Course Project - ML

Rafael Vanhoz

12/09/2020

Practice Machine learning project Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here

library

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

```
library(rpart)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

DATA

```
downloadcsv <- function(url, nastrings) {
    temp <- tempfile()
    download.file(url, temp, method = "curl")
    data <- read.csv(temp, na.strings = nastrings)
    unlink(temp)
    return(data)
}

trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
train.data.raw <- downloadcsv(trainurl, c("", "NA", "#DIV/0!"))

testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
test.data.raw <- downloadcsv(testurl, c("", "NA", "#DIV/0!"))
```

The training data has 19622 observations and 160 features, and the distribution of the five measured stances A,B,C,D,E is:

```
dim(train.data.raw)
```

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

```
table(train.data.raw$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
# Drop the first 7 columns as they're unnecessary for predicting.
train.data.clean1 <- train.data.raw[,8:length(colnames(train.data.raw))]
test.data.clean1 <- test.data.raw[,8:length(colnames(test.data.raw))]
# Drop colums with NAs
train.data.clean1 <- train.data.clean1[, colSums(is.na(train.data.clean1)) == 0]
test.data.clean1 <- test.data.clean1[, colSums(is.na(test.data.clean1)) == 0]
# Check for near zero variance predictors and drop them if necessary
nzv <- nearZeroVar(train.data.clean1,saveMetrics=TRUE)
zero.var.ind <- sum(nzv$nzv)
if ((zero.var.ind>0)) {
        train.data.clean1 <- train.data.clean1[,nzv$nzv==FALSE]
}
```

The training data is divided into two sets. This first is a training set with 70% of the data which is used to train the model. The second is a validation set used to assess model performance.

```
in.training <- createDataPartition(train.data.clean1$classe, p=0.70, list=F)
train.data.final <- train.data.clean1[in.training, ]
validate.data.final <- train.data.clean1[-in.training, ]
```

MODEL DEVELOPMENT ### Train the model
The training data-set is used to fit a Random Forest model because it automatically selects important variables and is robust to correlated covariates & outliers in general. 5-fold cross validation is used when applying the algorithm. A Random Forest algorithm is a way of averaging multiple deep decision trees, trained on different parts of the same data-set, with the goal of reducing the variance. This typically produces better performance at the expense of bias and interpret-ability. The Cross-validation technique assesses how the results of a statistical analysis will generalize to an independent data set. In 5-fold cross-validation, the original sample is randomly partitioned into 5 equal sized sub-samples. a single sample is retained for validation and the other sub-samples are used as training data. The process is repeated 5 times and the

results from the folds are averaged.

```
control.parms <- trainControl(method="cv", 5)
rf.model <- train(classe ~ ., data=train.data.final, method="rf",
                  trControl=control.parms, ntree=251)
rf.model
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10990, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9899546  0.9872914
##   27    0.9903184  0.9877513
##   52    0.9837668  0.9794611
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

ESTIMATE PERFORMANCE The model fit using the training data is tested against the validation data. Predicted values for the validation data are then compared to the actual values. This allows forecasting the accuracy and overall out-of-sample error, which indicate how well the model will perform with other data.

```
rf.predict <- predict(rf.model, validate.data.final)
confusionMatrix(as.factor(validate.data.final$classe), rf.predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    1    0    0
##          B    9 1127    3    0    0
##          C    0    8 1016    2    0
##          D    0    0   11  952    1
##          E    0    0    1    2 1079
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2858
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9930   0.9845   0.9958   0.9991
## Specificity            0.9998   0.9975   0.9979   0.9976   0.9994
## Pos Pred Value         0.9994   0.9895   0.9903   0.9876   0.9972
## Neg Pred Value         0.9979   0.9983   0.9967   0.9992   0.9998
## Prevalence             0.2858   0.1929   0.1754   0.1624   0.1835
## Detection Rate         0.2843   0.1915   0.1726   0.1618   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9972   0.9952   0.9912   0.9967   0.9992
```

```
accuracy <- postResample(rf.predict, as.factor(validate.data.final$classe))
acc.out <- accuracy[1]
overall.ose <-
        1 - as.numeric(confusionMatrix(as.factor(validate.data.final$classe), rf.predict)
                       $overall[1])
```

RESULTS The accuracy of this model is **0.9935429** and the Overall Out-of-Sample error is **0.0064571**.

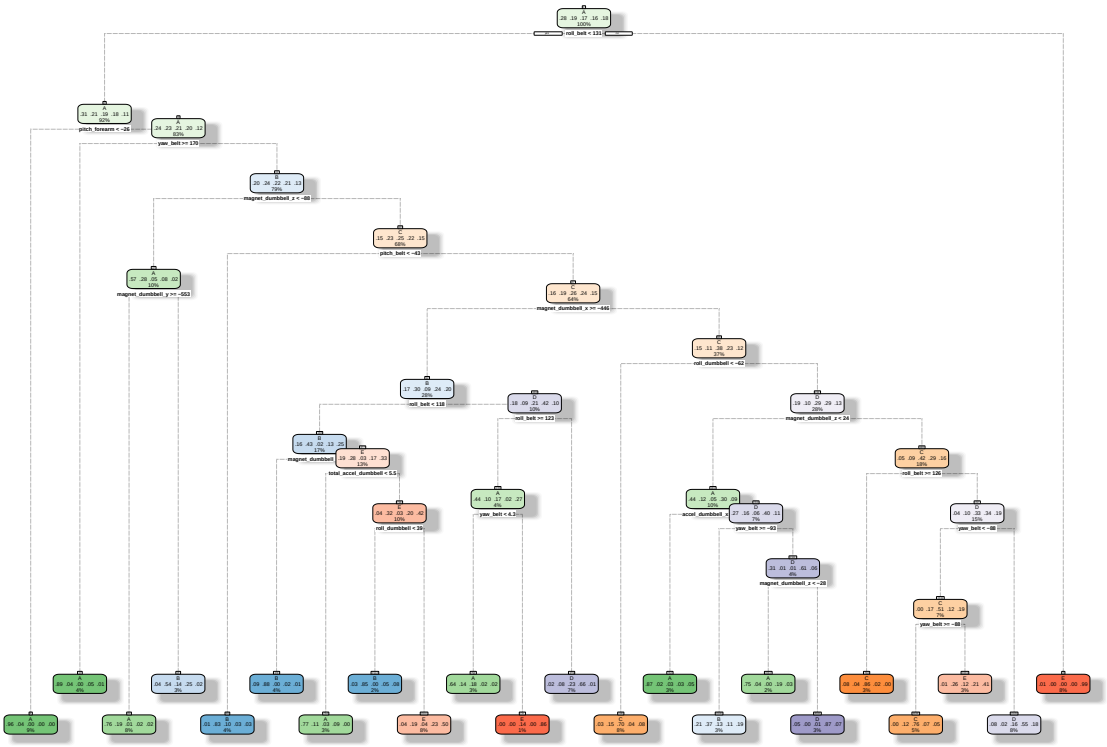
RUN THE MODEL The model is applied to the test data to produce the results.

```
results <- predict(rf.model,
                   test.data.clean1[, -length(names(test.data.clean1))])
results
```

```
##  [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
```

DECISION TREE VISUALIZATION

```
treeModel <- rpart(classe ~ ., data=train.data.final, method="class")
fancyRpartPlot(treeModel)
```



Rattle 2020−set−12 16:05:57 rafae