Practical Machine Learning Course Project

Summary

Two modeling methods were utilized for this project, Classification Tree and Random Forest. For both methods, 5-fold cross validation is used.

Load Packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

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

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

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

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

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.4.4
```

Import Data

It is assumed that data files are located in working directory

```
training <- read.csv("pml-training.csv", na.strings=c("NA",""), header=TRUE)
colnames_train <- colnames(training)
testing <- read.csv("pml-testing.csv", na.strings=c("NA",""), header=TRUE)
colnames_test <- colnames(testing)
```

Remove columns that aren't needed. NA data is cleaned, as well.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

Training data is separated into training and test sets

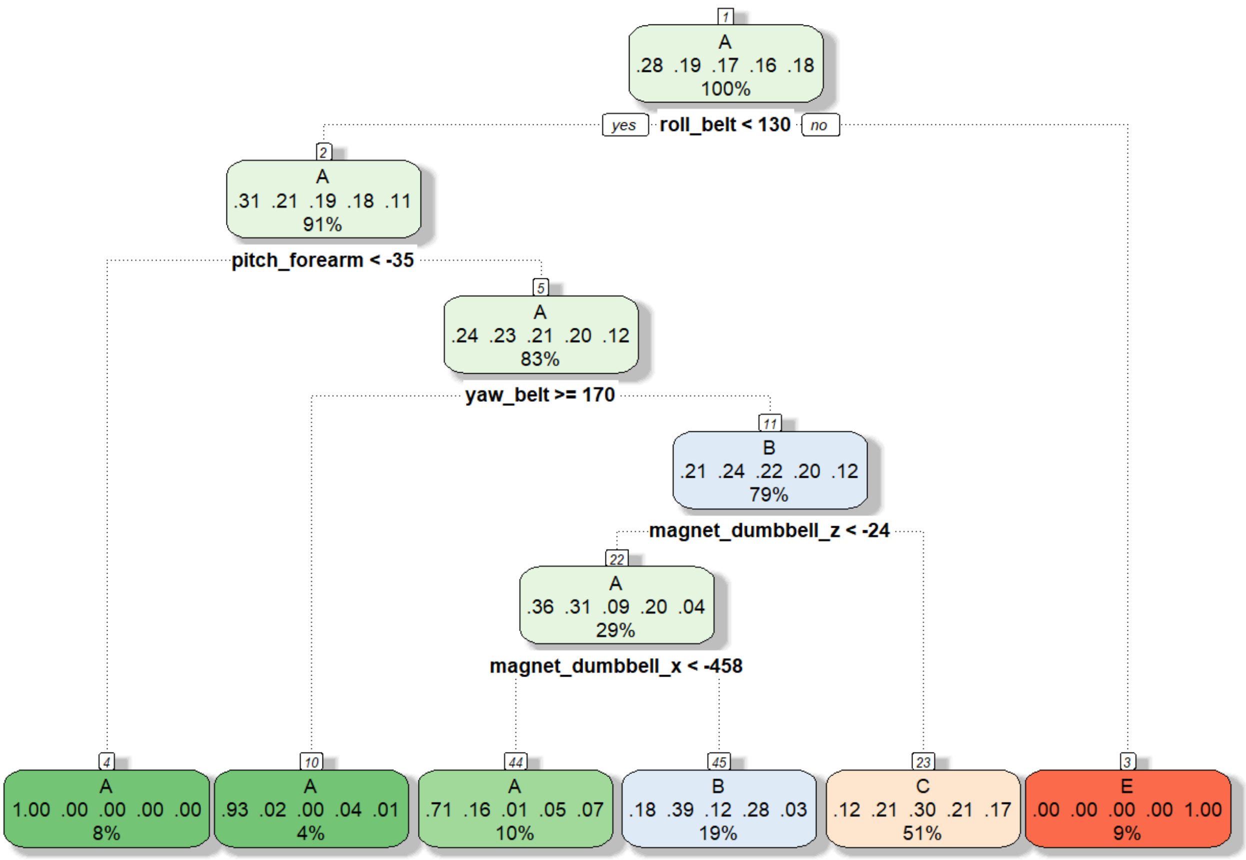
```
set.seed(1111)
ids_small <- createDataPartition(y=training$classe, p=0.25, list=FALSE)
small <- training[ids_small,]
remainder <- training[-ids_small,]

set.seed(1111)
Train <- createDataPartition(y=small$classe, p=0.6, list=FALSE)
small_training <- small[Train,]
small_testing <- small[-Train,]
```

Classification Tree

```r
set.seed(1111)
control <- trainControl(method = "cv", number = 5)
fit_rpart <- train(classe ~ ., data = small_training, method = "rpart", trControl =
control)
print(fit_rpart, digits=3)
```

```
## CART
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2356, 2358, 2357, 2356, 2357
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa
##   0.0365  0.537     0.4130
##   0.0408  0.516     0.3873
##   0.1214  0.334     0.0753
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0365.
```

```r
fancyRpartPlot(fit_rpart$finalModel)
```



Rattle 2018-Apr-22 18:26:13 vpwil

Classification Tree Prediction

```
predict_rpart <- predict(fit_rpart, newdata=small_testing)
print(confusionMatrix(predict_rpart, small_testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 357  47   3  12  17
##          B  76 108  47 116  17
##          C 125 225 292 193 148
##          D   0   0   0   0   0
##          E   0   0   0   0 178
## 
## Overall Statistics
## 
##                Accuracy : 0.4768
##                  95% CI : (0.4545, 0.4992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.3433
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6398  0.28421   0.8538   0.0000  0.49444
## Specificity            0.9437  0.83808   0.5732   1.0000  1.00000
## Pos Pred Value         0.8188  0.29670   0.2970      NaN  1.00000
## Neg Pred Value         0.8682  0.82968   0.9489   0.8363  0.89792
## Prevalence             0.2845  0.19378   0.1744   0.1637  0.18358
## Detection Rate         0.1820  0.05507   0.1489   0.0000  0.09077
## Detection Prevalence   0.2223  0.18562   0.5013   0.0000  0.09077
## Balanced Accuracy      0.7917  0.56114   0.7135   0.5000  0.74722
```

CT Accuracy 0.4768

Random Forest

```
set.seed(1111)
fit_rf <- train(classe ~ ., data = small_training, method = "rf", trControl = contr
ol)
print(fit_rf, digits = 4)
```

```
## Random Forest
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2356, 2358, 2357, 2356, 2357
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9569    0.9454
##   27    0.9603    0.9497
##   52    0.9525    0.9399
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Random Forest Prediction

```
predict_rf <- predict(fit_rf, newdata=small_testing)
print(confusionMatrix(predict_rf, small_testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 556  22   0   1   0
##          B   0 351   5   0   1
##          C   0   7 334   9   1
##          D   2   0   3 311   7
##          E   0   0   0   0 351
##
## Overall Statistics
##
##                Accuracy : 0.9704
##                  95% CI : (0.9619, 0.9775)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9625
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9237   0.9766   0.9688   0.9750
## Specificity            0.9836   0.9962   0.9895   0.9927   1.0000
## Pos Pred Value         0.9603   0.9832   0.9516   0.9628   1.0000
## Neg Pred Value         0.9986   0.9819   0.9950   0.9939   0.9944
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2835   0.1790   0.1703   0.1586   0.1790
## Detection Prevalence   0.2953   0.1820   0.1790   0.1647   0.1790
## Balanced Accuracy      0.9900   0.9599   0.9831   0.9808   0.9875
```

RF Accuracy 0.9704

After a head to head comparison, Random Forest method (0.9704) proved to be more accurate than Classification Tree method (0.4768).

Test set Prediction

```
prediction <- predict(fit_rf, newdata=testing)
prediction
```

```
##  [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
```