Coursera - Practical Machine Learning - Course Project

Executive summary

This project analyses the data provided in the Weight Lifting Exercise Dataset. Its goal is to build a model to predict how the observed subjects performed physical exercises (five levels from A to E).

Building the prediction model

Loading the data We assume that the data files are in the working directory.

```
train.raw <- read.csv("pml-training.csv", na.strings=c("NA",""), header=TRUE)
test.raw <- read.csv("pml-testing.csv", na.strings=c("NA",""), header=TRUE)

dim(train.raw)
```

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

We can see that the raw training data contains 19622 rows and 160 columns.

```
library(caret)
```

Load the required libraries

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

```
library(rpart)
set.seed(123) #set seed for reproducible results
```

Choosing relevant predictors Our model should predict the outcome "classe" (five levels from A to E). Having analyzed the raw data, we conclude that many columns have missing values (NA or are blank). In fact, almost all the values in these columns are missing. Some of these columns contain information which can be derived from other columns (for example, columns containing the standard deviation and variance) and are often left blank. We decided to exclude these variables from the list of potential predictors.

Also, we can see that some columns contain non-numeric data (such as 'user-name', 'new_window', etc.) or information which might not be relevant (such as 'timestamp'). We decided to exclude these columns too.

Below you will find the list of predictors which we decided to use (the last variable is 'classe' and is the outcome):

```
significant.features <- c("roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt",
                          "gyros_belt_x", "gyros_belt_y", "gyros_belt_z",
                          "accel_belt_x", "accel_belt_y", "accel_belt_z",
                          "magnet_belt_x", "magnet_belt_y", "magnet_belt_z",
                          "roll_arm", "pitch_arm", "yaw_arm",
                          "gyros_arm_x", "gyros_arm_y", "gyros_arm_z",
```

```
                           "accel_arm_x", "accel_arm_y", "accel_arm_z",
                           "magnet_arm_x", "magnet_arm_y", "magnet_arm_z",
                           "roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell",
                           "gyros_dumbbell_x", "gyros_dumbbell_y", "gyros_dumbbell_z",
                           "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z",
                           "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_dumbbell_z",
                           "roll_forearm", "pitch_forearm", "yaw_forearm",
                           "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm_z",
                           "accel_forearm_x", "accel_forearm_y", "accel_forearm_z",
                           "magnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z",
                           "classe")

training <- train.raw[,significant.features]
dim(training)
```

```
## [1] 19622    50
```

Splitting the training data set into two data sets for training and validation - cross validation

In order to perform cross validation we decided to split the training data set into two data sets: for training (60% of the data) and validation (testing inside the training data set) (40%).

```
inTrain = createDataPartition(training$classe, p = 0.6)[[1]]
training = training[inTrain,]
validation = train.raw[-inTrain,]
```

Choosing the prediction method

Taking into account that our models contains 49 predictors, we decided to choose the **Random Forest method** which is particlularly efficient in dealing with a large number of predictors.

Training the prediction model

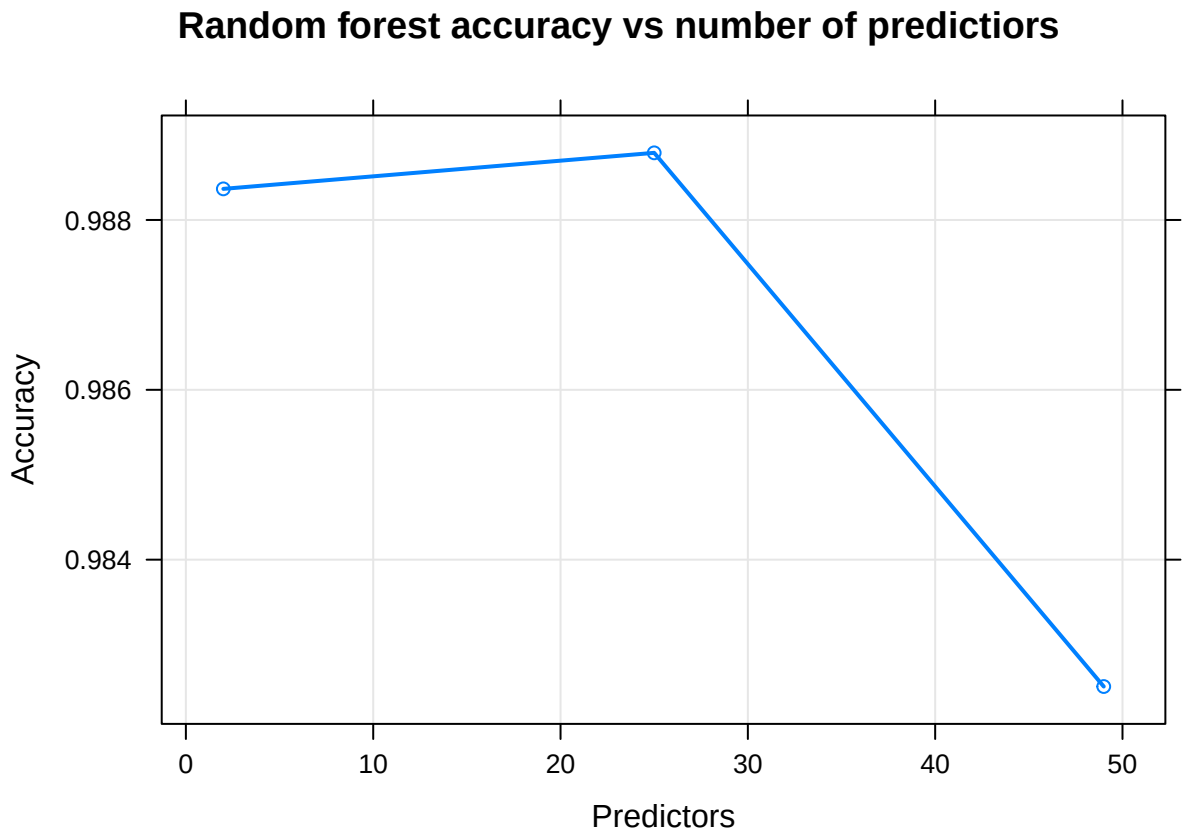
Using the Random Forest method, we decided to preprocess data (center and scale) and use training control (cross validation with number of iterations equal to 4) for efficiency (without these measures the model took far too long to be built on the entire training set):

```
modFit <- train(classe ~ ., method="rf", preProcess=c("center", "scale"),
                trControl=trainControl(method = "cv", number = 4), data=training)
```

```
## Loading required package: randomForest
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

Random Forest accuracy

```r
plot(modFit, log = "y", lwd = 2, main = "Random forest accuracy vs number of predictiors",
     xlab = "Predictors",  ylab = "Accuracy")
```




```r
print(modFit)
```

```
## Random Forest
##
## 11776 samples
##    49 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 8832, 8833, 8830, 8833
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     1         1      0.002        0.003
##   20    1         1      0.001        0.002
##   50    1         1      0.003        0.004
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 25.
```

We can see (on the above plot and in the model print out) that a **model with the number of variables per level (mtry) = 25 was selected as the optimal one**.

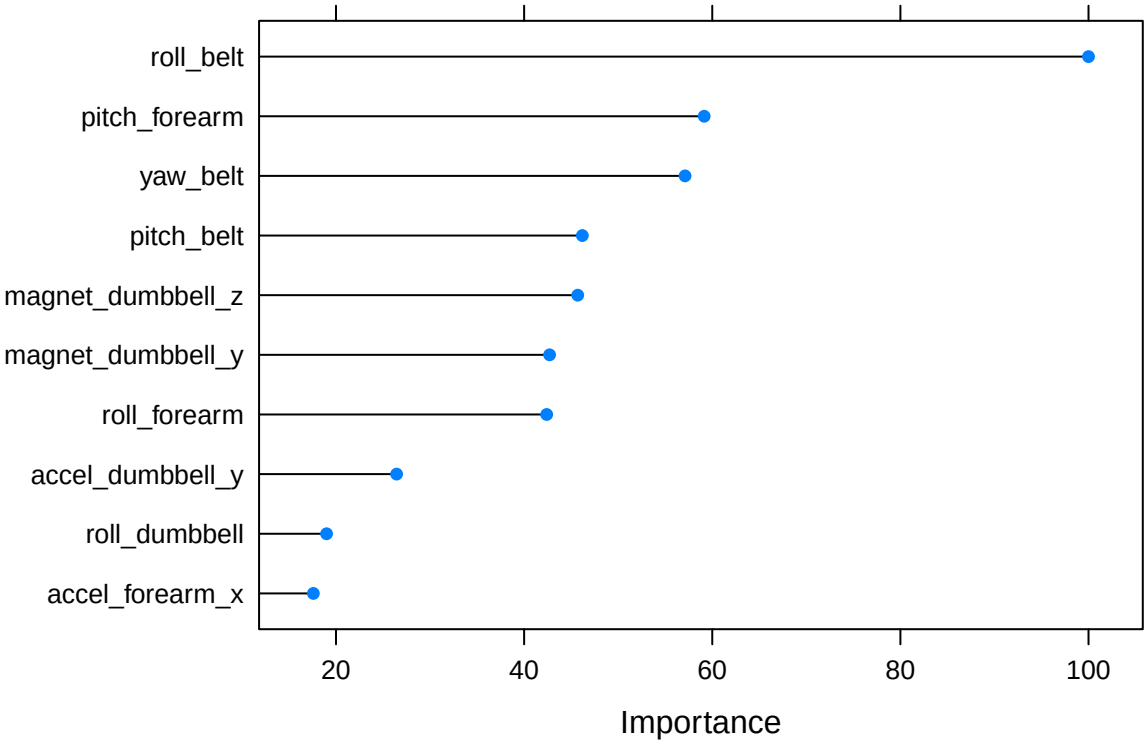
Estimating variable importance

```
vI <- varImp(modFit)
vI
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 49)
##
##                    Overall
## roll_belt           100.00
## pitch_forearm        59.14
## yaw_belt             57.11
## pitch_belt           46.19
## magnet_dumbbell_z    45.70
## magnet_dumbbell_y    42.71
## roll_forearm         42.40
## accel_dumbbell_y     26.44
## roll_dumbbell        19.00
## accel_forearm_x      17.59
## magnet_dumbbell_x    17.05
## accel_dumbbell_z     16.35
## magnet_belt_z        16.08
## accel_belt_z         14.02
## magnet_forearm_z     13.77
## magnet_belt_y        12.16
## yaw_arm              10.27
## gyros_belt_z         10.26
## magnet_belt_x         9.43
## roll_arm              9.00
```

Top 10 most important variables

```
plot(vI, main = "Top 10 Variable Importance", top = 10)
```



Applying the model to the validation set

Let us apply our model to the validation set in order to check its performance.

```
validation <- validation[,significant.features]
#exclude the last column 'classe' which is the outcome
predictions <- predict(modFit,newdata=validation[,-ncol(validation)])
cm <- confusionMatrix(predictions, validation$classe)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   11    1    0    0
##          B    4 1504    9    0    0
##          C    0    3 1345   16    4
##          D    0    0   13 1269    5
##          E    0    0    0    1 1433
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.989, 0.993)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.989
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.998    0.991    0.983    0.987    0.994
## Specificity             0.998    0.998    0.996    0.997    1.000
## Pos Pred Value          0.995    0.991    0.983    0.986    0.999
## Neg Pred Value          0.999    0.998    0.996    0.997    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.171    0.162    0.183
## Detection Prevalence    0.285    0.193    0.174    0.164    0.183
## Balanced Accuracy       0.998    0.994    0.990    0.992    0.997
```

Accuracy of the final Model

```
round(cm$overall['Accuracy'], 4)
```

```
## Accuracy
##   0.9915
```

The accuracy is approximately 99%.

Out of sample error

Error of the model on the new data.

```
1 - 0.9915
```

```
## [1] 0.0085
```

The out of sample error is approximately 1%

Prediction on the test set

```
testing <- test.raw[,significant.features[-length(significant.features)]]
predictions.testing <- predict(modFit, newdata=testing)
predictions.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
```

Conclusion

The Random Forest algorithm is rather efficient for predicting how the observed subjects performed physical exercises based on the selected features and using an unseen data set. The **accuracy on the validation (unseen) data is approximately 99% and the out of sample error is approximately 1%**. The model also performed well on the testing data which was submitted as the second part of the course project.