Machine learning prediction

Summary

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. The goal of this project is to predict in which manner an excercise was performed, by using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. First some exploratory analysis/ data cleaning is performed, next several models are fitted and tested, then the best fitting model is chosen and used to predict outcomes for the test data set.

Data analysis

Read in the data

Load required packages, read in the data

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
setwd("C:/Users/Dani/Documents")
# provide a character vector of which strings are to be interpreted as null
training <- read.csv("pml-training.csv", na.strings=c("", "NA", "NULL"))
testing <- read.csv("pml-testing.csv", na.strings = c("","NA","NULL"))
```

Exploratory data analysis / Clean up data

- Get data dimensions to show how large the dataset is

```
dim(training)
```

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

```
dim(testing)
```

```
## [1]  20 160
```

The dimensions of the training data show that there are 160 variables (including the outcome variable). Because there are many variables with NA's and plenty variables are available, unrelated and variables with NA's are removed from the dataset

- Remove variables with NA's

```
# Remove columns with NA's
training <- training[,colSums(is.na(training)) == 0]
dim(training)
```

```
## [1] 19622    60
```

Removal of variables with NAs results in a remaining 60 variables

- Remove other variables that are likely uncorrelated (time related variables) These are the first 7 variables, which are correlated to time, row number or the individual

```
training <- training[, c(-1:-7)]
```

After removal of these variables, a clean dataset remains, which can be used for training and cross validation.

Split the training data for cross validation

- Split the data into training and validation/testing 75/25 To test multiple different models, cross validation set is needed. Therefore the training data is split into a new training and validation/test data set. To make this exact approach reproducable, a seed was set on 101.

```
set.seed(101)
trainInd <- createDataPartition(training$classe, p = 3/4, list = FALSE)
traindat <- training[trainInd,]
testdat <- training[-trainInd,]

# Convert the outcome to factor
traindat$classe <- as.factor(traindat$classe)
testdat$classe <- as.factor(testdat$classe)
```

The resulting training and test data set can now be used to create different models for comparison.

Create prediction models

Different methods are available to create a prediction model, to analyze which of these methods results in the highest accuracy, three different models were tested. - Create prediction models with different methods (methods = random forest, lda, rpa)

```
# Use randomforest function instead of caret's rf method (randomforest is much faster)
randomFMod <- randomForest(classe ~ ., data = traindat)

# Use lda and rpart method from the caret package to create models
ldaMod <- train(classe ~ ., data = traindat, method ="lda")
```

```
## Loading required package: MASS
```

```
rpaMod <- train(classe ~ ., data = traindat, method ="rpart")
```

```
## Loading required package: rpart
```

The created prediction models can now be used

Validate prediction models

- Check which prediction model fits best (has the highest accuracy)

```
# Predict values using test/validation data
randomFPred <- predict(randomFMod, testdat)
ldaPred <- predict(ldaMod, testdat)
rpaPred <- predict(rpaMod, testdat)

# Calculate accuracy
confusionMatrix(randomFPred,testdat$classe)$overall[1]
```

```
##  Accuracy
## 0.9949021
```

```
confusionMatrix(ldaPred,testdat$classe)$overall[1]
```

```
##  Accuracy
## 0.6926998
```

```
confusionMatrix(rpaPred,testdat$classe)$overall[1]
```

```
##  Accuracy
## 0.4893964
```

The random forest model results in an accuracy of 99.5%, which is the highest of the 3 different methods used. Because 99.5% is already quite a high percentage, the random forest model will be used to predict the classe variable of the test set.

Use best model to predict the test set

- Predict classe variable of the training set

```
# Predict on the testing data
predictedV <- predict(randomFMod,testing)
predictedV
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```