4. Modelling

2022-06-20

Contents

split data 2
predictive models 3
Decision Tree 3
predict() 4
predict test set 5
accuracy rate 5
SVM (Support Vector Machines) 5
SVM model 1 5
SVM model 2 5
Logistic Regression 6
glm() 6
model comparison 7


```
# Load caTools package for data partitioning
library(caTools)

# Import spambase.csv and assing it to mydata
mydata <- read.csv('spambase.csv')

# Display the structure of the data file
str(mydata)
```

Aim: predict whether an e-mail is spam or not

```
## 'data.frame':    4601 obs. of  32 variables:
##  $ freq_make       : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ freq_address    : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ freq_all        : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ freq_our        : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ freq_over       : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ freq_remove     : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ freq_internet   : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ freq_order      : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
```

```
##  $ freq_mail       : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ freq_receive    : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ freq_will       : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ freq_people     : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ freq_free       : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ freq_business   : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ freq_email      : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ freq_you        : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ freq_credit     : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ freq_your       : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ freq_000        : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ freq_money      : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ freq_data       : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ freq_original   : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ freq_project    : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ freq_re         : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ freq_parentheses: num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ freq_exclamation: num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ freq_dollar     : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ freq_sharp      : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ length_average  : num  3.76 5.11 9.82 3.54 3.54 ...
##  $ length_longest  : int  61 101 485 40 40 15 4 11 445 43 ...
##  $ length_total    : int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ class           : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Change the data type of target variable if necessary
mydata$class <- as.factor(mydata$class)

levels(mydata$class)
```

```
## [1] "0" "1"
```

split data

70% of the data should be allocated to training set and the remaining should be allocated to the test set.

```r
#Set a seed of 123
set.seed(123)

#Generate a vector split
split <- sample.split(mydata, SplitRatio = 0.7)

# Create training set: training
training <- subset(mydata, split == TRUE)

# Create test set: test
test <- subset(mydata, split == FALSE)
```

To use sample.split() function, load caTools package.

predictive models

Decision Tree

```
# install.packages("tree")
# install.packages("maptree")

# Load tree library
library(tree)
```

In R, tree library is used to construct classification and regression trees.

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
# Load maptree library for plotting
library(maptree)
```

```
## Loading required package: cluster

## Loading required package: rpart
```

```
# tree(target~.,data)
```

Use tree() function to fit a classification tree in order to predict class of an e-mail.

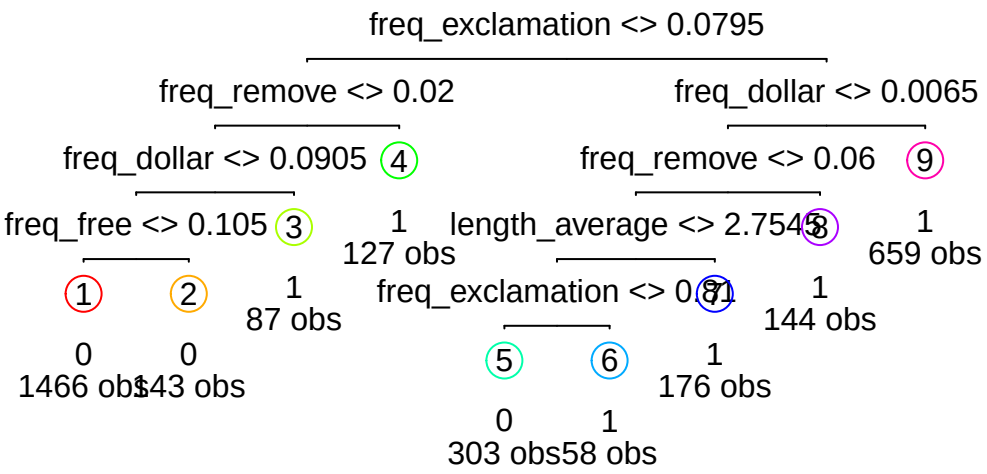
```
# Build the decision tree by using tree() function
tree_spam <- tree(class~. , training)

# Display the summary of your model and print the model
summary(tree_spam)
```

Use summary() function to list the features that are used in the tree, the number of terminal nodes, and the (training) error rate.

```
##
## Classification tree:
## tree(formula = class ~ ., data = training)
## Variables actually used in tree construction:
## [1] "freq_exclamation" "freq_remove"      "freq_dollar"      "freq_free"
## [5] "length_average"
## Number of terminal nodes:  9
## Residual mean deviance:  0.5893 = 1859 / 3154
## Misclassification error rate: 0.1065 = 337 / 3163
```

```
# Plot the model
draw.tree(tree_spam)
```




```
# tree.control(nobs, mincut, minsize, mindev)

# nobs: The number of observations in the training set.
# mincut: The minimum number of observations to have in a child node (the default value is 5).
# minsize: The smallest number of observations in a node (the default value is 10).
# mindev: In order to split a node, the deviance must be at least this times that of the root node.
```

Note that the decision tree parameters can be controled using the tree.control argument.

predict()

Use predict() function to predict the spam class in the test data.

```
# predict(modelname, testdata, type = "class")
```

predict() function for tree() model

predict test set

accuracy rate

```
# Predict the class of emails in test set
tree_predict = predict(tree_spam, test, type = "class")

# Find the percentage of correct predictions
accuracy_tree <- length(which(tree_predict == test$class))/nrow(test)
accuracy_tree
```

```
## [1] 0.8929068
```

SVM (Support Vector Machines)

Build an SVM model using training_set. Initially, set the kernel = "radial" and also try kernel = "linear".

```

# Load package e1071
library(e1071)
```

Predict e-mail classification (spam or not) for the test data.

SVM model 1

```
# kernel="radial"
# Build a SVM model by using svm() function
svm_spam_1  <- svm(class ~. , data = training, kernel = "radial", scale = TRUE)

# Predicting the Test set results
svm_predict_1 = predict(svm_spam_1, test)

# Find the percentage of correct predictions

accuracy_svm_1 <- length(which(svm_predict_1 == test$class))/nrow(test)
accuracy_svm_1
```

```
## [1] 0.9165508
```

SVM model 2

```
# kernel="linear"
# Build a SVM model by using svm() function
svm_spam_2  <- svm(class ~. , data = training, kernel = "linear", scale = TRUE)
```

```
# Predicting the Test set results
svm_predict_2 = predict(svm_spam_2 , test)

# Find the percentage of correct predictions

accuracy_svm_2 <- length(which(svm_predict_2 == test$class))/nrow(test)
accuracy_svm_2
```

```
## [1] 0.9012517
```

Logistic Regression

It is a classification technique which models the probability that a target variable belongs to a particular class.

glm()

```
# glm(formula, data, family = ...)

# Formula shows which features are used in modelling to predict target variable.
# Data is the dataset that will be used for model building.
# Family shows which type of model we want to develop. glm() function can be used to build generalised
```

```
# Build a logistic regression model assign it to LR_spam
LR_spam <- glm(class ~. , data = training, family = "binomial")
```

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

```
# predict(logistic regression model, test data, type="response")
```

```
# Predict the class probabilities of the test data
LR_prob <- predict(LR_spam, test, type="response")
```

Predict the class of the test data and store the result.

LR_prob will return the class scores (or probabilities). In order to predict the class of a test data, a cutoff value 0.5 is used.

```r
# Predict the class
LR_class <- ifelse(LR_prob >= 0.50, "1", "0")

# Save the predictions as factor variables
LR_class <- as.factor(LR_class)

# Find the percentage of correct predictions
accuracy_LR <- length(which(LR_class == test$class))/nrow(test)
accuracy_LR
```

If the probability of a record is greater than or equal to 0.5, it will be marked as "1", otherwise it will be marked as "0". These predictions need to be saved as factor variable.

```
## [1] 0.8929068
```

model comparison

```r
# Compare correct predictions obtained by these three models
# Return the total number of correct predictions for decision tree
accuracy_tree
```

```
## [1] 0.8929068
```

```r
# Return the total number of correct predictions for SVM
accuracy_svm_1
```

```
## [1] 0.9165508
```

```r
accuracy_svm_2
```

```
## [1] 0.9012517
```

```r
# Return the total number of correct predictions for logistic regression
accuracy_LR
```

```
## [1] 0.8929068
```