Peer-graded Assignment: Prediction project

Summary

This report uses data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to generate machine learning models to predict how well people perform a particular activity.

I tried to fit 3 different machine learning models, including the classification tree, random forest, and boosting model, and I found that the random forest model performed the best, with accuracy of 0.9992, followed by the boosting model (accuracy=0.9959). I then used the random forest model to predict the classe of the test data set.

Loading packages

I firstly loaded the packages that would be used in the analyses.

```
library(rattle)
library(caret)
```

Download the data files

1. I downloaded the data from the website and then load the data into r using read.csv.
2. I convert the main outcome "classe" into factor variable.
3. I checked the dimension of the training set and there are 160 different variables.

```
if (!file.exists("pml-training.csv")){
    url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
    download.file(url1, destfile ="./pml-training.csv", method = "curl")
}

if (!file.exists("pml-testing.csv")){
    url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
    download.file(url2, destfile ="./pml-testing.csv", method = "curl")
}

training <- read.csv("pml-training.csv",na.strings=c("","NA"))
testing <- read.csv("pml-testing.csv",na.strings=c("","NA"))

training$classe<-as.factor(training$classe)
dim(training)
```

```
## [1] 19622   160
```

Filter the column variables by removing NAs and near zero variables

Here, I removed the NA variables, the ID variables, or the near zero variables. The total number of variables decreased from 160 to 58.

```
# Remove columns that contain NAs
training <- training[,colSums(is.na(training))==0]

# Remove the columns of user ID
training<-training[,-1]

# Remove near zero variables from both the training and test datasets
training<-training[,!(nearZeroVar(training,saveMetrics = TRUE)$nzv)]
dim(training)
```

```
## [1] 19622    58
```

Split the training data set to sub-training and sub-testing data sets

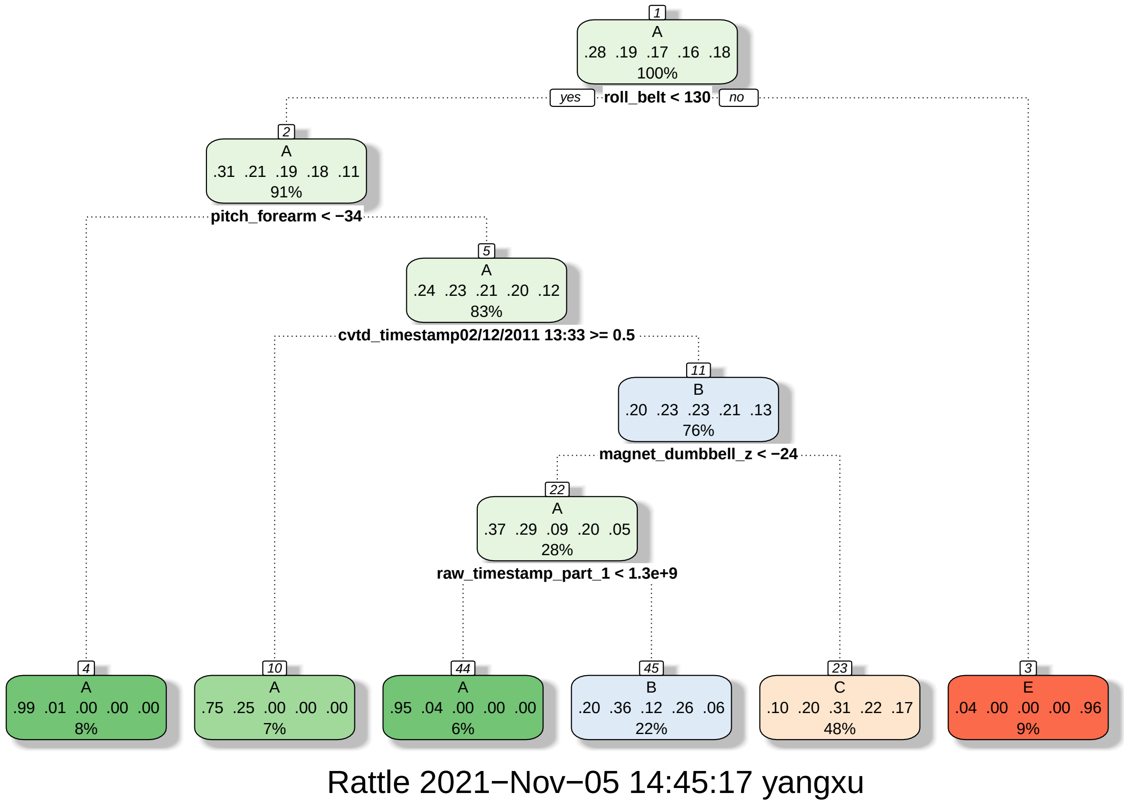
```
inTrain <- createDataPartition(training$classe,p=0.75,list=FALSE)
subtraining <- training[inTrain,]
subtesting <- training[-inTrain,]
```

Fit a classification tree

1. I performed 5-fold cross-validation for all of the machine learning models.
2. By fitting classification free model using the sub-training set, I predict the classe using the sub-testing set, yielding an accuracy of 0.4961, which is not ideal.

```
# For performing 5-fold cross validation
myControl <- trainControl(method = "cv", number = 5)

# Fit a classification tree
modelFit1 <- train(classe~., data=subtraining, method="rpart", trControl=myControl)
fancyRpartPlot(modelFit1$finalModel)
```




```
predict1<-predict(modelFit1, subtesting)
confusionMatrix(predict1, subtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 914 102   3   0   0
##          B 222 371 127 275  69
##          C 238 476 725 529 409
##          D   0   0   0   0   0
##          E  21   0   0   0 423
##
## Overall Statistics
##
##                Accuracy : 0.4961
##                  95% CI : (0.482, 0.5102)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3684
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6552  0.39094   0.8480   0.0000  0.46948
## Specificity            0.9701  0.82478   0.5920   1.0000  0.99475
## Pos Pred Value         0.8970  0.34868   0.3050      NaN  0.95270
## Neg Pred Value         0.8762  0.84948   0.9486   0.8361  0.89283
## Prevalence             0.2845  0.19352   0.1743   0.1639  0.18373
## Detection Rate         0.1864  0.07565   0.1478   0.0000  0.08626
## Detection Prevalence   0.2078  0.21697   0.4847   0.0000  0.09054
## Balanced Accuracy      0.8126  0.60786   0.7200   0.5000  0.73212
```

Fit a random forest model

By fitting random forest model using the sub-training set, I predict the classe using the sub-testing set, yielding an accuracy of 0.9992, which is the best prediction so far.

```
modelFit2 <- train(classe~., data=subtraining, method="rf",trControl=myControl)
predict2<-predict(modelFit2, subtesting)
confusionMatrix(predict2, subtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    1    0    0
##          C    0    0  853    2    0
##          D    0    0    1  802    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9979, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9977   0.9975   1.0000
## Specificity            1.0000   0.9997   0.9995   0.9998   1.0000
## Pos Pred Value         1.0000   0.9989   0.9977   0.9988   1.0000
## Neg Pred Value         1.0000   1.0000   0.9995   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1739   0.1635   0.1837
## Detection Prevalence   0.2845   0.1937   0.1743   0.1637   0.1837
## Balanced Accuracy      1.0000   0.9999   0.9986   0.9986   1.0000
```

Fit a boosting model

1. By fitting boosting model using the sub-training set, I predict the classe using the sub-testing set, yielding an accuracy of 0.9959, which is a little lower than that of the random forest model.
2. Overall, the random forest model is the best model to predict the sub-testing data.

```
modelFit3 <- train(classe~., data=subtraining, method="gbm",verbose=FALSE,trControl=myControl)
predict3<-predict(modelFit3, subtesting)
confusionMatrix(predict3, subtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    1    0    0
##          C    0    1  846    8    0
##          D    0    0    8  796    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9895   0.9900   0.9989
## Specificity            0.9997   0.9997   0.9978   0.9978   1.0000
## Pos Pred Value         0.9993   0.9989   0.9895   0.9888   1.0000
## Neg Pred Value         1.0000   0.9995   0.9978   0.9980   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1725   0.1623   0.1835
## Detection Prevalence   0.2847   0.1933   0.1743   0.1642   0.1835
## Balanced Accuracy      0.9999   0.9988   0.9936   0.9939   0.9994
```

Prediction of the test set using the random forest model

```
predict(modelFit2, 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
```