Practical Machine Learning - Assignment: Prediction Assignment Writeup

Date - 1/30/2015

1. Summary

Prediction Assignment Writeup

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, 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: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases. Peer Review Portion

2. Data

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

3. Data Set setup and Preparation

Load libraries

library(dplyr) library(ggplot2)
library(grid)
library(gridExtra)
library(scales)

Environment variables and Data Loading

setwd("C:/Data-Scientist/practical-machine-learning/PredictionAssignmentWriteup") training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", "")) testing <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", "")) str(training, list.len=15) table(training*classe)prop.table(table(training*user_name, training*classe), 1)prop.table(table(training*classe))

We can do data clean-up by removing columns 1 to 6, training <- training[, 7:160] testing <- testing[, 7:160]

is_data <- apply(!is.na(training), 2, sum) > 19621 # number of observations training <- training[, is_data] testing <- testing[, is_data]

Split training set into two for cross validation Also subsample 60% of the set for training purposes (actual model building) 40% remainder will be used only for testing, evaluation and accuracy measurement.

library(caret) set.seed(3141592) inTrain <- createDataPartition(y=training$classe, p=0.60, list=FALSE) train1 <- training[inTrain,] train2 <- training[-inTrain,] dim(train1) dim(train2)

train1 is the training data set (it contains 11776 observations, or about 60% of the entire training data set) train2 is the testing data set (it contains 7846 observations, or about 40% of the entire training data set). The dataset train2 will never be looked at, and will be used only for accuracy measurements.

Lets examine 1. Identify the "zero covariates"" from train1 2. Remove these "zero covariates"" from both train1 and train2:

nzv_cols <- nearZeroVar(train1) if(length(nzv_cols) > 0) { train1 <- train1[, -nzv_cols] train2 <- train2[, -nzv_cols] } dim(train1) dim(train2)

This didn't give any results anything as the earlier removal of NA was sufficient to clean the data.

We are satisfied that we now have 53 clean covariates to build a model for classe (which is the 54th column of the data set).

4. Graps and Plots

53 covariates is a lot of variables. Let's look at their relative importance using the output of a quick Random Forest algorithm (which we call directly using randomForest() rather than the caret package purely for speed purposes as we cannot specify the number of trees to use in caret), and plotting data importance using varImpPlot():

library(randomForest)

set.seed(3141592) fitModel <- randomForest(classe~., data=train1, importance=TRUE, ntree=100) varImpPlot(fitModel)

Using the Accuracy and Gini graphs above, we select the top 10 variables that we'll use for model building. If the accuracy of the resulting model is acceptable, limiting the number of variables is a good idea to ensure readability and interpretability of the model. A model with 10 parameters is certainly much more user friendly than a model with 53 parameters.

Our 10 covariates are: yaw_belt, roll_belt, num_window, pitch_belt, magnet_dumbbell_y, magnet_dumbbell_z, pitch_forearm, accel_dumbbell_y, roll_arm, and roll_forearm.

Let's analyze the correlations between these 10 variables. The following code calculates the correlation matrix, replaces the 1s in the diagonal with 0s, and outputs which variables have an absolute value correlation above 75%:

correl = cor(train1[,c("yaw_belt","roll_belt","num_window","pitch_belt","magnet_dumbbell_z","magnet_dumbbell_y","pi
diag(correl) <- 0 which(abs(correl)>0.75, arr.ind=TRUE)

So we may have a problem with roll_belt and yaw_belt which have a high correlation (above 75%) with each other:

cor(train1$roll_belt, train$1yaw_belt)

These two variables are on top of the Accuracy and Gini graphs.

By re-running the correlation script above (eliminating yaw_belt) and outputting max(correl), we find that the maximum correlation among these 9 variables is 50.57% so we are satisfied with this choice of relatively independent set of covariates.

We can identify an interesting relationship between roll_belt and magnet_dumbbell_y:

qplot(roll_belt, magnet_dumbbell_y, colour=classe, data=train1)

This graph suggests that we could probably categorize the data into groups based on roll_belt values.

Incidentally, a quick tree classifier selects roll_belt as the first discriminant among all 53 covariates (which explains why we have eliminated yaw_belt instead of roll_belt, and not the opposite: it is a "more important" covariate):

library(rpart.plot)

fitModel <- rpart(classe~., data=train1, method="class") prp(fitModel)

Random Forest algorithm are satisfactory.

5. Building Model

For medelling we will be using Random Forest algorithm, using the train() function from the caret package. Use 9 variables out of the 53 as model parameters which are most significant from initial Random Forest algorithm. Those are roll_belt, num_window, pitch_belt, magnet_dumbbell_y, magnet_dumbbell_z, pitch_forearm, accel_dumbbell_y, roll_arm, and roll_forearm.

These variable are relatively independent as the maximum correlation among them is 50.57%. We are using a 2-fold cross-validation control. This is the simplest k-fold cross-validation possible and it will give a reduced computation time. Because the data set is large, using a small number of folds is justified.

set.seed(3141592) fitModel <- train(classe~roll_belt+num_window+pitch_belt+magnet_dumbbell_y+magnet_dumbbell_z+
data=train1, method="rf", trControl=trainControl(method="cv",number=2), prox=TRUE, verbose=TRUE, allowParallel=TRUE)

saveRDS(fitModel, "modelRF.Rds") We can later use this tree, by allocating it directly to a variable using the command: fitModel <- readRDS("modelRF.Rds")

Accuracy of the model

We can use caret's confusionMatrix() function applied on train2 (the test set) to get an idea of the accuracy:

predictions <- predict(fitModel, newdata=train2) confusionMat <- confusionMatrix(predictions, train2$classe) confusionMat

99.77% accuracy which totally validates the idea / hypothesis we made to eliminate most variables and use only 9 relatively independent covariates.

Estimation of the out-of-sample error rate The train2 test set was removed and left untouched during variable selection, training and optimizing of the Random Forest algorithm. Therefore this testing subset gives an unbiased estimate of the Random Forest algorithm's prediction accuracy (99.77% as calculated above). The Random Forest's out-of-sample error rate is derived by the formula 100% - Accuracy = 0.23%, or can be calculated directly by the following lines of code:

```
missClass = function(values, predicted) { sum(predicted != values) / length(values) } OOS_errRate = missClass(train2$classe, predictions) OOS_errRate
```

The out-of-sample error rate is 0.23%.

Coursera's challenge and predict the 20 observations in testing (recall that testing corresponds to the data set pml-testing.csv)

Predict the classification of the 20 observations of the testing data set for Coursera's "Course Project: Submission" challenge page:

```
predictions <- predict(fitModel, newdata=testing) testing$classe <- predictions
```

Create one .CSV file with all the results, presented in two columns (named problem_id and classe) and 20 rows of data:

```
submit <- data.frame(problem_id = testing$problem_id, classe = predictions) write.csv(submit, file = "coursera-submission.csv", row.names = FALSE)
```

Create twenty .TXT file that we will upload one by one in the Coursera website (the 20 files created are called problem_1.txt to problem_20.txt):

```
answers = testing$classe write_files = function(x){ n = length(x) for(i in 1:n){ filename = paste0("problem_",i,".txt") write.table(x[i], file=filename, quote=FALSE, row.names=FALSE, col.names=FALSE) } } write_files(answers)
```

Result 20/20

6. Conclusion and Acknowledgement

From above project we did predict the classification of 20 observations using a Random Forest algorithm trained on a subset of data using less than 20% of the covariates.

The accuracy obtained (accuracy = 99.77%, and out-of-sample error = 0.23%) is obviously highly suspicious as it is never the case that machine learning algorithms are that accurate, and a mere 85% if often a good accuracy result.

Either the 6 participants are with high performance .

Ref / help from : Arnaud Desombre

————————————————EOF————————————-