Predict exercise manners based on data from the quantified self context

'Practical Machine Learning' course project of Hendrik L., September 2015

Download and read the training and test data

```
# download and prepare the subTrainData data
if (!file.exists("pmlTraining.csv")) {
    fileURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-subTrainData.csv"
    download.file(fileURL, "pmlTraining.csv", mode = "wb")
}
trainData <- read.csv("pmlTraining.csv",
    na.strings = c("NA", "NAs", "NULL", "#DIV/0!"," ", ""))

# download and prepare the test data
if (!file.exists("pmlTest.csv")) {
    fileURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-subTestData.csv"
    download.file(fileURL, "pmlTest.csv", mode = "wb")
}
testData <- read.csv("pmlTest.csv",
    na.strings = c("NA", "NAs", "NULL", "#DIV/0!"," ", ""))
```

Drop columns with NA values and unnecessary data

```
# drop the columns with NA values
trainData <- trainData[, -which(names(trainData) %in% c("X", "user_name",
    "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp",
    "new_window", "num_window"))]
trainData <- trainData[, colSums(is.na(trainData)) == 0]
```

Remove zero and near-zero variance predictors

```
# zero variance predictors: they have one unique value across samples
# near-zero variance predictors: they have both few unique values relative to the
#   number of samples and a large ratio of the frequency of the most common value
#   to the frequency of the second most common value
nearZV <- nearZeroVar(trainData[sapply(trainData, is.numeric)], saveMetrics = TRUE)
trainData <- trainData[, nearZV[, 'nzv'] == 0]
```

Remove highly correlated data (correlation cutoff = 0.7)

```
corMatrix <- cor(na.omit(trainData[sapply(trainData, is.numeric)]))
# correlationmatrixdegreesoffreedom <- expand.grid(row = 1:52, col = 1:52)
# correlationmatrixdegreesoffreedom$correlation <- as.vector(corMatrix)
corIndex <- findCorrelation(corMatrix, cutoff = 0.7, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  22 with corr  0.722
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 22  and column  4 with corr  0.759
##   Means:  0.224 vs 0.158 so flagging column 22
## Compare row 4  and column  3 with corr  0.762
##   Means:  0.2 vs 0.155 so flagging column 4
## Compare row 3  and column  8 with corr  0.708
##   Means:  0.2 vs 0.153 so flagging column 3
## Compare row 36  and column  29 with corr  0.849
##   Means:  0.257 vs 0.151 so flagging column 36
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.229 vs 0.146 so flagging column 8
## Compare row 2  and column  11 with corr  0.884
##   Means:  0.212 vs 0.143 so flagging column 2
## Compare row 37  and column  38 with corr  0.769
##   Means:  0.198 vs 0.139 so flagging column 37
## Compare row 35  and column  30 with corr  0.773
##   Means:  0.195 vs 0.137 so flagging column 35
## Compare row 38  and column  5 with corr  0.781
##   Means:  0.177 vs 0.134 so flagging column 38
## Compare row 21  and column  24 with corr  0.814
##   Means:  0.176 vs 0.133 so flagging column 21
## Compare row 34  and column  28 with corr  0.808
##   Means:  0.176 vs 0.13 so flagging column 34
## Compare row 23  and column  26 with corr  0.779
##   Means:  0.137 vs 0.129 so flagging column 23
## Compare row 25  and column  24 with corr  0.792
##   Means:  0.145 vs 0.128 so flagging column 25
## Compare row 12  and column  13 with corr  0.779
##   Means:  0.122 vs 0.127 so flagging column 13
## Compare row 48  and column  51 with corr  0.772
##   Means:  0.145 vs 0.127 so flagging column 48
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.095 vs 0.127 so flagging column 18
## Compare row 46  and column  45 with corr  0.846
##   Means:  0.131 vs 0.129 so flagging column 46
## Compare row 45  and column  31 with corr  0.71
##   Means:  0.098 vs 0.129 so flagging column 31
## Compare row 45  and column  33 with corr  0.716
##   Means:  0.078 vs 0.132 so flagging column 33
## All correlations <= 0.7
```

```
trainData <- trainData[, -corIndex]
```

Remove blank columns from training and test data

```
for (i in c(8:ncol(trainData) - 1)) {
    trainData[,i] <- as.numeric(as.character(trainData[,i]))
}
for (i in c(8:ncol(testData) - 1)) {
    testData[,i] <- as.numeric(as.character(testData[,i]))
}
```

Create the final feature set and model data

```
featureset <- colnames(trainData[colSums(is.na(trainData)) == 0])[-(1:7)]
modeldata <- trainData[featureset]
```

Subset the training data (60% for training, 40% for testing)

```
trainIndex <- createDataPartition(modeldata$classe, p = 0.6, list = FALSE )
subTrainData <- modeldata[trainIndex,]
subTestData <- modeldata[-trainIndex,]
```

Fit a random forest model with 5-fold cross validation

```
control <- trainControl(method = "cv", 5)
model <- train(classe ~ ., data = subTrainData, method = "rf",
    trControl = control, ntree = 250)
```

Evaluate the model performance

```
predict <- predict(model, subTestData)
confusionMatrix(subTestData$classe, predict)
```

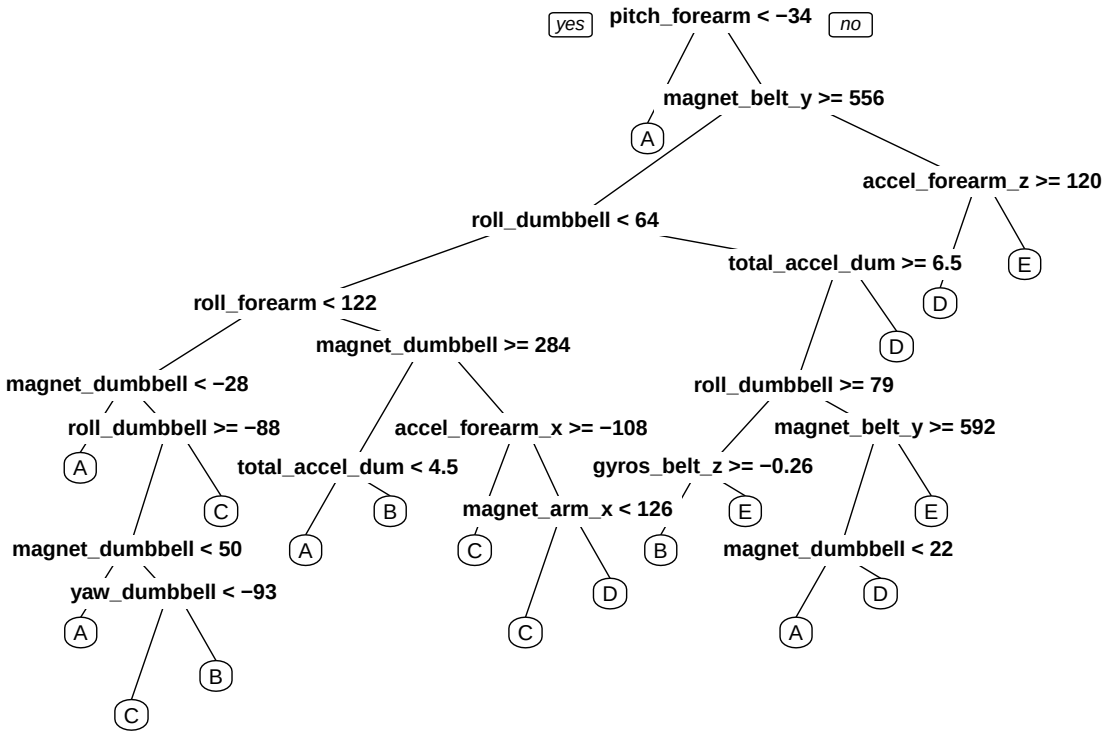
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2223    4    1    2    2
##          B   34 1468    9    6    1
##          C    1   23 1329   11    4
##          D    2    0   52 1222   10
##          E    3    9    2   11 1417
##
## Overall Statistics
##
##                Accuracy : 0.9762
##                  95% CI : (0.9725, 0.9794)
##     No Information Rate : 0.2884
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9698
##  Mcnemar's Test P-Value : 4.833e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9823   0.9761   0.9541   0.9760   0.9881
## Specificity            0.9984   0.9921   0.9940   0.9903   0.9961
## Pos Pred Value         0.9960   0.9671   0.9715   0.9502   0.9827
## Neg Pred Value         0.9929   0.9943   0.9901   0.9954   0.9973
## Prevalence             0.2884   0.1917   0.1775   0.1596   0.1828
## Detection Rate         0.2833   0.1871   0.1694   0.1557   0.1806
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9904   0.9841   0.9740   0.9832   0.9921
```

```
accuracy <- postResample(predict, subTestData$classe)
```

Estimated results:
The estimated **model accuracy** is **97.62%** and the estimated **out of sample error** is **2.38%**.

Show the tree model

```
treeModel <- rpart(classe ~ ., data = trainData, method = "class")
prp(treeModel)
```



Part 2 of the course project:

Predict the results for the test data and write them to files for submission

```
testData <- testData[featureset[featureset != 'classe']]
predictedResults <- predict(model, newdata = testData)
for (i in 1:length(predictedResults)) {
    filename = paste0("problem_id_", i, ".txt")
    write.table(predictedResults[i], file = filename, quote = FALSE,
        row.names = FALSE, col.names = FALSE)
}
```