Machine Learning

Antonio Ferreras

Thursday, August 21, 2014

Executive Summary

This document is the final delivery of the Assigment of the Coursera "Practice Machine Learn". A classification model is fitted from a train-test data from more than 19.000 records and and 160 variables. I followed a proccess of reducing the number of variables, mantaining only the most significatives, while preserving the accuracy of the model.

The final result is a **Random Forest** model which only uses **6 variables as predictors**, and obtained a **100% success** in predict the test set.

---

1. 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. 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**. The goal of this 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 available in http://groupware.les.inf.puc-rio.br/har

The goal of this project is to predict the manner in which they did the exercise. This is the *classe* variable in the training set.

2. Data inspection

```
train <- read.csv(".//pml-training.csv")
test <- read.csv(".//pml-testing.csv")
dim(train)
```

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

19622 records in the data set and 160 variables. The data is quite noisy, as we know for the cited paper, so the appropiate model could be a Random Forest classification, but there are too much data for fitting such a complex model (untractable calculation time). Then the strategy for fitting a correct model is, first reduce as much as possible the number of the variables, and fit the Random Forest model and, finally, fit the parameters.

The response variable is *classe* a factor variable with 5 values {"A" to "E"}.

```
library(caret)
library(rpart)
library(randomForest)
set.seed(1966)
```

---

3. Variable reduction

Several variables do not have to do with the measurements. Following the paper cited above we will not use them in the prediction model. Lets eliminate them, in several steps:

- **problem_id**. It is merely one enumeration of the test records.

I combined the two data sets in a new variable called *combi*, for a parallel processing of the test and data sets. Then I eliminated de predictors which do not variate enough to be consider good predictors. I used the function *nearZeroVar* form the *caret* package over the test dataset.

```
test$problem_id <- NULL
test$classe<- c(rep("A",10), rep("B",10))
combi <- rbind(test,train)
nzv <- nearZeroVar(test)
combi <- combi[,-nzv]
```

- **X**. An enumeration. Do not have anything to do with the response. However is also highly correlated.
- **raw_timestamp_part_1**, **raw_timestamp_part_2**, **cvtd_timestamp**. The model must be independent from the time when the measurements are taken
- **num_window**, not related to the experiment
- **user_name**. The user whom belong the data. It could be used as a good predictor, but I consider it would be be cheating the model, as it must be valid for other future data and users

```
combi$X <- NULL
combi$raw_timestamp_part_1 <- NULL
combi$raw_timestamp_part_2 <- NULL
combi$cvtd_timestamp <- NULL
combi$num_window <- NULL
combi$user_name <- NULL
combi$classe <- factor(combi$classe)
ncol(combi)
```

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

The variables has been reduced to **53**. Let's work a bit more with a simplified model of *Recursive Partioning and Regression Tree* to further eliminate variables, before apply the Random Forest model.

```
train_data <- combi[c(21:19642),]
modfit1 <- rpart(classe ~ ., data = train_data, method="class")
pred_train_1 <- predict(modfit1, newdata=train_data, type ="class")
confusionMatrix(pred_train_1, train_data$classe)$overall[1]
```

```
## Accuracy
##   0.7556
```

The accuracy is very low. However, we could choose the more significant variables discarding the ones not used in the model *Regression Tree* model; the cirteria has been discard the variables which *Importance* is equally zero.

```
vars <- varImp(modfit1)
combi <- combi[, c(rownames(vars)[vars$Overall>0],"classe")]
sum(vars > 0)
```

```
## [1] 36
```

17 additional variables discarded, and only **36 variables** left.

4. Model Tunning

It is time to aplly the *Random Forest* model whith the 36 predictors left.

```
train_data <- combi[c(21:19642),]
modfit3 <- randomForest(classe ~ ., data = train_data, importance=TRUE)
pred_train_3 <- predict(modfit3, newdata=train_data, type ="class")
confusionMatrix(pred_train_3, train_data$classe)$overall[1]
```

```
## Accuracy
##        1
```

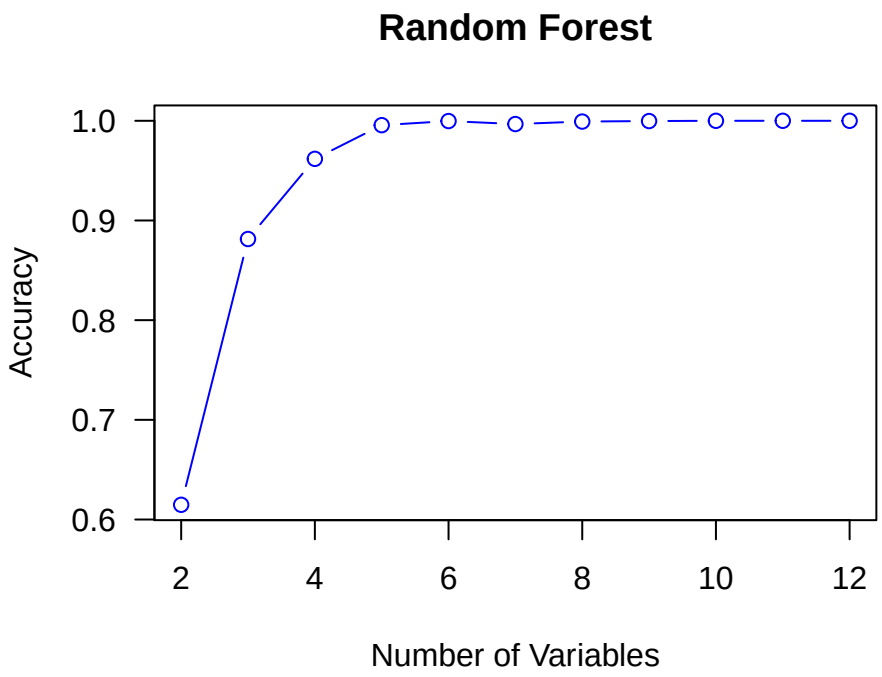
Using the 36 variables in the model we have achived a perfect Accuracy. The model is nearly perfect, however, perhaps we could further discard some of the variables. Again, we would use the *Importance* of the variables.

```
vars <- importance(modfit3)
summary(vars[,"MeanDecreaseGini"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     124     257     330     431     477    1450
```

We can see that the *MeanDecreaseGini* of the variable (the most suitable for a rf model) has a high variance among the predictors, specially on the 4th quartile, with very high values. First, we order the variables according to its importance. Then, step by step, we will fit different rf model, ranging from 2 predictors, up to 12. Then we will study the Accuracy of each of these models and its variation.

```
Giny <- vars[order(vars[,"MeanDecreaseGini"], decreasing = TRUE) ,"MeanDecreaseGini"]
for (i in c(1:11)) {
    train_data <- combi[c(21:19642),c(names(Giny[1:i+1]), "classe")]
    modfit4 <- randomForest(classe ~ ., data = train_data, importance=TRUE)
    pred_train_4 <- predict(modfit4, newdata = train_data, type ="class")
    y[i] <- confusionMatrix(pred_train_4, train_data$classe)$overall[1]
}
plot(c(2:12),y,type="b", las=1, col="blue",
     xlab="Number of Variables", ylab="Accuracy", main="Random Forest")
```



The model rapidly converge to a perfect Accuracy, with 10 variables or more, the Accuracy is perfect. And with only six variables the Accuracy is *0.9996942*. I chose this model as the solution of this assigment. Let's generate the answer for the 20 questions of the test data.

```
data1 = combi[,c(names(Giny[1:6]), "classe")]
modfit5 <- randomForest(classe ~ ., data = data1[c(21:19642),], importance=TRUE)
pred_test_5 <- predict(modfit5, newdata=data1[c(1:20),], type ="class")
pred_test_5
```

```
##  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
```

20 out of 20. A perfect match!

5. Conclusion

A Random Forest model with only six variables has been chosen. This variables are:

```
names(importance(modfit5)[,"MeanDecreaseGini"])
```

```
## [1] "roll_belt"         "yaw_belt"          "pitch_forearm"
## [4] "magnet_dumbbell_z" "pitch_belt"        "magnet_dumbbell_y"
```

The advantages of this model are:

- Very simple one. Only 6 variables out of 160. The experiment of taking thedata can be greatly simplified. Some sensoors can be avoided.

- All the variables are measurements. All external variables (time, user:name, windows) has been discarded
- Almost a perfect Accuracy
- 20 correct hits out of 20, in the course project submission

Appendix

A1. Final model

```
pred_train_5 <- predict(modfit5, newdata=data1[c(21:19642),], type ="class")
confusionMatrix(pred_train_5, data1[c(21:19642),"classe"])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5577    0    0    0    0
##          B    0 3797    2    0    0
##          C    2    0 3420    0    0
##          D    1    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    1.000    0.999    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    0.999    0.999    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```