Project Report of Practical Machine Learning

In this report, we try to build an classification model based on the *Weight Lifting Exercise Dataset*. We first read the TRAIN and TEST data set into the work space. TRAIN data set is a labeled data set used to train and validate (test) the model. TEST data set is an unlabeled data set which we want to predict the label for.

```
setwd("C:/Dropbox/Coursera/machinelearning")
TRAIN <- read.csv("pml-training.csv")
TEST <- read.csv("pml-testing.csv")
```

We select the predictors by observing the values and meanings of the variables. Basically, we omit the row number, time related variables and the variables with almost no non-NA values.

```
TRAIN <- TRAIN[,-c(1,3,4,5,6,12:36,50:59,69:83,87:101,103:112,125:139,140:150)]
TEST <- TEST[,-c(1,3,4,5,6,12:36,50:59,69:83,87:101,103:112,125:139,140:150)]
```

We split the TRAIN data set into training and testing.

```
library(caret)
```

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

```
inTrain <- createDataPartition(y=TRAIN$classe, p = 0.75, list=FALSE)
training <- TRAIN[inTrain,]
testing <- TRAIN[-inTrain,]
```

We then train a decision tree model to see if it works well. We also plot the tree out.

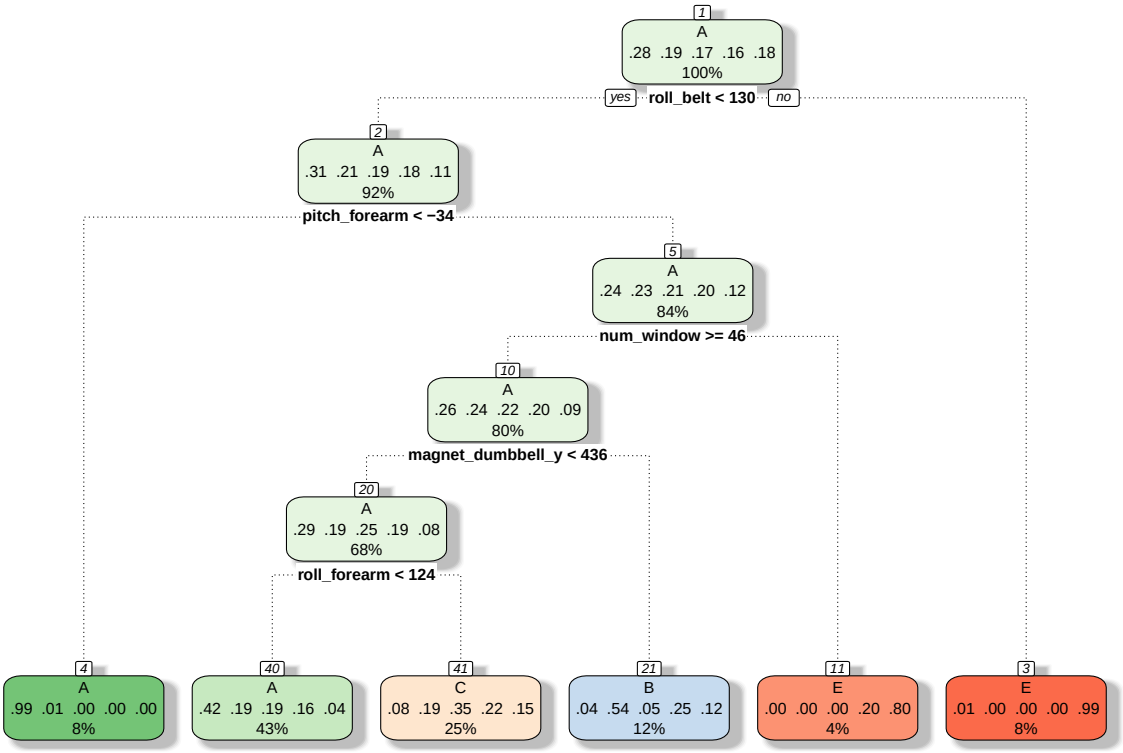
```
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
FitTREE <- train(classe~., data = training, method = "rpart")
```

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

```
fancyRpartPlot(FitTREE$finalModel)
```



Rattle 2015−Aug−22 23:13:54 Dingguo

We estimate the accuracy of the tree model using testing data.

```
pred <- predict(FitTREE, testing);
testing$predRight <-pred == testing$classe
table(pred,testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1257  380  412  328   90
##    B   22  344   24  144   69
##    C  111  225  419  298  185
##    D    0    0    0    0    0
##    E    5    0    0   34  557
```

Apparently, the accuracy is not very satisfying. So we decide to use random forest.

We train a random forest model

```
FitRF <- train(classe ~ ., method="rf", ntree = 10, data = training, trControl=trainControl(method = "cv
```

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

```
print(FitRF, digits = 3)
```

```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11776, 11774, 11775, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.981     0.976  0.00252      0.00319
##   29    0.995     0.993  0.00197      0.00249
##   57    0.992     0.990  0.00429      0.00543
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 29.
```

We estimate the accuracy of the random forest model using testing data.

```
pred <- predict(FitRF, newdata=testing)
```

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

```
print(confusionMatrix(pred, testing$classe), digits=3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  947    0    0    0
##          C    0    2  855    0    0
##          D    0    0    0  803    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.998    1.000    0.999    0.998
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    0.998    0.998    0.999
## Neg Pred Value          1.000    0.999    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.164    0.183
## Detection Prevalence    0.284    0.193    0.175    0.164    0.184
## Balanced Accuracy       1.000    0.999    1.000    0.999    0.999
```

We predict the classification for TEST data set.

```
print(predict(FitRF, newdata=TEST))
```

```
##  [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
```

In order to estimate the out of sample error, we carry out cross validation as follows.

First, we divide the whole TRAIN dataset into three parts.

```
folds <- createFolds(y=TRAIN$classe,k=3)
```

Then we use two parts as training data set and the other one as testing data set, which is as shown below.

Cross Validation 1

```
training1 <- TRAIN[c(folds[[2]],folds[[3]]),]
testing1  <- TRAIN[folds[[1]],]
FitRF1 <- train(classe ~ ., method="rf", ntree = 10 , data = training1, trControl=trainControl(method =
pred <- predict(FitRF1, newdata=testing1)
print(confusionMatrix(pred, testing1$classe), digits=3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1859    8    0    0    0
##          B    0 1253    4    0    5
##          C    0    3 1134    7    2
##          D    1    0    3 1062    4
##          E    0    1    0    3 1191
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.995)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.992
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.991    0.994    0.991    0.991
## Specificity             0.998    0.998    0.998    0.999    0.999
## Pos Pred Value          0.996    0.993    0.990    0.993    0.997
## Neg Pred Value          1.000    0.998    0.999    0.998    0.998
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.162    0.182
## Detection Prevalence    0.285    0.193    0.175    0.164    0.183
## Balanced Accuracy       0.999    0.994    0.996    0.995    0.995
```

Cross Validation 2

```
training2 <- TRAIN[c(folds[[1]],folds[[3]]),]
testing2  <- TRAIN[folds[[2]],]
FitRF2 <- train(classe ~ ., method="rf", ntree = 10 , data = training2, trControl=trainControl(method =
pred <- predict(FitRF2, newdata=testing2)
print(confusionMatrix(pred, testing2$classe), digits=3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1859    7    0    0    0
##          B    1 1249    7    0    0
##          C    0   10 1131   12    0
##          D    0    0    3 1059    5
##          E    0    0    0    1 1197
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.991, 0.995)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.987    0.991    0.988    0.996
## Specificity             0.999    0.998    0.996    0.999    1.000
## Pos Pred Value          0.996    0.994    0.981    0.993    0.999
## Neg Pred Value          1.000    0.997    0.998    0.998    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.191    0.173    0.162    0.183
## Detection Prevalence    0.285    0.192    0.176    0.163    0.183
## Balanced Accuracy       0.999    0.993    0.994    0.993    0.998
```

Cross Validation 3

```
training3 <- TRAIN[c(folds[[1]],folds[[2]]),]
testing3  <- TRAIN[folds[[3]],]
FitRF3 <- train(classe ~ ., method="rf", ntree = 10 , data = training3, trControl=trainControl(method =
pred <- predict(FitRF3, newdata=testing3)
print(confusionMatrix(pred, testing3$classe), digits=3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1859    5    0    0    0
##          B    0 1254    6    0    1
##          C    0    4 1134    2    1
##          D    0    3    0 1069    1
##          E    1    0    0    1 1200
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.991    0.995    0.997    0.998
## Specificity             0.999    0.999    0.999    0.999    1.000
## Pos Pred Value          0.997    0.994    0.994    0.996    0.998
## Neg Pred Value          1.000    0.998    0.999    0.999    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.163    0.183
## Detection Prevalence    0.285    0.193    0.174    0.164    0.184
## Balanced Accuracy       0.999    0.995    0.997    0.998    0.999
```

Conclusions

As can be seen from the results of the three cross validation, the accuracy of prediction is above 0.99. This is to say, if we are to predict the classification for TEST data set, the **out of sample error rate is going to be lower than 0.01**.