Machine Learning - Project

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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.
The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Building the learning model

We are going to use and compare two of the most extended accurate classifiers: Random forests and Boosting. The model with the highest accuracy will be chosen as the final model.

Loading libraries

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

Loading the data set

```
dataUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

data <- read.csv(url(dataUrl), na.strings=c("NA","#DIV/0!",""))
```

Preprocessing data

Remove (1) constant and almost constant predictors (zero and near-zero predictors) across samples and (2) predictors that have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large

```
NZV <- nearZeroVar(data)
data <- data[, -NZV]
```

Remove the identification variables

```
IDVariables <- names(data) %in% c("X", "user_name", "raw_timestamp_part_1", "raw_time
stamp_part_2", "cvtd_timestamp", "num_window")
data <- data[!IDVariables]
```

Remove variables with more than 50% missing values

```
varNA <- sapply(colnames(data), function(x) if(sum(is.na(data[, x])) > 0.50*nrow(data))
    {return(TRUE)
}else{ return(FALSE) })

data <- data[, !varNA]
```

Principal Component Analysis

In both models we are going to preprocess the variables using Principal Component Analysis). Therefore, we will remove variables highly correlated, including a new set of multivariate variables that explain as much variance and information as possible.

Cross validation

Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. Cross-validation will be performed by spliting our data set randomly into Training (60% of the data) and Testing (40% of the data) sets.

```
inTrain <- createDataPartition(y = data$classe, p=0.6, list=FALSE)
training <- data[inTrain, ]
testing <- data[-inTrain, ]
```

Random Forest

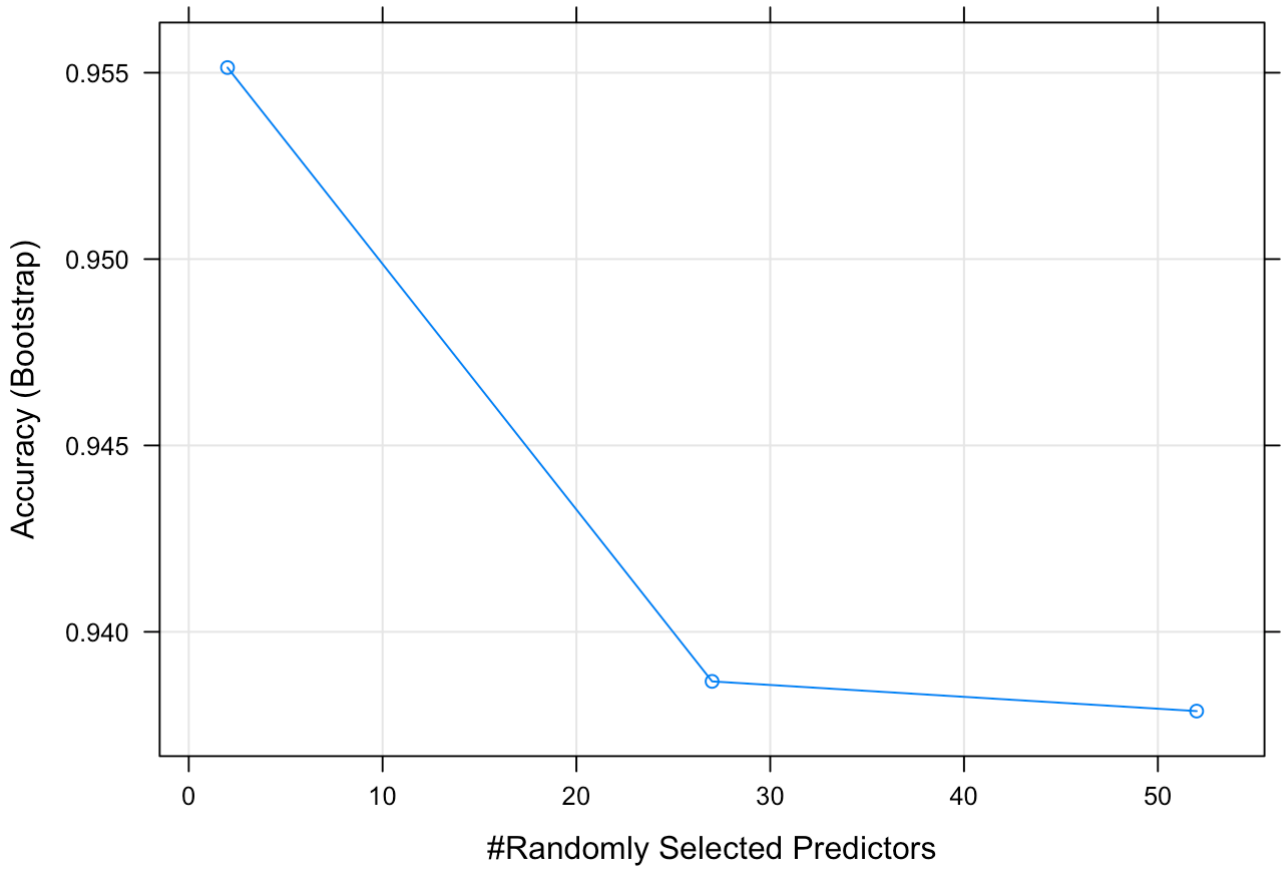
```
set.seed(1234)
#Fit the "Random Forest" Model with PCA preprocess
modFitRF <- train(classe ~., data = training, preProcess = "pca", method = "rf")
```

Boosting

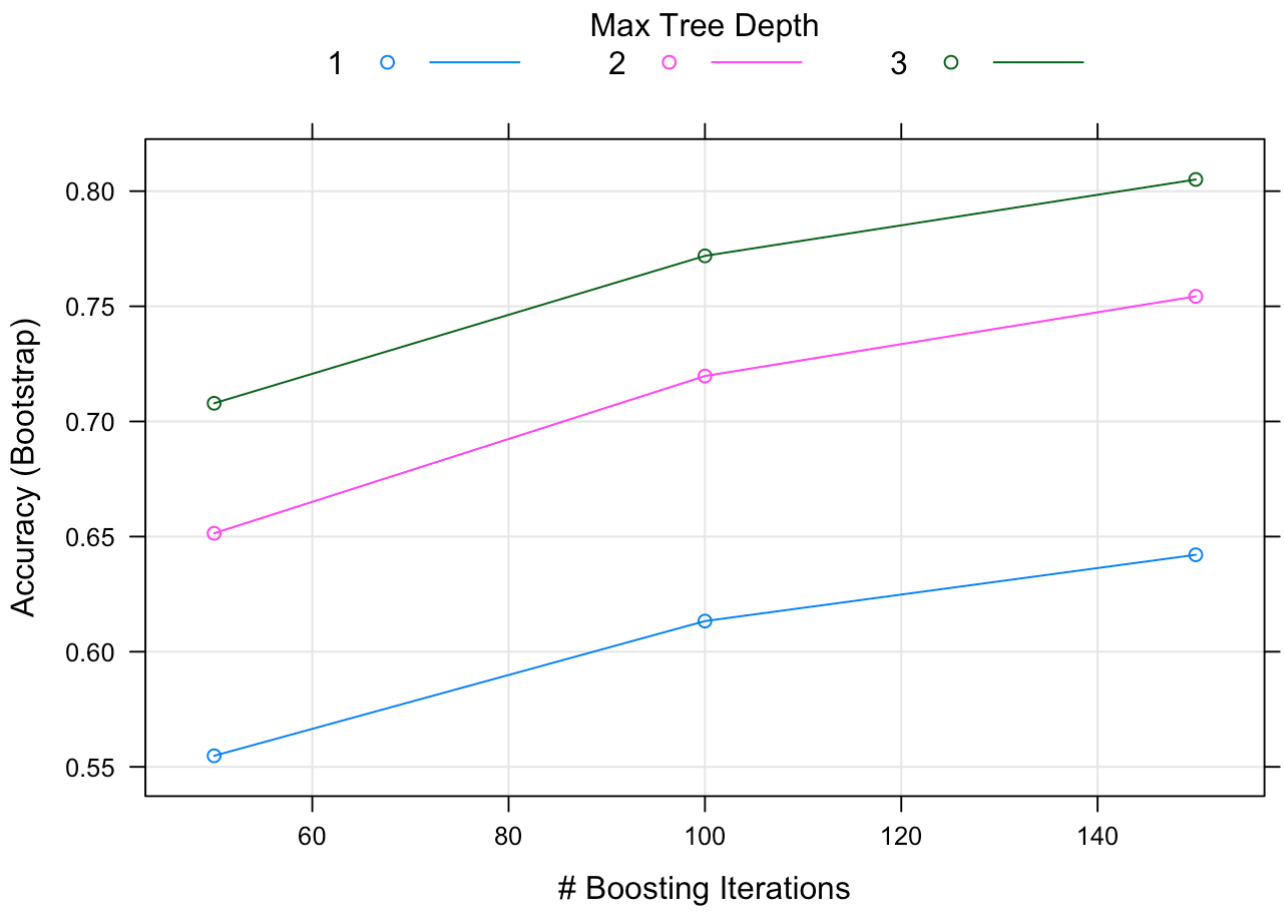
```
#Fit the "Boosting with trees" Model and with PCA preprocess
modFitBoost <- train(classe ~., data = training, preProcess = "pca", method = "gbm")
```

Plots

```
plot(modFitRF)
```




```
plot(modFitBoost)
```



Accuracy and out-of-sample error

Accuracy is the proportion of correct classified observation over the total sample of the test data set. The expected out-of-sample error will correspond to (1 - Accuracy) in the cross validation data. Therefore, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations in the test data set.

```
## Random Forest
predictionRF <- predict(modFitRF, testing)
cmRF <- confusionMatrix(predictionRF, testing$classe)
print(cmRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2209   29    2    4    0
##          B    8 1457   21    5    4
##          C    4   29 1333   61   15
##          D   11    0    9 1215   15
##          E    0    3    3    1 1408
##
## Overall Statistics
##
##                Accuracy : 0.9715
##                  95% CI : (0.9675, 0.975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9639
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9897   0.9598   0.9744   0.9448   0.9764
## Specificity            0.9938   0.9940   0.9832   0.9947   0.9989
## Pos Pred Value         0.9844   0.9746   0.9244   0.9720   0.9951
## Neg Pred Value         0.9959   0.9904   0.9945   0.9892   0.9947
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2815   0.1857   0.1699   0.1549   0.1795
## Detection Prevalence   0.2860   0.1905   0.1838   0.1593   0.1803
## Balanced Accuracy      0.9917   0.9769   0.9788   0.9697   0.9877
```

```
## Boosting
predictionBoost <- predict(modFitBoost, testing)
cmBoost <- confusionMatrix(predictionBoost, testing$classe)
print(cmBoost)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2008  176   61   24   35
##          B   45 1101  104   41  105
##          C   59  159 1113  175   81
##          D  103   37   56 1007   86
##          E   17   45   34   39 1135
##
## Overall Statistics
##
##                Accuracy : 0.8111
##                  95% CI : (0.8023, 0.8197)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7609
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8996   0.7253   0.8136   0.7830   0.7871
## Specificity            0.9473   0.9534   0.9268   0.9570   0.9789
## Pos Pred Value         0.8715   0.7887   0.7013   0.7812   0.8937
## Neg Pred Value         0.9596   0.9353   0.9593   0.9575   0.9533
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2559   0.1403   0.1419   0.1283   0.1447
## Detection Prevalence   0.2937   0.1779   0.2023   0.1643   0.1619
## Balanced Accuracy      0.9235   0.8393   0.8702   0.8700   0.8830
```

Conclusion

Results show that Random Forest algorithm performs better than Boosting, with a 97.15% of accuracy.

Prediction on the data set

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

dataTest <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

predictTest <- predict(modFitRF, dataTest, method = "class")
predictTest
```

```
##  [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
```