Coursera - Machine Learning - Course Project

peterkaj

27 01 2017

Executive Summary

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. The goal in this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise ("classe" variable in the training set).

After i have prepared the data and reduced unrelated features, i have trained/fitted several models and picked the best for predicting the "classe". Then i have used this model for predicting the "classe" on the 20 samples of the test data set.

- Trained Models: Tree, LDA, RF (RandomForest), GBM (Generalized Boosted Model)
- Best Model: GBM with an expected out-of-sample-error of 0.34%
- 20 samples classification outcome: "B A B A A E D B A A B C B A E E A B B B" with a prediction accuracy of 100%

Overview

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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 i will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

Loading the data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data (means the new data, which has to be classified) are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

If the data sets are actually not available in the local directory "./data", a download puts these two files into this directory (which is created, if it is not present).

```r
library(caret); library(randomForest); library(rpart); library(rattle); library(MASS); library(gbm)

if (!file.exists("./data/pml-training.csv")){
        if (!dir.exists("./data")) {dir.create("./data")}
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(fileUrl, destfile="./data/pml-training.csv", method = "curl")
}
traindata <- read.csv("./data/pml-training.csv")
# Test data set (20 samples of new data)
if (!file.exists("./data/pml-testing.csv")){
        if (!dir.exists("./data")) {dir.create("./data")}
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(fileUrl, destfile="./data/pml-testing.csv", method = "curl")
}
newdata_full <- read.csv("./data/pml-testing.csv")
dim(traindata); dim(newdata_full)
```

```
## [1] 19622   160

## [1]  20 160
```

Feature Selection

Goal of this part is the selection of features which have a relation to activity and removing the others from the data sets. Many of the features in the test data set contains only NA and are worthless for prediction. I will truncate this features without any loss of information in both of the datasets.

```r
# Select only Testdata variables(columns) without NA
nonaNew <- subset(newdata_full, select = !is.na(newdata_full[1,]))
# Select only Traindata variables(columns) where Testdata is available -> Traindata set without NA
nonaTrain <- subset(traindata, select = names(nonaNew[1:dim(nonaNew)[2]-1]))
nonaTrain$classe <- traindata$classe
dim(nonaTrain); dim(nonaNew)
```

```
## [1] 19622    60

## [1] 20 60
```

With this procedure i am able to reduce the number of variables/features from 159 to **59**.

Some of the remaining features doesn´t look like having a relation to activity and will also be truncated from data sets:

- X … only an index and contains no information about activity
- cvtd_timestamp, raw_timestamp_part_1, raw_timestamp_part_2 … timestamps with no relation to activity
- num_window, new_window … relation to timing, but none with activity

```r
redTrain <- subset(nonaTrain, select = -c(X, cvtd_timestamp, raw_timestamp_part_1, raw_timestamp_part_2
newdata <- subset(nonaNew, select = -c(X, cvtd_timestamp, raw_timestamp_part_1, raw_timestamp_part_2, nu
```

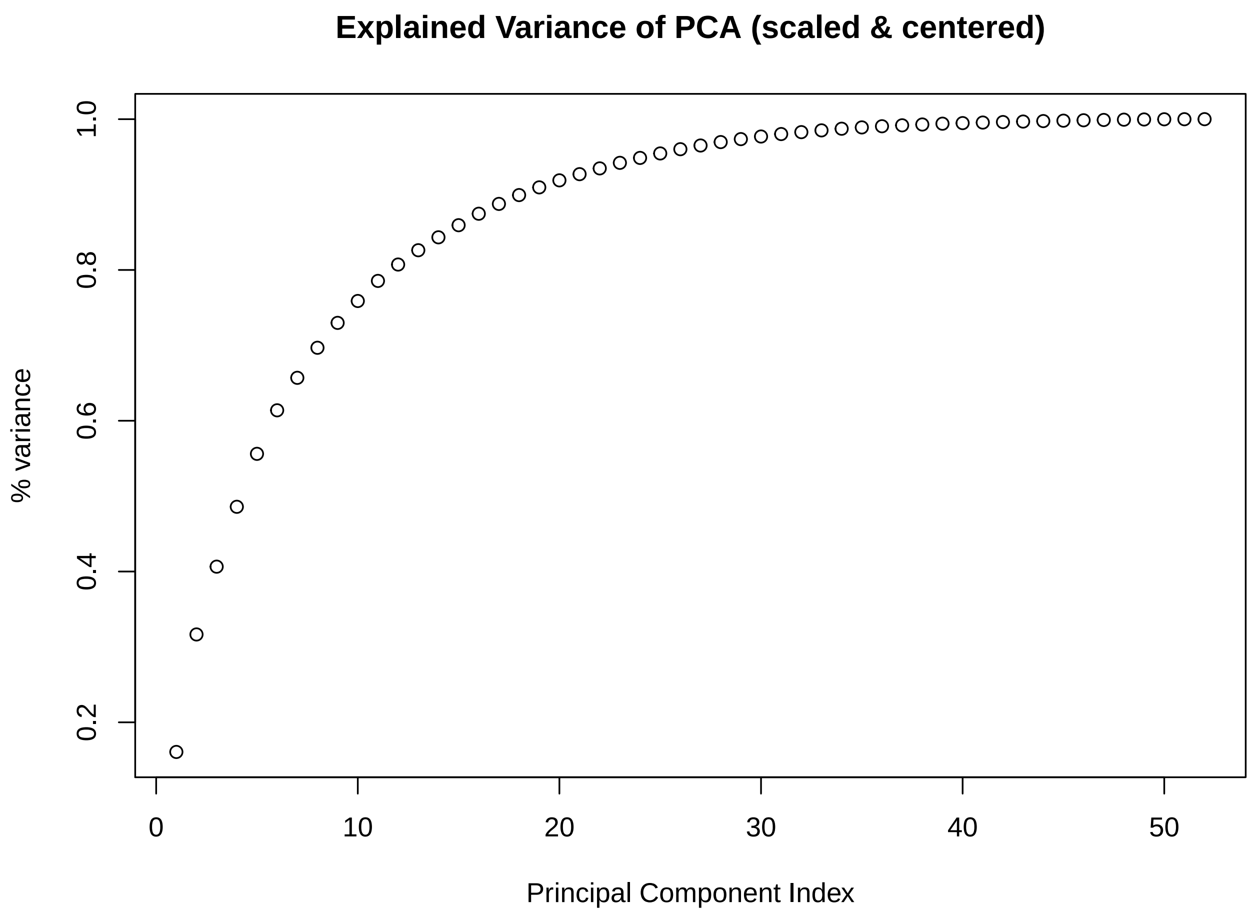
This procedure leads to a further reduction from 59 to **53** variables/features.

```r
nzVar <- nearZeroVar(redTrain, saveMetrics = TRUE)
```

A check for Near Zero Variables/Features results in **0** Variables, which leads to no further reduction of features.

For a possible other reduction of features i will perform a PCA (Principial Component Analysis) on the centered and scaled training set.

```
prComp <- prcomp(redTrain[-c(1,54)], center=TRUE, scale. = TRUE)
prComp_var <- cumsum(prComp$sdev^2) / sum(prComp$sdev^2)
plot(prComp_var, main = "Explained Variance of PCA (scaled & centered)", ylab = "% variance", xlab = "P
```



The cumulative sum of the variance explained through the several components of the PCA shows a monontone rising curve, where some of the variables could be reduced without significant loss in explaining the variance. But if i would proceed with PCA variables, i will loose the interpretation of features. So i decide in a first approach to proceed **without** PCA.

Creation of Training and Test data for model fitting and cross validation

For cross validation and model testing some test data is necessary. This is done via a split of the training data into 2 data sets - traininig and testing - in a relation of 70% to 30%. The split is done via random subsampling.

```
# Create a training and testing data set
set.seed(36826)
inTrain <- createDataPartition(y=redTrain$classe, p=0.7, list=FALSE)
training <- redTrain[inTrain,]
testing <- redTrain[-inTrain,]
dim(training); dim(testing); dim(newdata)
```

```
## [1] 13737    54

## [1] 5885   54

## [1] 20 54
```

Model

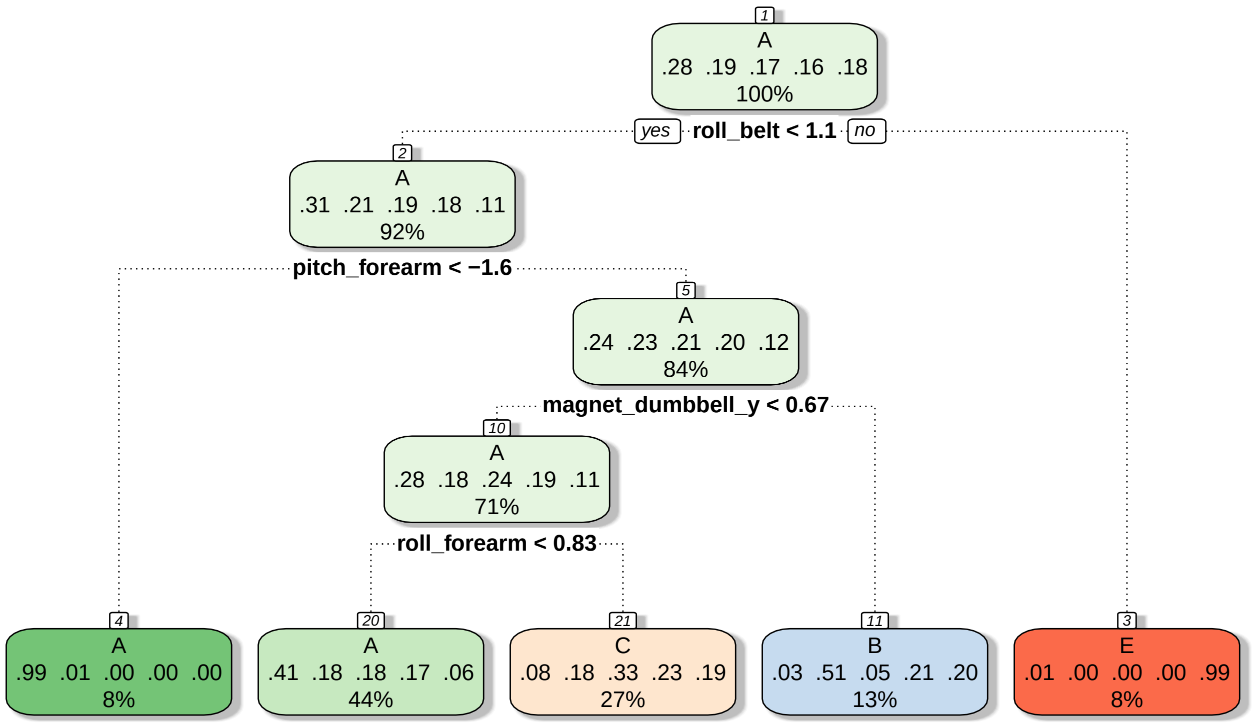
I will train (based on the training data set) several models to predict the manner in which the people did the exercise (classification into 5 different classes within the variable "classe"). Afterwards i will perform predictions on the test data set (cross validation) with every model and pick the model with the best results to perform a prediction based on the new data set.

Trees

```r
# Tree Model
mod_tree <- train(classe ~ ., data=training, method = "rpart", preProcess=c("center", "scale"))
pred_tree <- predict(mod_tree, testing)
acc_tree <- prettyNum(confusionMatrix(pred_tree, testing$classe)$overall)
confusionMatrix(pred_tree, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1517  473  466  422  146
##          B   33  386   28  195  135
##          C  118  280  532  347  285
##          D    0    0    0    0    0
##          E    6    0    0    0  516
##
## Overall Statistics
##
##                Accuracy : 0.5014
##                  95% CI : (0.4886, 0.5143)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3489
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9062  0.33889   0.5185   0.0000  0.47689
## Specificity            0.6421  0.91761   0.7880   1.0000  0.99875
## Pos Pred Value         0.5017  0.49678   0.3406      NaN  0.98851
## Neg Pred Value         0.9451  0.85258   0.8857   0.8362  0.89446
## Prevalence             0.2845  0.19354   0.1743   0.1638  0.18386
## Detection Rate         0.2578  0.06559   0.0904   0.0000  0.08768
## Detection Prevalence   0.5138  0.13203   0.2654   0.0000  0.08870
## Balanced Accuracy      0.7742  0.62825   0.6533   0.5000  0.73782
```

```r
fancyRpartPlot(mod_tree$finalModel)
```



Rattle 2017−Jan−31 18:58:21 JPE

LDA (Linear Discriminant Analysis)

```r
# LDA Model
mod_lda <- train(classe ~ ., data=training, method = "lda", preProcess=c("center", "scale"))
pred_lda <- predict(mod_lda, testing)
acc_lda <- prettyNum(confusionMatrix(pred_lda, testing$classe)$overall)
confusionMatrix(pred_lda, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction     A    B    C    D    E
##          A 1394  180  103   64   35
##          B   47  737   81   36  135
##          C  118  121  714  104   63
##          D  115   46  112  741  103
##          E    0   55   16   19  746
##
## Overall Statistics
##
##                Accuracy : 0.7361
##                  95% CI : (0.7246, 0.7473)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.6658
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8327   0.6471   0.6959   0.7687   0.6895
## Specificity            0.9093   0.9370   0.9164   0.9236   0.9813
## Pos Pred Value         0.7849   0.7114   0.6375   0.6634   0.8923
## Neg Pred Value         0.9319   0.9171   0.9345   0.9532   0.9335
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2369   0.1252   0.1213   0.1259   0.1268
## Detection Prevalence   0.3018   0.1760   0.1903   0.1898   0.1421
## Balanced Accuracy      0.8710   0.7920   0.8062   0.8461   0.8354
```

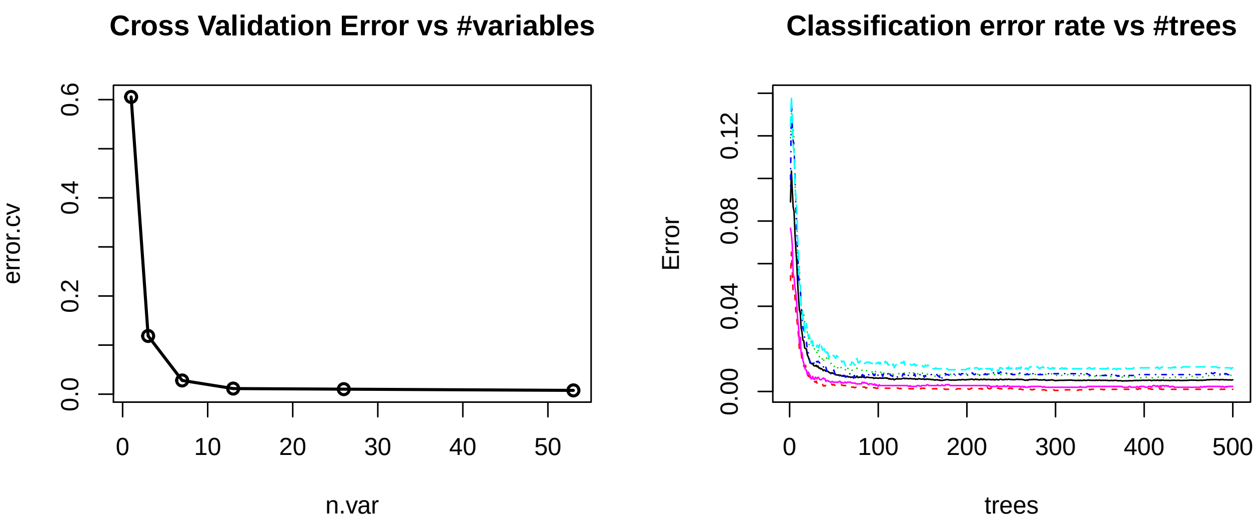
Random Forests

```
# Random Forest Model
mod_rf <- randomForest(classe ~ ., data=training, prox=TRUE, preProcess=c("center", "scale"))
result <- rfcv(training[,-54], training$classe)
pred_rf <- predict(mod_rf, testing)
acc_rf <- prettyNum(confusionMatrix(pred_rf, testing$classe)$overall)
confusionMatrix(predict(mod_rf, testing), testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    2 1131    1    0    0
##          C    0    1 1025    4    2
##          D    0    0    0  957    3
##          E    0    0    0    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9961
##                  95% CI : (0.9941, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9951
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9990   0.9927   0.9954
## Specificity            0.9983   0.9994   0.9986   0.9994   0.9994
## Pos Pred Value         0.9958   0.9974   0.9932   0.9969   0.9972
## Neg Pred Value         0.9995   0.9983   0.9998   0.9986   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1742   0.1626   0.1830
```

```
## Detection Prevalence   0.2853   0.1927   0.1754   0.1631   0.1835
## Balanced Accuracy      0.9986   0.9962   0.9988   0.9961   0.9974
```

```
par(mfrow=c(1,2))
with(result, plot(n.var, error.cv, type="o", lwd=2, main="Cross Validation Error vs #variables"))
plot(mod_rf, main="Classification error rate vs #trees")
```




```
par(mfrow=c(1,1))
```

The cross validation error in relation to the number of variables/features is a monotone decreasing curve, which means there is no overfitting with too many variables. For decreasing the cv error significantly only a few variables are nessessary. This outcome could be expected from the Principial Component Analyses above. So model tuning could be done with a tradeoff between number of variables and accuracy with respect to calculation time. I decide to proceed with all variables and the highest accuracy, spending some more calculation time. The classification error rate doesn´t decrease significantly above ~150 trees, thats why the tuning parameter "ntree" could be reduced, which decreases calculation time.

```
imp <- as.data.frame(mod_rf$importance)
imp$variable <- rownames(imp)
imp <- imp[order(imp[1], decreasing=TRUE),]
head(imp,10); tail(imp,10)
```

```
##                    MeanDecreaseGini          variable
## roll_belt                  852.3471         roll_belt
## yaw_belt                   619.5830          yaw_belt
## pitch_forearm              520.9174     pitch_forearm
## magnet_dumbbell_z          520.6034 magnet_dumbbell_z
## magnet_dumbbell_y          471.1790 magnet_dumbbell_y
## pitch_belt                 470.7864        pitch_belt
## roll_forearm               410.6556      roll_forearm
## magnet_dumbbell_x          324.8465 magnet_dumbbell_x
## roll_dumbbell              296.3383     roll_dumbbell
## accel_dumbbell_y           290.0169  accel_dumbbell_y
```

```
##                      MeanDecreaseGini            variable
## accel_belt_y                 88.55415        accel_belt_y
## accel_belt_x                 80.91573        accel_belt_x
## gyros_belt_y                 77.50304        gyros_belt_y
## total_accel_forearm          77.24254 total_accel_forearm
```

```
## total_accel_arm             71.94322     total_accel_arm
## gyros_belt_x                68.99250        gyros_belt_x
## gyros_forearm_z             58.92406     gyros_forearm_z
## gyros_dumbbell_z            57.62413    gyros_dumbbell_z
## gyros_forearm_x             55.68034     gyros_forearm_x
## gyros_arm_z                 42.94302         gyros_arm_z
```

The importance of the variables is listed above (Top10, Last10). It is a measure for the total decrease in node impuriies from splitting on the variable, averaged over all trees and measured by the Gini Index.

GBM Generalized Boosted Model

Beyond this report i have done some model performance tuning (different model parameter settings) to find a suitable tradeoff between accuracy and calculation time. Afterwards i selected the best parameter set and fitted the model. I have attached the tuning parameters without guarantee of reproducibility.

```r
# GBM Generalized Boosted Model
mod_gbm <- gbm(classe ~ ., data = training, distribution = "multinomial", n.trees=1000, shrinkage = 0.2
                    interaction.depth = 10, cv.folds=0, verbose=FALSE, n.cores=4)
pred_gbm <- predict(object=mod_gbm, newdata = testing[,-54], n.trees = gbm.perf(mod_gbm, plot.it = FALSE
```

```
## Using OOB method...
```

```r
pred_gbm_cat <- as.factor(apply(pred_gbm, 1, which.max)) # Classification = Class with highest probabil
levels(pred_gbm_cat) <- c("A","B","C","D","E") #Prediction output as Factor variable
acc_gbm <- prettyNum(confusionMatrix(pred_gbm_cat, testing$classe)$overall)
confusionMatrix(pred_gbm_cat, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    5    0    0    0
##          B    4 1132    0    0    0
##          C    0    2 1026    1    1
##          D    0    0    0  959    3
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9939   1.0000   0.9948   0.9963
## Specificity            0.9988   0.9992   0.9992   0.9994   0.9992
## Pos Pred Value         0.9970   0.9965   0.9961   0.9969   0.9963
## Neg Pred Value         0.9990   0.9985   1.0000   0.9990   0.9992
```

```
## Prevalence            0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate        0.2838   0.1924   0.1743   0.1630   0.1832
## Detection Prevalence  0.2846   0.1930   0.1750   0.1635   0.1839
## Balanced Accuracy     0.9982   0.9965   0.9996   0.9971   0.9977
```

```
# gbm() Performance tuning
# Accuracy n.trees shrinkage interaction.depth cv.folds calc_time[sec]
#   0.7293    250    0.001          5             3     1.51769 min
#   0.8226   1000    0.001          5             3     5.83640 min
#   0.9730   1000    0.01           5             3     5.58111 min
#   0.9096   3000    0.001          5             3    17.28626 min
#   0.7325    250    0.001          5             0    47.4728 sec
#   0.5412    250    0.001          1             0    12.2535 sec
#   0.6386    250    0.001          2             0    21.5739 sec
#   0.6780    250    0.001          3             0    31.9862 sec
#   0.6783    250    0.001          3             3     1.00142 min
#   0.8425    250    0.01           3             0    31.7111 sec
#   0.9592    250    0.05           3             0    30.5457 sec
#   0.9806    250    0.1            3             0    31.8826 sec
#   0.9895    250    0.2            3             0    30.4324 sec
#   0.9567    250    0.5            3             0    31.8826 sec
#   0.9934    250    0.2            7             0     1.03756 min
#   0.9951    250    0.2           10             0     1.45154 min
#   0.9952    250    0.2           15             0     2.00576 min
#   0.9922   1000    0.2            3             0     1.99028 min
#   0.9951   1000    0.2            5             0     3.04000 min
#   0.9964   1000    0.2            7             0     3.97441 min
#   0.9975   1000    0.2           10             0     5.67936 min  #Selected Parameterset
```

Model Selection

```
test_sum <- rbind(tree=acc_tree, LDA=acc_lda, RandomForest=acc_rf, GBM=acc_gbm)
as.data.frame(test_sum)[1:4]
```

```
##                 Accuracy     Kappa AccuracyLower AccuracyUpper
## tree           0.5014444 0.3488883     0.4885869     0.5143004
## LDA            0.7361088 0.6658321      0.724648      0.747335
## RandomForest 0.9960918 0.9950561     0.9941415     0.9975209
## GBM            0.9966015 0.9957013     0.9947562     0.9979229
```

For predicting the 20 samples in the test data i will choose the best fitted model with the highest accuracy and the lowest out-of-sample-error, which leads to the GBM (Generalized Boosted Model) with an estimated out-of-sample-error of 0.33985%.

Prediction and Submission of test data

Now i am using the GBM Model to predict the 20 test samples.

```
pred20 <- predict(object=mod_gbm, newdata = newdata[,-54], n.trees = gbm.perf(mod_gbm, plot.it = FALSE)
```

```
## Using OOB method...
```

```r
pred20_cat <- as.factor(apply(pred20, 1, which.max)) # Classification = Class with highest probability
levels(pred20_cat) <- c("A","B","C","D","E") #Prediction output as Factor variable
pred20_cat
```

```
##  [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
```

The predicted outcome is “B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B”, which was also my submission to the “Course Project Prediction Quiz” with the outcome of 100% success, which means an prediction accuracy of 100% :-)

End of Report