Practical machine learning course project

VictorButen

2 4 2021

Introduction/Executive Summary

The goal for this project is to build a machine learning model to predict how an excercise was done using data from six participants doing the exercise for 5 sets of ten repetitions each. Each one of the sets performed the exercise in a different way, with one (class A) being the correct one, and the other ones being common ways of doing the exercise wrongly.

Data Cleaning

I start by loading up the data, as well as the packages I'll be using throughout the analysis.

```
## Link for the training dataset: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
training <- read.csv("training.csv")

##Link for the test dataset: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
## I call the test dataset validation as I'll be making my own testing set from the training data
validation <- read.csv("testing.csv")

library(caret);library(ggplot2);library(rpart);library(maptree)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: cluster
```

```
set.seed(11111)

dim(training)
```

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

We see that the training data contains 19622 observations of 160 variables, leading to the first part of the analysis: cleaning up the data. I first notice that a lot of columns (like the kurtosis ones) have missing values as "" rather than NA, so I start by changing that. I then remove columns 1:7 as these contain metadata like subject name that don't tell us much about the excercise being performed. Lastly, I then remove the columns that contain NA values.

```
training[training==""] <- NA

## I remove the columns of metadata, then the columns containing NA values
training <- training[,-c(1:7)]
training <- training[,colSums(is.na(training))==0]

validation <- validation[,-c(1:7)]
validation <- validation[,colSums(is.na(validation))==0]


dim(training)
```

```
## [1] 19622    53
```

```
dim(validation)
```

```
## [1] 20 53
```

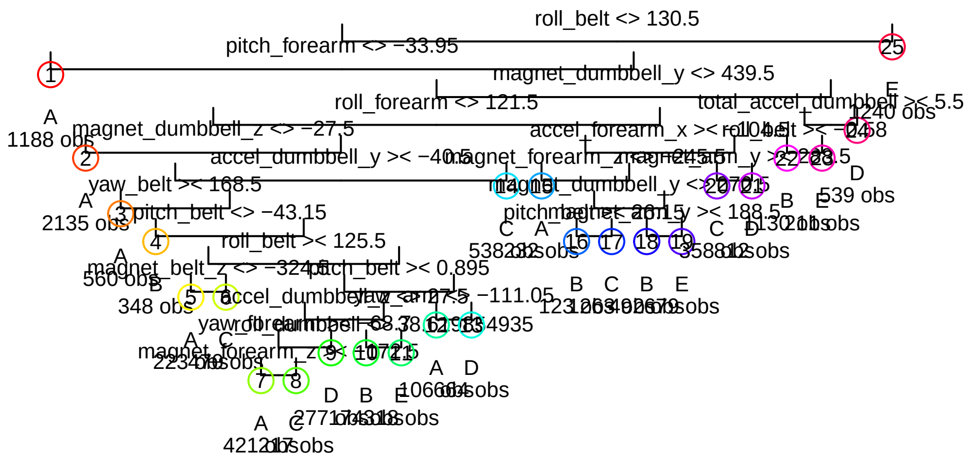
As we see from the dim printout, we are not down to 53 variables for both the training and validation dataset. Next, I split the training dataset into two new datasets, one for training and one for testing.

```
inTrain <- createDataPartition(training$classe, p=0.75, list=F)
testing <- training[-inTrain,]
training <- training[inTrain,]
```

Model training/prediction

Now that we have our datasets, we can get started making our ML models. Since we're dealing with a classification problem, I'll start off trying to use a classification tree.

```
treeModel <- rpart(classe~.,data=training, method="class")
draw.tree(treeModel, cex=.7)
```



```
pTree <- predict(treeModel, testing, type="class")
confusionMatrix(pTree, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1285  199   15   75   28
##          B   34  562   84   41   66
##          C   37  100  656   75   79
##          D   24   50   70  548   76
##          E   15   38   30   65  652
##
## Overall Statistics
##
##                Accuracy : 0.7551
##                  95% CI : (0.7428, 0.7671)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6887
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9211   0.5922   0.7673   0.6816   0.7236
## Specificity            0.9097   0.9431   0.9281   0.9463   0.9630
## Pos Pred Value         0.8021   0.7141   0.6927   0.7135   0.8150
## Neg Pred Value         0.9667   0.9060   0.9497   0.9381   0.9393
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2620   0.1146   0.1338   0.1117   0.1330
## Detection Prevalence   0.3267   0.1605   0.1931   0.1566   0.1631
## Balanced Accuracy      0.9154   0.7677   0.8477   0.8140   0.8433
```

With an overall accuracy of .75 our model isn't the worst, but it could be better. Since the tree seems to be doing well, I'll move on to either a random forest or a gbm model, both of which use multiple trees as their basis.

```
ctrl <- trainControl(method="repeatedcv", number=5, repeats = 3)
forestModel <- train(classe~.,data=training, method="rf", trControl=ctrl)
pForest <- predict(forestModel, testing)
confusionMatrix(pForest, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    6    0    0    0
##          B    1  940    1    0    0
##          C    0    3  850    4    1
##          D    0    0    4  799    4
##          E    0    0    0    1  896
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9905   0.9942   0.9938   0.9945
## Specificity            0.9983   0.9995   0.9980   0.9980   0.9998
## Pos Pred Value         0.9957   0.9979   0.9907   0.9901   0.9989
## Neg Pred Value         0.9997   0.9977   0.9988   0.9988   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1917   0.1733   0.1629   0.1827
## Detection Prevalence   0.2855   0.1921   0.1750   0.1646   0.1829
## Balanced Accuracy      0.9988   0.9950   0.9961   0.9959   0.9971
```

Our random forest is doing extremely well with an overall accuracy of .994, and the 95% confidence interval for that accuracy also being entirely above 0.99. Now for the gbm:

```r
gbmModel <- train(classe~.,data=training, method="gbm", trControl=ctrl, verbose=F)
pGbm <- predict(gbmModel, testing)
confusionMatrix(pGbm, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1375   29    0    0    2
##          B   17  896   22    3   13
##          C    1   24  819   17   14
##          D    2    0   11  782   15
##          E    0    0    3    2  857
##
## Overall Statistics
##
##                Accuracy : 0.9643
##                  95% CI : (0.9587, 0.9693)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9549
##
##  Mcnemar's Test P-Value : 5.955e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9442   0.9579   0.9726   0.9512
## Specificity            0.9912   0.9861   0.9862   0.9932   0.9988
## Pos Pred Value         0.9780   0.9422   0.9360   0.9654   0.9942
## Neg Pred Value         0.9943   0.9866   0.9911   0.9946   0.9891
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2804   0.1827   0.1670   0.1595   0.1748
## Detection Prevalence   0.2867   0.1939   0.1784   0.1652   0.1758
## Balanced Accuracy      0.9884   0.9651   0.9720   0.9829   0.9750
```

Our gbm model also has a very high overall accuracy, though not quite as high as the random forest. For my final prediction of the validators I'll therefore be using the random forest, which I'll in turn use to answer the quiz that goes along with this assignment.

```r
pValidation <- predict(forestModel, validation)
pValidation
```

```
##  [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
```