Final Project for "Practical Machine Learning"

Setup Project

Download the training and test data using these links:

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

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

Next, load all necessary libraries:

```
library(caret)
library(AppliedPredictiveModeling)
library(randomForest)
```

Load and preprocess data

Next, I loaded the data.I noticed that there where many columns that appear to be empty

```
train_data_raw <- read.csv("pml-training.csv")
test_data_raw <- read.csv("pml-testing.csv")


#frist impression: lot of NAs and empty cells, fill empty cells with NA, confirm with:
train_data_raw[train_data_raw == ''] <- NA
test_data_raw[test_data_raw == ''] <- NA
number_na <- colSums(is.na(train_data_raw))
unique(number_na)
```

```
## [1]     0 19216
```

From the unique values (0 or 19216) we can see, that the columns are either complete or empty, there are no "half-filled" columns. So it's safe to remove them.

```
#remove all columns that only contain NAs
train_data_complete <- train_data_raw[,(colSums(is.na(train_data_raw))==0)]
test_data_complete <- test_data_raw[,(colSums(is.na(test_data_raw))==0)]
```

Since the description of the dataset says: "This human activity recognition research has traditionally focused on discriminating between different activities, i.e. to predict"which" activity was performed at a specific point in time (like with the Daily Living Activities dataset above)" I understand that we have to make a prediction for each line, disregarding any potential time component.

```
#remove all columns that contain information about time or user
train_data_reduced <- train_data_complete[,8:ncol(train_data_complete)]
test_data_reduced <- test_data_complete[,8:ncol(test_data_complete)]


#scale all columns except for "Classe" because they have different orders of magnitude
train_data_scaled <- as.data.frame(scale(train_data_reduced[,1:(ncol(train_data_reduced)-1)]))
test_data_scaled <- as.data.frame(scale(test_data_reduced[,1:(ncol(test_data_reduced)-1)]))
```

```r
#make "Classe" a factor variable
train_data_scaled$classe <- factor(train_data_reduced$classe)

#Split data
inTrain = createDataPartition(train_data_scaled$classe , p = 0.7)[[1]]
data_training = train_data_scaled[ inTrain,]
data_testing = train_data_scaled[-inTrain,]
```

Train Models

First I tried to use a "normal" tree.

```r
#Classification Tree
class_tree <- train(classe ~ ., method = "rpart", data = data_training)
tree_pred <- predict(class_tree, newdata = data_testing)
confusionMatrix(tree_pred, data_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1503  483  470  421  134
##          B   27  386   43  166  139
##          C  115  270  513  377  288
##          D    0    0    0    0    0
##          E   29    0    0    0  521
##
## Overall Statistics
##
##                Accuracy : 0.4967
##                  95% CI : (0.4838, 0.5095)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3429
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978  0.33889  0.50000   0.0000  0.48152
## Specificity            0.6419  0.92099  0.78391   1.0000  0.99396
## Pos Pred Value         0.4992  0.50723  0.32821      NaN  0.94727
## Neg Pred Value         0.9405  0.85304  0.88130   0.8362  0.89485
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2554  0.06559  0.08717   0.0000  0.08853
## Detection Prevalence   0.5116  0.12931  0.26559   0.0000  0.09346
## Balanced Accuracy      0.7699  0.62994  0.64195   0.5000  0.73774
```

As you can see, the accuracy is around 0.5, which is quite bad. So I tried to use a random forest next:

```r
class_forest <- train(classe ~ ., method = "rf", data = data_training)
forest_pred <- predict(class_forest, newdata = data_testing)
confusionMatrix(forest_pred, data_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   16    0    0    0
##          B    3 1118   18    0    0
##          C    0    5 1007   12    0
##          D    0    0    1  952    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9901
##                  95% CI : (0.9873, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9816   0.9815   0.9876   0.9972
## Specificity            0.9962   0.9956   0.9965   0.9992   1.0000
## Pos Pred Value         0.9905   0.9816   0.9834   0.9958   1.0000
## Neg Pred Value         0.9993   0.9956   0.9961   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1900   0.1711   0.1618   0.1833
## Detection Prevalence   0.2867   0.1935   0.1740   0.1624   0.1833
## Balanced Accuracy      0.9972   0.9886   0.9890   0.9934   0.9986
```

The accuracy is 0.99, which is very good. So I use this model to predict.

```r
#predict with test data
predict(class_forest, test_data_scaled)
```