

Practical Machine Learning project

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages (click the **Help** toolbar button for more details on using R Markdown).

When you click the **Knit HTML** button a web page will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(caret)
```

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

```
library(ggplot2)
library(randomForest) # I'm using the package directly as it is much faster than through caret
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
setwd("/Users/krochek/Copy/Coursera/Practical Machine Learning/project")
pmltraining <- read.csv("./pml-training.csv")
pmltesting <- read.csv("./pml-testing.csv")
```

Now, lets clean the data up a bit:

The following code makes sure there are no predictors that have very low variance due to missing values mostly. I must admit that the test data given (20 cases) - helped to better understand which are the ones not needed as well - Given the fact that if a variable is empty or NA in all 20 rows - it can't add any information for prediction, therefore it's not significant for the prediction model.

By using nearZeroVar on the 20 testing rows - you can easily "cheat" and get rid of all the columns the same way as written above. The logic behind it is if it has constant values in the test set for all the 20 rows - it means they have no or very very little influence on the classification.

The drops list includes all the colnames that I took off as well based on general logic:

- X - is just an index
- user_name - shouldn't have any effect
- The 3 timestamp variables - these all represent the time at the moment of the exercise and is only there to help with the windows which are relative to one another
- new window - by the testing data - adds no information to the model (all values the same)
- problem_id - same as X.

```
numnas <- sapply(pmltesting,function(x) sum(is.na(x)))
numnas <- data.frame(numnas)
isnas <- ifelse(numnas$numnas>4,row(numnas),NA)
isnas <- isnas[!is.na(isnas)]
trainingtemp <- pmltraining[,-isnas]
testingtemp <- pmltesting[,-isnas]
drops <-
c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","new_window","problem_id")

trainingtemp <- trainingtemp[,!names(trainingtemp) %in% drops]
testingtemp <- testingtemp[,!names(testingtemp) %in% drops]

#
```

partitioning the data

I've partition the processes pmltraining data into ¾ training and ¼ testing for the cross validation - to understand the out of sample error rate and see that the model doesn't overfit.

```
inTrain <- createDataPartition(y=trainingtemp$classe, p =0.75, list = FALSE)
training <- trainingtemp[inTrain,]
testing <- trainingtemp[-inTrain,]
```

Fitting the model

I've tried a few models that had little to no succes (a regression tree got me 55% accurace which seemed too low for me). Eventually I've struck gold with random forest and ran it directly from the randomForest package - the implementation must be a lot more efficient as it takes a fraction of the time it would running through caret's train function.

```
set.seed(1234) # for reproducible research
rfmodFit100 <- randomForest(training$classe ~., data = training[,-54], ntree = 100)

predict20 <- predict(rfmodFit100,testingtemp)
print(predict20)
```

```
##  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
```

#The following function helps write the prediction results into 20 seperate txt file for the submission of the answers

```
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)
  }
}

pml_write_files(predict20)
```

Cross validation and out of sample error expectation

From the confusion Matrix presented below, It's easy to see that the accuracy is very very high - Which scared me at first when I did it on the training data as I thought i had a clear case of overfitting but after running it on the cross validation data (¼ of the pml training data), it seems that the accurace is above 99.5%. It's actually is closer to 99.7%.

My expectation for the out of sample error- theoretically it should be around 0.3%.

Actual out of sample error:

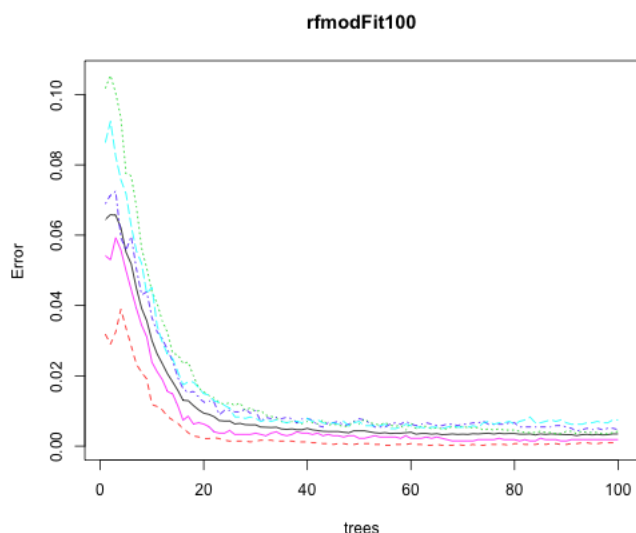
We only had 20 out of samples to work with - **I got all of them right 20/20**. so the empirical error is zero.

clarification - If i had a 1000 out of sample entries, I would have expected to see 0.3% errors - meaning 3 errors more or less.

Some Plots

You can see the error is already very low with only 50 trees but I took 100 to be on the very very safe side :)

```
plot(rfmodFit100)
```



```
confusionMatrix(testing$classe,predict(rfmodFit100,testing[,-54]))
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A    B    C    D    E
##    A 1395    0    0    0    0
##    B     2  946    1    0    0
##    C     0    2  850    3    0
##    D     0    0    6  798    0
##    E     0    0    0    2  899
##
## Overall Statistics
##
##           Accuracy : 0.997
##           95% CI : (0.995, 0.998)
##    No Information Rate : 0.285
##    P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.996
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.999    0.998    0.992    0.994    1.000
## Specificity      1.000    0.999    0.999    0.999    1.000
## Pos Pred Value    1.000    0.997    0.994    0.993    0.998
## Neg Pred Value    0.999    0.999    0.998    0.999    1.000
## Prevalence        0.285    0.193    0.175    0.164    0.183
## Detection Rate    0.284    0.193    0.173    0.163    0.183
## Detection Prevalence 0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy 0.999    0.999    0.995    0.996    1.000

```