Prediction Assignment Writeup

Shresth Juyal

25/06/2022

Introduction

This is a Prediction Assignment Writeup for the Coursera course “Practical Machine Learning”. This report will attempt to preduct the classe variable in the training set, which will then be very verified by the testing set. This report will explain how I built the model and also show the expected sample error. In the end, I will use my predication model to preduct 20 test cases given in the test case file.

Loading Libraries

```
library(lattice)
library(ggplot2)
library(caret)
library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(corrplot)

## corrplot 0.92 loaded

set.seed(4234)
```

Importing Data sets and Cleaning Data Set

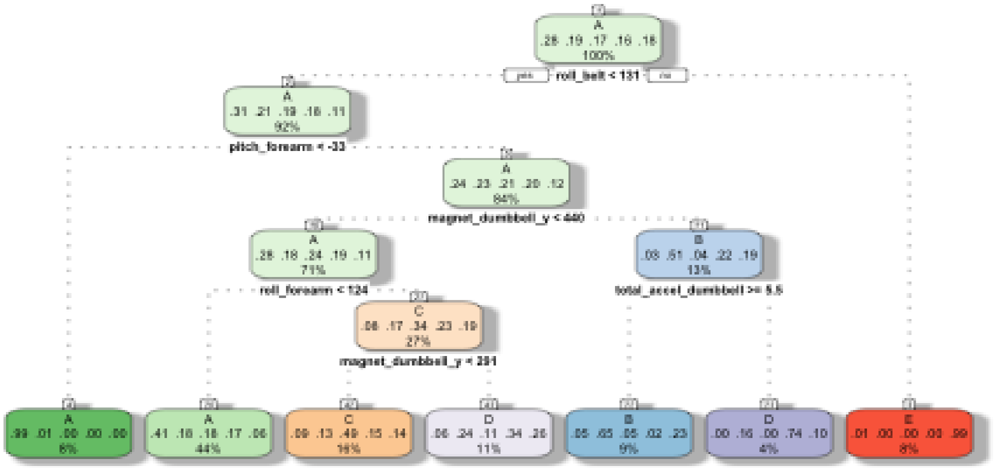
After loading the csv files, I will atempt the clean the data set by removing certain varialbes which have no significance and will not contribute to my prediction model.

```
training_set = read.csv("/Users/shresthjuyal/data/pml-training.csv")
test_set = read.csv("/Users/shresthjuyal/data/pml-testing.csv")
```

```
dim(test_set)

## [1]  20 160

training_set = training_set[,colMeans(is.na(training_set)) < .9]
training_set = training_set[,-c(1:7)]

zerovariance = nearZeroVar(training_set)
training_set <- training_set[,-zerovariance]
dim(training_set)

## [1] 19622    53

inTrain = createDataPartition(y=training_set$classe, p=0.7, list=F)
train = training_set[inTrain,]
valid = training_set[-inTrain,]
```

Creating Decision Tree

```
control = trainControl(method="cv", number=3, verboseIter=F)

decision_tree <- train(classe~., data=train, method="rpart", trControl = cont
rol, tuneLength = 5)
fancyRpartPlot(decision_tree$finalModel)
```



Rattle 2022-Jun-25 22:56:38 shresthjuyal

The decision tree indicates that the ‘A’ is the best model to make with .41 .18 .18 .16 and .06.

Creating First Prediction Model

```
pred_trees = predict(decision_tree, valid)
cmtrees = confusionMatrix(pred_trees, factor(valid$classe))
cmtrees

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1527  459  485  416  151
##          B   23  359   39   16  127
##          C   84  129  423  132  128
##          D   37  192   79  400  173
##          E    3    0    0    0  503
## 
## Overall Statistics
## 
##                Accuracy : 0.5458
##                  95% CI : (0.533, 0.5586)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4084
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9122  0.31519  0.41228  0.41494  0.46488
## Specificity            0.6412  0.95681  0.90265  0.90226  0.99938
## Pos Pred Value         0.5026  0.63652  0.47210  0.45403  0.99407
## Neg Pred Value         0.9484  0.85341  0.87913  0.88729  0.89236
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2595  0.06100  0.07188  0.06797  0.08547
## Detection Prevalence   0.5162  0.09584  0.15225  0.14970  0.08598
## Balanced Accuracy      0.7767  0.63600  0.65747  0.65860  0.73213
```

```
mod = train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)
```

```
tree_prediction <- predict(mod, valid)
tree <- confusionMatrix(tree_prediction, factor(valid$classe))
tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    3    0    0    0
##          B    2 1130   14    0    0
##          C    0    6 1008    6    3
##          D    0    0    4  956    3
##          E    0    0    0    2 1076
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9902, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9921   0.9825   0.9917   0.9945
## Specificity            0.9993   0.9966   0.9969   0.9986   0.9996
## Pos Pred Value         0.9982   0.9860   0.9853   0.9927   0.9981
## Neg Pred Value         0.9995   0.9981   0.9963   0.9984   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1920   0.1713   0.1624   0.1828
## Detection Prevalence   0.2846   0.1947   0.1738   0.1636   0.1832
## Balanced Accuracy      0.9990   0.9944   0.9897   0.9951   0.9970
```

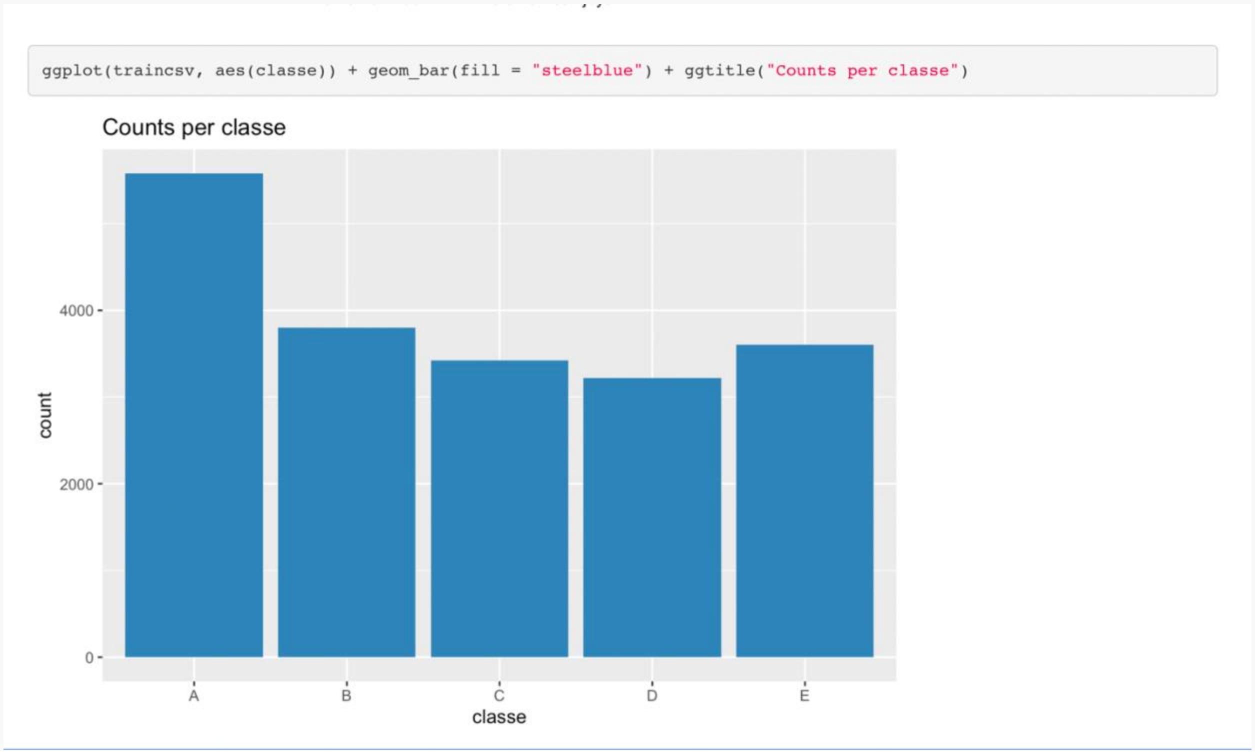
Creating Second Prediction Model

```
second <- train(classe~., data=train, method="gbm", trControl = control, tune
Length = 5, verbose = F)

second_prediction <- predict(second, valid)
plot_second <- confusionMatrix(second_prediction, factor(valid$classe))
plot_second
```

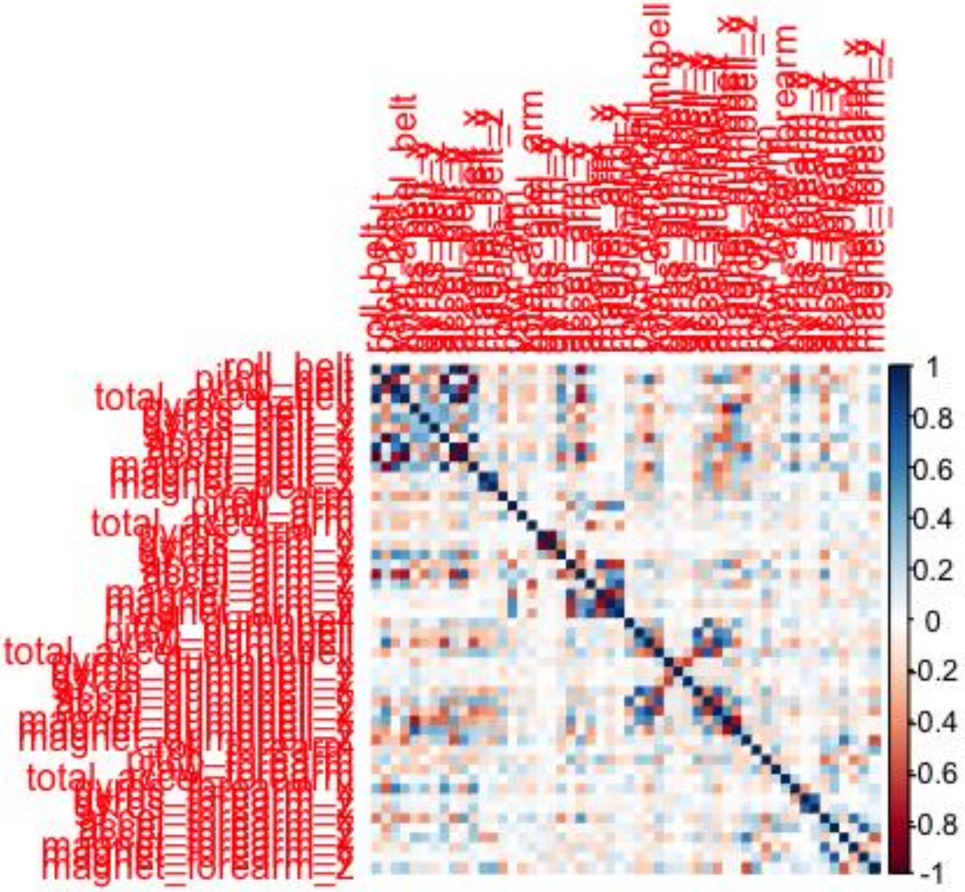
```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction    A    B    C    D    E
##          A 1664    5    0    0    0
##          B    9 1121   16    1    0
##          C    0   13  999    8    3
##          D    0    0   10  952    5
##          E    1    0    1    3 1074
## 
## Overall Statistics
## 
##                Accuracy : 0.9873
##                  95% CI : (0.9841, 0.99)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9839
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9940   0.9842   0.9737   0.9876   0.9926
## Specificity            0.9988   0.9945   0.9951   0.9970   0.9990
## Pos Pred Value         0.9970   0.9773   0.9765   0.9845   0.9954
## Neg Pred Value         0.9976   0.9962   0.9944   0.9976   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2828   0.1905   0.1698   0.1618   0.1825
## Detection Prevalence   0.2836   0.1949   0.1738   0.1643   0.1833
## Balanced Accuracy      0.9964   0.9894   0.9844   0.9923   0.9958
```

Our first prediction model is the best because its has a .99 accuracy and the lowest sample error of 0.04. This should be a good enough model to predict the 'classe' variable and test our 20 test cases.

```
predicted <- predict(mod, test_set)
print(predicted)

##  [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

correlation = cor(train[, -length(names(train))])
corrplot(correlation, method="color")
```



```
ggplot(traincsv, aes(classe)) + geom_bar(fill = "steelblue") + ggtitle("Counts per classe")
```



My prediction for my 20 test cases is {r print(pred)}