Untitled

Ivo Pinheiro

2024-09-06

Practical Machine Learning Course Project

To predict the manner in which participants performed barbell lifts using sensor data, that was the task at hand. To accomplish this, I cleaned and analysed data to built a model that predicts the value of a target variable ($classe) based on input variables (53 features).

```
# Libraries
set.seed(123)
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

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

```
library(caret)
```

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

Data preprocessing

I started by assigning NA to any missing value in the data, and then I removed any column that had over 90% NA. As it didn't make sense to include the first six columns, these were removed. Lastly, the target variable ($classe) was converted into a factor variable.

These steps reduced the number of columns from 160 to 54, and ensured that our data was now ready for analysis and building our model.

```
# Importing
url_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
# Reading
training_data <- read.csv(url_training)
testing_data <- read.csv(url_testing)
# Cleaning training set
dim(training_data)
```

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

```
training_data[training_data == ""] <- NA
proportion_NA_train <- colSums(is.na(training_data))/nrow(training_data)
```

```r
training_data1 <- training_data[, proportion_NA_train < 0.9]
training_data2 <- training_data1[, -c(1:6)]
training_data2$classe <- as.factor(training_data2$classe)
```

Model building and evaluation

Since this is a classification problem and we already know the target variable ($classe), the model that made the most sense to me was random forests. I trained the model on 70% of the training set and then I tested it on the other 30%.

```r
# Model with training set
in_Train <- createDataPartition(y=training_data2$classe, p=0.7, list=FALSE)
trainingset <- training_data2[in_Train, ]
testingset <- training_data2[-in_Train, ]
model_1 <- randomForest(classe ~ ., data = trainingset)
print(model_1)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingset)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.31%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2648    3    0    0 0.0037622272
## C    0   10 2386    0    0 0.0041736227
## D    0    0   14 2237    1 0.0066607460
## E    0    0    0    6 2519 0.0023762376
```

Happy with the results, I applied the model to the testing set.

```r
# Cleaning test set
dim(testing_data)
```

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

```r
testing_data[testing_data == ""] <- NA
proportion_NA_test <- colSums(is.na(testing_data))/nrow(testing_data)
testing_data1 <- testing_data[, proportion_NA_test < 0.9]
testing_data2 <- testing_data1[, -c(1:6)]
# Applying model to test set
testset_predictions <- predict(model_1, newdata = testing_data2)
print(testset_predictions)
```

```
##  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
```

So what is the percentage of the target variable in the test set that are correctly classified by the model?

```r
predictions <- predict(model_1, testingset)
confusionMatrix(predictions, testingset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1137    1    0    0
##          C    0    2 1025    2    0
##          D    0    0    0  962    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9971, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9990   0.9979   0.9963
## Specificity            1.0000   0.9998   0.9992   0.9992   1.0000
## Pos Pred Value         1.0000   0.9991   0.9961   0.9959   1.0000
## Neg Pred Value         1.0000   0.9996   0.9998   0.9996   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1932   0.1742   0.1635   0.1832
## Detection Prevalence   0.2845   0.1934   0.1749   0.1641   0.1832
## Balanced Accuracy      1.0000   0.9990   0.9991   0.9986   0.9982
```

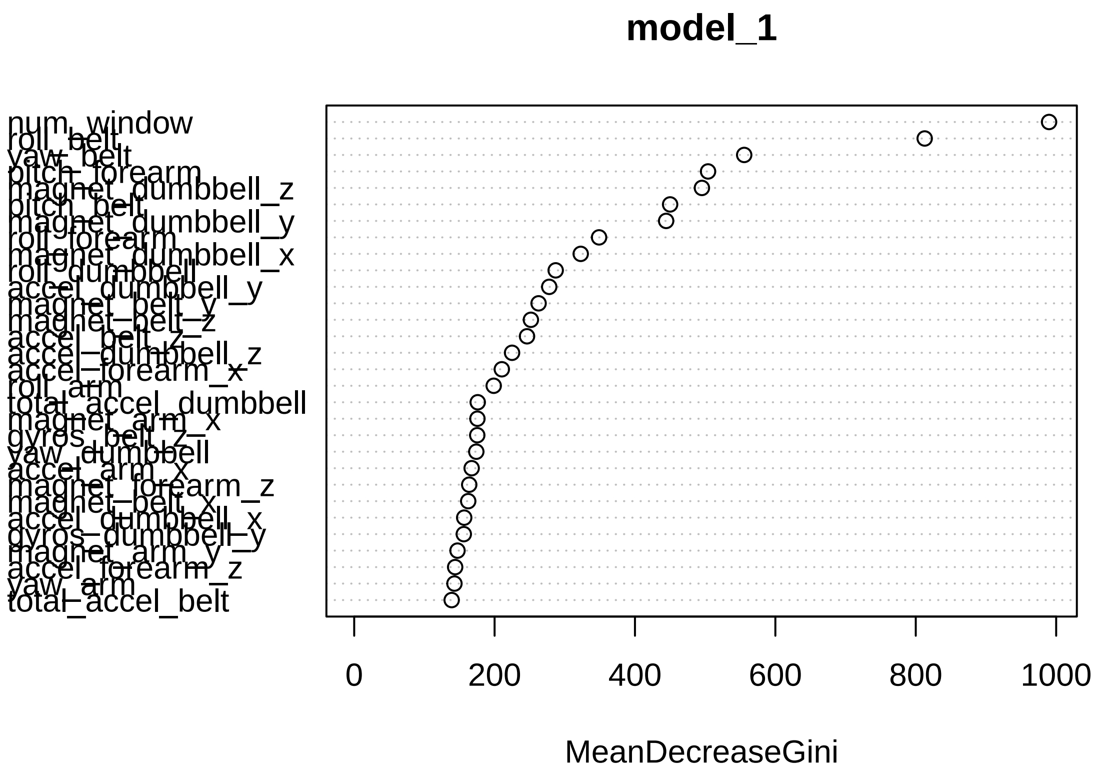
According to the confusion matrix, the model correctly classified 99.85% of the instances overall.

Given the high accuracy, the expected out-of-sample error is likely to be very low (1-0.9985=0.0015).

Thus, the expected out-of-sample error is approximately 0.15%.

Plots

```r
varImpPlot(model_1)
```




```
ggplot(training_data2, aes(x = classe)) +
        geom_bar(fill = "white", color = "grey") +
        labs(title = "How many of each $classe in the training set", x = NULL, y = NULL)
```

How many of each $classe in the training set