Practical Machine Learning

Project

Submitted by: Pete Ostergren
——————————————-

12/30/2019

Introduction

This project is an example of categorical prediction. Given a data set with a cataegorical output variable (classe) and a large number (over 100) of input variables. Additionally given is a second data set with only input variables. A model must be created to be used on the second data set to predict the "classe" variable.

Load Library list

```
library(caret)
library(corrplot)
library(lattice)
library(ggplot2)
library(randomForest)
library(rattle)
library(RColorBrewer)
library(rpart)
library(rpart.plot)
```

1. Load data and perform data wrangling

Set the working directory and read in data files

```
setwd("C:/Coursera/PracticalMachineLearning")
trainRaw = read.csv("./data/pml-training.csv")
testRaw = read.csv("./data/pml-testing.csv")
```

Remove Near Zero Variance variables

```
NZV <- nearZeroVar(trainRaw, saveMetrics = TRUE)
head(NZV, 20)
```

```
##                          freqRatio percentUnique zeroVar   nzv
## X                         1.000000  100.00000000   FALSE FALSE
## user_name                 1.100679    0.03057792   FALSE FALSE
```

```
## raw_timestamp_part_1    1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2    1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp          1.000668    0.10192641   FALSE FALSE
## new_window             47.330049    0.01019264   FALSE  TRUE
## num_window              1.000000    4.37264295   FALSE FALSE
## roll_belt               1.101904    6.77810621   FALSE FALSE
## pitch_belt              1.036082    9.37722964   FALSE FALSE
## yaw_belt                1.058480    9.97349913   FALSE FALSE
## total_accel_belt        1.063160    0.14779329   FALSE FALSE
## kurtosis_roll_belt   1921.600000    2.02323922   FALSE  TRUE
## kurtosis_picth_belt   600.500000    1.61553358   FALSE  TRUE
## kurtosis_yaw_belt      47.330049    0.01019264   FALSE  TRUE
## skewness_roll_belt   2135.111111    2.01304658   FALSE  TRUE
## skewness_roll_belt.1  600.500000    1.72255631   FALSE  TRUE
## skewness_yaw_belt      47.330049    0.01019264   FALSE  TRUE
## max_roll_belt           1.000000    0.99378249   FALSE FALSE
## max_picth_belt          1.538462    0.11211905   FALSE FALSE
## max_yaw_belt          640.533333    0.34654979   FALSE  TRUE
```

```r
training01 <- trainRaw[, !NZV$nzv]
testing01 <- testRaw[, !NZV$nzv]
dim(training01)
```

```
## [1] 19622   100
```

```r
dim(testing01)
```

```
## [1]  20 100
```

```r
rm(trainRaw)
rm(testRaw)
rm(NZV)
```

Remove first five columns which are not needed

```r
regex <- grepl("^X|timestamp|user_name", names(training01))
training <- training01[, !regex]
testing <- testing01[, !regex]
rm(regex)
rm(training01)
rm(testing01)
dim(training)
```

```
## [1] 19622    95
```

```r
dim(testing)
```

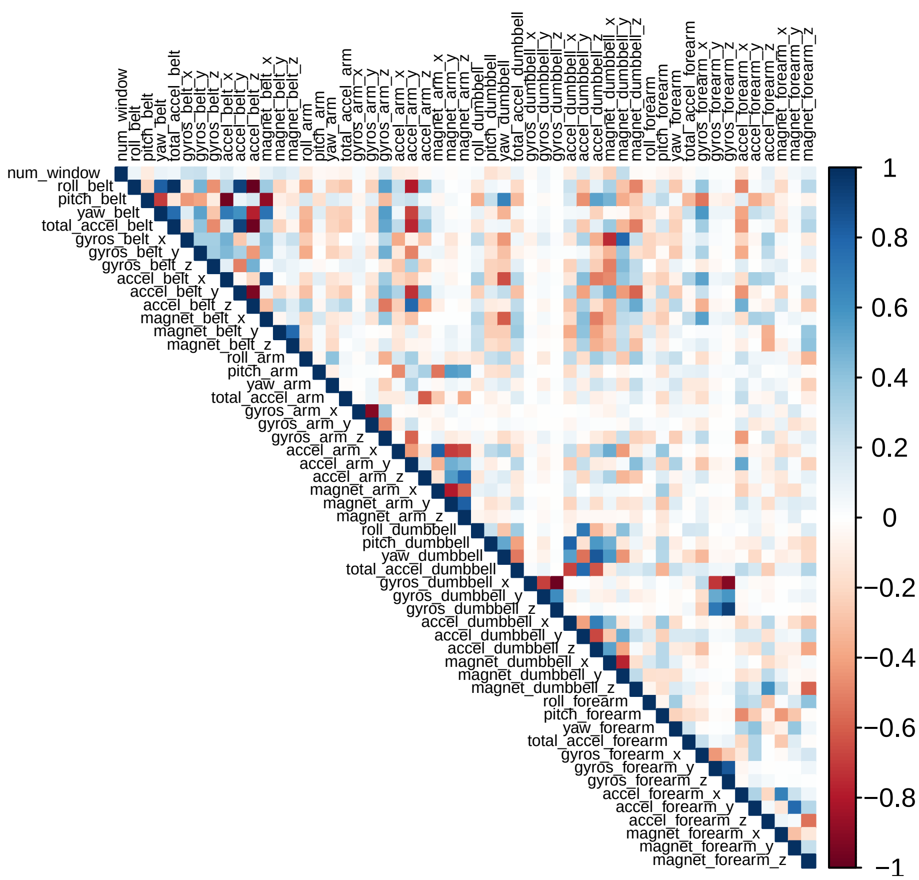
```
## [1] 20 95
```

Remove NA columns

```r
cond <- (colSums(is.na(training)) == 0)
training <- training[, cond]
testing <- testing[, cond]
rm(cond)
```

Explore the variables

```r
corrplot(cor(training[, -length(names(training))])
            , method = "color"
            , type = "upper"
            , tl.cex = 0.5
            , tl.col = rgb(0,0,0)
            )
```



Partitioning of our data

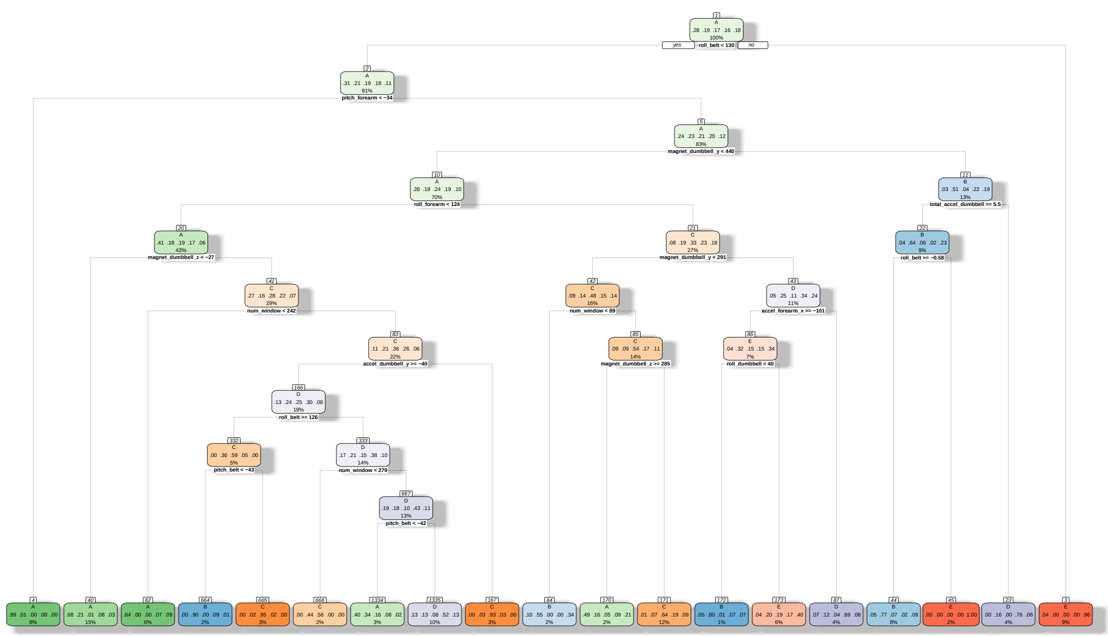
```r
set.seed(56789) # For reproducibile purpose
inTrain <- createDataPartition(training$classe, p = 0.70, list = FALSE)
validation <- training[-inTrain, ]
training <- training[inTrain, ]
rm(inTrain)
```

MODELING

Decision Tree

Create

```r
modelTree <- rpart(classe ~ ., data = training, method = "class")
fancyRpartPlot(modelTree)
```



Rattle 2019−Dec−31 13:27:18 Pete

```r
#Check
predictTree <- predict(modelTree, validation, type = "class")
confusionMatrix(validation$classe, predictTree)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1492   37   10   84   51
##          B  270  551  120  134   64
##          C   55   32  818   49   72
##          D  116   17  117  655   59
##          E   84   89   61  140  708
##
## Overall Statistics
##
##                Accuracy : 0.7178
##                  95% CI : (0.7061, 0.7292)
##     No Information Rate : 0.3427
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6409
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7397  0.75895   0.7265   0.6168   0.7421
## Specificity            0.9529  0.88602   0.9563   0.9359   0.9242
## Pos Pred Value         0.8913  0.48376   0.7973   0.6795   0.6543
## Neg Pred Value         0.8753  0.96313   0.9366   0.9173   0.9488
## Prevalence             0.3427  0.12336   0.1913   0.1805   0.1621
## Detection Rate         0.2535  0.09363   0.1390   0.1113   0.1203
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8463  0.82249   0.8414   0.7763   0.8331
```

```
accuracy <- postResample(predictTree, validation$classe)
ose <- 1 - as.numeric(confusionMatrix(validation$classe
                                      , predictTree)$overall[1])
rm(predictTree)
rm(modelTree)
```

Estimates for Decision Tree

Accuracy is 72% and Out-of-Sample Error is 28%.

Random Forests

Create

```
modelRF <- train(classe ~ ., data = training, method = "rf", trControl = trainControl(method = "cv", 5)
modelRF
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10990, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9933762  0.9916204
##   27    0.9972341  0.9965015
##   53    0.9939581  0.9923576
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```r
#Check
predictRF <- predict(modelRF, validation)
confusionMatrix(validation$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    1 1136    2    0    0
##          C    0    2 1024    0    0
##          D    0    0    0  964    0
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9982   0.9981   0.9979   1.0000
## Specificity            1.0000   0.9994   0.9996   1.0000   0.9996
## Pos Pred Value         1.0000   0.9974   0.9981   1.0000   0.9982
## Neg Pred Value         0.9998   0.9996   0.9996   0.9996   1.0000
## Prevalence             0.2846   0.1934   0.1743   0.1641   0.1835
## Detection Rate         0.2845   0.1930   0.1740   0.1638   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Balanced Accuracy      0.9997   0.9988   0.9988   0.9990   0.9998
```

```r
accuracy <- postResample(predictRF, validation$classe)
ose <- 1 - as.numeric(confusionMatrix(validation$classe, predictRF)$overall[1])
```

Estimates for Random Forests

Accuracy is 99.9% and Out-of-Sample Error is 0.1%.

As expected Random Forests out perform simple Decision Trees.

Go Live, use top performing model to predict!

```r
predict(modelRF, testing)
```

```
##  [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
```