MLproject

jesussuniaga

2 de abril de 2016

Background

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, my goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were performing barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (the section on the Weight Lifting Exercise Dataset).

Data Analysis & Features Selection

Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. from whom we are very greatful in allowing their data to be used for this assignment. After downloading the two datasets, next step is to read and save that raw data into two variables:rtraining and rtesting

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
```

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

```r
rtraining <- read.csv("C:\\Users\\Jesus\\AppData\\Local\\Temp\\RtmpkBG1gN\\data12ec109b286f", na.strings="c(\"NA\",\"#DIV/0\",\"\")")

rtesting <- read.csv("C:\\Users\\Jesus\\AppData\\Local\\Temp\\RtmpkBG1gN\\data12ec53b51627", na.strings="c(\"NA\",\"#DIV/0\",\"\"")
```

cleaning Data

We will remove first columns. since them contain values just for indexing the samples, and they do not give valuable information for predicting the sample's class. These are: "X" "raw_timestamp_part_1" "raw_timestamp_part_2" "cvtd_timestamp"

```r
rtraining <- subset(rtraining, select = -c(X))
rtraining <- subset(rtraining, select = -c(raw_timestamp_part_1))
rtraining <- subset(rtraining, select = -c(raw_timestamp_part_2))
rtraining <- subset(rtraining, select = -c(num_window))
```

The next step is to reassign NAs values . The mean of column values will be assigned to NAs. ``` **calculate mean for each integer and numerics columns assigns means to NAs**

```r
for (colName in names(rtraining)){

  if(class(rtraining[[colName]]) %in% c("integer", "numeric")){
    m<-mean(rtraining[[colName]], na.rm = TRUE)
    rtraining[[colName]][is.na(rtraining[[colName]])]<-m
  }

}
```

Now, let's remove the columns where variance is near zero. By default, a predictor is classified as near-zero variance if the percentage of unique values in the samples is less than {10%} and when the frequency ratio mentioned above is greater than 19 (95/5)

```r
rtraining<-rtraining[,-nearZeroVar(rtraining)]
which( colnames(rtraining) == "classe")
```

```
## [1] 55
```

```r
good <- colnames(rtraining[, -55])  # remove the classe column
```

```
rtesting <- rtesting[good]  # allow only predictors variables in testing that are also in rtraining

dim(rtesting)

## [1] 20 54
```

Splitting Dataset Extracting Training & Cross-Validation Data

After the data has been filtered, the remaining features can be used for training the model. For efficient prediction, the training data will be split into two subsets: training and cross-validation. Dividing the training set in training and validation will help to overcome the problem of overfiting the samples. First, training samples are used for fitting the model, but in order to overcome overfitting and to generalize the prediction error (also to measure the accuracy), cross-validation samples will be used

```
set.seed(123)

index <- createDataPartition(y=rtraining$classe, p=0.6, list=FALSE)

trainingData <- rtraining[index, ]

testingData <- rtraining[-index, ]
```

Creating the Model

Before proposing a machine learning algorithm for predicting a value based on a set of features as an input, the problem (or the question) that the algorithm will be trying to answer must be understand. This current project belongs in classification problems, so algorithms like logistic regression, prediction with trees, random forest or even neural network can be used.

In this project, predictions are made using the Random Forests algorithm. Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

```
train_control <- trainControl(method="cv", number=3)

model <- train(classe~., data=trainingData, method="rf", trControl = train_control)
```

From the confusion matrix after training the model, we can see that The model is predicting the training samples with great accuracy, and almost 0 error rate.

This means that the model has fitted the data very well. But to be sure it doesn’t overfit them, prediction with cross-validation samples will be made.

```
pred_cv <- predict(model, newdata = testingData)

confusionMatrix(pred_cv,testingData$classe)

## Confusion Matrix and Statistics

##

##             Reference

## Prediction    A    B    C    D    E
##           A 2229   10    0    0    0
##           B    3 1504    7    0    0
##           C    0    4 1350    3    3
##           D    0    0   11 1281    2
##           E    0    0    0    2 1437
## 
## Overall Statistics
## 
##                Accuracy : 0.9943          
##                  95% CI : (0.9923, 0.9958)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9927          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9908   0.9868   0.9961   0.9965
## Specificity            0.9982   0.9984   0.9985   0.9980   0.9997
## Pos Pred Value         0.9955   0.9934   0.9926   0.9900   0.9986
## Neg Pred Value         0.9995   0.9978   0.9972   0.9992   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1917   0.1721   0.1633   0.1832
## Detection Prevalence   0.2854   0.1930   0.1733   0.1649   0.1834
## Balanced Accuracy      0.9984   0.9946   0.9926   0.9971   0.9981
```

With accuracy of 99.43% on the cross-validation data set, this prediction algorithm is ready to be used for predicting the labels (classes) on the test samples.

Prediction on the Test Data

The model is used to predict the labels on the samples from the test set. The same features that were used for training the model, are extracted from the test set as an input to the model. The following predictions are obtained for the test samples.

```
prediction <- predict(model, rtesting, type="raw")
```

Generate Data for Submitting

pml_write_files = function(x){ n = length(x) for(i in 1:n){ filename = paste0("problem_id_",i,".txt")

```
write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE) } } x <-
testing

answers <- predict(model, newdata=x) answers
```