Practical Machine Learning Course Project

The goal is to predict the manner in which exercise is done.

Synopsis

Using devices 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,goal is 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. This is the "classe" variable in the training set.

Objectives

1.Create a report describing how you built your model? 2.How you used cross validation? 3.What you think the expected out of sample error is? 4.Why you made the choices you did. 5.Use your prediction model to predict 20 different test cases.

Data Sources

The training data for this project is taken from below link:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data is taken from below link:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```r
## Load required packages
library(AppliedPredictiveModeling)
library(caret)
```

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

```r
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

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

```r
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Load and Clean Data

```r
##Data is downloaded and saved in current working directory. Below commands load data in R.
df_training<-read.csv("pml-training.csv", na.strings=c("NA",""))
colnames_train<-colnames(df_training)
df_testing<-read.csv("pml-testing.csv", na.strings=c("NA",""))
colnames_test<-colnames(df_testing)
```

```r
## Preliminary test is perrformed to check if column names are same in test and training data
all.equal(colnames_train[1:length(colnames_train)-1],colnames_test[1:length(colnames_train)-1])
```

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

```r
##Remove NAs data cleaning
df_training_nonNA <-df_training[,which(as.numeric(colSums(is.na(df_training)))==0)]
df_testing_nonNA <-df_testing[,which(as.numeric(colSums(is.na(df_testing)))==0)]

## Remove Non-numeric Variables
df_training_nonNA <- df_training_nonNA[,-(1:7)]
df_testing_nonNA <- df_testing_nonNA[,-(1:7)]
```

```r
## Check variablility of covariates
nsv <-nearZeroVar(df_training_nonNA, saveMetrics=TRUE)
```

Non zero variance variables are FALSE, so there is no need to eliminate any covariates due to lack of variability

Data Partitioning

We were provided with a large training set (19,622 entries) and a small testing set (20 entries). Instead of performing the algorithm on the entire training set, as it would be time consuming and wouldn't allow for an attempt on a testing set, I chose to divide the given training set into four roughly equal sets, each of which was then split into a training set (comprising 60% of the entries) and a testing set (comprising 40% of the entries).

```r
set.seed(888)
subData <- createDataPartition(y=df_training_nonNA$classe,p=0.25,list=FALSE)
subData1<-df_training_nonNA[subData,]
subTest <- df_training_nonNA[-subData,]
set.seed(888)
subData <-createDataPartition(y=df_remainder$classe, p= 0.33,list=FALSE)
subData2 <- df_remainder[subData,]
subTest <-df_remainder[-subData,]
set.seed(888)
```

```r
subData <- createDataPartition(y=df_remainder$classe,p=0.5,list=FALSE)
subData3 <- df_remainder[subData,]
subData4<- df_remainder[-subData,]
set.seed(888)
subTrain <- createDataPartition(y=df_small1$classe, p=0.6, list=FALSE)
subData_training1 <-subData1[subTrain,]
subData_testing1<- subData1[-subTrain,]
set.seed(888)
subTrain <- createDataPartition(y=df_small2$classe, p=0.6, list=FALSE)
subData_training2 <-subData2[subTrain,]
subData_testing2 <-subData2[-subTrain,]
set.seed(888)
subTrain <- createDataPartition(y=df_small3$classe, p=0.6, list=FALSE)
subData_testing3 <-subData3[subTrain,]
subData_training3 <-subData3[-subTrain,]
set.seed(888)
subTrain <- createDataPartition(y=df_small3$classe, p=0.6, list=FALSE)
subData_testing4 <-subData4[subTrain,]
subData_training4 <-subData4[-subTrain,]
```

Trial 1 Decision Tree Model Fitting

```r
set.seed(888)
modFit <- train(subData_training1$classe ~.,data=subData_training1,method="rpart")
print(modFit)
```

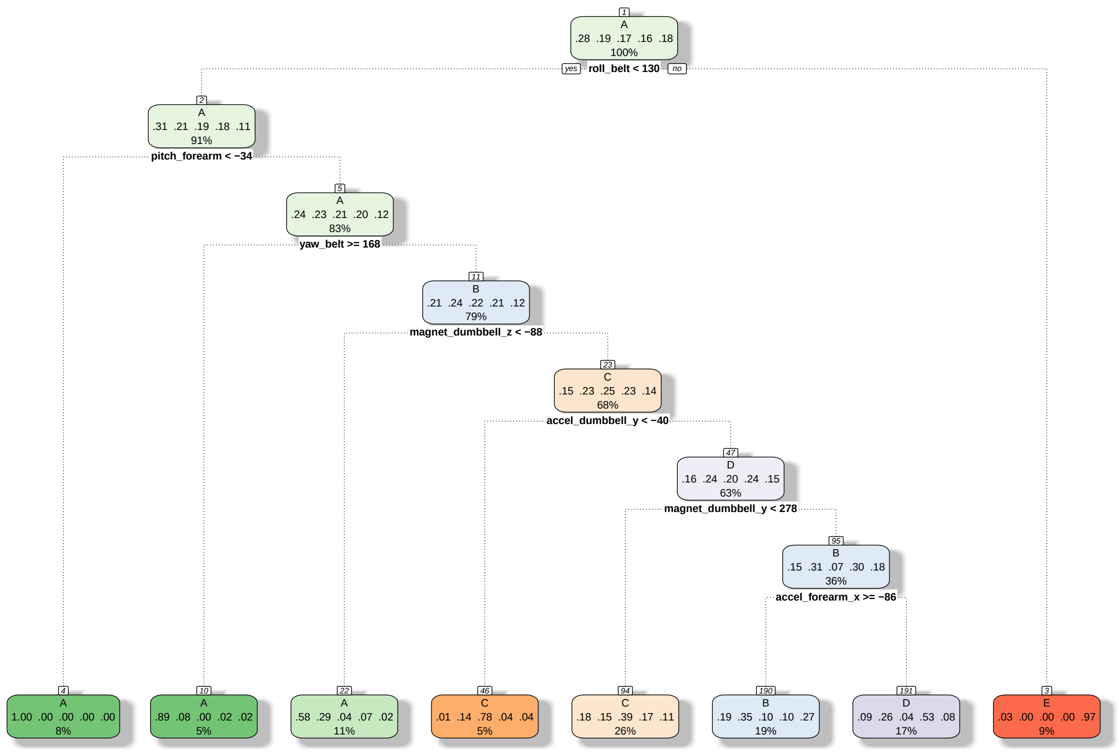
```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 2946, 2946, 2946, 2946, 2946, 2946, ...
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.02797534  0.5761250  0.46349709  0.02336491   0.03135466
##   0.04343291  0.4653918  0.29769957  0.07957532   0.13578839
##   0.11806543  0.3273469  0.06487751  0.04382772   0.06373561
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02797534.
```

```r
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 2683 1850 A (0.31 0.21 0.19 0.18 0.11)
##       4) pitch_forearm< -34.3 234    1 A (1 0.0043 0 0 0) *
##       5) pitch_forearm>=-34.3 2449 1850 A (0.24 0.23 0.21 0.2 0.12)
##        10) yaw_belt>=168 133   15 A (0.89 0.075 0 0.023 0.015) *
##        11) yaw_belt< 168 2316 1760 B (0.21 0.24 0.22 0.21 0.12)
##          22) magnet_dumbbell_z< -88.5 313  133 A (0.58 0.29 0.045 0.073 0.019) *
##          23) magnet_dumbbell_z>=-88.5 2003 1500 C (0.15 0.23 0.25 0.23 0.14)
##            46) accel_dumbbell_y< -40.5 161   36 C (0.012 0.14 0.78 0.037 0.037) *
##            47) accel_dumbbell_y>=-40.5 1842 1390 D (0.16 0.24 0.2 0.24 0.15)
##              94) magnet_dumbbell_y< 278 767  466 C (0.18 0.15 0.39 0.17 0.11) *
##              95) magnet_dumbbell_y>=278 1075  744 B (0.15 0.31 0.069 0.3 0.18)
##               190) accel_forearm_x>=-86.5 566  368 B (0.19 0.35 0.099 0.095 0.27) *
##               191) accel_forearm_x< -86.5 509  239 D (0.094 0.26 0.035 0.53 0.079) *
##     3) roll_belt>=130 263    7 E (0.027 0 0 0 0.97) *
```

```r
fancyRpartPlot(modFit$finalModel)
```



Rattle 2015−Apr−25 14:13:39 Mandeep

```r
# Run against testing set 1
predictions<-predict(modFit, newdata=subData_testing1)
print(confusionMatrix(predictions, subData_testing1$classe),digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 358  70  10  23   4
##          B  57 142  46  52  96
##          C  83  81 270  86  65
##          D  52  87  16 160  19
##          E   8   0   0   0 176
##
## Overall Statistics
##
##                Accuracy : 0.564
##                  95% CI : (0.5417, 0.5861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4526
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6416  0.37368   0.7895  0.49844  0.48889
## Specificity            0.9237  0.84124   0.8054  0.89390  0.99500
## Pos Pred Value         0.7699  0.36132   0.4615  0.47904  0.95652
## Neg Pred Value         0.8663  0.84821   0.9477  0.90104  0.89645
## Prevalence             0.2845  0.19378   0.1744  0.16369  0.18358
## Detection Rate         0.1826  0.07241   0.1377  0.08159  0.08975
## Detection Prevalence   0.2371  0.20041   0.2983  0.17032  0.09383
## Balanced Accuracy      0.7827  0.60746   0.7975  0.69617  0.74195
```

Accuracy rate is very low as .564 and needs improvement.

```
# 1a. Train Decision Tree Model with preprocessing.
set.seed(888)
modFit <- train(subData_training1$classe ~ ., preProcess= c("center","scale"), data=subData_training1,m
print(modFit,digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## 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
##   0.0280  0.576     0.4635  0.0233       0.0312
##   0.0434  0.465     0.2977  0.0795       0.1357
##   0.1181  0.327     0.0649  0.0438       0.0637
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.028.
```

```
# 1b. Train Decision Tree Modelwith cross validation.
modFit <- train(subData_training1$classe ~ ., trControl= trainControl(method= "cv",number=4), data=subDa

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: 2209, 2209, 2210, 2210
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0280  0.561     0.4418  0.0360       0.0563
##   0.0434  0.488     0.3327  0.0841       0.1413
##   0.1181  0.323     0.0592  0.0448       0.0684
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.028.
```

```
# 1c. Train Decision Tree Modelwith both preprocessing and cross validation.
set.seed(888)
modFit <- train(subData_training1$classe ~.,preProcess=c("center","scale"),trControl=trainControl(metho
print(modFit, digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 2207, 2210, 2210, 2211
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0280  0.574     0.4652  0.0172       0.0202
##   0.0434  0.486     0.3366  0.0907       0.1528
##   0.1181  0.322     0.0576  0.0436       0.0665
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.028.
```

```
# Run against testing set 1 of 4 with both preprocessing and cross validation.
predictions<- predict(modFit,newdata=subData_testing1)
print(confusionMatrix(predictions,subData_testing1$classe),digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 358  70  10  23   4
##          B  57 142  46  52  96
##          C  83  81 270  86  65
##          D  52  87  16 160  19
##          E   8   0   0   0 176
##
## Overall Statistics
##
##                Accuracy : 0.564
##                  95% CI : (0.5417, 0.5861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4526
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6416  0.37368   0.7895  0.49844  0.48889
## Specificity            0.9237  0.84124   0.8054  0.89390  0.99500
## Pos Pred Value         0.7699  0.36132   0.4615  0.47904  0.95652
## Neg Pred Value         0.8663  0.84821   0.9477  0.90104  0.89645
## Prevalence             0.2845  0.19378   0.1744  0.16369  0.18358
## Detection Rate         0.1826  0.07241   0.1377  0.08159  0.08975
## Detection Prevalence   0.2371  0.20041   0.2983  0.17032  0.09383
## Balanced Accuracy      0.7827  0.60746   0.7975  0.69617  0.74195
```

2 Random forest Model Fitting

```
## 2a Train on training set 1 of 4 with only cross validation.
modFit<-train(subData_training1$classe ~.,method="rf",trControl=trainControl(method="cv",number=4),data
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: 2210, 2210, 2208, 2210
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.944     0.930  0.0115       0.0146
##   27    0.950     0.936  0.0101       0.0128
##   52    0.941     0.926  0.0151       0.0191
##
## 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=subData_testing1)
print(confusionMatrix(predictions,subData_testing1$classe),digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 554  15   2   1   0
##          B   1 353  11   2   2
##          C   0  11 325  16   4
##          D   3   1   4 302   5
##          E   0   0   0   0 349
##
## Overall Statistics
##
##                Accuracy : 0.9602
##                  95% CI : (0.9506, 0.9684)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9496
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9928   0.9289   0.9503   0.9408   0.9694
## Specificity            0.9872   0.9899   0.9809   0.9921   1.0000
## Pos Pred Value         0.9685   0.9566   0.9129   0.9587   1.0000
## Neg Pred Value         0.9971   0.9830   0.9894   0.9885   0.9932
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2825   0.1800   0.1657   0.1540   0.1780
## Detection Prevalence   0.2917   0.1882   0.1815   0.1606   0.1780
## Balanced Accuracy      0.9900   0.9594   0.9656   0.9664   0.9847
```

```r
print(predict(modFit, newdata=df_testing_nonNA))
```

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

```r
# 2b Train Random forest Model with both preprocessing and cross validation.
set.seed(888)
modFit <- train(subData_training1$classe ~., method="rf",preProcess=c("center","scale"),trControl=train
print(modFit,digits=3)
```

```
## Random Forest
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 2207, 2210, 2210, 2211
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.945     0.930  0.00347      0.00436
##   27    0.949     0.936  0.00963      0.01219
##   52    0.946     0.932  0.01216      0.01539
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```r
# Run against testing set 1 of 4.
predictions <- predict(modFit, newdata=subData_testing1)
print(confusionMatrix(predictions, subData_testing1$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 554  14   1   2   0
##          B   1 355  11   1   2
##          C   0  11 327  17   3
##          D   3   0   3 300   6
##          E   0   0   0   1 349
##
## Overall Statistics
##
##                Accuracy : 0.9612
##                  95% CI : (0.9517, 0.9693)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9509
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9928   0.9342   0.9561   0.9346   0.9694
## Specificity            0.9879   0.9905   0.9809   0.9927   0.9994
## Pos Pred Value         0.9702   0.9595   0.9134   0.9615   0.9971
## Neg Pred Value         0.9971   0.9843   0.9906   0.9873   0.9932
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2825   0.1810   0.1668   0.1530   0.1780
## Detection Prevalence   0.2912   0.1887   0.1826   0.1591   0.1785
## Balanced Accuracy      0.9904   0.9624   0.9685   0.9636   0.9844
```

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

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

```
# 2c Train Random forest Model with preprocessing and cross validation training data2.
set.seed(888)
modFit <- train(subData_training2$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=t
print(modFit, digits=3)
```

```
## Random Forest
##
## 2917 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 1464, 1464, 1466, 1465
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.931     0.913  0.0175       0.0221
##   27    0.931     0.913  0.0199       0.0252
##   52    0.926     0.906  0.0207       0.0262
##
## 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 2 of 4.
predictions <- predict(modFit, newdata=subData_testing2)
print(confusionMatrix(predictions, subData_testing2$classe), digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 373   6   0   1   0
##          B   4 228   5   0   2
##          C   0  11 212  16   5
##          D   1   1   3 199   4
##          E   0   1   0   0 230
##
## Overall Statistics
##
##                Accuracy : 0.9539
##                  95% CI : (0.9411, 0.9647)
##     No Information Rate : 0.2903
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9416
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9868   0.9231   0.9636   0.9213   0.9544
## Specificity            0.9924   0.9896   0.9704   0.9917   0.9991
## Pos Pred Value         0.9816   0.9540   0.8689   0.9567   0.9957
## Neg Pred Value         0.9946   0.9821   0.9924   0.9845   0.9897
## Prevalence             0.2903   0.1897   0.1690   0.1659   0.1851
## Detection Rate         0.2865   0.1751   0.1628   0.1528   0.1767
## Detection Prevalence   0.2919   0.1836   0.1874   0.1598   0.1774
## Balanced Accuracy      0.9896   0.9563   0.9670   0.9565   0.9767
```

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

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

```
# 2d Train on training set 3 of 4 with preprocessing and cross validation.
set.seed(888)
modFit <- train(subData_training3$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=t
print(modFit, digits=3)
```

```
## Random Forest
##
## 1970 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 1478, 1479, 1477, 1476
##
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.939     0.923  0.00299      0.00381
##   27    0.937     0.920  0.01457      0.01854
##   52    0.925     0.905  0.01029      0.01312
##
## 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 3 of 4.
predictions <- predict(modFit, newdata=subData_testing3)
print(confusionMatrix(predictions, subData_testing3$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 829  31   1  10   1
##          B   4 509  28   0  16
##          C   3  32 483  34  17
##          D   5   1   4 438   8
##          E   1   0   0   3 502
##
## Overall Statistics
##
##                Accuracy : 0.9328
##                  95% CI : (0.9231, 0.9415)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9149
##  Mcnemar's Test P-Value : 9.335e-14
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9846   0.8883   0.9360   0.9031   0.9228
## Specificity            0.9797   0.9799   0.9648   0.9927   0.9983
## Pos Pred Value         0.9507   0.9138   0.8489   0.9605   0.9921
## Neg Pred Value         0.9938   0.9734   0.9862   0.9812   0.9829
## Prevalence             0.2845   0.1936   0.1743   0.1639   0.1838
## Detection Rate         0.2801   0.1720   0.1632   0.1480   0.1696
## Detection Prevalence   0.2946   0.1882   0.1922   0.1541   0.1709
## Balanced Accuracy      0.9821   0.9341   0.9504   0.9479   0.9606
```

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

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

There is significant difference in Accuracy using Decison Tree Modeling and Random Forest. Using Random Forest model and cross Validation Accuracy approaches 1.

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

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

```
predictfinal <- predict(modFit, df_testing)
```

Out Of Sample Error

Random Forest (preprocessing and cross validation) Testing Set 1: 1 - .9612 = 0.0388 Random Forest (preprocessing and cross validation) Testing Set 2: 1 - .9539 = 0.0461 Random Forest (preprocessing and cross validation) Testing Set 3: 1 - .9328 = 0.0672

Average Out Of Sample Error: 0.0169

Function for Course Project submission

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictfinal)
```