MachLearnProj

Corey Neff

11/4/2020

Packages

Install packages I think we'll need.

```
if (!require("librarian")) install.packages("librarian")
```

```
## Loading required package: librarian
```

```
library(librarian)
shelf(caret, randomForest, rpart, ggplot2, dplyr)
```

```
##
##   The 'cran_repo' argument in shelf() was not set, so it will use
##   cran_repo = 'https://cran.r-project.org' by default.
##
##   To avoid this message, set the 'cran_repo' argument to a CRAN
##   mirror URL (see https://cran.r-project.org/mirrors.html) or set
##   'quiet = TRUE'.
```

Download and read data

Read csvs into R, making sure to specify NA strings from CSV file (found by opening in Excel). Also kept only compelte cases. We'll also get rid of the columns that are irrelevant (i.e. the ones dealing with time, etc.)

```
set.seed(69)
setwd('/users/coreyneff/downloads/')
train_file <- download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',
                            destfile = 'train.csv')
test_file <- download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',
                            destfile = 'test.csv')
train <- read.csv('train.csv', na.strings = c('#DIV/0!', 'NA',''))
test <- read.csv('test.csv', na.strings = c('#DIV/0!', 'NA', ''))
train <-train[,colSums(is.na(train)) == 0]
test <-test[,colSums(is.na(test)) == 0]
train <- train[,-c(1:7)]
test <- test[,-c(1:7)]
train$classe <- as.factor(train$classe)
```

Look at data

Next we'll look at our training dataset

```
summary(train)
head(train)
```

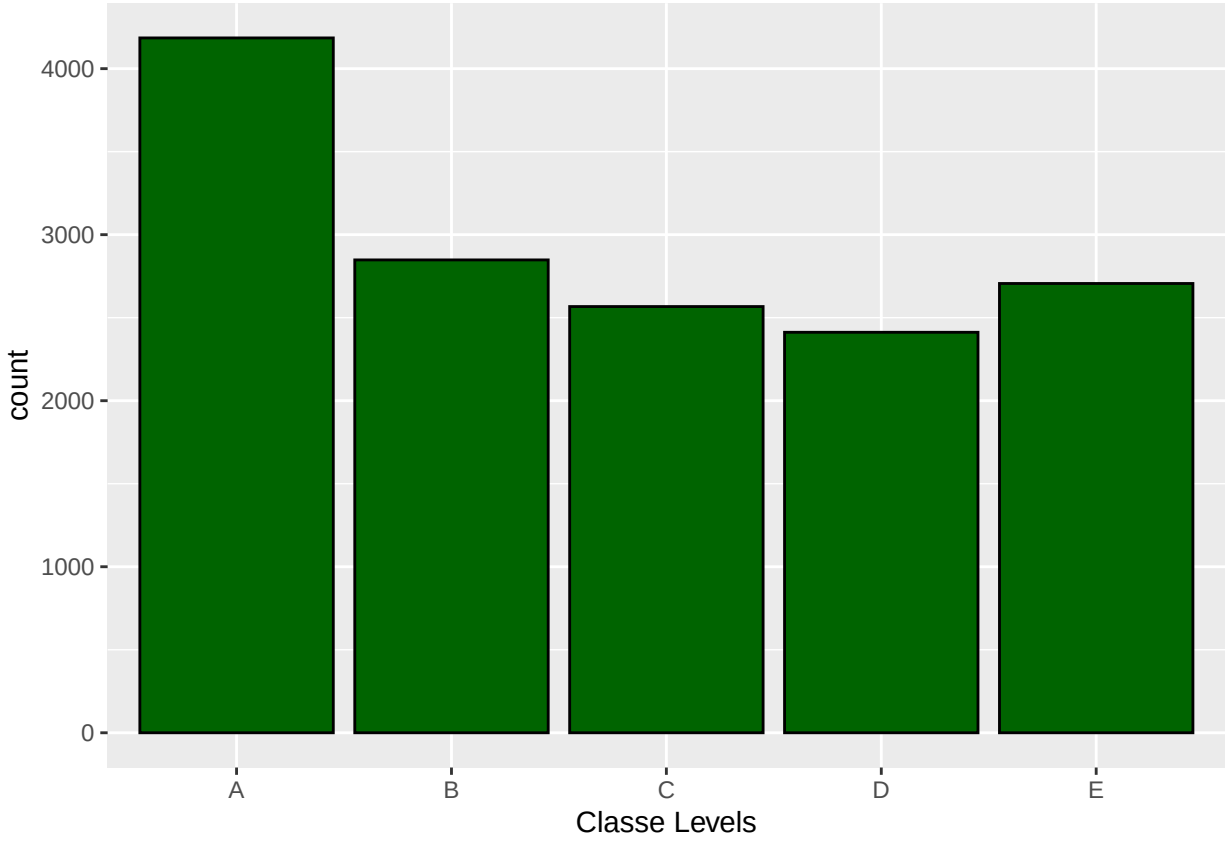
Partition data

```
partition <- createDataPartition(train$classe, p=0.75, list=FALSE)
train_train <- train[partition,]
train_test <- train[-partition,]
```

First look

According to the directions, "Classe" is the outcome variable, so let's see its distribution.

```
train_train$classe <- as.factor(train_train$classe)
ggplot(train_train) +
      geom_bar(aes(classe), col="black", fill="darkgreen") +
      xlab("Classe Levels")
```



First go at a prediction model

I like CART, personally, so I'm going to start there.

```
train_cart <- rpart(classe ~., train_train, method = "class")
cart_predict <- predict(train_cart, train_test, type = "class")
confusion<- confusionMatrix(cart_predict,as.factor(train_test$classe))
confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1253  212   14   74   46
##          B   34  540   78   29   56
##          C   35   78  688  120   93
##          D   41   66   49  523   51
##          E   32   53   26   58  655
##
## Overall Statistics
##
##                Accuracy : 0.7461
##                  95% CI : (0.7337, 0.7583)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6773
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8982   0.5690   0.8047   0.6505   0.7270
## Specificity            0.9014   0.9502   0.9195   0.9495   0.9578
## Pos Pred Value         0.7836   0.7327   0.6785   0.7164   0.7949
## Neg Pred Value         0.9570   0.9018   0.9571   0.9327   0.9397
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2555   0.1101   0.1403   0.1066   0.1336
## Detection Prevalence   0.3261   0.1503   0.2068   0.1489   0.1680
## Balanced Accuracy      0.8998   0.7596   0.8621   0.8000   0.8424
```

Next is randomForest

Next we should look at a randomForest model.

```
train_forest <- randomForest(classe ~., train_train, method = "class")
forest_predict <- predict(train_forest, train_test, method = "class")
confusion2 <- confusionMatrix(forest_predict, as.factor(train_test$classe))
confusion2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  944   13    0    0
##          C    0    0  842    8    1
##          D    0    0    0  795    2
##          E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9913, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9848   0.9888   0.9967
## Specificity            0.9986   0.9967   0.9978   0.9995   0.9998
## Pos Pred Value         0.9964   0.9864   0.9894   0.9975   0.9989
## Neg Pred Value         1.0000   0.9987   0.9968   0.9978   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1717   0.1621   0.1831
## Detection Prevalence   0.2855   0.1951   0.1735   0.1625   0.1833
## Balanced Accuracy      0.9993   0.9957   0.9913   0.9942   0.9982
```

Comparison

Lets compare the accuracy of the two.

```
list(CART = confusion$overall["Accuracy"],
        randomForest = confusion2$overall["Accuracy"])
```

```
## $CART
##  Accuracy
## 0.7461256
##
## $randomForest
##  Accuracy
## 0.9938825
```

Final verdict

It seems like the randomForest model's preductive power is much higher, so let's use that. Here is our prediction for the test set.

```
predict(train_forest, test, type = "class")
```

```
##  1  2  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
```