Prediction Assignment Write up for Practical Machine Learning

Steven's write up for the Johns Hopkins Coursera Data Science specialization

January 23, 2016

Summary

The goal of this project is to use machine learning to predict the qualitative activity recognition of weight lifting exercises (the classes) from research activity recognition data hosted at http://groupware.les.inf.puc-rio.br/har. 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. The data was collected from six participants, where class A represents the correct execution of the exercise, and the remaining five classes (B-E) are considered to have mistakes in the execution of the exercise.

The analysis uses two different learners, one fast (**Random Forrest**), and one slow (**Extream Graident Boosting**) with the intent to combine predictors if they both do not perform optimally.

Getting and Cleaning the Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

It is important to examine the data fully for missing values. It became apparent that there was a second code used for missing values (*#DIV/0!*). Now, that we know all of the codes used for missing values, the data is read into a data frame:

```
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv","training.csv")
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","testing.csv")
training <- read.csv("training.csv", na.strings = c("NA", "#DIV/0!"));testing <- read.csv("testing.csv"
```

Using the paper by Velloso, E., et. al. as a reference, there are only a handful of variables that are useful for the prediction. The variables of interest have **belt**, **arm**, **dumbbell**, and **forearm** in their names. Therefore, reducing the variables for us.

```
dim(training)
```

```
## [1] 19622   160
```

```
sensorColumns = grep(pattern = "_belt|_arm|_dumbbell|_forearm", names(training))
training = training[, c(sensorColumns,160)]
dim(training)
```

```
## [1] 19622   153
```

After the four types of variables are selected, the next course of action is to deal with the missing values in the data frame. Unfortunately, the vast amount of missing values does not allow us to impute the data. Consequently, columns with missing values were removed as part of the data cleaning process. Next, variable selection was applied to the testing set to avoid errors.

```
missingData = is.na(training)
omitColumns = which(colSums(missingData) > 19000)
training = training[, -omitColumns]
dim(training)
```

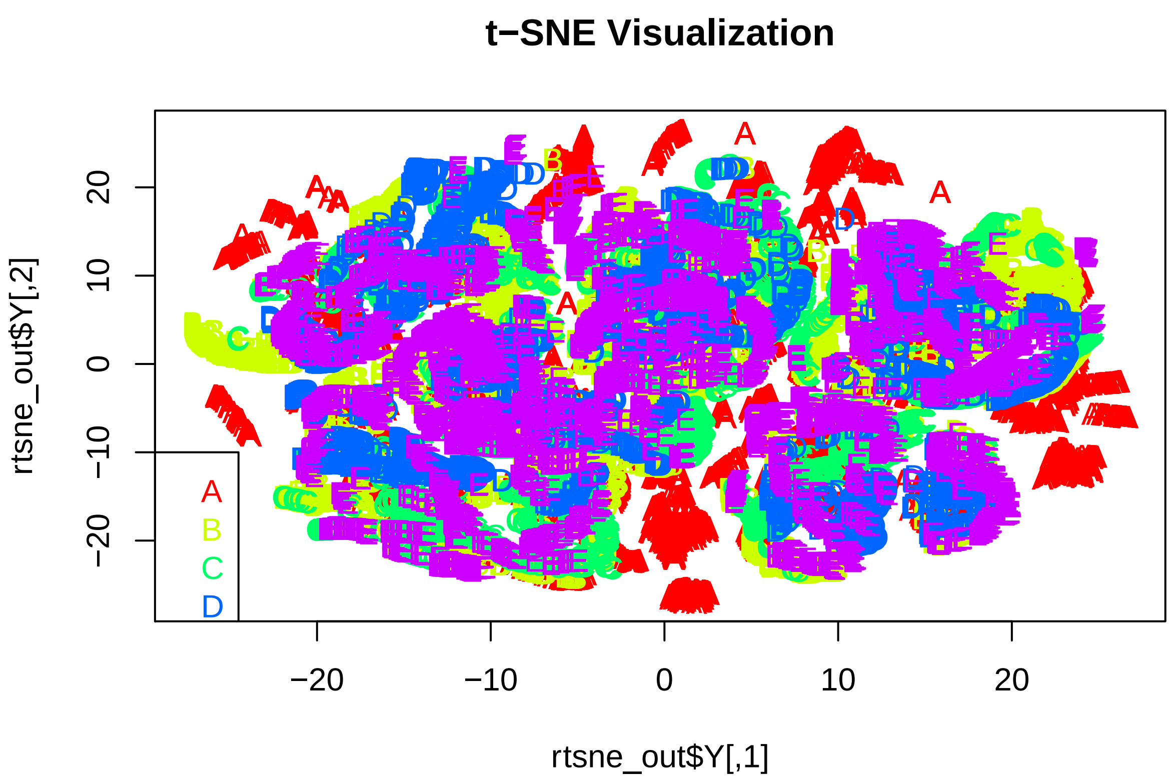
```
## [1] 19622    53
```

```
testing <- testing[colnames(testing) %in% colnames(training)]
```

Exploring the Data

With a high dimension data set, t-Distributed Stochastic Neighbor Embedding (**t-SNE**) was chosen to see and identify any patterns in the data. High dimensional visualization methods provide insight as to which machine learning techniques to utilize.

```
suppressMessages(library(Rtsne))
set.seed(323)
rtsne_out <- Rtsne(data.matrix(training), max_iter=500)
colors = rainbow(length(unique(training$classe)))
names(colors) = unique(training$classe)
plot(rtsne_out$Y, t='n', main="t-SNE Visualization")
text(rtsne_out$Y, labels=training$classe, col=colors[training$classe])
legend(-30,-10,unique(training$classe),text.col=colors )
```



This visualization technique preserves the high dimension structure, and projects that structure into two or three-dimensional space. The figure does not distinguish nice clusters, therefore, a cluster approach will not be applied to this data set.In addition, a decision tree would yield poor results on this data set. More on **t-SNE** can be found in this paper http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

Machine Learning

Random Forrest

No preprocessing, feature engineering, parameter tuning, or dimension reduction was needed, outside of removing missing values, for Random Forrest. The caret package was utilized for it's robust train function to fit predictors to the training set. The training data was not partitioned, because repeated cross-validation will be used to train the model and estimate the out of sample error by holding out a portion of the data, and using it to estimate the error.

```
suppressMessages(library(caret))
suppressMessages(library(doMC))
registerDoMC(cores=7)
set.seed(323)
cvCtrl <- trainControl(method = "repeatedcv", classProbs = TRUE, summaryFunction = multiClassSummary, re
rf_fit <- train(classe~.,method="rf",data=training, trainControl=cvCtrl, metric = "Accuracy")
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
predictions <- predict(rf_fit,newdata = training)
print(confusionMatrix(predictions,training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Repeated cross validation, 10 fold, was done to validate the model for assessing how the results of the analysis will generalize to our testing set. Three repeats were selected to reduce variability, and the miss-classification error rate was low. therefore, based on the confusion matrix, it is expected that the out of sample error rate will be less than 1%, closer to 0.

XG Boost

In addition, no preprocessing, feature engineering, parameter tuning, or dimension reduction was needed, outside of removing missing values, for the boosting method. Again, the caret package was utilized for it's robust train function to fit predictors to the training set.

```
set.seed(323)
xg_fit <- train(classe~.,method="xgbTree",data=training, trainControl=cvCtrl, metric = "Accuracy")
```

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

```
predictions <- predict(xg_fit,newdata = training)
print(confusionMatrix(predictions,training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

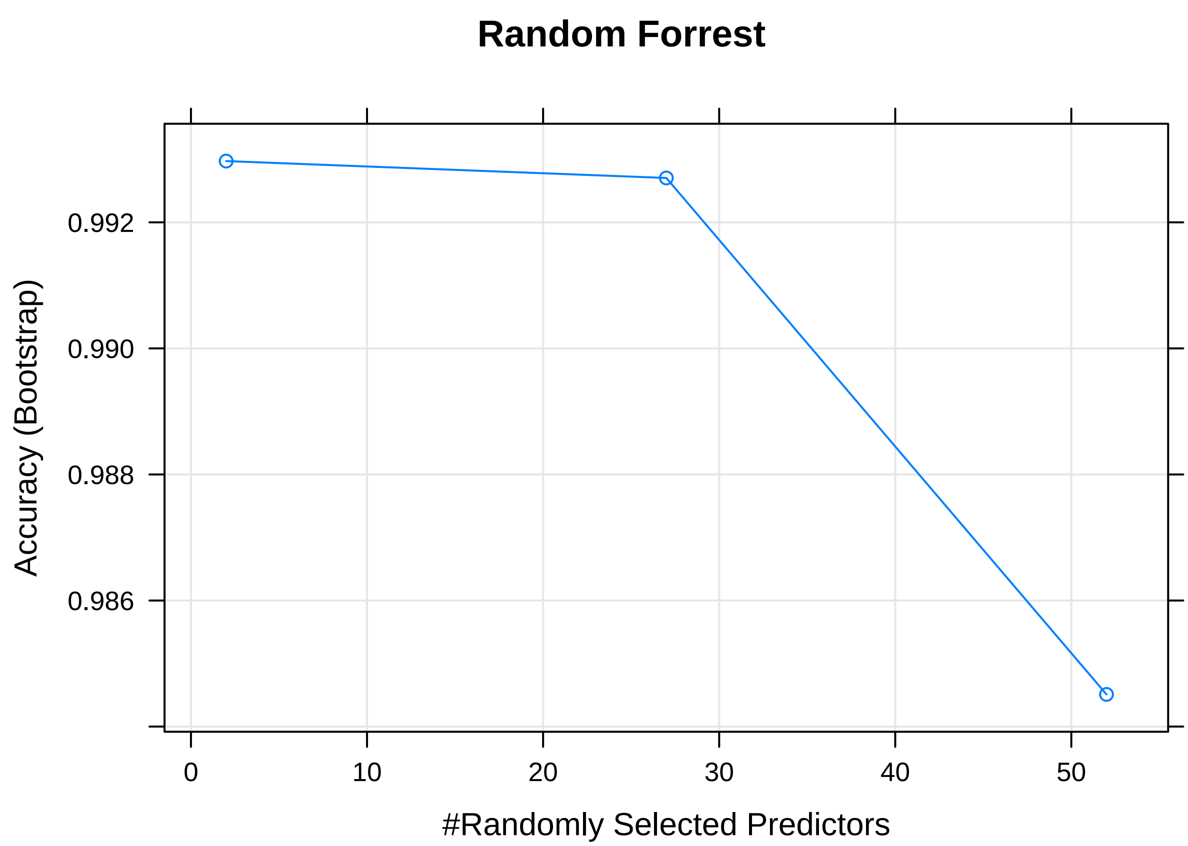
```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Additionally, the confusion matrix for the boosted model also suggest an out of sample error rate less than 1%. Based on the metrics for both models, it is not necessary to combine predictors to increase the accuracy.

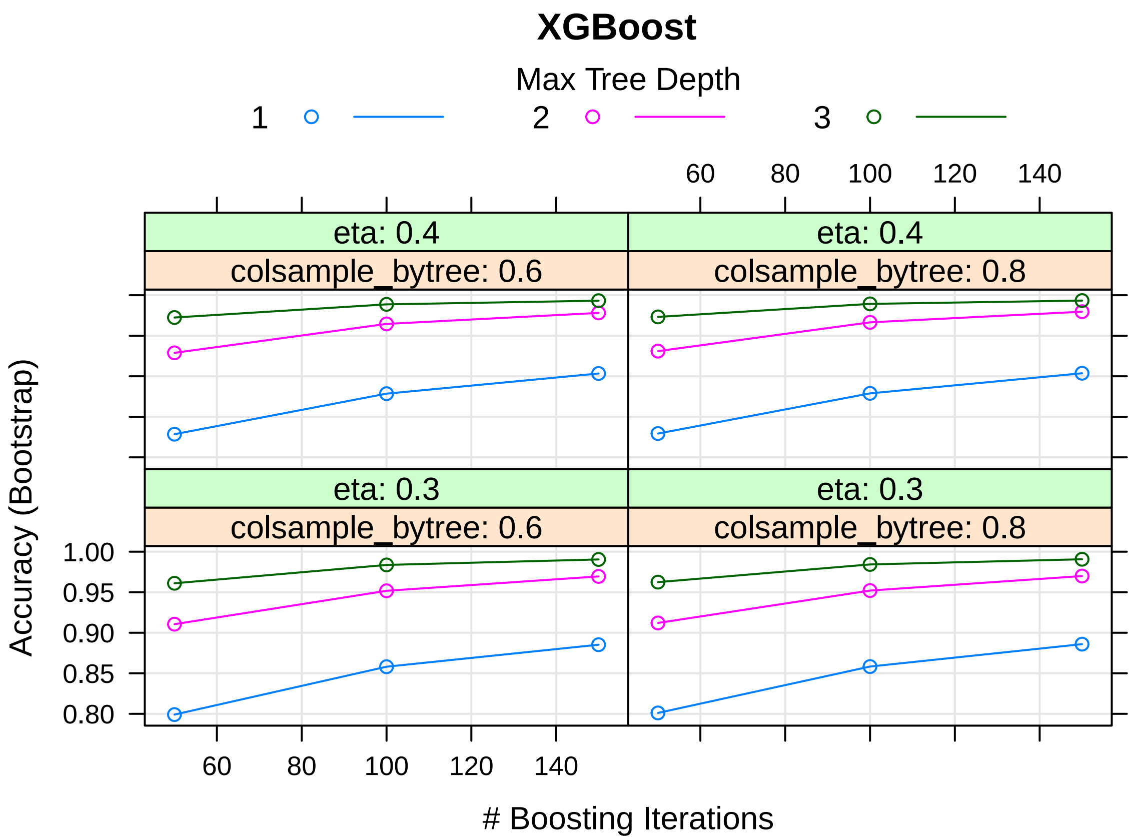
Investigation and Applying the Models to the Testing Set

Here we briefly discuss the model selection process and its performance.

```r
plot(rf_fit, main = "Random Forrest")
```




```r
plot(xg_fit, main="XGBoost")
```



For Random Forrest, the models performance decreased with increasing predictors. While, boosting, a max tree depth of 3 is optimal for achieving the desired performance.

Furthermore, the function varImp can be used to characterize the general effect of the predictors on the models. These features have large contributions to the models performance:

```
caret::varImp(xg_fit)
```

```
## xgbTree variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                   Overall
## yaw_belt           100.00
## roll_belt           92.58
## pitch_forearm       67.72
## magnet_dumbbell_y   67.71
## roll_forearm        62.01
## magnet_dumbbell_z   50.39
## accel_belt_z        46.93
## pitch_belt          40.11
## magnet_belt_z       39.44
## gyros_belt_z        31.48
## magnet_forearm_z    28.27
## accel_dumbbell_y    27.68
## yaw_arm             25.88
## magnet_belt_y       23.28
```

```
## accel_forearm_z     18.13
## accel_forearm_x     16.26
## roll_arm            15.13
## accel_dumbbell_z    14.58
## accel_dumbbell_x    14.16
## magnet_dumbbell_x   13.67
```

```
caret::varImp(rf_fit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               76.48
## magnet_dumbbell_z      65.57
## magnet_dumbbell_y      61.23
## pitch_forearm          60.02
## pitch_belt             59.92
## magnet_dumbbell_x      51.51
## roll_forearm           50.71
## accel_belt_z           45.33
## accel_dumbbell_y       43.69
## roll_dumbbell          43.02
## magnet_belt_z          42.42
## magnet_belt_y          38.32
## accel_dumbbell_z       37.97
## roll_arm               34.36
## accel_forearm_x        32.49
## gyros_belt_z           29.98
## total_accel_dumbbell   28.02
## accel_dumbbell_x       27.53
## gyros_dumbbell_y       27.05
```

```
predictions_tree <- predict(rf_fit,newdata = testing)
predictions_xg <- predict(xg_fit,newdata = testing)
predictions_tree
```

```
##  [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
```

```
predictions_xg
```

```
##  [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
```

Conclusion

Both models were tested against the testing set, and the results were submitted to validate the models out of sample predictions capability. The Kappa statistic is a metric that compares an observed accuracy with an expected accuracy by taking into account random chance. The Kappa value of 1 lets us know that both models performed perfectly on the training sets. Because multiple cross validation was selected, the out of error rate is expected to be 0%< x<1% as the accuracy of the models are 100%.

Reference:

Qualitative Activity Recognition of Weight Lifting Exercises

- Author: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H.
- Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13)

http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz3y6WoPH9r