Practical Machine Learning Project

Micheal Copeland

November 18, 2015

This is the course project for Practical Machine Learning from Coursera.

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.1.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.1.3
```

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

```
## Warning: package 'rpart' was built under R version 3.1.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

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

```
# Download data.
trainUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Import the data treating empty values as NA.
df_training <- read.csv(url(trainUrl), na.strings=c("NA", "#DIV/0!",""))
colnames_train <- colnames(df_training)
df_testing <- read.csv(url(testUrl), na.strings=c("NA", "#DIV/0!",""))
colnames_test <- colnames(df_testing)

# Verify that the column names (excluding classe and problem_id) are identical in the train
ing and test set.
all.equal(colnames_train[1:length(colnames_train)-1], colnames_test[1:length(colnames_train
)-1])
```

```
## [1] TRUE
```

```
# Count the number of non-NAs in each col.
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.
colcnts <- nonNAs(df_training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(df_training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}

# Drop NA data and the first 7 columns as they're unnecessary for predicting.
df_training <- df_training[,!(names(df_training) %in% drops)]
df_training <- df_training[,8:length(colnames(df_training))]

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

# Show remaining columns.
colnames(df_training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
colnames(df_testing)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "problem_id"
```

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

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt                1.102       6.77811   FALSE FALSE
## pitch_belt               1.036       9.37723   FALSE FALSE
## yaw_belt                 1.058       9.97350   FALSE FALSE
## total_accel_belt         1.063       0.14779   FALSE FALSE
## gyros_belt_x             1.059       0.71348   FALSE FALSE
## gyros_belt_y             1.144       0.35165   FALSE FALSE
## gyros_belt_z             1.066       0.86128   FALSE FALSE
## accel_belt_x             1.055       0.83580   FALSE FALSE
## accel_belt_y             1.114       0.72877   FALSE FALSE
## accel_belt_z             1.079       1.52380   FALSE FALSE
## magnet_belt_x            1.090       1.66650   FALSE FALSE
## magnet_belt_y            1.100       1.51870   FALSE FALSE
## magnet_belt_z            1.006       2.32902   FALSE FALSE
## roll_arm                52.338      13.52563   FALSE FALSE
## pitch_arm               87.256      15.73234   FALSE FALSE
## yaw_arm                 33.029      14.65702   FALSE FALSE
## total_accel_arm          1.025       0.33636   FALSE FALSE
## gyros_arm_x              1.016       3.27693   FALSE FALSE
## gyros_arm_y              1.454       1.91622   FALSE FALSE
## gyros_arm_z              1.111       1.26389   FALSE FALSE
## accel_arm_x              1.017       3.95984   FALSE FALSE
## accel_arm_y              1.140       2.73672   FALSE FALSE
## accel_arm_z              1.128       4.03629   FALSE FALSE
## magnet_arm_x             1.000       6.82397   FALSE FALSE
## magnet_arm_y             1.057       4.44399   FALSE FALSE
## magnet_arm_z             1.036       6.44685   FALSE FALSE
## roll_dumbbell            1.022      83.78351   FALSE FALSE
## pitch_dumbbell           2.277      81.22516   FALSE FALSE
## yaw_dumbbell             1.132      83.14137   FALSE FALSE
## total_accel_dumbbell     1.073       0.21914   FALSE FALSE
## gyros_dumbbell_x         1.003       1.22821   FALSE FALSE
## gyros_dumbbell_y         1.265       1.41678   FALSE FALSE
## gyros_dumbbell_z         1.060       1.04984   FALSE FALSE
## accel_dumbbell_x         1.018       2.16594   FALSE FALSE
## accel_dumbbell_y         1.053       2.37489   FALSE FALSE
## accel_dumbbell_z         1.133       2.08949   FALSE FALSE
## magnet_dumbbell_x        1.098       5.74865   FALSE FALSE
## magnet_dumbbell_y        1.198       4.30129   FALSE FALSE
## magnet_dumbbell_z        1.021       3.44511   FALSE FALSE
## roll_forearm            11.589      11.08959   FALSE FALSE
## pitch_forearm           65.983      14.85577   FALSE FALSE
## yaw_forearm             15.323      10.14677   FALSE FALSE
## total_accel_forearm      1.129       0.35674   FALSE FALSE
## gyros_forearm_x          1.059       1.51870   FALSE FALSE
```

```
## gyros_forearm_y          1.037      3.77637   FALSE FALSE
## gyros_forearm_z          1.123      1.56457   FALSE FALSE
## accel_forearm_x          1.126      4.04648   FALSE FALSE
## accel_forearm_y          1.059      5.11161   FALSE FALSE
## accel_forearm_z          1.006      2.95587   FALSE FALSE
## magnet_forearm_x         1.012      7.76679   FALSE FALSE
## magnet_forearm_y         1.247      9.54031   FALSE FALSE
## magnet_forearm_z         1.000      8.57711   FALSE FALSE
## classe                   1.470      0.02548   FALSE FALSE
```

Algorithm

```
# Divide the given training set into 4 roughly equal sets.
set.seed(666)
ids_small <- createDataPartition(y=df_training$classe, p=0.25, list=FALSE)
df_small1 <- df_training[ids_small,]
df_remainder <- df_training[-ids_small,]
ids_small <- createDataPartition(y=df_remainder$classe, p=0.33, list=FALSE)
df_small2 <- df_remainder[ids_small,]
df_remainder <- df_remainder[-ids_small,]
ids_small <- createDataPartition(y=df_remainder$classe, p=0.5, list=FALSE)
df_small3 <- df_remainder[ids_small,]
df_small4 <- df_remainder[-ids_small,]
# Divide each of these 4 sets into training (60%) and test (40%) sets.
inTrain <- createDataPartition(y=df_small1$classe, p=0.6, list=FALSE)
df_small_training1 <- df_small1[inTrain,]
df_small_testing1 <- df_small1[-inTrain,]
inTrain <- createDataPartition(y=df_small2$classe, p=0.6, list=FALSE)
df_small_training2 <- df_small2[inTrain,]
df_small_testing2 <- df_small2[-inTrain,]
inTrain <- createDataPartition(y=df_small3$classe, p=0.6, list=FALSE)
df_small_training3 <- df_small3[inTrain,]
df_small_testing3 <- df_small3[-inTrain,]
inTrain <- createDataPartition(y=df_small4$classe, p=0.6, list=FALSE)
df_small_training4 <- df_small4[inTrain,]
df_small_testing4 <- df_small4[-inTrain,]
```

Evaluation

Classification Tree

```
# Train on training set 1 of 4 with no extra features.
modFit <- train(df_small_training1$classe ~ ., data = df_small_training1, method="rpart")
print(modFit, digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2946, 2946, 2946, 2946, 2946, 2946, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0313  0.538     0.4099  0.0278       0.0395
##   0.0477  0.466     0.2945  0.0642       0.1100
##   0.1162  0.328     0.0643  0.0454       0.0632
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0313.
```

```
print(modFit$finalModel, digits=3)
```

```
## n= 2946
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2946 2110 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130 2699 1860 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34 225    2 A (0.99 0.0089 0 0 0) *
##      5) pitch_forearm>=-34 2474 1860 A (0.25 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436 2080 1470 A (0.29 0.17 0.24 0.19 0.11)
##         20) roll_forearm< 124 1313  762 A (0.42 0.18 0.19 0.16 0.049) *
##         21) roll_forearm>=124 767  523 C (0.072 0.17 0.32 0.23 0.21) *
##       11) magnet_dumbbell_y>=436 394  186 B (0.018 0.53 0.036 0.24 0.18) *
##    3) roll_belt>=130 247    1 E (0.004 0 0 0 1) *
```

```
# Run against testing set 1 of 4 with no extra features.
predictions <- predict(modFit, newdata=df_small_testing1)
print(confusionMatrix(predictions, df_small_testing1$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 498 172 158 152  49
##          B  13 116   8  55  46
##          C  45  92 176 114 102
##          D   0   0   0   0   0
##          E   2   0   0   0 163
##
## Overall Statistics
##
##                Accuracy : 0.486
##                  95% CI : (0.4636, 0.5084)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3278
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8925  0.30526  0.51462   0.0000  0.45278
## Specificity            0.6215  0.92283  0.78196   1.0000  0.99875
## Pos Pred Value         0.4840  0.48739  0.33270      NaN  0.98788
## Neg Pred Value         0.9356  0.84678  0.88408   0.8363  0.89031
## Prevalence             0.2845  0.19378  0.17440   0.1637  0.18358
## Detection Rate         0.2540  0.05915  0.08975   0.0000  0.08312
## Detection Prevalence   0.5247  0.12137  0.26976   0.0000  0.08414
## Balanced Accuracy      0.7570  0.61405  0.64829   0.5000  0.72576
```

```
# Train on training set 1 of 4 with only preprocessing.
modFit <- train(df_small_training1$classe ~ .,  preProcess=c("center", "scale"), data = df_
small_training1, method="rpart")
print(modFit, digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2946, 2946, 2946, 2946, 2946, 2946, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0313  0.550     0.4206  0.0371       0.0552
##   0.0477  0.469     0.2988  0.0542       0.0929
##   0.1162  0.332     0.0713  0.0380       0.0598
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0313.
```

```
# Train on training set 1 of 4 with only cross validation.
modFit <- train(df_small_training1$classe ~ .,  trControl=trainControl(method = "cv", numbe
r = 4), data = df_small_training1, method="rpart")
print(modFit, digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2210, 2210, 2209, 2209
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0313  0.530     0.3907  0.0658       0.0960
##   0.0477  0.425     0.2257  0.0673       0.1149
##   0.1162  0.345     0.0927  0.0406       0.0618
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0313.
```

```
# Train on training set 1 of 4 with both preprocessing and cross validation.
modFit <- train(df_small_training1$classe ~ .,  preProcess=c("center", "scale"), trControl=
trainControl(method = "cv", number = 4), data = df_small_training1, method="rpart")
print(modFit, digits=3)
```

```
## CART
##
## 2946 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: 2209, 2207, 2210, 2212
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0313  0.519     0.3783  0.0255       0.0370
##   0.0477  0.429     0.2309  0.0625       0.1075
##   0.1162  0.321     0.0561  0.0422       0.0648
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0313.
```

```
# Run against testing set 1 of 4 with both preprocessing and cross validation.
predictions <- predict(modFit, newdata=df_small_testing1)
print(confusionMatrix(predictions, df_small_testing1$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 498 172 158 152  49
##          B  13 116   8  55  46
##          C  45  92 176 114 102
##          D   0   0   0   0   0
##          E   2   0   0   0 163
##
## Overall Statistics
##
##                Accuracy : 0.486
##                  95% CI : (0.4636, 0.5084)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3278
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8925  0.30526  0.51462   0.0000  0.45278
## Specificity            0.6215  0.92283  0.78196   1.0000  0.99875
## Pos Pred Value         0.4840  0.48739  0.33270      NaN  0.98788
## Neg Pred Value         0.9356  0.84678  0.88408   0.8363  0.89031
## Prevalence             0.2845  0.19378  0.17440   0.1637  0.18358
## Detection Rate         0.2540  0.05915  0.08975   0.0000  0.08312
## Detection Prevalence   0.5247  0.12137  0.26976   0.0000  0.08414
## Balanced Accuracy      0.7570  0.61405  0.64829   0.5000  0.72576
```

Random Forest

```
# Train on training set 1 of 4 with only cross validation.
modFit <- train(df_small_training1$classe ~ ., method="rf", trControl=trainControl(method =
"cv", number = 4), data=df_small_training1)
print(modFit, digits=3)
```

```
## Random Forest
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2209, 2209, 2210, 2210
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.946     0.932  0.00476      0.00605
##   27    0.952     0.939  0.00973      0.01232
##   52    0.942     0.927  0.00953      0.01204
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Run against testing set 1 of 4.
predictions <- predict(modFit, newdata=df_small_testing1)
print(confusionMatrix(predictions, df_small_testing1$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 557  24   0   0   0
##          B   0 343  12   3   3
##          C   1   8 326  11   4
##          D   0   3   4 307   6
##          E   0   2   0   0 347
##
## Overall Statistics
##
##                Accuracy : 0.9587
##                  95% CI : (0.9489, 0.9671)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9477
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9026   0.9532   0.9564   0.9639
## Specificity            0.9829   0.9886   0.9852   0.9921   0.9988
## Pos Pred Value         0.9587   0.9501   0.9314   0.9594   0.9943
## Neg Pred Value         0.9993   0.9769   0.9901   0.9915   0.9919
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2840   0.1749   0.1662   0.1566   0.1770
## Detection Prevalence   0.2963   0.1841   0.1785   0.1632   0.1780
## Balanced Accuracy      0.9906   0.9456   0.9692   0.9742   0.9813
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(modFit, newdata=df_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
```

```
# Train on training set 1 of 4 with only both preprocessing and cross validation.
modFit <- train(df_small_training1$classe ~ ., method="rf", preProcess=c("center", "scale")
, trControl=trainControl(method = "cv", number = 4), data=df_small_training1)
print(modFit, digits=3)
```

```
## Random Forest
##
## 2946 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: 2210, 2208, 2210, 2210
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.948     0.934  0.00597      0.00758
##   27    0.948     0.935  0.00299      0.00383
##   52    0.942     0.927  0.00454      0.00573
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Run against testing set 1 of 4.
predictions <- predict(modFit, newdata=df_small_testing1)
print(confusionMatrix(predictions, df_small_testing1$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 557  25   0   0   0
##          B   0 343  12   2   3
##          C   1   7 325  12   4
##          D   0   3   5 307   8
##          E   0   2   0   0 345
##
## Overall Statistics
##
##                Accuracy : 0.9572
##                  95% CI : (0.9472, 0.9657)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9457
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9026   0.9503   0.9564   0.9583
## Specificity            0.9822   0.9892   0.9852   0.9902   0.9988
## Pos Pred Value         0.9570   0.9528   0.9312   0.9505   0.9942
## Neg Pred Value         0.9993   0.9769   0.9895   0.9915   0.9907
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2840   0.1749   0.1657   0.1566   0.1759
## Detection Prevalence   0.2968   0.1836   0.1780   0.1647   0.1770
## Balanced Accuracy      0.9902   0.9459   0.9677   0.9733   0.9785
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(modFit, newdata=df_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
```

```
# Train on training set 2 of 4 with only cross validation.
modFit <- train(df_small_training2$classe ~ ., method="rf", preProcess=c("center", "scale")
, trControl=trainControl(method = "cv", number = 4), data=df_small_training2)
print(modFit, digits=3)
```

```
## Random Forest
##
## 2917 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: 2189, 2188, 2186, 2188
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.951     0.938  0.00984      0.01244
##   27    0.961     0.951  0.00643      0.00814
##   52    0.956     0.944  0.00585      0.00743
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Run against testing set 2 of 4.
predictions <- predict(modFit, newdata=df_small_testing2)
print(confusionMatrix(predictions, df_small_testing2$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 548  12   0   1   0
##          B   2 351  13   0   0
##          C   1   9 318   3   3
##          D   0   1   7 314   3
##          E   1   3   0   0 351
##
## Overall Statistics
##
##                Accuracy : 0.9696
##                  95% CI : (0.961, 0.9768)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9615
##  Mcnemar's Test P-Value : 0.01288
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9928   0.9335   0.9408   0.9874   0.9832
## Specificity            0.9906   0.9904   0.9900   0.9932   0.9975
## Pos Pred Value         0.9768   0.9590   0.9521   0.9662   0.9887
## Neg Pred Value         0.9971   0.9841   0.9876   0.9975   0.9962
## Prevalence             0.2844   0.1937   0.1741   0.1638   0.1839
## Detection Rate         0.2823   0.1808   0.1638   0.1618   0.1808
## Detection Prevalence   0.2890   0.1886   0.1721   0.1674   0.1829
## Balanced Accuracy      0.9917   0.9620   0.9654   0.9903   0.9903
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(modFit, newdata=df_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
```

```
# Train on training set 3 of 4 with only cross validation.
modFit <- train(df_small_training3$classe ~ ., method="rf", preProcess=c("center", "scale")
, trControl=trainControl(method = "cv", number = 4), data=df_small_training3)
print(modFit, digits=3)
```

```
## Random Forest
##
## 2960 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: 2220, 2221, 2219, 2220
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.945     0.930  0.01619      0.02046
##   27    0.956     0.945  0.00676      0.00856
##   52    0.950     0.937  0.00207      0.00263
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Run against testing set 3 of 4.
predictions <- predict(modFit, newdata=df_small_testing3)
print(confusionMatrix(predictions, df_small_testing3$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 555  10   0   0   0
##          B   0 361  11   0   4
##          C   4  10 332   8   7
##          D   0   0   1 315   2
##          E   1   0   0   0 349
##
## Overall Statistics
##
##                Accuracy : 0.9706
##                  95% CI : (0.9621, 0.9776)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9628
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9911   0.9475   0.9651   0.9752   0.9641
## Specificity            0.9929   0.9906   0.9822   0.9982   0.9994
## Pos Pred Value         0.9823   0.9601   0.9197   0.9906   0.9971
## Neg Pred Value         0.9964   0.9875   0.9925   0.9952   0.9920
## Prevalence             0.2843   0.1934   0.1746   0.1640   0.1838
## Detection Rate         0.2817   0.1832   0.1685   0.1599   0.1772
## Detection Prevalence   0.2868   0.1909   0.1832   0.1614   0.1777
## Balanced Accuracy      0.9920   0.9690   0.9736   0.9867   0.9817
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(modFit, newdata=df_testing))
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

```
# Train on training set 4 of 4 with only cross validation.
modFit <- train(df_small_training4$classe ~ ., method="rf", preProcess=c("center", "scale")
, trControl=trainControl(method = "cv", number = 4), data=df_small_training4)
print(modFit, digits=3)
```

```
## Random Forest
##
## 2958 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: 2219, 2218, 2219, 2218
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.953     0.941  0.00650      0.00825
##   27    0.953     0.941  0.00570      0.00719
##   52    0.941     0.925  0.00392      0.00496
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
# Run against testing set 4 of 4.
predictions <- predict(modFit, newdata=df_small_testing4)
print(confusionMatrix(predictions, df_small_testing4$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 552   6   0   0   0
##          B   3 372  10   0   1
##          C   1   3 330  15   1
##          D   2   0   3 304   3
##          E   2   0   0   4 357
##
## Overall Statistics
##
##                Accuracy : 0.9726
##                  95% CI : (0.9644, 0.9793)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9653
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9764   0.9621   0.9412   0.9862
## Specificity            0.9957   0.9912   0.9877   0.9951   0.9963
## Pos Pred Value         0.9892   0.9637   0.9429   0.9744   0.9835
## Neg Pred Value         0.9943   0.9943   0.9920   0.9885   0.9969
## Prevalence             0.2844   0.1935   0.1742   0.1640   0.1838
## Detection Rate         0.2803   0.1889   0.1676   0.1544   0.1813
## Detection Prevalence   0.2834   0.1960   0.1778   0.1585   0.1844
## Balanced Accuracy      0.9907   0.9838   0.9749   0.9682   0.9912
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(modFit, newdata=df_testing))
```

```
##  [1] B A B A A E D D A A B C B A E E A B B B
## Levels: A B C D E
```

Out of Sample Error

Conclusion