# HW 08

TUAN BUI

11/16/2021

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.
3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.4     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   2.0.1     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

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

spambase_data <- read.table('~/OneDrive - Stony Brook University/SBU/MAT + AM
S/Fall 2021/AMS 380/hw/08/spambase.data', sep = ",")

spambase_names <- read.delim('~/OneDrive - Stony Brook University/SBU/MAT + A
MS/Fall 2021/AMS 380/hw/08/spambase.names')

spambase_names <- spambase_names[-c(1:29),]
spambase_names <- as.data.frame(spambase_names)
spambase_names <- spambase_names %>%
  separate(spambase_names, c("Variable", "Type"), sep = ":")
names(spambase_data) <- spambase_names$Variable
names(spambase_data)[58] <- 'class'

# clean data
spambase_data <- na.omit(spambase_data)

spambase_data$class <- as.factor(spambase_data$class)
```

## Question 01:

```
# Use the random seed 123 to divide the cleaned data into 75% training and 25
% testing
set.seed(123)
training.samples <- spambase_data$class %>%
  createDataPartition(p = 0.75, list = FALSE)
train.data  <- spambase_data[training.samples, ]
test.data <- spambase_data[-training.samples, ]
```

## Question 02:

```
set.seed(123)
model <- train(
  class ~., data = train.data, method = "rf",
  trControl = trainControl("cv", number = 10),
  importance = TRUE
  )

predicted.classes_1 <- model %>% predict(train.data)
```

```
# Confusion matrix of the training data
table(predicted.classes_1, train.data$class)

##
## predicted.classes_1    0    1
##                   0 2091    2
##                   1    0 1358

mean(predicted.classes_1 == train.data$class)

## [1] 0.9994205

# The overall accuracy of the training data is 0.9994205

sum((train.data$class == 1)*(predicted.classes_1 == 1))/sum(train.data$class
==
1)

## [1] 0.9985294

# The sensitivity of the test data is 0.9985294

sum((train.data$class == 0)*(predicted.classes_1 == 0))/sum(train.data$class
==
0)

## [1] 1

# The specificity of the test data is 1
```

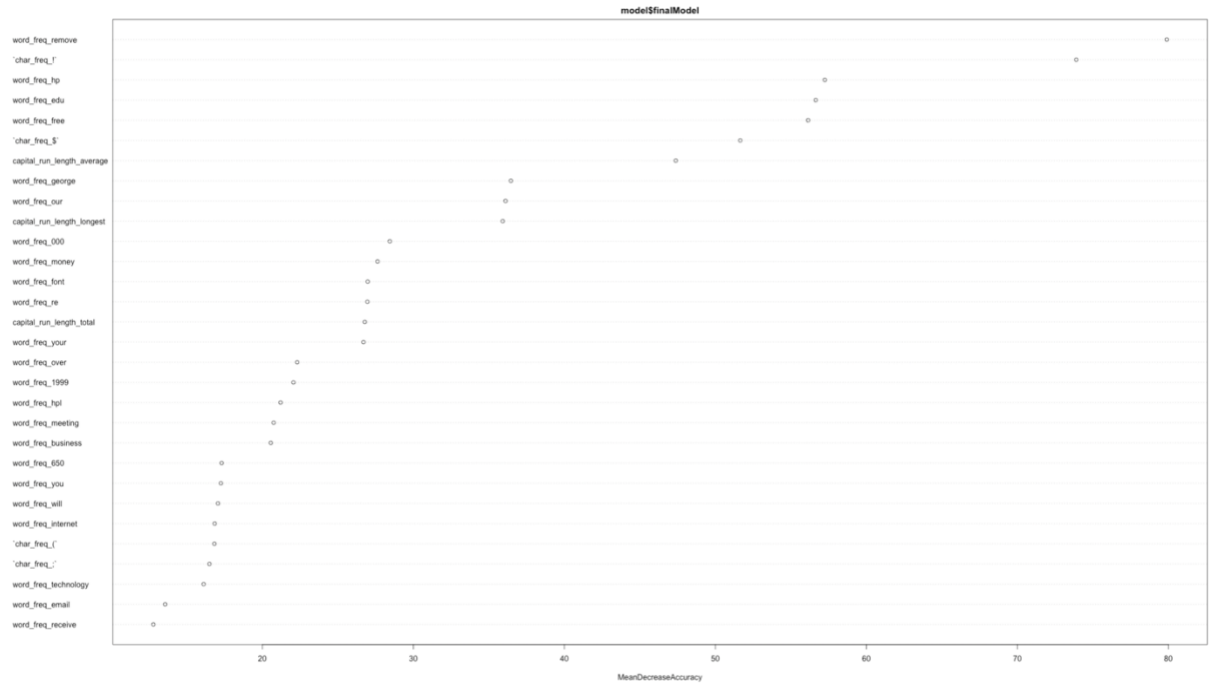
## Question 03:

```
predicted.classes_2 <- model %>% predict(test.data)

# Confusion matrix of the testing data
table(predicted.classes_2, test.data$class)

##
## predicted.classes_2   0   1
##                   0 674  34
##                   1  23 419

mean(predicted.classes_2 == test.data$class)

## [1] 0.9504348

# The overall accuracy of the testing data is 0.9504348

sum((test.data$class == 1)*(predicted.classes_2 == 1))/sum(test.data$class ==
1)
```

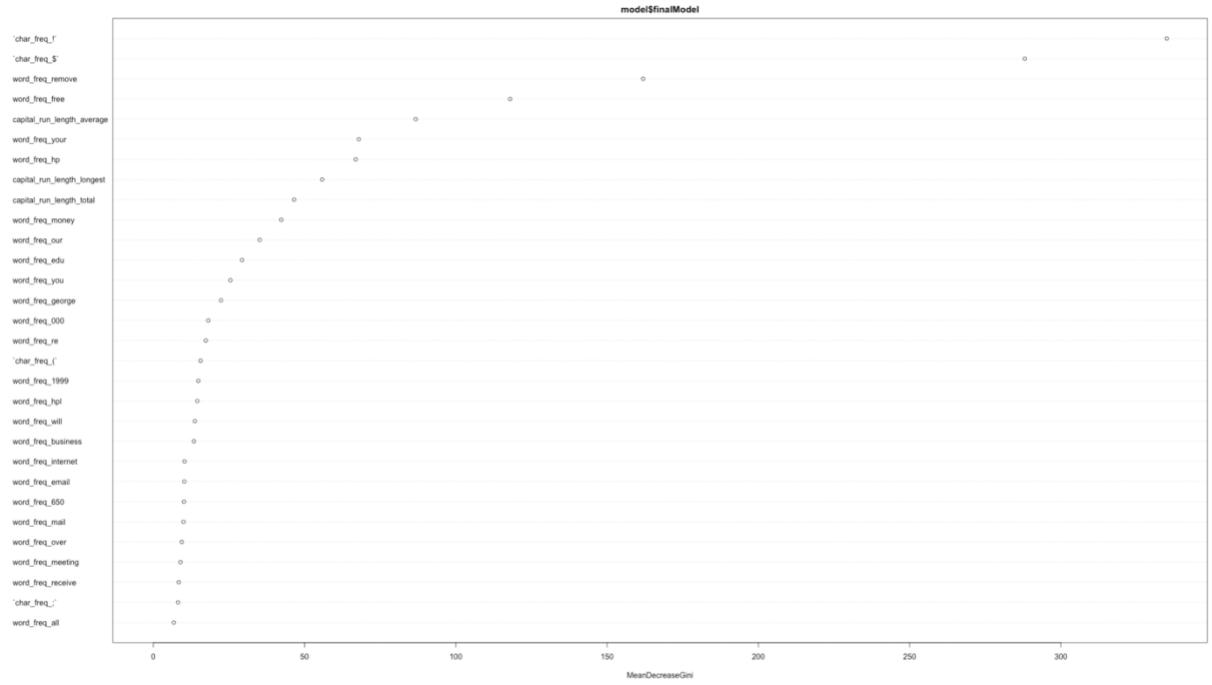
```
## [1] 0.9249448

# The sensitivity of the testing data is 0.9249448

sum((test.data$class == 0)*(predicted.classes_2 == 0))/sum(test.data$class == 0)

## [1] 0.9670014

# The specificity of the testing data is 0.9670014
```

## Question 04:

```
# Plot MeanDecreaseAccuracy
varImpPlot(model$finalModel, type = 1)
```




```
# Plot MeanDecreaseGini
varImpPlot(model$finalModel, type = 2)
```




## Question 05:

```
# The importance of each variable in percentage based on MeanDecreaseAccuracy
varImp(model$finalModel, type = 1)
```

```
##                              Overall
## word_freq_make              8.153946
## word_freq_address           9.367384
## word_freq_all               6.902195
## word_freq_3d               11.807186
## word_freq_our              36.103842
## word_freq_over             22.307910
## word_freq_remove           79.892670
## word_freq_internet         16.843599
## word_freq_order            12.148703
## word_freq_mail             12.358148
## word_freq_receive          12.781801
## word_freq_will             17.058283
## word_freq_people            8.937479
## word_freq_report            8.684960
## word_freq_addresses         6.577593
## word_freq_free             56.136717
## word_freq_business         20.560001
## word_freq_email            13.563499
## word_freq_you              17.255713
## word_freq_credit            8.753119
## word_freq_your             26.697597
## word_freq_font             26.977270
```

```
## word_freq_000               28.440091
## word_freq_money             27.628663
## word_freq_hp                57.245508
## word_freq_hpl               21.207529
## word_freq_george            36.460092
## word_freq_650               17.305336
## word_freq_lab               12.332069
## word_freq_labs              10.002784
## word_freq_telnet             5.097685
## word_freq_857                6.688856
## word_freq_data               5.389617
## word_freq_415                5.611032
## word_freq_85                11.228468
## word_freq_technology        16.113064
## word_freq_1999              22.059059
## word_freq_parts              3.843755
## word_freq_pm                12.666298
## word_freq_direct             4.832203
## word_freq_cs                 5.346080
## word_freq_meeting           20.748908
## word_freq_original           7.560766
## word_freq_project            6.885084
## word_freq_re                26.956730
## word_freq_edu               56.642240
## word_freq_table              1.073460
## word_freq_conference        10.188196
## `char_freq_;`               16.495985
## `char_freq_(`               16.824041
## `char_freq_[`                6.488239
## `char_freq_!`               73.894087
## `char_freq_$`               51.642349
## `char_freq_#`                7.792698
## capital_run_length_average 47.378136
## capital_run_length_longest 35.917454
## capital_run_length_total   26.784196
```

## Question 6:

```
mtry <- 26
p <- mtry*3
p
```

```
## [1] 78
```

```
# The number of variables we should (as commonly recommended) to select, at r
andom, to be considered for that node split are 78
```