Machine Learning Project

Steven Toews

05/04/2020

Setting Up

Set up, including installing necessary libraries and setting the seed.

Getting, Cleaning Data

Create train URL variable:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

Create test URL variable:

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Loading data into RAM:

```
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Partition

Partition data into training and testing set (60/40)

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

Clean The Data

Transformation 1: Cleaning Near Zero Variance Variables

```
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"
                                      "max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_a
                                      "var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_a
                                      "stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_p
                                      "kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "s
                                      "max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_a
                                      "kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw
                                      "skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbb
                                      "amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth
                                      "skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_
                                      "max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitu
                                      "amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm
                                      "avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm",
                                      "stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
```

Transformation 2: Removing first ID variable

```
myTraining <- myTraining[c(-1)]
```

Transformation 3: Removing variables with > 60% NA values

```
trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
    if( sum(is.na(myTraining[, i]))/nrow(myTraining)>=0.6 ) {
        for(j in 1:length(trainingV3)) {
            if( length(grep(names(myTraining[i]), names(trainingV3)[j])) ==1)  {
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}

myTraining <- trainingV3
rm(trainingV3)
```

Same transformations done for myTesting and testing sets:

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]
```

Coercing data into the same type:

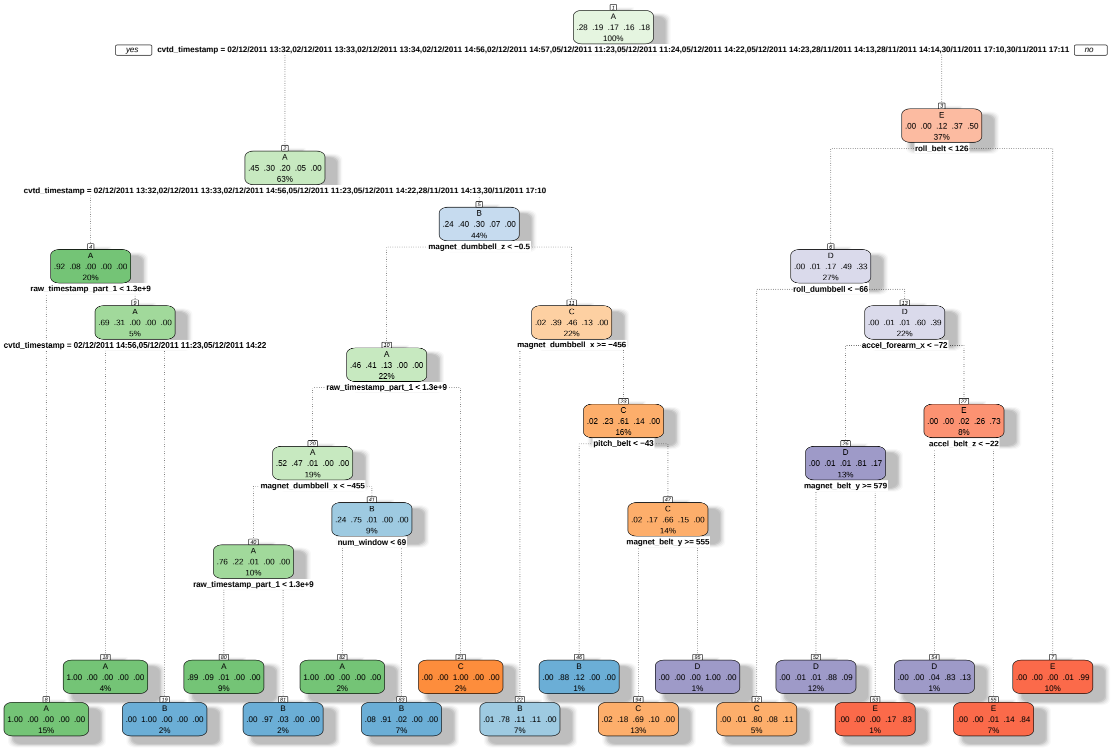
```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

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

Decision Tree

Run Decision Tree and view with Fancy

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



Rattle 2020−Apr−05 19:04:09 Steve

Prediction

Running prediction

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 2157   68   10    1    0
##          B   60 1265   73   67    0
##          C   15  177 1261  141   70
##          D    0    8   15  962  111
##          E    0    0    9  115 1261
##
```

```
## Overall Statistics
##
##                Accuracy : 0.8802
##                  95% CI : (0.8728, 0.8873)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8484
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9664   0.8333   0.9218   0.7481   0.8745
## Specificity            0.9859   0.9684   0.9378   0.9796   0.9806
## Pos Pred Value         0.9647   0.8635   0.7578   0.8777   0.9105
## Neg Pred Value         0.9866   0.9604   0.9827   0.9520   0.9720
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2749   0.1612   0.1607   0.1226   0.1607
## Detection Prevalence   0.2850   0.1867   0.2121   0.1397   0.1765
## Balanced Accuracy      0.9762   0.9009   0.9298   0.8638   0.9276
```

Random Forest

Run Random Forest and compare with Decision Tree

```
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    5    0    0    0
##          B    0 1513    2    0    0
##          C    0    0 1361    6    0
##          D    0    0    5 1279    1
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9961, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                       Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9967   0.9949   0.9946   0.9993
## Specificity            0.9991   0.9997   0.9991   0.9991   0.9998
## Pos Pred Value         0.9978   0.9987   0.9956   0.9953   0.9993
## Neg Pred Value         1.0000   0.9992   0.9989   0.9989   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1928   0.1735   0.1630   0.1837
## Detection Prevalence   0.2851   0.1931   0.1742   0.1638   0.1838
## Balanced Accuracy      0.9996   0.9982   0.9970   0.9968   0.9996
```

Random Forests yield more accurate results.

Generate files for project submission

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

pml_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)
  }
}

pml_write_files(predictionsB2)
```