Practical Machine Learning: Prediction

N. Lakhani

27 January 2018

Executive Summary

This project involves analysis of wearable fitness trackers. “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, my goal is to 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)

1. Loading libraries

The libraries needed: caret, rpart, randomForest, reshape2, AppliedPredictiveModeling are loaded in workspace.

2. Data cleaning and preparation

Input data from url’s provided. The training data is split (70:30) into training & validation data (to be used as out of sample data) for cross validation. The following observations are made regarding the data:

- Since data from belt,forearm,arm, and dumbell are to be examined - filter out the rest
- There are dummy variables with no measurements for each observation, but these are summary stats for each time sliding window.
- The ‘X’ variable (row number) and ‘new window’ (marker for summary data), timestamp are not relevant in the current analysis, hence we drop columns (1:5)
- Several variables have near zero values (NZV) & ’NA’s. Variables with NZV’s & NA’s over 70% are dropped. Interestingly even with 70%, we find that all variables with any NA’s are dropped.

```r
set.seed(12345)
testing <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',na.strings=c(
training <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',na.strings=


# determine nearzero variables for elimination
nzv <- nearZeroVar(training,saveMetrics = TRUE)
training <- training[,nzv$nzv==FALSE]

nzv <- nearZeroVar(testing,saveMetrics = TRUE)
testing <- testing[,nzv$nzv==FALSE]


# partition training data into training (70% and validation (30%)
inTrain <- createDataPartition(training$classe,p=0.7,list=FALSE)
```

```
mytraining <- training[inTrain,]
myvalidation <- training[-inTrain,]


# identify variables with > 70% NA's, the remaining variables are eliminated. Also variables 1:5 are dr
NAvars <- sapply(mytraining, function (x) mean(is.na(x))) > 0.70
mytraining <- mytraining[,NAvars == F]
mytraining <- mytraining[,-(1:5)]

NAvars <- sapply(myvalidation, function (x) mean(is.na(x))) > 0.70
myvalidation <- myvalidation[,NAvars == F]
myvalidation <- myvalidation[,-(1:5)]

NAvars <- sapply(training, function (x) mean(is.na(x))) > 0.70
training <- training[,NAvars == F]
training <- training[,-(1:5)]


NAvars <- sapply(testing, function (x) mean(is.na(x))) > 0.70
testing <- testing[,NAvars == F]
testing <- testing[,-(1:5)]

filter <- grepl('forearm|belt|arm|dumbell',names(mytraining))
mytraining <- mytraining[filter,]
myvalidation <- myvalidation[filter,]
testing <- testing[filter,]
```

3. Cleaned data and features

The cleaned training data set has:

- 54 variables including **classe** with 9,926 observations.The testing dataset has 20 observations and 54 variables.
- The absence of NA's in any variables is also validated.
- The corr plot on the cleaned dataset indicates that very few of the variables have strong correlation (values close to -1 or 1, ignoring the squares along the diagonal in the plot, which shows cor for variables to themselves). *Also looking at the cor values and not too strong relationship, I feel there is no need for PCA and further variable elimination.

```
print(names(mytraining))
```

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```
print(names(myvalidation))
```

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```
dim(mytraining)
```

```
## [1] 9926   54
```

```
colSums(is.na(mytraining))
```

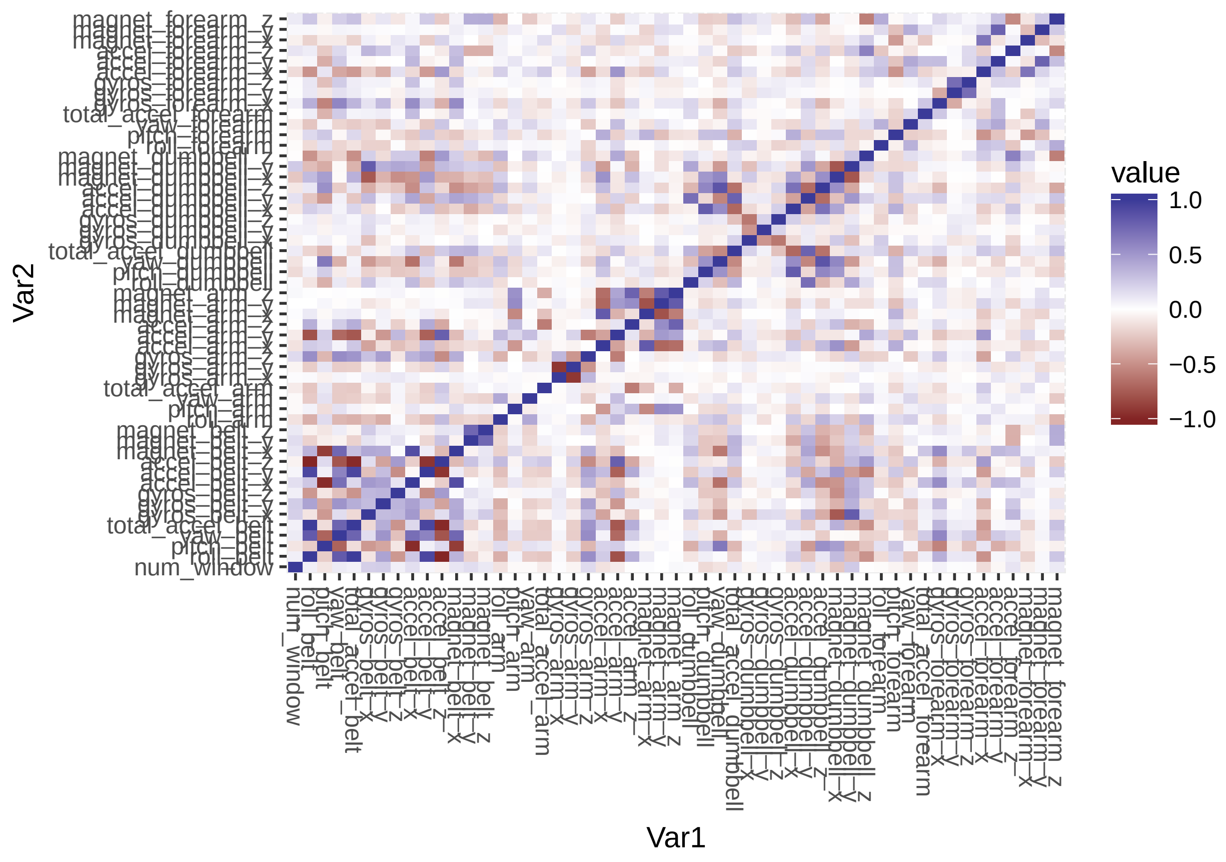
```
##           num_window            roll_belt           pitch_belt
##                    0                    0                    0
##             yaw_belt     total_accel_belt         gyros_belt_x
##                    0                    0                    0
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##                    0                    0                    0
##         accel_belt_y         accel_belt_z        magnet_belt_x
##                    0                    0                    0
##        magnet_belt_y        magnet_belt_z             roll_arm
##                    0                    0                    0
##            pitch_arm              yaw_arm      total_accel_arm
##                    0                    0                    0
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##                    0                    0                    0
##          accel_arm_x          accel_arm_y          accel_arm_z
##                    0                    0                    0
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##                    0                    0                    0
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##                    0                    0                    0
## total_accel_dumbbell     gyros_dumbbell_x     gyros_dumbbell_y
##                    0                    0                    0
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##                    0                    0                    0
##    accel_dumbbell_z     magnet_dumbbell_x    magnet_dumbbell_y
##                    0                    0                    0
##   magnet_dumbbell_z          roll_forearm        pitch_forearm
##                    0                    0                    0
##         yaw_forearm   total_accel_forearm      gyros_forearm_x
##                    0                    0                    0
##     gyros_forearm_y       gyros_forearm_z      accel_forearm_x
##                    0                    0                    0
##     accel_forearm_y       accel_forearm_z     magnet_forearm_x
##                    0                    0                    0
##    magnet_forearm_y      magnet_forearm_z               classe
##                    0                    0                    0
```

```
unique(mytraining$classe)
```

```
## [1] A B C D E
## Levels: A B C D E
```

```
# get corelation vlues and print plot to visually check strength of relationships

cormat <- cor(mytraining[sapply(mytraining, is.numeric)])
cormat <- melt(cormat)
qplot(x=Var1, y=Var2, data=cormat, fill=value, geom="tile") + scale_fill_gradient2(limits=c(-1, 1)) + t
```



4. Building the model and parameters

I have taken the following approach:

- The variable being predicted **classe** is a factor with 5 levels, so this is a classification problem.
- The 3 models evaluated for best accuracy are ++a) Decision Trees (rpart), ++b) Stochastic gradient boosting trees (gbm), ++c) Random forest decision trees (rf).
- I do a 3-fold cross validation using the function train to build the model

4a. Decision trees model results

The first model we explore is Decision Trees. As can be seen visually from the plot and confusion matrix (for classe values):

- the accuracy is quite low at 52% and
- the confusion matrix is highly populated across the matrix indicating many false postives/negatives; not a great model.

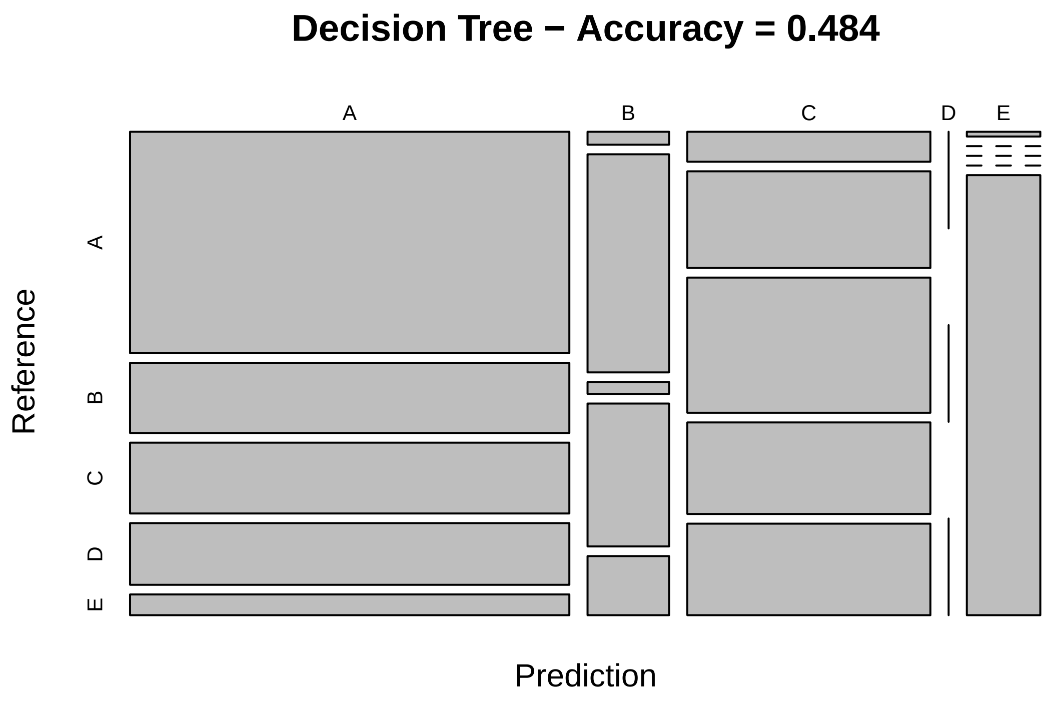
```
# define control parameters to be cross validation & create rpart model
mod_control <- trainControl(method='cv',number=3)
fit_rpart <- train(classe~.,data=mytraining,method='rpart',trControl=mod_control)

# predict classe values on validation data
pred_rpart <- predict(fit_rpart,newdata=myvalidation)
cm_tree <- confusionMatrix(pred_rpart,myvalidation$classe)
cm_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1110  352  355  309  104
##          B   12  203   11  133   55
##          C   83  268  375  254  254
##          D    0    0    0    0    0
##          E    4    0    0    0  369
##
## Overall Statistics
##
##                Accuracy : 0.4839
##                  95% CI : (0.4688, 0.499)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3255
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9181  0.24666  0.50607   0.0000  0.47187
## Specificity            0.6318  0.93845  0.75527   1.0000  0.99885
## Pos Pred Value         0.4978  0.49034  0.30389      NaN  0.98928
## Neg Pred Value         0.9510  0.83842  0.87869   0.8363  0.89350
## Prevalence             0.2844  0.19360  0.17431   0.1637  0.18396
## Detection Rate         0.2611  0.04775  0.08821   0.0000  0.08680
## Detection Prevalence   0.5246  0.09739  0.29028   0.0000  0.08774
## Balanced Accuracy      0.7750  0.59255  0.63067   0.5000  0.73536
```

```
plot(cm_tree$table, fill = cm_tree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(cm_tree$overall['Accuracy'], 3)))
```

```
## Warning: In mosaicplot.default(x, xlab = xlab, ylab = ylab, ...) :
##  extra argument 'fill' will be disregarded
```



4b. Random Forest model results

The random forest model shows a significant improvement over the rpart model. As can be seen in output below, both from the plot and confusion matrix (for classe values): *The accuracy is above 99%.* The confusion matrix is quite clean with the diagonal of the matrix having majority of the matches. *The accuracy also peaks at about 27 predictors. The variables are listed in the order of importance.

```
set.seed(12345)
fit_rf <- train(classe ~ ., data=mytraining,method='rf',trControl=mod_control)
fit_rf$finalModel
```

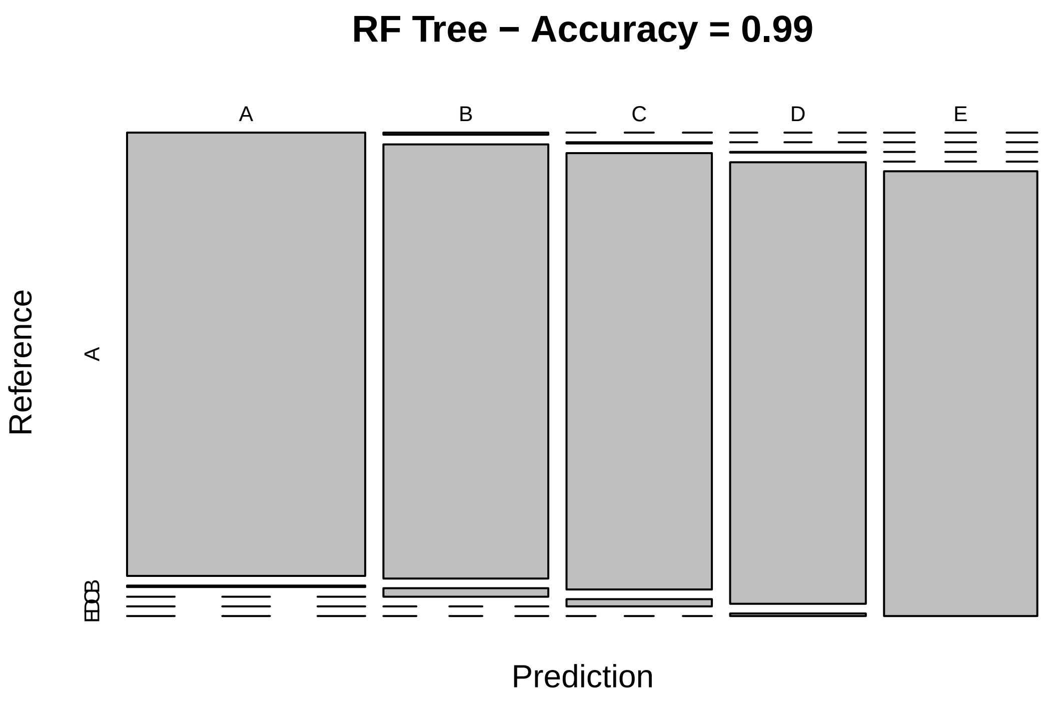
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
```

```
##
##         OOB estimate of  error rate: 0.33%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 2822    3    0    0    0 0.001061947
## B    5 1912    3    0    0 0.004166667
## C    0    5 1719    2    0 0.004055620
## D    0    0    9 1619    0 0.005528256
## E    0    0    0    6 1821 0.003284072
```

```
pred_rf <- predict(fit_rf, myvalidation)
cm_rf <- confusionMatrix(pred_rf, myvalidation$classe)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1205    4    0    0    0
##          B    4  817   16    0    0
##          C    0    2  724   12    0
##          D    0    0    1  684    4
##          E    0    0    0    0  778
##
## Overall Statistics
##
##                Accuracy : 0.9899
##                  95% CI : (0.9864, 0.9927)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9872
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9967   0.9927   0.9771   0.9828   0.9949
## Specificity            0.9987   0.9942   0.9960   0.9986   1.0000
## Pos Pred Value         0.9967   0.9761   0.9810   0.9927   1.0000
## Neg Pred Value         0.9987   0.9982   0.9952   0.9966   0.9988
## Prevalence             0.2844   0.1936   0.1743   0.1637   0.1840
## Detection Rate         0.2835   0.1922   0.1703   0.1609   0.1830
## Detection Prevalence   0.2844   0.1969   0.1736   0.1621   0.1830
## Balanced Accuracy      0.9977   0.9934   0.9865   0.9907   0.9974
```

```
plot(cm_rf$table, fill = cm_rf$byClass,
     main = paste("RF Tree - Accuracy =",
                  round(cm_rf$overall['Accuracy'], 3)))
```



4c. gbm model results

The gbm model shows: *Significant improvement over the rpart model and is slightly better than the rf model.* As can be seen in the output below, both from the plot and confusion matrix (for classe values), the accuracy is 98%. Of the 53 predictors, only 12 have influence

```
fit_gbm <- train(classe ~.,data=mytraining,method='gbm',trControl=mod_control,verbose=FALSE)
fit_gbm$finalModel
```

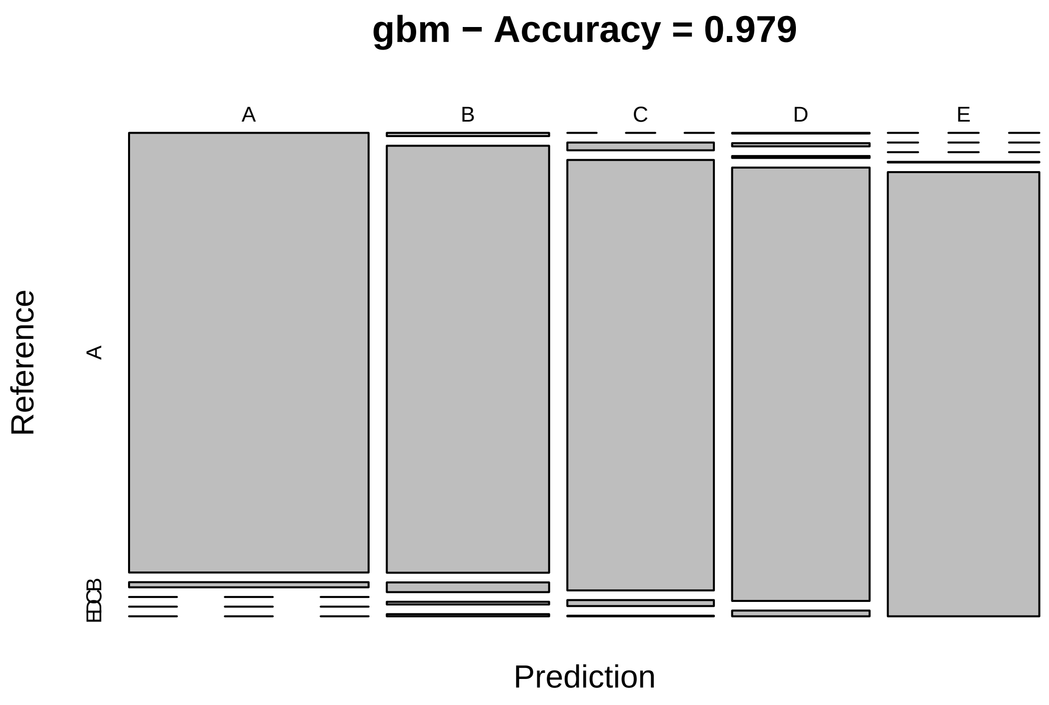
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

```
pred_gbm <- predict(fit_gbm, myvalidation)
cm_gbm <- confusionMatrix(pred_gbm, myvalidation$classe)
cm_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1202   14    0    0    0
##          B    6  791   18    5    4
##          C    0   13  720   10    1
##          D    1    5    3  680    9
##          E    0    0    0    1  768
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9788
##                  95% CI : (0.974, 0.9829)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9732
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9942   0.9611   0.9717   0.9770   0.9821
## Specificity            0.9954   0.9904   0.9932   0.9949   0.9997
## Pos Pred Value         0.9885   0.9600   0.9677   0.9742   0.9987
## Neg Pred Value         0.9977   0.9907   0.9940   0.9955   0.9960
## Prevalence             0.2844   0.1936   0.1743   0.1637   0.1840
## Detection Rate         0.2828   0.1861   0.1694   0.1600   0.1807
## Detection Prevalence   0.2861   0.1938   0.1750   0.1642   0.1809
## Balanced Accuracy      0.9948   0.9757   0.9824   0.9860   0.9909
```

```r
plot(cm_gbm$table, fill = cm_gbm$byClass,
     main = paste("gbm - Accuracy =",
                  round(cm_gbm$overall['Accuracy'], 3)))
```



4. Final model and evaluation

Based on the results, I have picked the **rf** model as the final model with accuracy of 0.996. The top 5 features in order of inluence are shown below along with the accuracy predictions for the test dataset

```r
AccuracyResults <- data.frame(
  Model = c('CART', 'GBM', 'RF'),
  Accuracy = rbind(cm_tree$overall[1], cm_gbm$overall[1], cm_rf$overall[1])
)
print(AccuracyResults)
```

```
##   Model  Accuracy
## 1  CART 0.4838861
## 2   GBM 0.9788285
## 3    RF 0.9898847
```

5. Out of sample error

- The optimum rf model (mtry=27) has accuracy of 0.988. Hence in sample error is 0.012% (1-.988).
- Out of sample error is calculated used the myvalidation dataset below as the number of classe matches/total observations in the prediction

```r
fit_rf
```

```
## Random Forest
##
## 9926 samples
##   53 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 6617, 6618, 6617
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9889179  0.9859789
##   27    0.9935521  0.9918429
##   53    0.9897239  0.9870000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```r
predvalid <- predict(fit_rf,newdata=myvalidation)
length(predvalid)
```

```
## [1] 4251
```

```r
oos_acc <- sum(predvalid == myvalidation$classe)/length(predvalid)
oos_error <- 1- oos_acc
paste0 ('Out of sample error ',round(oos_error,5))
```

```
## [1] "Out of sample error 0.01012"
```

6. Prediction on test cases and output submission

```
imp <- varImp(fit_rf)
imp$importance$Overall <- sort(imp$importance$Overall, decreasing=TRUE)
featureDF <- data.frame(FeatureName=row.names(imp$importance), Importance=imp$importance$Overall)


print(featureDF[1:5,])
```

```
##        FeatureName Importance
## 1       num_window  100.00000
## 2        roll_belt   65.86912
## 3       pitch_belt   41.85809
## 4         yaw_belt   34.19024
## 5 total_accel_belt   31.12868
```

```
predtest <- predict(fit_rf,newdata=testing)
predtest <- as.character(predtest)

pml_write <- function (x) {
  n <- length(x)
  for (i in 1:n) {
    filename <- paste0('problem_id_',i,'.txt')
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
  }

pml_write(predtest)
predtest
```

```
##  [1] "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B"
## [18] "B" "B"
```