Practical Data Science with R - Tidyverse style

Indrajeet Patil

2019-02-24

Contents

Chapter 1

Introduction

This is my attempt to convert all R code encountered in *Practical Data Science with R* to use tidyverse packages.

Chapter 2

Choosing and evaluating models

2.1 Building and applying a logistic regression spam model

```
set.seed(123)
library(tidyverse, warn.conflicts = FALSE)
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
```

```
## -- Attaching packages -------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0.9000     v purrr   0.3.0
## v tibble  2.0.1          v dplyr   0.8.0.9000
## v tidyr   0.8.2          v stringr 1.4.0
## v readr   1.3.1          v forcats 0.4.0
```

```
## -- Conflicts ----------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Using logistic regression to classify emails into spam or non-spam:

```
# reading the file containing spam data
spamD <- readr::read_tsv("https://raw.githubusercontent.com/WinVector/zmPDSwR/master/Spambase/spamD.tsv
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   spam = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
# creating training and testing datasets
spamTrain <- dplyr::filter(.data = spamD, rgroup >= 10)
spamTest <- dplyr::filter(.data = spamD, rgroup < 10)

# training the model
spamModel <- stats::glm(formula = spam =="spam" ~ .,
           family = stats::binomial(link = "logit"),
           data = dplyr::select(spamTrain, -rgroup))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# looking at the result
broom::tidy(spamModel)
```

```
## # A tibble: 58 x 5
##    term               estimate std.error statistic  p.value
##    <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)          -1.62     0.151     -10.7  1.24e-26
##  2 word.freq.make       -0.327    0.237      -1.38 1.68e- 1
##  3 word.freq.address    -0.155    0.0771     -2.00 4.51e- 2
##  4 word.freq.all         0.149    0.123       1.22 2.23e- 1
##  5 word.freq.3d          2.19     1.56        1.40 1.60e- 1
##  6 word.freq.our         0.476    0.102       4.68 2.91e- 6
##  7 word.freq.over        0.744    0.252       2.95 3.13e- 3
##  8 word.freq.remove      2.34     0.349       6.70 2.08e-11
##  9 word.freq.internet    0.801    0.220       3.63 2.83e- 4
## 10 word.freq.order       0.645    0.300       2.15 3.14e- 2
## # ... with 48 more rows
```

```
# looking at the model summary
broom::glance(spamModel)
```

```
## # A tibble: 1 x 7
##   null.deviance df.null logLik   AIC   BIC deviance df.residual
##           <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int>
## 1         5556.    4142  -807. 1730. 2097.    1614.        4085
```

```
# with predicted response on training data
spamTrain <- broom::augment(
  x = spamModel,
  newdata = spamTrain,
  type.predict = "response"
)

# with predicted response on test data
spamTest <- broom::augment(
  x = spamModel,
  newdata = spamTest,
  type.predict = "response"
)

# performance with the training data
table(y = spamTrain$spam, glmPred = spamTrain$.fitted > 0.5)
```

```
##           glmPred
## y           FALSE TRUE
##   non-spam  2396  114
##   spam       178 1455
```

```
# performance with the test data
table(y = spamTest$spam, glmPred = spamTest$.fitted > 0.5)
```

```
##           glmPred
## y           FALSE TRUE
##   non-spam   264   14
##   spam        22  158
```

Looking at actual and predicted sample responses

```
sample <- spamTest[c(7,35,224,327), c('spam', '.fitted')]
print(sample)
```

```
## # A tibble: 4 x 2
##   spam      .fitted
##   <chr>       <dbl>
## 1 spam     0.990
## 2 spam     0.480
## 3 non-spam 0.000685
## 4 non-spam 0.000143
```

Spam confusion matrix

```
# performance with the test data
(cM <- table(truth = spamTest$spam, prediction = spamTest$.fitted > 0.5))
```

```
##           prediction
## truth      FALSE TRUE
##   non-spam   264   14
##   spam        22  158
```

Entering data by hand (example of a good spam filter)

```
t <- as.table(matrix(data = c(288 - 1, 17, 1, 13882 - 17), nrow = 2, ncol = 2))
rownames(t) <- rownames(cM)
colnames(t) <- colnames(cM)
print(t)
```

```
##          FALSE  TRUE
## non-spam   287     1
## spam        17 13865
```