Machine Learning Coursera Project

Ed Song

March 5, 2016

This report will cover how my machine learning model built, trained, and tested. We will be looking at the provided data sets and will be predicting for the "classe" variable.

Packages that need loaded

THe caret and AppliedPredictive Modeling packages are loaded in order us the functions need for machine learning model construction.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(AppliedPredictiveModeling)
```

Load training and test data

Grabbing the data from the assignments source.

```
URL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
download.file(url = URL, destfile = '~/training.csv'
              , method = 'auto')

URL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
download.file(url = URL
              , destfile = '~/test.csv'
              , method = 'auto')
```

Data Exploration

Placing the data into the training and test as well as storing the columns names for limiting data in the future.

```
training <- read.csv(file = 'training.csv', header = TRUE, sep = ",", na.strings = c('NA', ''))
test     <- read.csv(file = 'test.csv', header = TRUE, sep = ",", na.strings = c('NA', ''))

colnames_train<-colnames(training)
colnames_test<-colnames(test)
```

Cleaning traning data

Below we created a function to identify all columns without "NA" present. Then we looped through each column and if the row count of nonNAs was less then the total rows then we will disregard that column. We also dropped the first 8 columns because they didn't add to the model.

```
nonNAs <- function(x) {
  as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))}

#Build vector of missing data  or NA columns to drop
colcnt <- nonNAs(training)
drops  <- c()
for (cnt in 1:length(colcnt)){
  if(colcnt[cnt] < nrow(training)){
    drops <- c(drops, colnames_train[cnt])
    }
  }

#Drop NA data and the first 7 columns as they are unneccessary for prediciting
training <- training[,!(names(training) %in% drops)]
training <- training[,8:length(colnames(training))]

test <- test[,!(names(test) %in% drops)]
test <- test[,8:length(colnames(test))]

nsv <- nearZeroVar(training, saveMetrics = TRUE)
nsv
```

```
##                     freqRatio percentUnique zeroVar   nzv
## roll_belt            1.101904     6.7781062   FALSE FALSE
## pitch_belt           1.036082     9.3772296   FALSE FALSE
## yaw_belt             1.058480     9.9734991   FALSE FALSE
## total_accel_belt     1.063160     0.1477933   FALSE FALSE
## gyros_belt_x         1.058651     0.7134849   FALSE FALSE
## gyros_belt_y         1.144000     0.3516461   FALSE FALSE
## gyros_belt_z         1.066214     0.8612782   FALSE FALSE
## accel_belt_x         1.055412     0.8357966   FALSE FALSE
## accel_belt_y         1.113725     0.7287738   FALSE FALSE
## accel_belt_z         1.078767     1.5237998   FALSE FALSE
## magnet_belt_x        1.090141     1.6664968   FALSE FALSE
## magnet_belt_y        1.099688     1.5187035   FALSE FALSE
## magnet_belt_z        1.006369     2.3290184   FALSE FALSE
## roll_arm            52.338462    13.5256345   FALSE FALSE
## pitch_arm           87.256410    15.7323412   FALSE FALSE
## yaw_arm             33.029126    14.6570176   FALSE FALSE
## total_accel_arm      1.024526     0.3363572   FALSE FALSE
## gyros_arm_x          1.015504     3.2769341   FALSE FALSE
## gyros_arm_y          1.454369     1.9162165   FALSE FALSE
## gyros_arm_z          1.110687     1.2638875   FALSE FALSE
## accel_arm_x          1.017341     3.9598410   FALSE FALSE
## accel_arm_y          1.140187     2.7367241   FALSE FALSE
## accel_arm_z          1.128000     4.0362858   FALSE FALSE
## magnet_arm_x         1.000000     6.8239731   FALSE FALSE
## magnet_arm_y         1.056818     4.4439914   FALSE FALSE
```

```
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

```
dim(training)
```

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

Seeing that our columns are all sufficently unique we will proceed to build the model.

Break up training data to have a smaller subset to build and test models on

```
inTrain <- createDataPartition(y=training$classe, p=.75, list = FALSE)
big_train <- training[inTrain,]
big_test <- training[-inTrain,]
```

Building the Model

We built two models after setting the seed (for reproducability). The first was regression partitioning and the second was random forest.

```
set.seed(32556)

#recursive partitioning
rpart_modelFit <- train(classe~., data = big_train, method = 'rpart')
```

```
## Loading required package: rpart
```

```
rpart_modelFit
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.03503275  0.5134299  0.37049240  0.04245618   0.07174235
##   0.04183044  0.5013266  0.35439991  0.04604966   0.07565161
##   0.11649103  0.3410124  0.08505413  0.03954407   0.05963896
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03503275.
```

```
#random forest model
rf_modelFit <- train(classe~., data = big_train, method = 'rf'
                     , preProcess = c('center', 'scale')
                     , trControl = trainControl(method = 'cv',
                                                number = 4))
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf_modelFit
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
```

```
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 11038, 11038, 11040, 11038
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9910313  0.9886541  0.0012948566  0.001637968
##   27    0.9915069  0.9892562  0.0004095079  0.000517271
##   52    0.9881776  0.9850445  0.0010074576  0.001273636
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

From these models it looks like the random forest will be significantly more accurate. Below we will test the models to verify that it is.

Model testing

```
prediction <- predict(rpart_modelFit, newdata = big_test)
print(confusionMatrix(prediction, big_test$classe), digits = 4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 847 153  24  45  12
##          B 166 549  40 103 220
##          C 286 206 671 434 224
##          D  94  41 120 222  53
##          E   2   0   0   0 392
##
## Overall Statistics
##
##                Accuracy : 0.5467
##                  95% CI : (0.5326, 0.5607)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4316
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6072   0.5785   0.7848  0.27612  0.43507
## Specificity            0.9333   0.8662   0.7160  0.92488  0.99950
## Pos Pred Value         0.7835   0.5093   0.3685  0.41887  0.99492
## Neg Pred Value         0.8567   0.8955   0.9403  0.86694  0.88714
## Prevalence             0.2845   0.1935   0.1743  0.16395  0.18373
## Detection Rate         0.1727   0.1119   0.1368  0.04527  0.07993
## Detection Prevalence   0.2204   0.2198   0.3713  0.10808  0.08034
## Balanced Accuracy      0.7702   0.7224   0.7504  0.60050  0.71729
```

```r
prediction <- predict(rf_modelFit, newdata = big_test)
print(confusionMatrix(prediction, big_test$classe), digits = 4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
##          B    0  940    4    0    0
##          C    0    0  849    6    4
##          D    0    0    2  798    4
##          E    0    0    0    0  893
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9915, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9905   0.9930   0.9925   0.9911
## Specificity            0.9974   0.9990   0.9975   0.9985   1.0000
## Pos Pred Value         0.9936   0.9958   0.9884   0.9925   1.0000
## Neg Pred Value         1.0000   0.9977   0.9985   0.9985   0.9980
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1917   0.1731   0.1627   0.1821
## Detection Prevalence   0.2863   0.1925   0.1752   0.1639   0.1821
## Balanced Accuracy      0.9987   0.9948   0.9953   0.9955   0.9956
```

Thus we can say that our regression partitioning had an accuracy of .4925 while the random forest was .9947 accurate.