Machine Learning

Kyle

June 11, 2017

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

R Markdown

The goal of your project is to predict the manner in how well 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

Load the testing and training pml data and remove the index column.

```
dftrain <- read.csv("C:/Users/Kyle/Downloads/CourseraData/pml_training.csv",stringsAsFactors = FALSE)

dftest <- read.csv("C:/Users/Kyle/Downloads/CourseraData/pml_testing.csv", stringsAsFactors = FALSE)

dftrain <- dftrain[,-1]
dftest <- dftest[,-1]

dftrain$classe <- factor(dftrain$classe)
```

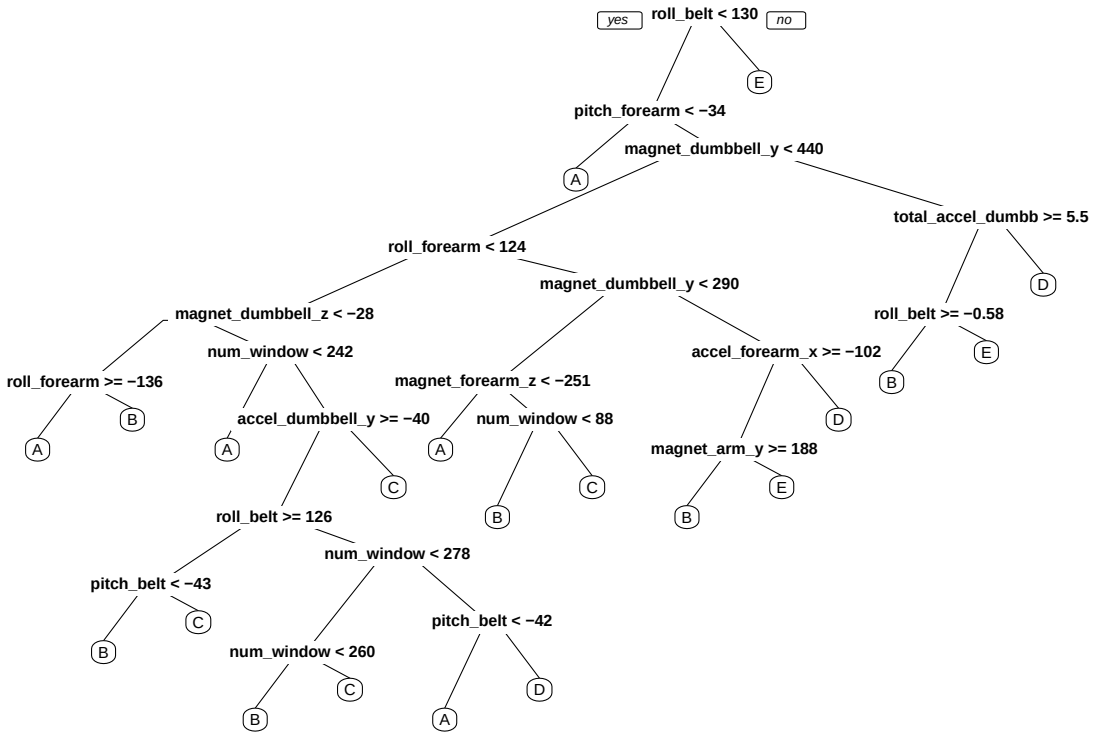
Remove the variables with near zero variance and columns with greater than 90% NA.

```
nzv <- nearZeroVar(dftrain)
dftrain <- dftrain[,-nzv]
dftest <- dftest[,-nzv]

rmna <- sapply(dftrain,function(x)mean(is.na(x))>.9)
dftrain <- dftrain[,rmna==FALSE]
dftest <- dftest[,-rmna==FALSE]
dftrain <- dftrain[,-(1:4)]
```

Make a tree to see a logical and easy to follow breakdown of how to classify each sample. I will still use randomforest for the final model as it will provide over 99% accuracy.

```
fitModel1 <- rpart(classe~., data=dftrain, method="class")
prp(fitModel1)
```




```
dftest <- dftest[,-(1:4)]
```

Making a plot to loo at the densities of the IVs and DV. For some of the IVs it is very easy to see differences between the classes and in others not so much. Since there are so many variables is spans a couple of pages so I will not include the output in this paper.

```
featurePlot(x=dftrain[,1:53],
            y=dftrain$classe, plot="density", scales=list(x=list(relation="free"),
            y = list(relation="free")), adjust = 1.5, pch = "|",
            layout = c(4, 1), auto.key = list(columns = 5))
```

In the following code I make two different models. One is random forest and the other bayesian. The bayesian model had roughly 90% accuracy and the random forest model had over 99% accuracy.

```
##  [1] 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
```

```
##  [1] E C C A B A C B A C C C B A B B C B D B
## Levels: A B C D E
```

Random Forest cross table, unfortunately I had to comment the CrossTables out as the knit process kept having an error at these lines.

```
#CrossTable(dftest$classe,pred_rf,
#            prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
#            dnn=c('Category','Predicted'))
```

```
dftest$classe <- c("B","A","C","A","A","E","D","D","A","A","C",
                   "C","B","A","E","E","A","B","B","B")
```

Bayes cross table

```
#CrossTable(dftest$classe,pred_nb,
#           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
#           dnn=c('Category','Predicted'))
```

Out of sample error calcuation. In a larger dataset the error would be greater than 0.

```
error = function(values, predicted) {
  sum(predicted != values) / length(values)
}
errnb = error(dftest$classe, pred_nb)
errRF = error(dftest$classe, pred_rf)
errnb
```

```
## [1] 0.55
```

```
errRF
```

```
## [1] 0.15
```

Conclusion

The random forest had a 100% success rate on the quiz. The bayesian model predicted 12 out of the 20 correct. The out of sample error rate was 55% for naiveBayes and 0% for random forest.