Assignment 2

Chew Yoong, Chang

December 27, 2015

Data

The training data for this project are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv]

The test data are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv]

Citation

The data for this project come from this source: [http://groupware.les.inf.puc-rio.br/har].

Goal

The goal is to find a model that can predicht the classes below based on the sensor data of an activity.

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes." [1] Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction. Two models will be tested using decision tree and random forest algorithms. The model with the highest accuracy will be chosen as our final model.

Cross-validation Cross-validation will be performed by subsampling our training data set randomly without replacement into 2 subsamples: subTraining data (75% of the original Training data set) and subTesting data (25%). Our models will be fitted on the subTraining data set, and tested on the subTesting data. Once the most accurate model is choosen, it will be tested on the original Testing data set.

Expected out-of-sample error The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Reasons for my choices

Our outcome variable “classe” is an unordered factor variable. Thus, we can choose our error type as 1-accuracy. We have a large sample size with N= 19622 in the Training data set. This allow us to divide our Training sample into subTraining and subTesting to allow cross-validation. Features with all missing values will be discarded as well as features that are irrelevant. All other features will be kept as relevant

variables. Decision tree and random forest algorithms are known for their ability of detecting the features that are important for classification [2]. Feature selection is inherent, so it is not so necessary at the data preparation phase. Thus, there won't be any feature selection section in this report.

Loading data

Below the code for loading the data (which was already downloaded to my harddrive).

```
library("dplyr")
library("caret")
library("tidyr")
library("rpart.plot")
library("randomForest")

set.seed(54356)

pml.training <- read.csv("pml-training.csv", na.strings = c("NA","#DIV/0!", ""), dec = ".")
```

Cleaning data

The data needs to be cleaned before it can be used for modelling. I tried several different ways of cleanng the data before I came up with the following steps:

1. Remove new_window == yes observations because these seem to be aggragates of other column.
2. Remove the first columns (id, timestamps, subject name) because they are not usefull in predicting.
3. Remove all columns with NA values.

```
x <- pml.training %>% filter(new_window == "no")
x <- x[8:length(x)]
x <- x[ , ! apply(x ,2 ,function(x) any(is.na(x)) ) ]
```

Creating training and testset for cross validation

The assignment provides a training and testset, however, the testset is not really a testset, but more a submission set. To be able to validate the model the provided trainingset will be split in a training and testset for the modelling.

```
inTrain <- createDataPartition(y=x$classe,
                               p=0.6, list=FALSE)
trainingset <- subset(x[inTrain,])
testset <- subset(x[-inTrain,])
```

Cross validation

The default resampling scheme for the caret train function is bootstrap. I have used custom settings instead by setting the below `trainControl`.

The out of sample error should be higher than the in sample error because the the model is based on the training set and will therefor most likely have a slightly worst performance on the testset. This will be shown further in the project.

Selecting variables

First I made a model on a small part of the training set (for speed). Then I selected the 20 most important variables with `varImp` and run the model again. I repeated this, meanwhile balancing the accurancy and number of variables. With 10 variables I still got very good accuracy and speed on the model with the full training set.

A look at the Data

The variable "classe" contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the subTraining data set and compare one another.

```
dim (trainingset)
```

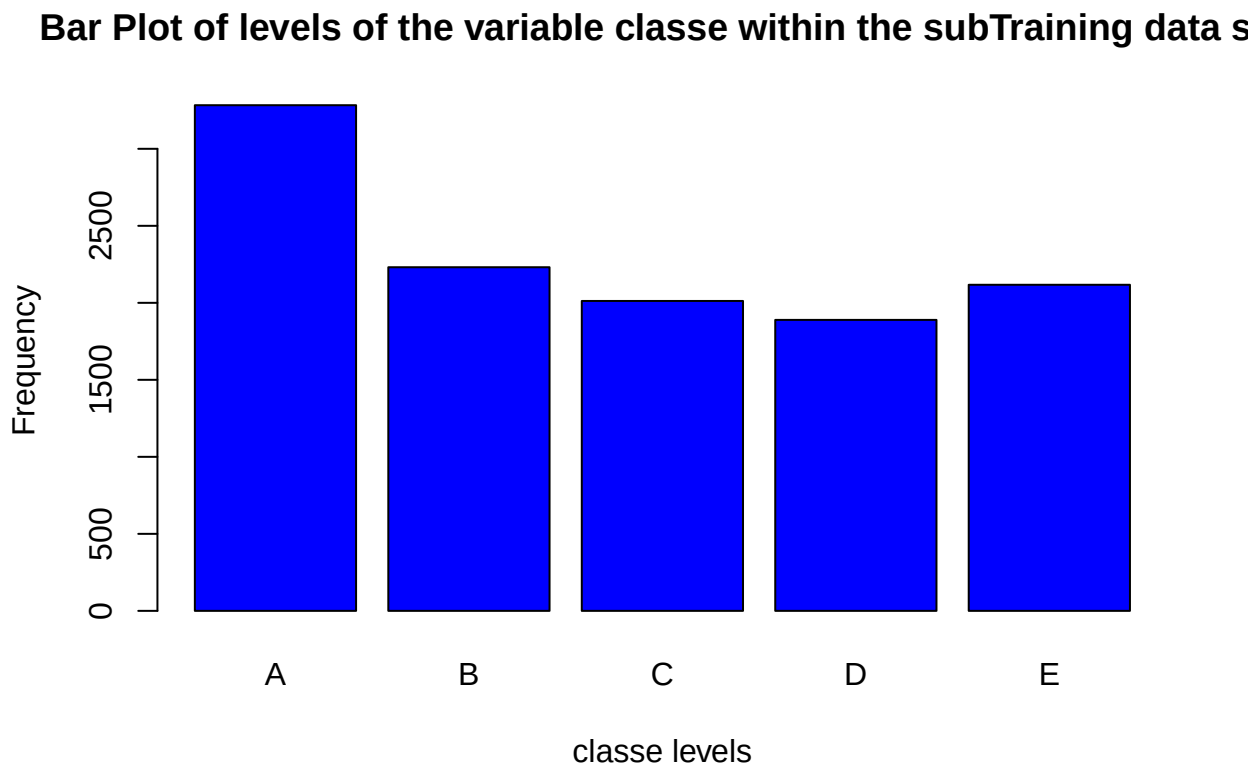
```
## [1] 11532    53
```

```
head(trainingset)
```

```
##    roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x
## 1       1.41       8.07    -94.4                3         0.00
## 5       1.48       8.07    -94.4                3         0.02
## 6       1.45       8.06    -94.4                3         0.02
## 7       1.42       8.09    -94.4                3         0.02
## 8       1.42       8.13    -94.4                3         0.02
## 10      1.45       8.17    -94.4                3         0.03
##    gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 1          0.00        -0.02          -21            4           22
## 5          0.02        -0.02          -21            2           24
## 6          0.00        -0.02          -21            4           21
## 7          0.00        -0.02          -22            3           21
## 8          0.00        -0.02          -22            4           21
## 10         0.00         0.00          -21            4           22
##    magnet_belt_x magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm
## 1             -3           599          -313     -128      22.5    -161
## 5             -6           600          -302     -128      22.1    -161
## 6              0           603          -312     -128      22.0    -161
## 7             -4           599          -311     -128      21.9    -161
## 8             -2           603          -313     -128      21.8    -161
## 10            -3           609          -308     -128      21.6    -161
##    total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x
## 1               34        0.00        0.00       -0.02        -288
## 5               34        0.00       -0.03        0.00        -289
## 6               34        0.02       -0.03        0.00        -289
## 7               34        0.00       -0.03        0.00        -289
## 8               34        0.02       -0.02        0.00        -289
## 10              34        0.02       -0.03       -0.02        -288
##    accel_arm_y accel_arm_z magnet_arm_x magnet_arm_y magnet_arm_z
## 1          109        -123         -368          337          516
## 5          111        -123         -374          337          506
## 6          111        -122         -369          342          513
## 7          111        -125         -373          336          509
## 8          111        -124         -372          338          510
```

```
## 10         110        -124         -376          334          516
##    roll_dumbbell pitch_dumbbell yaw_dumbbell total_accel_dumbbell
## 1       13.05217      -70.49400    -84.87394                   37
## 5       13.37872      -70.42856    -84.85306                   37
## 6       13.38246      -70.81759    -84.46500                   37
## 7       13.12695      -70.24757    -85.09961                   37
## 8       12.75083      -70.34768    -85.09708                   37
## 10      13.33034      -70.85059    -84.44602                   37
##    gyros_dumbbell_x gyros_dumbbell_y gyros_dumbbell_z accel_dumbbell_x
## 1                 0            -0.02                0             -234
## 5                 0            -0.02                0             -233
## 6                 0            -0.02                0             -234
## 7                 0            -0.02                0             -232
## 8                 0            -0.02                0             -234
## 10                0            -0.02                0             -235
##    accel_dumbbell_y accel_dumbbell_z magnet_dumbbell_x magnet_dumbbell_y
## 1                47             -271              -559                293
## 5                48             -270              -554                292
## 6                48             -269              -558                294
## 7                47             -270              -551                295
## 8                46             -272              -555                300
## 10               48             -270              -558                291
##    magnet_dumbbell_z roll_forearm pitch_forearm yaw_forearm
## 1                -65         28.4         -63.9        -153
## 5                -68         28.0         -63.9        -152
## 6                -66         27.9         -63.9        -152
## 7                -70         27.9         -63.9        -152
## 8                -74         27.8         -63.8        -152
## 10               -69         27.7         -63.8        -152
##    total_accel_forearm gyros_forearm_x gyros_forearm_y gyros_forearm_z
## 1                   36            0.03            0.00           -0.02
## 5                   36            0.02            0.00           -0.02
## 6                   36            0.02           -0.02           -0.03
## 7                   36            0.02            0.00           -0.02
## 8                   36            0.02           -0.02            0.00
## 10                  36            0.02            0.00           -0.02
##    accel_forearm_x accel_forearm_y accel_forearm_z magnet_forearm_x
## 1              192             203            -215              -17
## 5              189             206            -214              -17
## 6              193             203            -215               -9
## 7              195             205            -215              -18
## 8              193             205            -213               -9
## 10             190             205            -215              -22
##    magnet_forearm_y magnet_forearm_z classe
## 1               654              476      A
## 5               655              473      A
## 6               660              478      A
## 7               659              470      A
## 8               660              474      A
## 10              656              473      A
```

```r
plot(trainingset$classe, col="blue", main="Bar Plot of levels of the variable classe within the subTrai
```

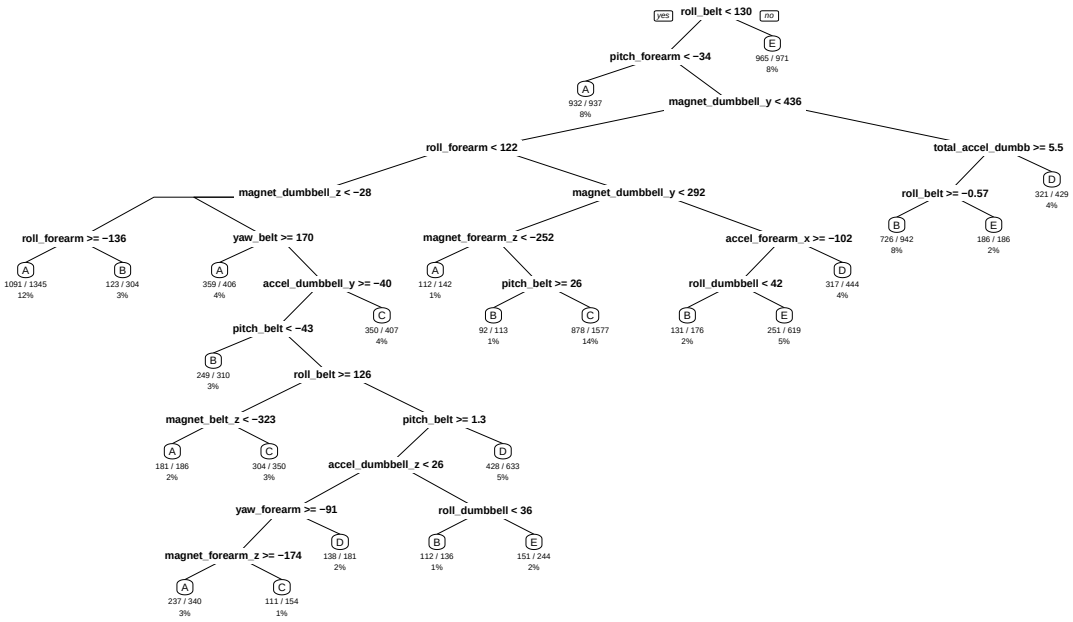


First prediction model :- Decision Tree Model

```r
model1 <- rpart(classe ~ ., data=trainingset, method="class")

# Predicting:
prediction1 <- predict(model1, testset, type = "class")

# Plot of the Decision Tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Test results on our subTesting data set:

```
confusionMatrix(prediction1, testset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1924  211   18   63   25
##          B   76  900   77  104  104
##          C   55  156 1075  186  180
##          D   93  102   85  801   79
##          E   40  118   85  104 1023
##
## Overall Statistics
##
##                Accuracy : 0.7448
##                  95% CI : (0.7349, 0.7545)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.6771
##  Mcnemar's Test P-Value : < 0.00000000000000022
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8793   0.6052   0.8022   0.6367   0.7250
## Specificity            0.9423   0.9417   0.9090   0.9441   0.9447
## Pos Pred Value         0.8585   0.7137   0.6507   0.6905   0.7467
## Neg Pred Value         0.9515   0.9086   0.9561   0.9300   0.9385
## Prevalence             0.2847   0.1935   0.1744   0.1637   0.1836
## Detection Rate         0.2504   0.1171   0.1399   0.1042   0.1331
## Detection Prevalence   0.2916   0.1641   0.2150   0.1510   0.1783
## Balanced Accuracy      0.9108   0.7735   0.8556   0.7904   0.8349
```

Second prediction model :- Using Random Forest

```
model2 <- randomForest(classe ~. , data=trainingset, method="class")

# Predicting:
prediction2 <- predict(model2, testset, type = "class")

# Test results on subTesting data set:
confusionMatrix(prediction2, testset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2183   14    0    0    0
##          B    4 1465    9    0    0
##          C    0    8 1328   17    0
##          D    0    0    3 1239    7
##          E    1    0    0    2 1404
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9892, 0.9935)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.9893
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9977   0.9852   0.9910   0.9849   0.9950
## Specificity            0.9975   0.9979   0.9961   0.9984   0.9995
## Pos Pred Value         0.9936   0.9912   0.9815   0.9920   0.9979
## Neg Pred Value         0.9991   0.9965   0.9981   0.9970   0.9989
## Prevalence             0.2847   0.1935   0.1744   0.1637   0.1836
```

```
## Detection Rate         0.2841   0.1907   0.1728   0.1612   0.1827
## Detection Prevalence   0.2859   0.1923   0.1761   0.1625   0.1831
## Balanced Accuracy      0.9976   0.9916   0.9936   0.9917   0.9973
```

Decision

As expected, Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. The random Forest model is choosen. The accuracy of the model is 0.995. The expected out-of-sample error is estimated at 0.005, or 0.5%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified.