Coursera Project - ML

iac

22 novembre 2015

Grab the data from url and leaving out NA values

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Create data partition.

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Transformate the data, leaving out near zero var predictors.

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
```

```
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")

myTraining <- myTraining[!myNZVvars]
myTraining <- myTraining[c(-1)]
trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  {
                trainingV3 <- trainingV3[ , -j]
            }
        }
    }
}

dim(trainingV3)
```

```
## [1] 11776    58
```

```
myTraining <- trainingV3
rm(trainingV3)
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])
myTesting <- myTesting[clean1]
testing <- testing[clean2]

dim(myTesting)
```

```
## [1] 7846   58
```

```
dim(testing)
```

```
## [1] 20 57
```

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Do some ML. After many trials (not shown), Random Forests is the algorithm that provides best results.

```r
modFitB1 <- randomForest(classe ~. , data=myTraining)

predictionsB1 <- predict(modFitB1, myTesting, type = "class")

confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    1    0    0
##          C    0    0 1363    4    0
##          D    0    0    4 1282    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.9977, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9984
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9963   0.9969   1.0000
## Specificity            0.9998   0.9998   0.9994   0.9994   1.0000
## Pos Pred Value         0.9996   0.9993   0.9971   0.9969   1.0000
## Neg Pred Value         1.0000   0.9998   0.9992   0.9994   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1737   0.1634   0.1838
## Detection Prevalence   0.2846   0.1935   0.1742   0.1639   0.1838
## Balanced Accuracy      0.9999   0.9996   0.9979   0.9981   1.0000
```

Submit to Coursera and generate files.

```r
predictionsB2 <- predict(modFitB1, testing, type = "class")
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(predictionsB2)
```