_belt         600.500000 1.61553358 FALSE TRUE
## kurtosis_yaw_belt           47.330049 0.01019264 FALSE TRUE
## skewness_roll_belt          2135.111111 2.01304658 FALSE TRUE
## skewness_roll_belt.1         600.500000 1.72255631 FALSE TRUE
## skewness_yaw_belt            47.330049 0.01019264 FALSE TRUE
## max_roll_belt               1.000000  0.99378249 FALSE FALSE
## max_picth_belt              1.538462  0.11211905 FALSE FALSE
## max_yaw_belt                640.533333 0.34654979 FALSE TRUE

training1 <- trainRaw[, !NZV$nzv]
testing1 <- testRaw[, !NZV$nzv]
dim(training1); dim(testing1)

## [1] 19622   100

## [1] 20 100
```

Removing additional columns not needed: X, user_name and timestamp columns which will leave 95 columns

```
removeColumns <- grep("X|timestamp|user_name", names(training1))
training <- training1[, !removeColumns]
testing <- testing1[, !removeColumns]
dim(training); dim(testing)

## [1] 19622    95

## [1] 20 95
```

Finally remove columns that contain NAs

```

goodCols <- (colSums(is.na(training)) == 0)
training <- training[, goodCols]
testing <- testing[, goodCols]
dim(training); dim(testing)

## [1] 19622      54

## [1] 20 54

```

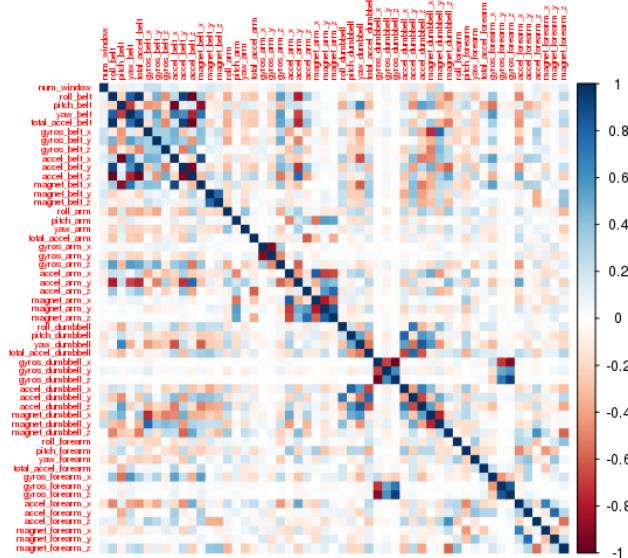
Final cleaned training data contains 19,622 observations and cleaned test data contains 20 observations. Both the training data and testing data have 54 variables (columns).

Correlation Matrix of the Training data.

```

corrplot(cor(training[, -length(names(training))]), method="color",
        tl.cex=.5)

```



Splitting of the Training Data

Need to now split the clean training dataset into a training dataset and a validation dataset. I will be splitting into 70% training and 30% for the validation dataset which we will conduct cross validation later.

```

set.seed(12345)
inTrain <- createDataPartition(training$classe, p=0.70, list=FALSE)
validation <- training[-inTrain, ]
training <- training[inTrain, ]
dim(validation); dim(training)

## [1] 5885      54

## [1] 13737      54

```

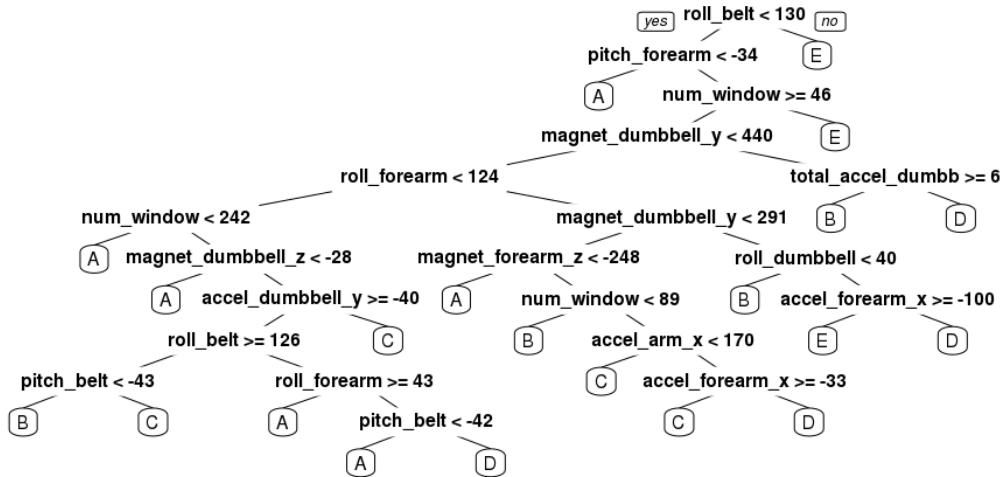
We have 3 datasets now, Training with 13,737 observations, Validation with 5,885 observations and Testing dataset still has 20 observations.

Model Exploration

Decision Tree Model

I will do a predictive model for activity recognition by doing a classification tree

```
modelDecisionTree <- rpart(classe ~ ., data=training, method="class")
prp(modelDecisionTree)
```



Now need to estimate the performance of this model using the validation dataset.

```
predictDecisionTree <- predict(modelDecisionTree, validation, type="class")
confusionMatrix(validation$classe, predictDecisionTree)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   A     B     C     D     E
##           A 1530   35   17   39   53
##           B  269   575   73  146   76
##           C   51   31  743  130   71
##           D   79   25   68  702   90
##           E   16   68   84  128  786
##
## Overall Statistics
##
##                 Accuracy : 0.7368
##                 95% CI : (0.7253, 0.748)
##      No Information Rate : 0.3305
##      P-Value [Acc > NIR] : < 2.2e-16
```

```

##                                     Kappa : 0.6656
##
##   Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity                  0.7866  0.78338  0.7543   0.6131   0.7305
## Specificity                  0.9635  0.89051  0.9422   0.9447   0.9384
## Pos Pred Value                0.9140  0.50483  0.7242   0.7282   0.7264
## Neg Pred Value                0.9014  0.96650  0.9502   0.9100   0.9396
## Prevalence                     0.3305  0.12472  0.1674   0.1946   0.1828
## Detection Rate                 0.2600  0.09771  0.1263   0.1193   0.1336
## Detection Prevalence          0.2845  0.19354  0.1743   0.1638   0.1839
## Balanced Accuracy              0.8750  0.83694  0.8483   0.7789   0.8345

accuracy <- postResample(predictDecisionTree, validation$classe)
est_OSE <- 1 - as.numeric(confusionMatrix(validation$classe,
predictDecisionTree)$overall[1])
accuracy

## Accuracy      Kappa
## 0.7367884 0.6656280

est_OSE

## [1] 0.2632116

```

The estimated accuracy is 73.67884% and the estimated out-of sample error is 26.32116%.

Random Forest Model

I will fit a predictive model for activity recognition using Random Forest whihc it automatically selects the important variables. I will use 5-fold cross validation.

```

modelRandomForest <- train(classe ~ ., data=training, method="rf",
trControl=trainControl(method="cv", 5), ntree = 250)
modelRandomForest

## Random Forest
##
## 13737 samples
##      53 predictor
##      5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10990, 10990
## Resampling results across tuning parameters:
##

```

```

##   mtry Accuracy Kappa
##    2    0.9940306 0.9924488
##   27    0.9967969 0.9959483
##   53    0.9949042 0.9935540
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.

```

We need to now see what the performance is on the validation dataset.

```

predictRandomForest <- predict(modelRandomForest, validation)
confusionMatrix(validation$classe, predictRandomForest)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   A     B     C     D     E
##           A 1674    0     0     0     0
##           B    5 1132    2     0     0
##           C    0    4 1022    0     0
##           D    0    0    9 955    0
##           E    0    0    0    3 1079
##
## Overall Statistics
##
##                 Accuracy : 0.9961
##                           95% CI : (0.9941, 0.9975)
##   No Information Rate : 0.2853
##   P-Value [Acc > NIR] : < 2.2e-16
##
##                 Kappa : 0.9951
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##                                Class: A Class: B Class: C Class: D Class: E
## Sensitivity                  0.9970  0.9965  0.9894  0.9969  1.0000
## Specificity                   1.0000  0.9985  0.9992  0.9982  0.9994
## Pos Pred Value                 1.0000  0.9939  0.9961  0.9907  0.9972
## Neg Pred Value                  0.9988  0.9992  0.9977  0.9994  1.0000
## Prevalence                      0.2853  0.1930  0.1755  0.1628  0.1833
## Detection Rate                  0.2845  0.1924  0.1737  0.1623  0.1833
## Detection Prevalence              0.2845  0.1935  0.1743  0.1638  0.1839
## Balanced Accuracy                  0.9985  0.9975  0.9943  0.9975  0.9997

accuracy <- postResample(predictRandomForest, validation$classe)
est_OSE <- 1 - as.numeric(confusionMatrix(validation$classe,
predictRandomForest)$overall[1])
accuracy

```

```
## Accuracy      Kappa
## 0.9960918 0.9950560

est_OSE

## [1] 0.003908241
```

The estimated accuracy is 99.62617% and the estimated out-of sample error is 0.003738318%.

The Random Forest Model worked better than the Decision Tree Model!

Results - Prediction on Testing Set

Applying the Random Forest to predict the outcome variable classe for the test set.

```
predict(modelRandomForest, testing[, -length(names(testing))])

## [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
```