FinalProject_PML

Synopsis: The goal of this report is to predict the manner in which they did the exercise (classe).

Data loading and processing

Load training and testing data

```
rm(list=ls())
ls()

## character(0)

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
#library(rattle)
library(randomForest)

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.3.1

library(knitr)
### Training data
training <-
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv", header=T,na.strings =c("NA","#DIV/0!"," ") , strip.white =T )
dim(training)

## [1] 19622   160

#summary of classe
summary(training$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607

### Testing data
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", header=T,na.strings =c("NA","#DIV/0!"," ") , strip.white =T )
dim(testing)

## [1]  20 160

#check if the variables in testing data set is the same as in training data set
library(compare)

##
## Attaching package: 'compare'

## The following object is masked from 'package:base':
##
##     isTRUE

all.equal(colnames(subset(training,select=-classe)) ,colnames(testing))

## [1] "Lengths (159, 160) differ (string compare on first 159)"

#The variables in both training and testing datasets are not exactly the same
comparison <- compare(subset(training,select=-classe),colnames(testing),allowAll=TRUE)

## Warning in compareCoerce.integer(comp$tMpartial, comp$tCpartial, comp
## $partialTransform, : NAs introduced by coercion

## Warning in compareCoerce.integer(comp$tMpartial, comp$tCpartial, comp
## $partialTransform, : NAs introduced by coercion

## Warning in compareCoerce.integer(comp$tMpartial, comp$tCpartial, comp
## $partialTransform, : NAs introduced by coercion
```

Exporing and cleaning data

```
#summary of percentage missing by variable
cat("Total number of variables in the training dataset: ", dim(training)[2]-1
)

## Total number of variables in the training dataset:  159

missing.var <- apply(training, 2, function(x)
{sum(is.na(x)/nrow(training)*100)})
head(missing.var)

##                    X            user_name raw_timestamp_part_1
##                    0                    0                    0
```

```
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                    0                    0                    0

#exclude the variables with %missing>90
cat("#Variables with %missing<90% :
",dim(training[,which(missing.var<90)])[2] )

## #Variables with %missing<90% :   60

trainingd <- training[,which(missing.var==0)]
training1 <- trainingd[,-(1:7)]
dim(training1)

## [1] 19622    53

#Testing data set
missingt.var <- apply(testing, 2, function(x)
{sum(is.na(x)/nrow(testing)*100)})
testing2<- testing[,colnames(subset(training1,select=-classe)) ]
all.equal(colnames(subset(training1,select=-classe)) ,colnames(testing2))

## [1] TRUE
```

Training and Validating datasets

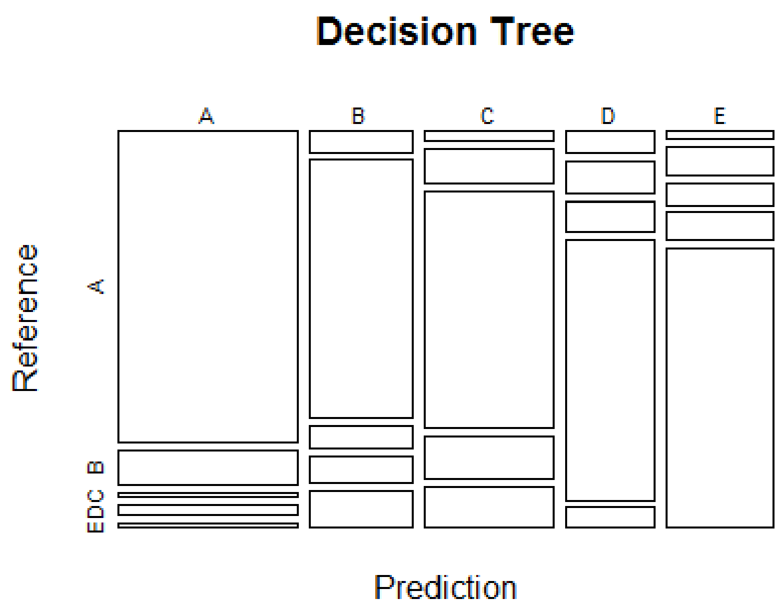
```
set.seed(13234)
inTrain <- createDataPartition(y=training1$classe, p=0.7, list=F)
trainingd2 <- training1[inTrain,]
validat2 <- training1[-inTrain,]
dim(trainingd2)

## [1] 13737    53

table(trainingd2$classe)

##
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

Prediction

Mode1 1: Build prediction model with decision tree

```
set.seed(13234)
modfit1 <-rpart(classe ~ ., data=trainingd2, method="class")
pred.train <- predict(modfit1, trainingd2, type="class")
pred.validate <- predict(modfit1, validat2, type="class")

accut.fit1 <- confusionMatrix(pred.train, trainingd2$classe)
cat("accuracy of training dataset ", round(accut.fit1$overall['Accuracy'],4))

## accuracy of training dataset  0.754
```

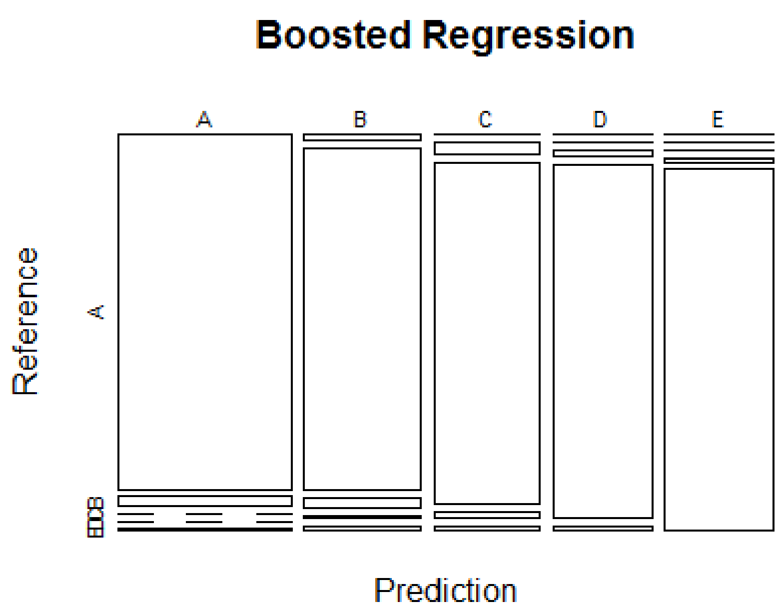
```
accuv.fit1 <- confusionMatrix(pred.validate, validat2$classe)
cat("accuracy of validation dataset: ",
round(accuv.fit1$overall['Accuracy'],4))

## accuracy of validation dataset:  0.7497

plot(accuv.fit1$table,col=accuv.fit1$byClass, main="Decision Tree")
```



Mode1 2:Build prediction model with Stochastic gradient boosting trees (gbm)

```
set.seed(13234)
modfit2 <- train(classe~., data=trainingd2,
method="gbm",trControl=trainControl(method="repeatedcv",number=5,repeats=1),v
erbose=FALSE)

## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel
```

```
## Loaded gbm 2.1.1

## Loading required package: plyr

pred2 <- predict(modfit2, newdata=validat2)
accu.fit2 <-confusionMatrix(pred2, validat2$classe)
#accuray of validation data set
round(accu.fit2$overall['Accuracy'],4)

## Accuracy
##   0.9631

plot(accu.fit2$table, col=accu.fit2$byClass, main="Boosted Regression")
```



Mode1 3: Build prediction model with random forest

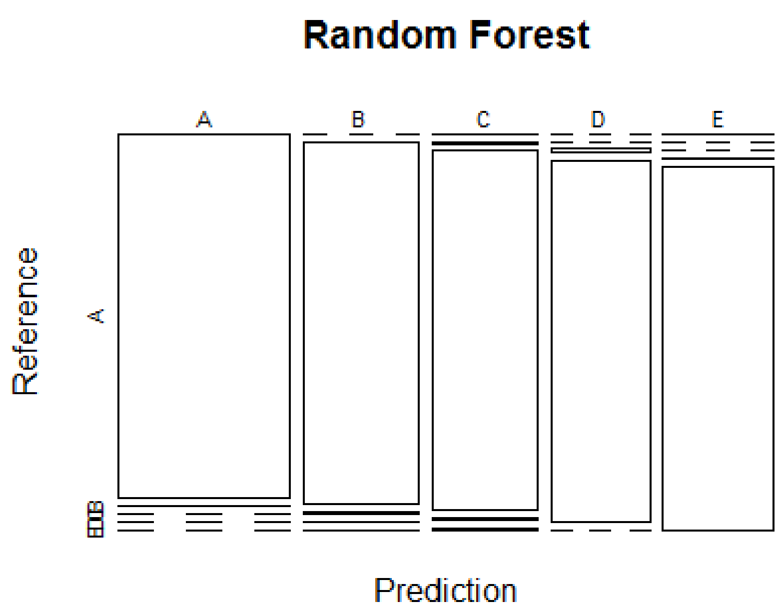
```
set.seed(13234)
fitControl <- trainControl(method='cv', number = 3)
modfit3 <- train(classe ~ ., data=trainingd2,trControl=fitControl,
method='rf', ntree=100)
print(modfit3)

## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9863143  0.9826853
##   27    0.9889347  0.9860008
##   52    0.9843485  0.9801962
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
pred3 <- predict(modfit3, validat2)

accu.fit3 <-confusionMatrix(pred3, validat2$classe)
#accuray of validation dataset
round(accu.fit3$overall['Accuracy'],4)
```

```
## Accuracy
##   0.9934
```

```
plot(accu.fit3$table, col=accu.fit3$byClass, main="Random Forest")
```



compare the accuracy of Decision tree vs Random forest model

```
Accuracy.all <- data.frame(Model = c('DecisionTree', 'GBM', 'RF'),Accuracy =
rbind(accuv.fit1$overall[1], accu.fit2$overall[1], accu.fit3$overall[1] ))
Accuracy.all

##          Model  Accuracy
## 1 DecisionTree 0.7497026
## 2          GBM 0.9631266
## 3           RF 0.9933730

# The Random forest model has the highest accuracy comparing to the other two
models.
```

Prediction

Predict 20 different test cases in testing data with random forest

```
set.seed(13234)
pred.final <- predict(modfit3,
newdata=testing[,colnames(subset(trainingd2,select=-classe))])
# The final prediction result
data.frame( problem_id=testing$problem_id,predicted=pred.final)

##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Summary: Three different models: Decision tree, gbm and random forest are compared, and the best model is Random Forest based on accuracy.Then Random forest is used to predict the classe of the testing data set.