Practical Machine Learning - Course Project

Paul Y. Ke

2/5/2018

Overview

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Files, Exploratory Analysis, and Cleanup

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

Looking at the website documentation and the raw data, it is clear that there are many NA/blank values that we'll clean on import.

The training dataset contains 19,622 observations across 160 variables and testing has 20 observations. We will need to further clean the data by removing factors within the data sets that have missing values as well as time-series values

```
## [1] 11776    53
```

```
## [1] 7846   53
```

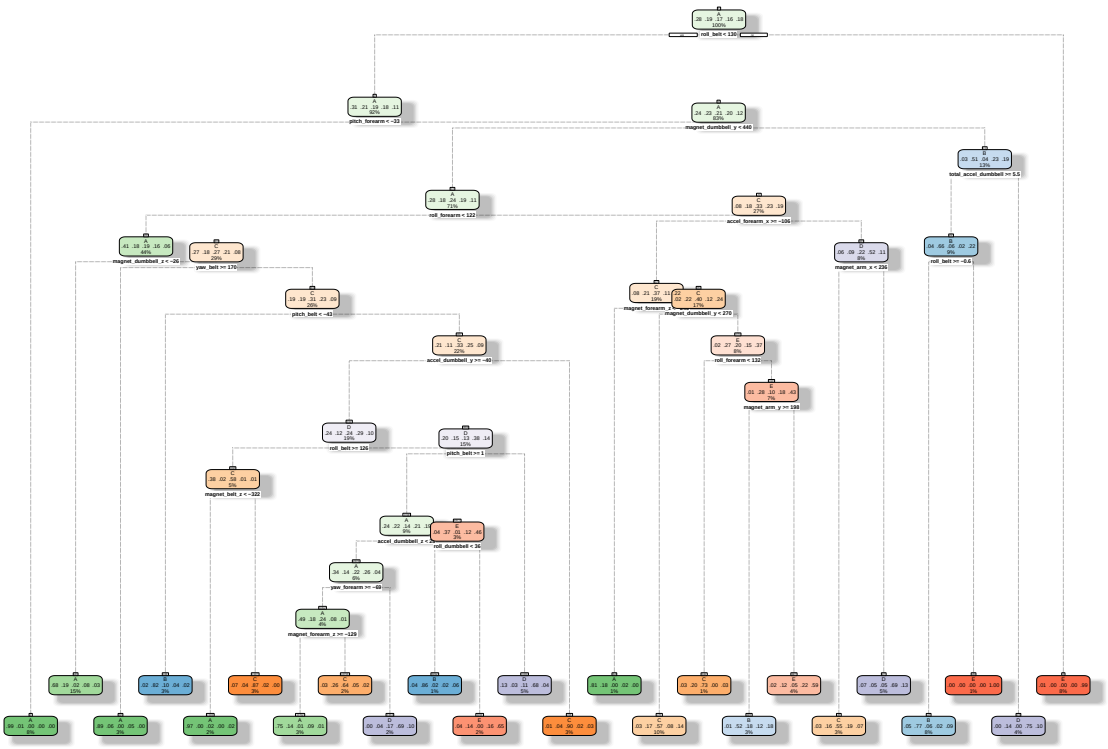
Analysis and model training

Decision Tree Model

Following examples from our quizzes, I thought the final 20 question quiz would provide us specific values for parameters and we were supposed to determine which activity (A, B, C, D, or E) correlated based on a decision tree. It wasn't until later that I realized that the testing data provided the 20 samples that the quiz used. However, it was interesting to see how the factors impacted the sepecific activity.

```
model_rpart <- rpart(classe ~ ., data=training, method = "class", control=rpart.control(method="cv", nu
fancyRpartPlot(model_rpart)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2018−Feb−08 16:57:17 paulke

```
predict_rpart <- predict(model_rpart, sample, type="class")
confusionMatrix(predict_rpart, sample$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2012  347   29  139   44
##          B   42  772  105   53  102
##          C   57  227 1119  124  113
##          D  103   97   91  871  105
##          E   18   75   24   99 1078
##
## Overall Statistics
##
##                Accuracy : 0.7459
##                  95% CI : (0.7361, 0.7555)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6771
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9014  0.50856   0.8180   0.6773   0.7476
## Specificity            0.9004  0.95228   0.9196   0.9396   0.9663
```

```
## Pos Pred Value         0.7826  0.71881   0.6823   0.6875   0.8331
## Neg Pred Value         0.9583  0.88984   0.9599   0.9369   0.9444
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2564  0.09839   0.1426   0.1110   0.1374
## Detection Prevalence   0.3277  0.13689   0.2090   0.1615   0.1649
## Balanced Accuracy      0.9009  0.73042   0.8688   0.8085   0.8569
```

With a 75% accuracy using Decision Tree and RPart, my thought was to investiage with Random Forest and Boosting models to compare and possibly combine them to produce a prediction.
Boosting

```
model_gbm <- train(classe ~ ., method="gbm", data=training, verbose=FALSE, trControl=trainControl(metho
model_gbm
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10599, 10598, 10599, 10598, 10599, 10597, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7508510  0.6840872
##   1                  100      0.8199718  0.7720562
##   1                  150      0.8525801  0.8133848
##   2                   50      0.8519866  0.8124382
##   2                  100      0.9087108  0.8844856
##   2                  150      0.9307037  0.9123185
##   3                   50      0.8960581  0.8684001
##   3                  100      0.9430169  0.9279028
##   3                  150      0.9611900  0.9509015
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
predict_gbm <- predict(model_gbm, sample)
confusionMatrix(predict_gbm, sample$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2207   50    0    1    1
##          B   17 1423   43    5   13
##          C    5   43 1308   49   11
##          D    1    0   14 1212   19
##          E    2    2    3   19 1398
##
```

```
## Overall Statistics
##
##                Accuracy : 0.962
##                  95% CI : (0.9576, 0.9661)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9519
##  Mcnemar's Test P-Value : 6.465e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9888   0.9374   0.9561   0.9425   0.9695
## Specificity            0.9907   0.9877   0.9833   0.9948   0.9959
## Pos Pred Value         0.9770   0.9480   0.9237   0.9727   0.9817
## Neg Pred Value         0.9955   0.9850   0.9907   0.9888   0.9931
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2813   0.1814   0.1667   0.1545   0.1782
## Detection Prevalence   0.2879   0.1913   0.1805   0.1588   0.1815
## Balanced Accuracy      0.9898   0.9625   0.9697   0.9686   0.9827
```

We also see Accuracy increase as the number of trees increase (50, 100, 150) but at best a 96.11% model accuracy.

Random Forest

In the website / documentation, the researches mentioned use of Random Forest so I also modeled with "rf" and the training dataset, the caret package defaults the number of trees to 500 and runs 3 different sizes of variable sampling which causes the program to run for a very long time (at least over an hour before I gave up). The caret package function wraps the randomForest function and allows for tuning of 2 parameters: mtry, The number of variables randomly sampled as candidates at each split and ntree, the number of trees to grow. Even running across a portion of the dataset, the training time was reduced at the sake of a lower ntree value and at the expense of accuracy.

```
#model_rf <- train(classe ~ ., data=training, ntree=50, method="rf")
```

caret is just a convenience wrapper for the "randomForest" method which I called directly and found it to run much faster by adding in the option to run the calculation in parallel, this will allow me to establish greater accuracy.

```
model_rf <- randomForest(classe ~., data=training, method="rf", importance=TRUE, trControl=trainControl
model_rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, method = "rf",      importance = TRUE, trContro
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3345    2    0    0    1 0.0008960573
```

```
## B   18 2255    6    0    0 0.0105309346
## C    0   20 2033    1    0 0.0102239533
## D    0    0   25 1905    0 0.0129533679
## E    0    1    3    5 2156 0.0041570439
```

Prediction using Random Forest model

```
predict_rf<- predict(model_rf, sample, type="class")
confusionMatrix(predict_rf, sample$classe )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    6    0    0    0
##          B    3 1504    9    0    0
##          C    0    8 1356   25    0
##          D    0    0    3 1261    2
##          E    1    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9906, 0.9945)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9908   0.9912   0.9806   0.9986
## Specificity            0.9989   0.9981   0.9949   0.9992   0.9998
## Pos Pred Value         0.9973   0.9921   0.9762   0.9961   0.9993
## Neg Pred Value         0.9993   0.9978   0.9981   0.9962   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1917   0.1728   0.1607   0.1835
## Detection Prevalence   0.2847   0.1932   0.1770   0.1614   0.1837
## Balanced Accuracy      0.9986   0.9944   0.9931   0.9899   0.9992
```

A 99.27% accuracy model, Random Forest seems the best model to use to predict our testing data records for the 20 question Quiz.

Testing

```
predict_quiz <- predict(model_rf, data_testing)
predict_quiz
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

The prediction results were entered into the quiz and received 100% (20/20) match.