Barebell lifts prediction using machine learning

Overview

The main purpose of this document is to create a model to predict the classe of barbell lift performed by 6 participants, there is a total of 5 different ways of barbell lifs.

Data Load and data cleaning

The data is loaded from 2 different dataset, one for training and testing and a second one for validation. First we load the data, apply some cleaning on the variables.

```
library(caret)
```

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

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

```
setwd("./Data")
csvtraining <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
csvtesting <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
```

Initially there is a total of 160 columns and 19622 records on training data and 20 records on testing data.

```
dim(csvtraining)
```

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

```
dim(csvtesting)
```

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

I apply some cleaning on the data, deleting the variables witch has more than 95% of NA values.

```
ColumnIndexNA <- colSums(is.na(csvtraining))/nrow(csvtraining) < 0.95
TrainingDataNA <- csvtraining[,ColumnIndexNA]
```

Then, I remove the columns related to the time, because there is no need of these variables in our prediction model.

```
FinalTraining <- TrainingDataNA[, -c(1,3:7)]
#Convert the classe column to a factor variable
FinalTraining$classe <- factor(FinalTraining$classe)
dim(FinalTraining)
```

```
## [1] 19622    54
```

At the end, the final dataset contains a total of 54 variables including the predicted variable classe. Then, I apply the same cleaning on the validation data.

```r
ColumnNames <- names(FinalTraining)
Testing <- csvtesting[,ColumnNames[1:53]]
```

Cross Validacion and Model selection

After loading the data, the next step is to divide the testing data in 2 datasets, needed for cross validation, one for training and the second one for validating the accuracy of the model.

```r
set.seed(12345)
TrainIndex <- createDataPartition(FinalTraining$classe, p=0.75,list = FALSE)
Training <- FinalTraining[TrainIndex,]
Validation <- FinalTraining[-TrainIndex,]
```

Having created all the datasets needed for the model creation, I selected random forest as a prediction model. This kind of model fits really well on the kind of problem we are trying to solve.

```r
set.seed(12345)
FitRF <- train(classe~.,method='rf',data=Training,ntree=100)
RFPrediction <- predict (FitRF,Validation)
#Showing the sample error on the model
confusionMatrix(Validation$classe, RFPrediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    1  946    2    0    0
##          C    0    1  849    5    0
##          D    0    0   10  790    4
##          E    0    0    2    2  897
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9989   0.9838   0.9912   0.9956
```

```
## Specificity            1.0000   0.9992   0.9985   0.9966   0.9990
## Pos Pred Value         1.0000   0.9968   0.9930   0.9826   0.9956
## Neg Pred Value         0.9997   0.9997   0.9965   0.9983   0.9990
## Prevalence             0.2847   0.1931   0.1760   0.1625   0.1837
## Detection Rate         0.2845   0.1929   0.1731   0.1611   0.1829
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9996   0.9991   0.9911   0.9939   0.9973
```

Predict on validation data

The result of predicting on testing data:

```r
RFPrediction <- predict (FitRF,Testing)
RFPrediction
```

```
##  [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

I am getting a 0.9945% in sample accuracy witch is a high level of accuracy, we can accept this value and conclude that using random forest to solve this problem is a good choice.