Course Project: Machine Learning

Gustavo Sánchez

3/7/2020

Introduction: We will be using machine learning to predict the outcome of given data, when the training data is given.

Description:

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, our 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. The main goal of the project is to predict the manner in which 6 participants performed the exercise.

Set Folder

```
getwd()
```

```
## [1] "C:/Users/GUSTAVO/Desktop/Data analisys/Taller 1/R"
```

Reading the File:

```
Dtrain<-read.csv("pml-training.csv")
Dtest<-read.csv("pml-testing.csv")
```

Importing libraries:

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

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

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

```
library(glm2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
##
## Attaching package: 'ggplot2'
```

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

```
library(ggplot2)
library(lattice)
```

Data cleaning: Data contains a lot of NA, which we will remove. Variables that have near zero variance are also removed to make prediction more accurate.

```
# Removing near zero variance
NZV <- nearZeroVar(Dtrain)
Dtrain <- Dtrain[, -NZV]
Dtest  <- Dtest[, -NZV]
# Removing variables that contain mostly NA
AllNA    <- sapply(Dtrain, function(x) mean(is.na(x))) > 0.95
Dtrain <- Dtrain[, AllNA==FALSE]
Dtest  <- Dtest[, AllNA==FALSE]
# Remove variables that are used as identification only,
# these should not impact the prediction in any way, but it will,
# if we don't remove it.
Dtrain <- Dtrain[, -(1:5)]
Dtest  <- Dtest[, -(1:5)]
dim(Dtrain)
```

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

Exploratory Analysis:

Let us start with splitting data for cross validation.

```
set.seed(1230) #To make the results reproducible
intrain<-createDataPartition(Dtrain$classe,p=0.7,list=FALSE)
subtrain<-Dtrain[intrain,]
subtest<-Dtrain[-intrain,]
```

Prediction: We will use below 3 methods to predict the result for test set.

1.Boosting

2.Decision Trees

3.Random Forest

We will use all variables as predictors.

Accuracy of each prediction model is determined by cross-validation method. This works by splitting training data into two sub groups where one is for training and other for testing. We have used 70:30 ratio for splitting training data.

1. Boosted prediction model:

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.2
```

```
## Loaded gbm 2.1.5
```

```
library(glm2)
set.seed(1230)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
##Careful: Below line of code takes a long time to run
gbm_model<-train(classe~.,data=subtrain,method="gbm",trControl = control,verbose=FALSE)
pred_values3<-predict(gbm_model,newdata=subtest)
confusionMatrix(as.factor(subtest$classe),pred_values3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B   22 1095   22    0    0
##          C    0    6 1014    6    0
##          D    1    7    6  950    0
##          E    0    4    1   11 1066
##
## Overall Statistics
##
##                Accuracy : 0.9849
##                  95% CI : (0.9814, 0.9878)
##     No Information Rate : 0.2879
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9809
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9864   0.9821   0.9722   0.9824   1.0000
## Specificity            0.9993   0.9908   0.9975   0.9972   0.9967
## Pos Pred Value         0.9982   0.9614   0.9883   0.9855   0.9852
## Neg Pred Value         0.9945   0.9958   0.9940   0.9965   1.0000
## Prevalence             0.2879   0.1895   0.1772   0.1643   0.1811
## Detection Rate         0.2839   0.1861   0.1723   0.1614   0.1811
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9929   0.9864   0.9849   0.9898   0.9983
```

This model has given us 98.49% accuracy

2. Decision Trees for prediction:

```
library(rpart) # ignore this line
set.seed(1230)
linear_model<-rpart(classe~.,data=subtrain,method="class")
pred_values2<-predict(linear_model,newdata=subtest,type="class")
confusionMatrix(as.factor(subtest$classe),pred_values2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1510   52    6   83   23
##          B  264  632   68  133   42
##          C   43   73  828   50   32
##          D   79   33  135  623   94
##          E   54   31   65  138  794
##
## Overall Statistics
##
##                Accuracy : 0.7455
##                  95% CI : (0.7341, 0.7565)
##     No Information Rate : 0.3314
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6765
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7744   0.7698   0.7514   0.6066   0.8061
## Specificity            0.9583   0.8999   0.9586   0.9298   0.9412
## Pos Pred Value         0.9020   0.5549   0.8070   0.6463   0.7338
## Neg Pred Value         0.8955   0.9602   0.9436   0.9179   0.9602
## Prevalence             0.3314   0.1395   0.1873   0.1745   0.1674
## Detection Rate         0.2566   0.1074   0.1407   0.1059   0.1349
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8663   0.8348   0.8550   0.7682   0.8737
```

This method gives on 74.55% accuracy

3. Random Forest Method

```
set.seed(1230)
control <- trainControl(method="cv", number=3, verboseIter=FALSE)
##Careful: Below line of code takes a long time to run
forest_model<-train(classe~.,data=subtrain,method="rf",trControl=control)
```

```
pred_values1<-predict(forest_model,newdata=subtest)
confusionMatrix(as.factor(subtest$classe),pred_values1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    4 1133    2    0    0
##          C    0    1 1025    0    0
##          D    0    0    2  962    0
##          E    0    3    0    2 1077
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9965   0.9961   0.9979   1.0000
## Specificity            1.0000   0.9987   0.9998   0.9996   0.9990
## Pos Pred Value         1.0000   0.9947   0.9990   0.9979   0.9954
## Neg Pred Value         0.9991   0.9992   0.9992   0.9996   1.0000
## Prevalence             0.2851   0.1932   0.1749   0.1638   0.1830
## Detection Rate         0.2845   0.1925   0.1742   0.1635   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9988   0.9976   0.9980   0.9988   0.9995
```

We have achieved an accuracy of 99.75%

Prediction Results: Accuracy of each method:

Random Forests: 99.75% accuracy

Boosted model : 99.49% accuracy

Decision Trees: 74.55% accuracy

The best model for prediction is random forest

So we will use that method for predicting test set

Results of test data:

```
predict(forest_model,Dtest)
```

```
##  [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
```