Data science Final project

스펨 메일 데이터
분류 분석

e – Business 황혜린

201821479

About

CONTENTS

with R

1. 데이터 설명

https://archive.ics.uci.edu/ml/index.php

Spambase Data Set

UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

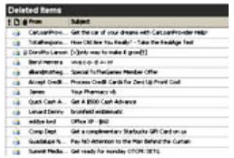
Search

Repository Web

View ALL Data Sets

Spambase Data Set

Download: Data Folder, Data Set Description



Abstract: Classifying Email as Spam or Non-Spam

Data Set Characteristics:	Multivariate	**Number of Instances:**	4601	**Area:**	Computer
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	57	**Date Donated**	1999-07-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	371587

Source:

Creators:

Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt
Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304

1. 데이터 설명

4601개의 관측치와 58개의 변수

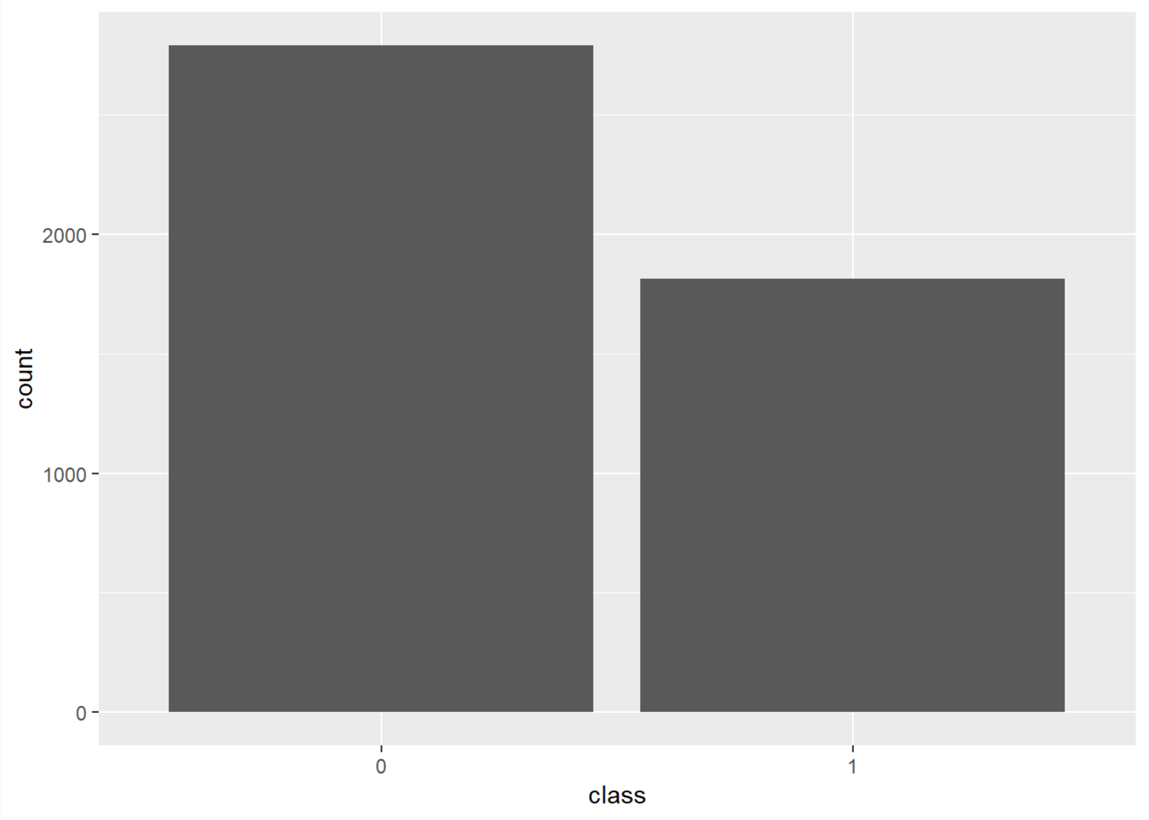
1~57th
스펨데이터의 예측변수

word_freq_make
word_freq_address
word_freq_all
word_freq_3d
word_freq_our
.
.
.
.
char_freq_;
char_freq_(
char_freq_[
.
.
.
capital_run_length_average
capital_run_length_longest
capital_run_length_total

58th
반응변수

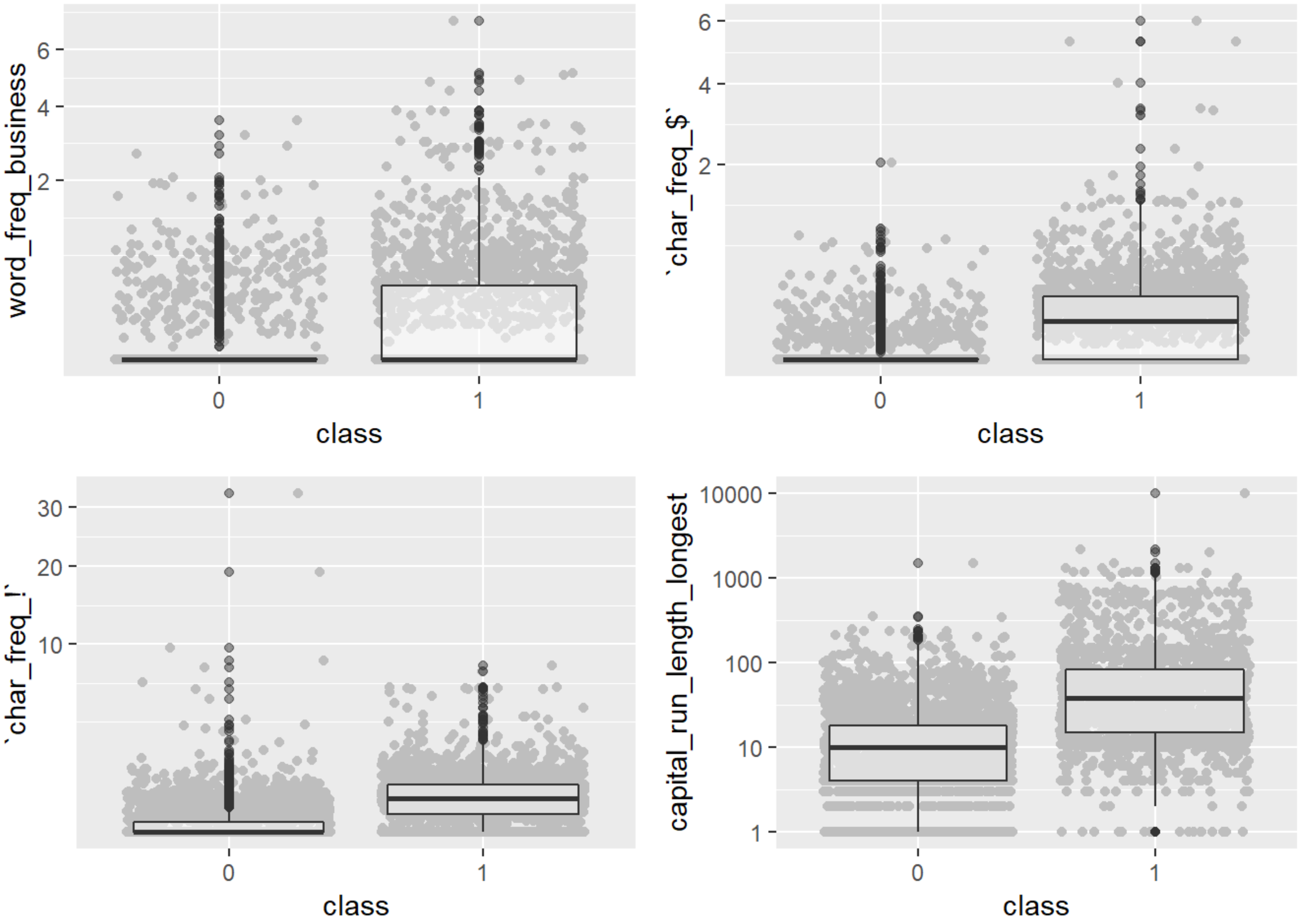
class

```
data %>% ggplot(aes(class)) + geom_bar()
```



57개의 예측변수들 중에서 스팸메일과 상관관계가 높은 변수는 무엇일까?

```r
library(ggplot2)
library(dplyr)
library(gridExtra)
p1 <- data %>% ggplot(aes(class, word_freq_business)) +
  geom_jitter(col='gray') +
  geom_boxplot(alpha=.5) +
  scale_y_sqrt()
p2 <- data %>% ggplot(aes(class, `char_freq_$`)) +
  geom_jitter(col='gray') +
  geom_boxplot(alpha=.5) +
  scale_y_sqrt()
p3 <- data %>% ggplot(aes(class, word_freq_credit)) +
  geom_jitter(col='gray') +
  geom_boxplot(alpha=.5) +
  scale_y_sqrt()
p4 <- data %>% ggplot(aes(class, capital_run_length_longest)) +
  geom_jitter(col='gray') +
  geom_boxplot(alpha=.5) +
  scale_y_log10()
grid.arrange(p1, p2, p3, p4, ncol=2)
```



3-0. 변수명의 특수문자 처리

- 일부 함수는 입력데이터의 변수명에 특수문자가 들어가면 에러를 일으키므로 make.names()함수를 사용하여 변수명을 변경해주었습니다.

```
old_names <- names(data)
new_names <- make.names(names(data), unique = TRUE)
cbind(old_names, new_names) [old_names!=new_names, ]
```

```
##      old_names     new_names
## [1,] "char_freq_;" "char_freq_."
## [2,] "char_freq_(" "char_freq_..1"
## [3,] "char_freq_[" "char_freq_..2"
## [4,] "char_freq_!" "char_freq_..3"
## [5,] "char_freq_$" "char_freq_..4"
## [6,] "char_freq_#" "char_freq_..5"
```

```
names(data) <- new_names
```

3-1. 데이터 나누기: 세트 구분

- 훈련:검증:테스트 세트 = 60:20:20

```
set.seed(1999) #재현 가능한 연구를 위해 seed설정
n <- nrow(data)
idx <- 1:n

#훈련 세트
training_idx <- sample(idx, n * .60)
idx <- setdiff(idx, training_idx)

#검증세트
validate_idx <- sample(idx, n * .20)

#테스트 세트
test_idx <- setdiff(idx, validate_idx)

training <- data[training_idx,]
validation <- data[validate_idx,]
test <- data[test_idx,]
```

3-2-1. 로지스틱 회귀모형에 훈련세트 적합

```
#glm() 함수 사용
data_lm_full <- glm(class ~ ., data=training, family=binomial)
```

glm() 을 사용하여
Training set를 선형 로지스틱 회귀 모형에 적합

3. 데이터 분석

3-2. 로지스틱 회귀

```
## Call:
## glm(formula = class ~ ., family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8744  -0.2331   0.0000   0.1000   4.0361
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.612e+00  1.869e-01  -8.623  < 2e-16 ***
## word_freq_make             -5.252e-01  3.026e-01  -1.736 0.082618 .
## word_freq_address          -1.358e-01  8.392e-02  -1.618 0.105658
## word_freq_all               3.194e-01  1.527e-01   2.091 0.036487 *
## word_freq_3d                2.250e+00  4.811e+00   0.468 0.640081
## word_freq_our               6.249e-01  1.424e-01   4.389 1.14e-05 ***
## word_freq_over              5.848e-01  2.727e-01   2.145 0.031990 *
## word_freq_remove            1.980e+00  3.860e-01   5.130 2.89e-07 ***
## word_freq_internet          2.679e-01  1.904e-01   1.407 0.159533
## word_freq_order             7.681e-01  3.940e-01   1.950 0.051223 .
## word_freq_mail              1.205e-01  9.682e-02   1.245 0.213253
## word_freq_receive          -2.378e-01  3.808e-01  -0.624 0.532368
## word_freq_will             -1.034e-01  9.124e-02  -1.133 0.257275
## word_freq_people           -6.180e-02  3.188e-01  -0.194 0.846304
## word_freq_report            9.627e-02  1.555e-01   0.619 0.535805
## word_freq_addresses         3.129e+00  1.416e+00   2.209 0.027145 *
## word_freq_free              1.157e+00  1.927e-01   6.005 1.91e-09 ***
## word_freq_business          1.017e+00  3.039e-01   3.345 0.000822 ***
## word_freq_email             7.042e-02  1.484e-01   0.475 0.635092
## word_freq_you               5.085e-02  4.440e-02   1.145 0.252103
## word_freq_credit            7.437e-01  5.516e-01   1.348 0.177537
## word_freq_your              2.051e-01  6.775e-02   3.027 0.002468 **
## word_freq_font              6.306e-02  1.765e-01   0.357 0.720829
## word_freq_000               2.367e+00  6.466e-01   3.660 0.000252 ***
## word_freq_money             2.895e-01  1.433e-01   2.021 0.043293 *
## word_freq_hp               -1.550e+00  3.006e-01  -5.156 2.52e-07 ***
## word_freq_hpl              -9.938e-01  4.643e-01  -2.140 0.032327 *
## word_freq_george           -2.005e+01  4.351e+00  -4.608 4.06e-06 ***
## word_freq_650               6.624e-01  3.431e-01   1.931 0.053529 .
## word_freq_lab              -1.916e+00  1.302e+00  -1.472 0.141136
## word_freq_labs             -7.920e-01  6.740e-01  -1.175 0.239961
## word_freq_telnet           -4.230e+00  2.823e+00  -1.499 0.133996
## word_freq_857              -3.933e+01  1.364e+03  -0.029 0.977006
## word_freq_data             -7.396e-01  4.216e-01  -1.754 0.079381 .
## word_freq_415              -8.107e+00  1.299e+01  -0.624 0.532479
## word_freq_85               -2.512e+00  1.350e+00  -1.861 0.062788 .
## word_freq_technology        7.395e-01  3.874e-01   1.909 0.056294 .
## word_freq_1999              8.929e-02  2.801e-01   0.319 0.749917
## word_freq_parts            -7.054e-01  4.964e-01  -1.421 0.155356
## word_freq_pm               -9.322e-01  4.320e-01  -2.158 0.030935 *
## word_freq_direct           -4.593e-01  4.384e-01  -1.048 0.294782
## word_freq_cs               -4.561e+01  3.913e+01  -1.166 0.243774
## word_freq_meeting          -2.030e+00  6.771e-01  -2.998 0.002714 **
## word_freq_original         -1.730e+00  1.133e+00  -1.527 0.126707
## word_freq_project          -1.325e+00  6.151e-01  -2.154 0.031205 *
## word_freq_re               -7.592e-01  2.100e-01  -3.615 0.000301 ***
## word_freq_edu              -1.574e+00  3.543e-01  -4.442 8.90e-06 ***
## word_freq_table            -2.913e+00  2.283e+00  -1.276 0.201924
## word_freq_conference       -3.347e+00  2.238e+00  -1.496 0.134779
## char_freq_.                -9.836e-01  4.801e-01  -2.049 0.040486 *
## char_freq_..1              -1.148e-01  2.777e-01  -0.413 0.679443
## char_freq_..2              -8.119e-01  1.175e+00  -0.691 0.489560
## char_freq_..3               2.968e-01  1.035e-01   2.868 0.004137 **
## char_freq_..4               5.572e+00  9.548e-01   5.835 5.36e-09 ***
## char_freq_..5               2.582e+00  1.076e+00   2.400 0.016383 *
## capital_run_length_average  3.498e-02  3.357e-02   1.042 0.297416
## capital_run_length_longest  1.165e-02  3.616e-03   3.222 0.001272 **
## capital_run_length_total    7.166e-04  2.869e-04   2.497 0.012509 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3669.7  on 2759  degrees of freedom
## Residual deviance: 1092.8  on 2702  degrees of freedom
## AIC: 1208.8
##
## Number of Fisher Scoring iterations: 20
```

–어떤 변수들이 스팸 여부에 대한 예측력이 높은지에 대한 정보를 얻을 수 있다

–뒤에 있는 '*'의 개수가 많을수록 예측력이 높은 변수

3-2-2. 검증세트 사용하여 모형 평가

```
y_obs <- as.numeric(as.character(validation$class))
yhat_lm <- predict(data_lm_full, newdata = validation, type='response')
pred_lm <- prediction(yhat_lm, y_obs)
performance(pred_lm, "auc")@y.values[[1]]
```

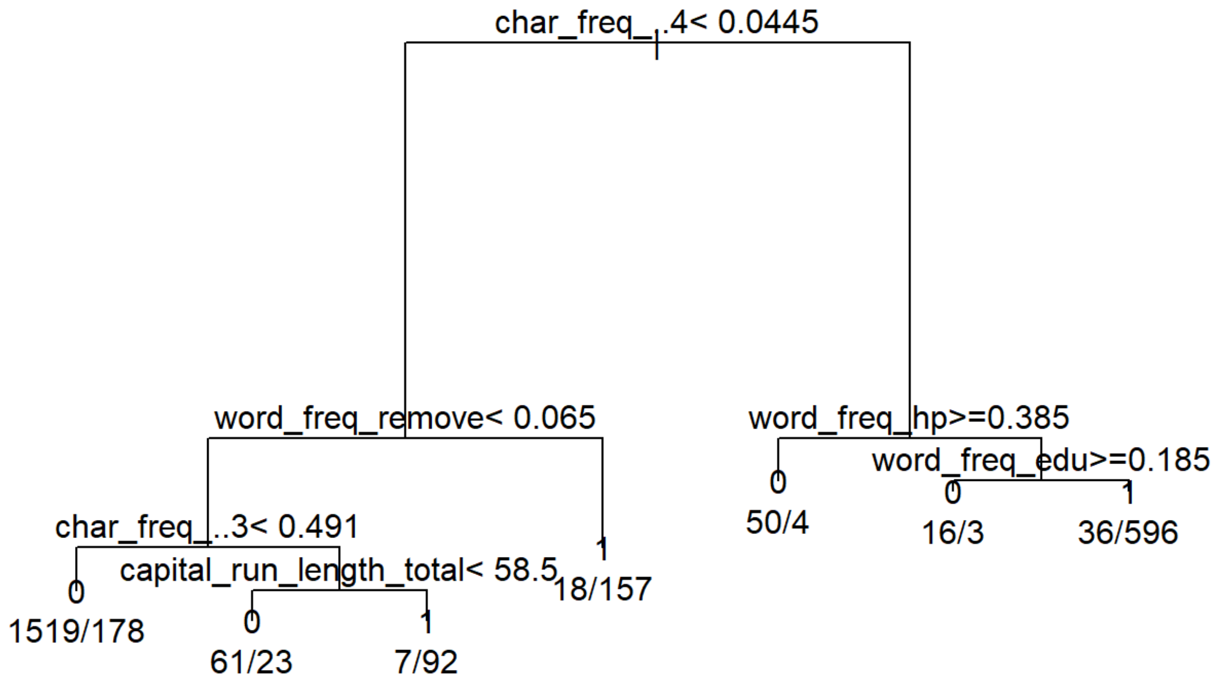
```
## [1] 0.9762398
```

높은 AUC값

rpart() 을 사용하여
Training set를 나무 모형에 적합

3-3-1. 나무 모형에 훈련세트 적합

```
# 나무모형
data_tr <- rpart(class ~ ., data = training)
data_tr
```



3-3-2. 검증세트 사용하여 모형 평가

```
yhat_tr <- predict(data_tr, validation)
yhat_tr <- yhat_tr[,"1"]
pred_tr <- prediction(yhat_tr, y_obs)
performance(pred_tr, "auc")@y.values[[1]]
```

```
## [1] 0.893021
```

```
#예측력이 약한 편이다
```

3. 데이터 분석

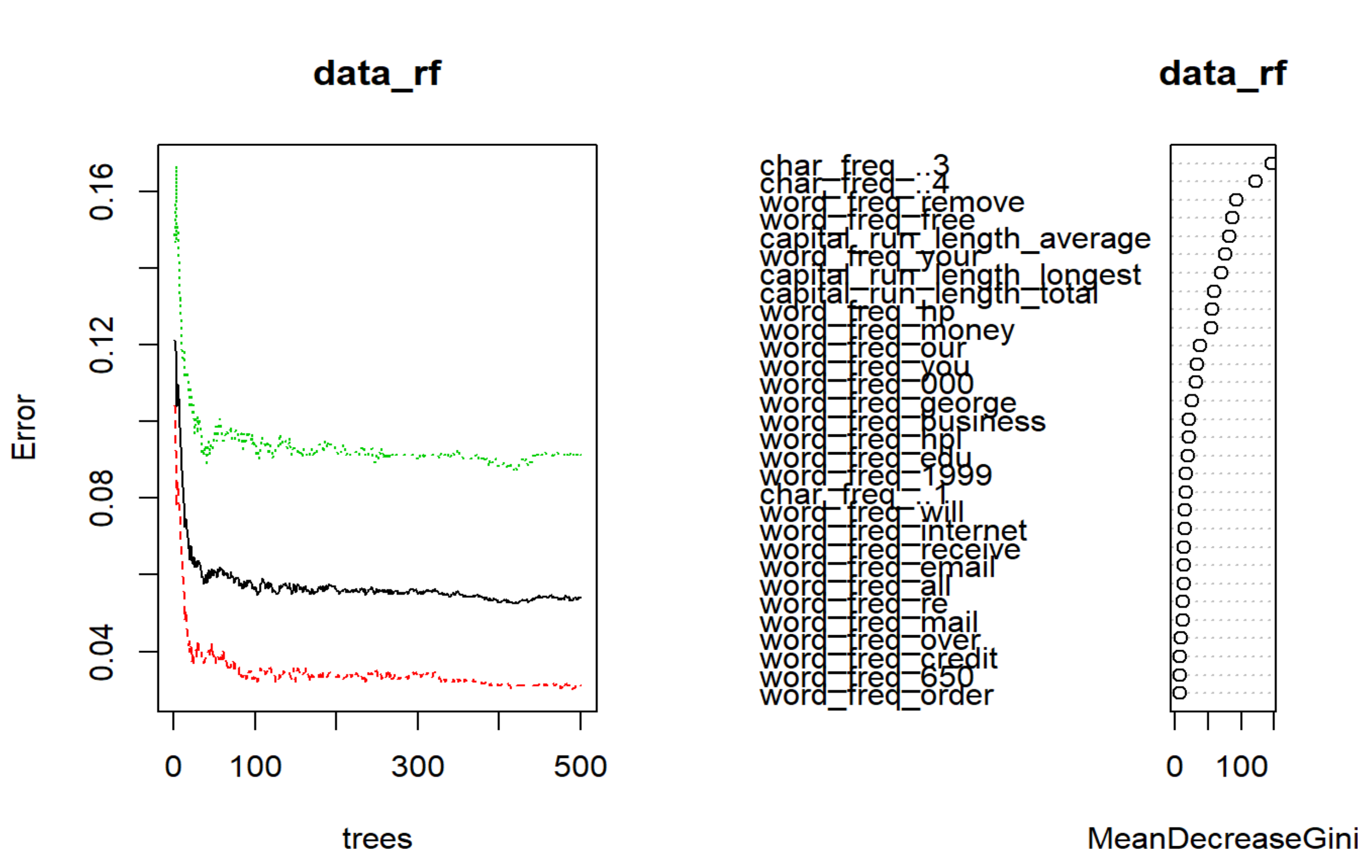
3-4. 랜덤 포레스트

3-4-1. 랜덤 포레스트 모형에 훈련세트 적합

```
set.seed(2018)
data_rf <- randomForest(class ~ ., data=training)
data_rf
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 5.4%
## Confusion matrix:
##      0   1 class.error
## 0 1654  53  0.03104862
## 1   96 957  0.09116809
```

```
opar <- par(mfrow=c(1,2))
plot(data_rf) #데이터에서 나무 수에 따른 오차율의 감소
#tree가 100개면 꽤 괜찮은 측정값을 얻을 수 있음
varImpPlot(data_rf)#각 예측변수의 중요도
```



3-4-2. 검증세트 사용하여 모형 평가

```
yhat_rf <- predict(data_rf, newdata=validation, type='prob')[,'1']
pred_rf <- prediction(yhat_rf, y_obs)
performance(pred_rf, "auc")@y.values[[1]]
```

```
## [1] 0.9886151
```

4. 모형평가

```
data.frame(method=c('rf', 'lm', 'tr'),
           auc = c(performance(pred_rf, "auc")@y.values[[1]],
                   performance(pred_lm, "auc")@y.values[[1]],

                   performance(pred_tr, "auc")@y.values[[1]]
                   ))
```

```
##   method       auc
## 1     rf 0.9886151
## 2     lm 0.9762398
## 3     tr 0.8930210
```

랜덤포레스트 〉 로지스틱 회귀 〉 decision Tree

랜덤 포레스트 in Test set

```
# 랜덤 포레스트
y_obs_test <- as.numeric(as.character(test$class))
yhat_rf_test <- predict(data_rf, newdata=test, type='prob')[,'1']
pred_rf_test <- prediction(yhat_rf_test, y_obs_test)
performance(pred_rf_test, "auc")@y.values[[1]]
```

```
## [1] 0.9885861
```

THANK YOU

Reference

1. 따라하며 배우는 데이터 과학 (저자: 권재명)
2. http://redhorse046.tistory.com/
3. https://cran.r-project.org/