2020_0828_Prac_Mach_Learn_FP

John Nguyen

8/28/2020

Include the libraries and then clean the Data

To complete this project we need to include specific libaries.

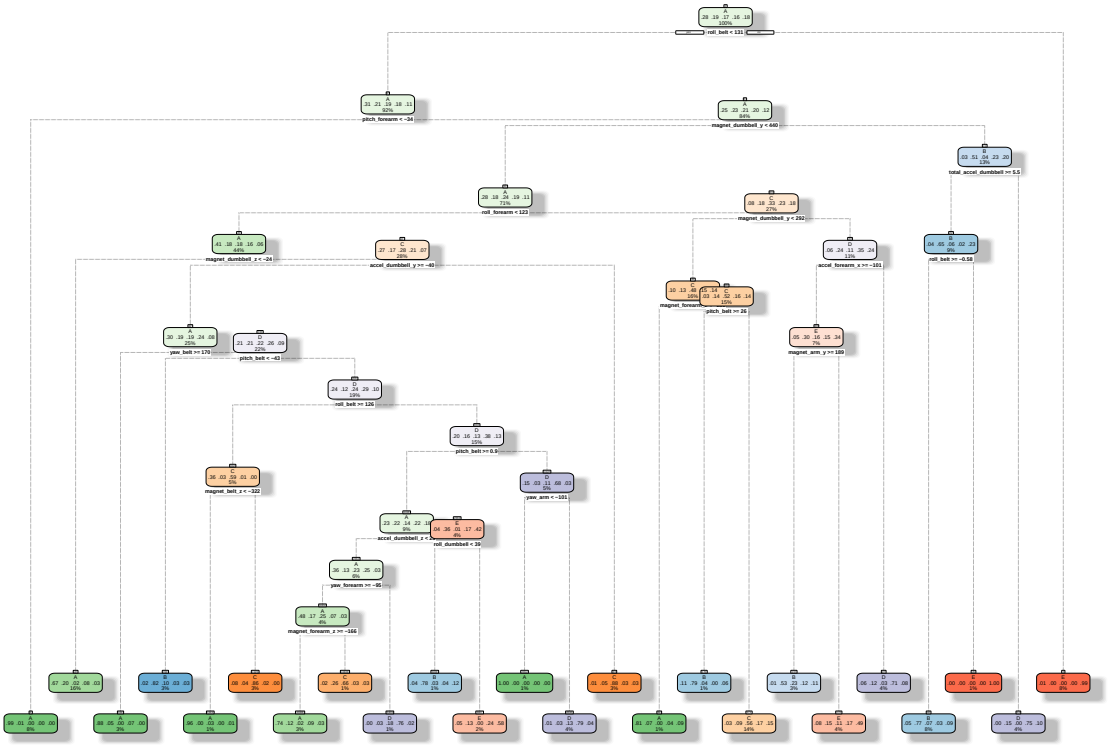
Then we need to get rid of spurious variables. Any variable that include the words, such as kurtosis, Max, Min, var, stddev, avg,skewness, timestamp, raw, name, and window can be removed because they will not help with prediction modeling. We further clean the data with nearzerovar function. This removes variables with approximately 0 variance.

```
library(lattice)
library(ggplot2)
library(caret)
library(dplyr)
library(rpart)
library(rpart.plot)
library(tibble)
library(bitops)
library(rattle)
library(randomForest)

train <- read.csv(file = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", header=
final_test<-read.csv(file = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", head
train<-data.frame(train)

new_train<-train%>%
    select(-contains('kurtosis'))%>%
    select(-contains('skewness'))%>%
    select(-contains('stddev'))%>%
    select(-contains('max'))%>%
    select(-contains('min'))%>%
    select(-contains('var'))%>%
    select(-contains('avg'))%>%
    select(-contains('raw'))%>%
    select(-contains('timestamp'))%>%
    select(-contains('name'))%>%
    select(-contains('window'))
new_train<- new_train[, colSums(is.na(new_train)) == 0]
nearZvar <- nearZeroVar(new_train)
new_train<-new_train[,-nearZvar]
new_train<-new_train[,-1]

inValidation = createDataPartition(new_train$classe, p = 3/4)[[1]]
training = new_train[ inValidation,]
validation = new_train[-inValidation,]
```

Tree Method

```
set.seed(11122)
Tree_mod <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(Tree_mod)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020−Aug−29 19:11:41 johnk

```
predictTreeVal<-predict(Tree_mod, newdata=validation, type="class")
testclassvalid<-as.factor(validation$classe)
confusionMatrix(predictTreeVal,factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1280  195   21   85   40
##          B   43  570   76   25   60
##          C   31   86  686  125  114
##          D   22   65   50  510   54
##          E   19   33   22   59  633
##
## Overall Statistics
##
##                Accuracy : 0.7502
```

```
##                  95% CI : (0.7378, 0.7623)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6823
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9176   0.6006   0.8023   0.6343   0.7026
## Specificity            0.9028   0.9484   0.9121   0.9534   0.9668
## Pos Pred Value         0.7896   0.7364   0.6583   0.7275   0.8264
## Neg Pred Value         0.9650   0.9082   0.9562   0.9300   0.9352
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2610   0.1162   0.1399   0.1040   0.1291
## Detection Prevalence   0.3305   0.1578   0.2125   0.1429   0.1562
## Balanced Accuracy      0.9102   0.7745   0.8572   0.7939   0.8347
```

```
out_sample_error_tree<-1-confusionMatrix(predictTreeVal,factor(validation$classe))$overall[[1]]
out_sample_error_tree
```

```
## [1] 0.2497961
```

Linear Discriminant Analysis

```
set.seed(87234)
modelda  <- train(classe ~ ., data=training, method = "lda")
predictlda <- predict(modelda, newdata=validation)
confusionMatrix(predictlda,factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1135  134   78   47   27
##          B   34  622   82   27  154
##          C  114  116  579  111   83
##          D  108   39   96  582  105
##          E    4   38   20   37  532
##
## Overall Statistics
##
##                Accuracy : 0.7035
##                  95% CI : (0.6905, 0.7163)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6251
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8136   0.6554   0.6772   0.7239   0.5905
## Specificity            0.9185   0.9249   0.8953   0.9151   0.9753
## Pos Pred Value         0.7987   0.6768   0.5773   0.6258   0.8431
## Neg Pred Value         0.9254   0.9179   0.9292   0.9441   0.9136
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2314   0.1268   0.1181   0.1187   0.1085
## Detection Prevalence   0.2898   0.1874   0.2045   0.1896   0.1287
## Balanced Accuracy      0.8661   0.7902   0.7862   0.8195   0.7829
```

```
out_sample_error_lda<-1-confusionMatrix(predictlda,factor(validation$classe))$overall[[1]]
out_sample_error_lda
```

```
## [1] 0.2964927
```

Random Forest

```
set.seed(98427)
controlfunction <- trainControl(method="cv", number=3, verboseIter = FALSE)
rand_for <- train(classe ~ ., data=training, method="rf", trControl=controlfunction)
predrf<-predict(rand_for, newdata=validation)
confusionMatrix(predict(rand_for, newdata=validation), factor(validation$classe))
```

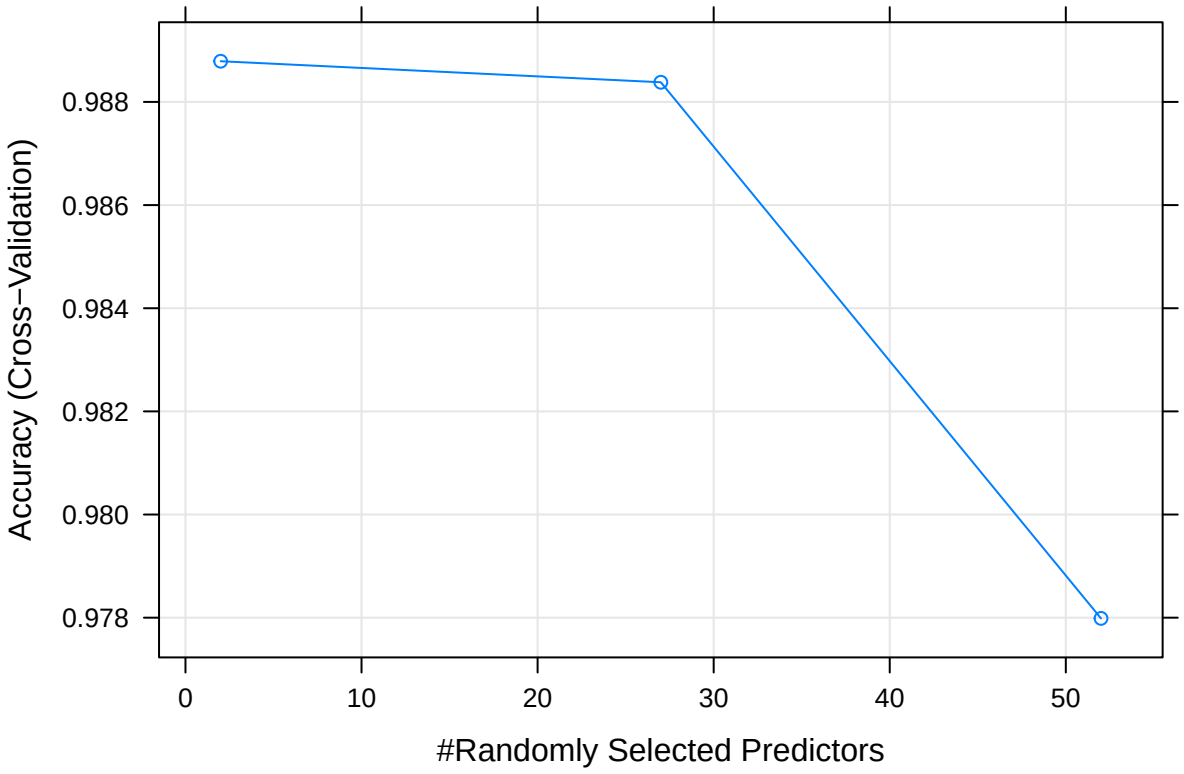
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  941    4    0    0
##          C    0    3  851    9    0
##          D    0    0    0  794    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9916   0.9953   0.9876   0.9978
## Specificity            0.9986   0.9990   0.9970   0.9995   0.9998
## Pos Pred Value         0.9964   0.9958   0.9861   0.9975   0.9989
## Neg Pred Value         1.0000   0.9980   0.9990   0.9976   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1919   0.1735   0.1619   0.1833
## Detection Prevalence   0.2855   0.1927   0.1760   0.1623   0.1835
## Balanced Accuracy      0.9993   0.9953   0.9962   0.9935   0.9988
```

```
out_sample_error_rf<-1-confusionMatrix(predict(rand_for, newdata=validation), factor(validation$classe))
out_sample_error_rf
```

```
## [1] 0.004893964
```

Random Forest has the highest accuracy and lowest out of sample error of the 3 models. Linear Discriminant Analysis has lowest accuracy and the highest out of sample error out of the 3 models. The reason for Random Forest to have a high accuracy for is maybe do to over fitting.

```
plot(rand_for)
```



Go with Random Forest for Final Prediction

```
set.seed(80275)
pred_model<-predict(rand_for, newdata=final_test)
pred_model
```

```
##  [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
```