Practical Machine Learning

Wellintton Perez

September 2019

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

DATA PROCESSING

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. The data processing I conducted consists of eliminating predictors with very low variance as well as removing columns with more than 65% NA.

DATA LOADING AND CLEANING

The code below describes step-by-step how I loaded the data, split the training set into two sets training=0.70, testing=0.30.

```
train_file<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_file<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training<-read.csv(train_file ,sep=",",nrows=-1,na.strings=c("NA","#DIV/0!",""))
testing<-read.csv(test_file ,sep=",",nrows=-1,na.strings=c("NA","#DIV/0!",""))

train_part<-createDataPartition(training$classe,p=0.70,list=F)
train_ds<-training[train_part,]
test_ds<-training[-train_part,]

dim(train_ds)
```

```
## [1] 13737   160
```

```
dim(test_ds)
```

```
## [1] 5885  160
```

In this of code snippet I remove columns with near zero variance using the function nearZeroVar() from the caret package. I also remove the first column from the training an testing datasets.

```
nzv=nearZeroVar(train_ds,saveMetrics = T)
train_ds<-train_ds[,nzv$nzv==FALSE]

nz=nearZeroVar(test_ds,saveMetrics = T)
test_ds<-test_ds[,nz$nzv==FALSE]
```

```
train_ds<-train_ds[,-1] #remove fist column
test_ds<-test_ds[,-1] #remove fist column

testing<-testing[,-1]
```

Following is removing NA from the training dataset for any covariate that has more than 65% NA. This will help me run the model functions without using the default method to remove NA which I think this gives me more control and better prediction.

```
remove_cols<-c()
for(i in 1:length(train_ds)){
  if((sum(is.na(train_ds[,i])) / nrow(train_ds)) > .65){
    remove_cols<-c(remove_cols,i)
  }
}
train_ds<-train_ds[,-c(remove_cols)]

remove_cols<-c()
for(i in 1:length(test_ds)){
  if((sum(is.na(test_ds[,i])) / nrow(test_ds)) > .65){
    remove_cols<-c(remove_cols,i)
  }
}
test_ds<-test_ds[,-c(remove_cols)]
```

In this section I am cleaning the data sets to contain the clean columns.

```
clean_test_ds<-colnames(train_ds)
clean_test<-colnames(train_ds[,-58])

test_ds<-test_ds[clean_test_ds]
testing<-testing[clean_test]


# To get the same class between testing and train_ds
testing <- rbind(train_ds[2, -58] , testing)
testing <- testing[-1,]
```

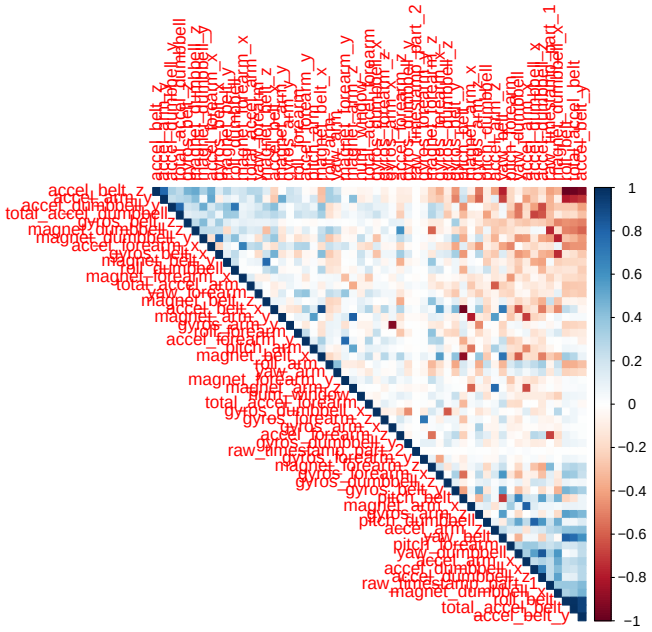
Investigating correlated predictors

```
corrtable = cor(train_ds[, c(-1,-4,-58)]) # remove the non-numeric columns
summary(corrtable[upper.tri(corrtable)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.991950 -0.108455  0.003347  0.001684  0.098575  0.980696
```

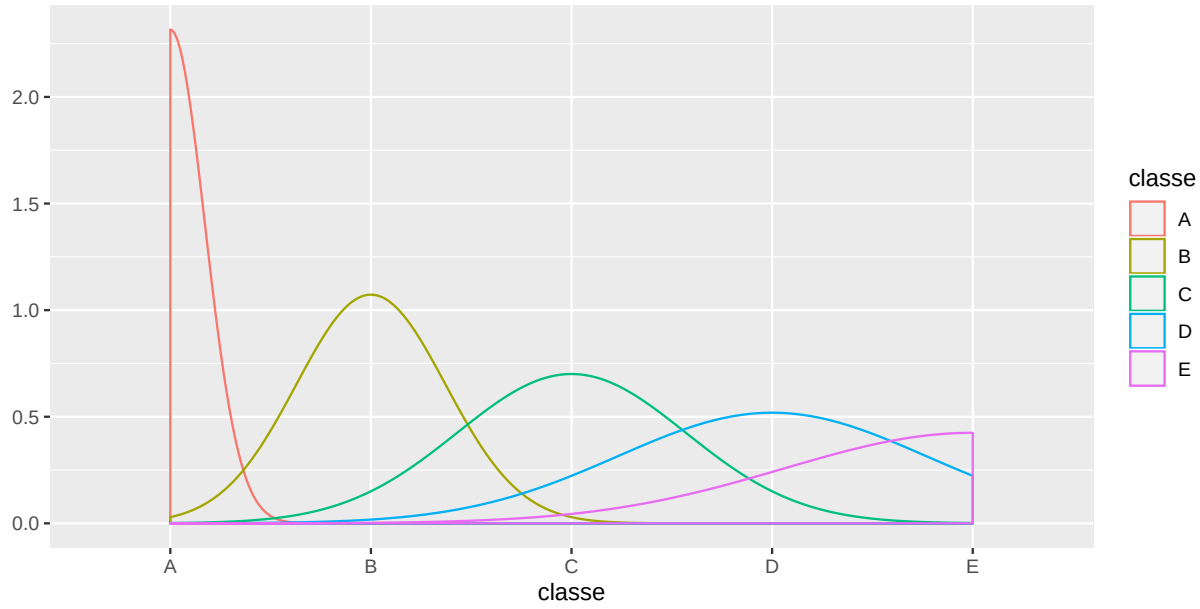
This shows the correlation between the predictors

```
corrplot(corrtable, order="FPC", method="color", type = "upper")
```



The following density plot shows that classe A should be the prevalent prediction from this data set.

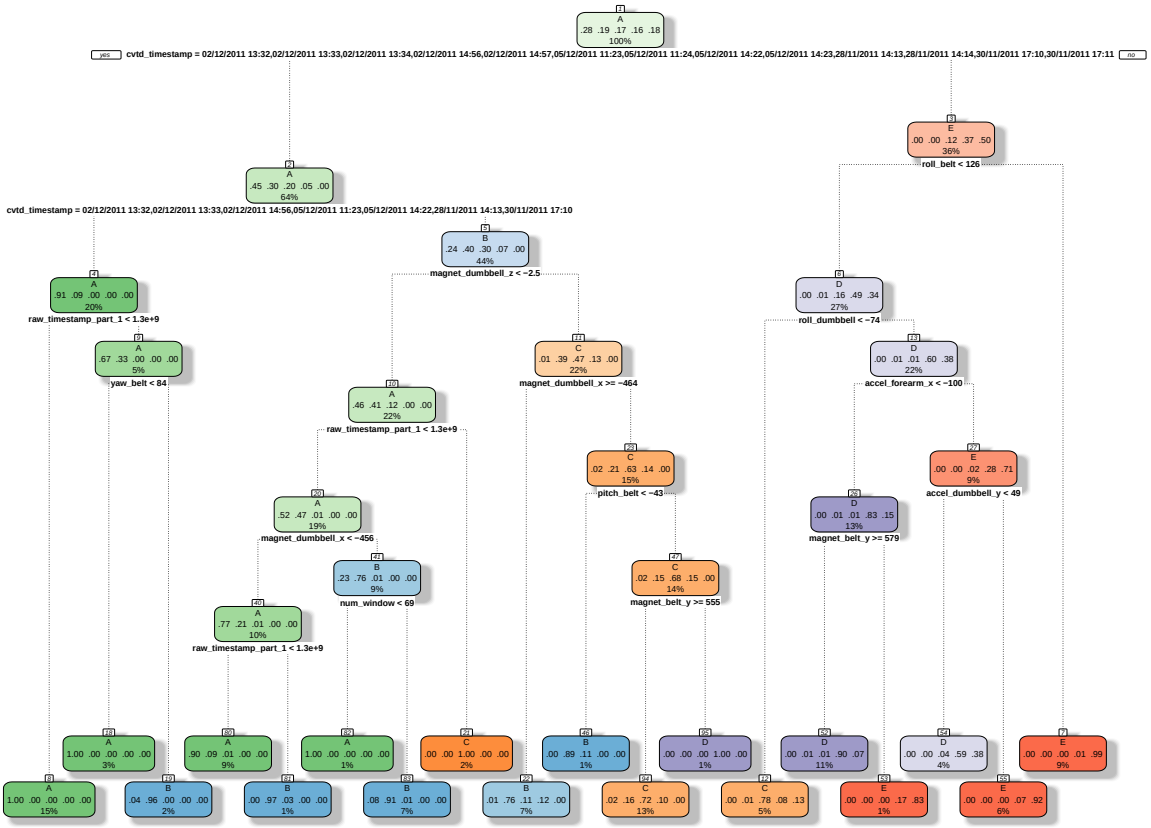
```
qplot(classe, colour=classe, data=train_ds[, c(-1,-4)],geom="density")
```



Prediction with decision trees

I am going to start by analyzing the data set using decision trees.

```
set.seed(423423)
fit1 <- rpart(classe ~ ., data=train_ds, method="class")
fancyRpartPlot(fit1)
```



Rattle 2019−Sep−29 12:16:31 saigsa

The decision trees prediction has an accuracy of about 88%. This is a very good prediction rate and we also have a very low P-value. But the table below shows misses on all predictors.

```
predict1 <- predict(fit1, test_ds, type = "class")
cmtree <- confusionMatrix(predict1, test_ds$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1609   49    4    2    0
##          B   48  930   63   41    0
##          C   17  152  929  112   40
##          D    0    8   30  756  128
##          E    0    0    0   53  914
##
## Overall Statistics
##
##                Accuracy : 0.8731
##                  95% CI : (0.8643, 0.8815)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8395
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9612   0.8165   0.9055   0.7842   0.8447
## Specificity            0.9869   0.9680   0.9339   0.9663   0.9890
## Pos Pred Value         0.9669   0.8595   0.7432   0.8200   0.9452
## Neg Pred Value         0.9846   0.9565   0.9791   0.9581   0.9658
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2734   0.1580   0.1579   0.1285   0.1553
## Detection Prevalence   0.2828   0.1839   0.2124   0.1567   0.1643
## Balanced Accuracy      0.9741   0.8922   0.9197   0.8752   0.9168
```

Prediction with random forest

The next method I will be using is random forest. Random forest uses bagging internally to come up with the best prediction.

```
set.seed(423423)
rfMod1 <- randomForest(classe ~ ., data=train_ds)
rfPredict1 <- predict(rfMod1, test_ds, type = "class")
cmrf <- confusionMatrix(rfPredict1, test_ds$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.998, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9979   1.0000
## Specificity            0.9998   0.9996   0.9996   1.0000   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   1.0000   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   1.0000
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1839
## Detection Prevalence   0.2846   0.1937   0.1743   0.1635   0.1839
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9990   1.0000
```

Predicting results

Random Forest has a 99.9% accuracy on the testing data partition of the data set. The result is more than 10% better than using decision trees therefore I will use random forest for my final prediction model.

```
rfPredict2 <- predict(rfMod1, testing, type = "class")
rfPredict2
```

```
##  1 21  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
```

Write the results to a file for submission.

```
write.table(rfPredict2,file="results.txt", col.names=F)
```