Machine Learning in R

Chenshu Liu

April 2022

Packages

```
# for data splitting into training and testing (general data manipulation)
library(caTools)

# dataset for logistic regression
# install.packages("mlbench")
library(mlbench)

# Decision tree
# install.packages("FSelector")
# install.packages("caret", dependencies = T)
# install.packages("rpart.plot")
# install.packages("data.tree")
# install.packages("rJava")

# for constructing decision tree
system("java -version")
library(FSelector)
library(rpart)
# for test train set split
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
library(dplyr)
# plotting the decision tree
library(rpart.plot)
library(data.tree)
```

Linear Regression

Linear regression is a way to find the best fit linear expression that can show the observed trend(s). The parameters of the best fit line $y = mx + c$ can be calculated by:

$$m = \frac{(n \times \Sigma(x \times y)) - (\Sigma(x) \times \Sigma(y))}{(n \times \Sigma(x^2)) - (\Sigma(x)^2)}$$

$$c = \frac{(\Sigma(y) \times \Sigma(x^2)) - (\Sigma(x) \times \Sigma(x \times y))}{(n \times \Sigma(x^2)) - (\Sigma(x)^2)}$$

Data

```
sales <- read.csv("/Users/chenshu/Documents/Programming/R/Machine Learning in R/datasets/revenue.csv")

# split into training and testing data
set.seed(2)
# SplitRatio means the percentage of data for training
split <- caTools::sample.split(sales$Profit, SplitRatio = 0.7)
train <- sales[split, ]
test <- sales[!split, ]
```

Modeling

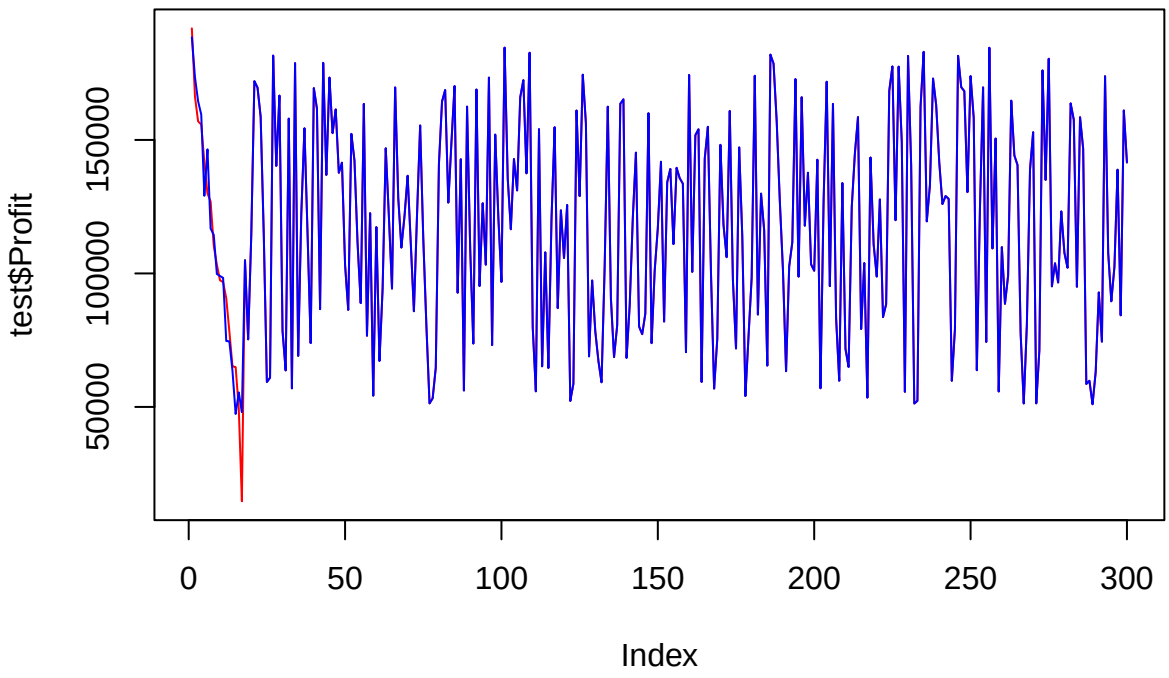
```
Model <- lm(Profit ~., data = train)
summary(Model)
```

```
##
## Call:
## lm(formula = Profit ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16134.9    -32.2    -17.7     -6.5  10133.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.395e+04  1.145e+03  47.126  < 2e-16 ***
## Paid         8.155e-01  6.982e-03 116.802  < 2e-16 ***
## Organic     -6.007e-02  9.547e-03  -6.292 5.53e-10 ***
## Social       2.488e-02  3.247e-03   7.661 6.21e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1480 on 696 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9986
## F-statistic: 1.693e+05 on 3 and 696 DF,  p-value: < 2.2e-16
```

Predict

```
pred <- predict(Model, test)

# comparing predicted vs. actual values
plot(test$Profit, type = 'l', lty = 1.8, col = "red")
lines(pred, type = 'l', lty = 1.8, col = "blue")
```




```
# determining prediction accuracy
rmse <- sqrt(mean(pred - test$Profit)^2)
rmse
```

```
## [1] 61.56887
```

Logistic Regression

Logistic regression is a **classification algorithm**, not a linear prediction algorithm. Different from linear regression, which is usually used to determine the magnitude of the effect, logistic regression is used to predict binary outcome.

Data

```
data(PimaIndiansDiabetes)
log_df <- PimaIndiansDiabetes
head(log_df)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35       0 33.6    0.627  50      pos
## 2        1      85       66      29       0 26.6    0.351  31      neg
## 3        8     183       64       0       0 23.3    0.672  32      pos
```

```
## 4        1      89       66       23      94 28.1    0.167  21       neg
## 5        0     137       40       35     168 43.1    2.288  33       pos
## 6        5     116       74        0       0 25.6    0.201  30       neg
```

```r
# splitting data
split <- caTools::sample.split(log_df$diabetes, SplitRatio = 0.7)
train <- log_df[split, ]
test <- log_df[!split, ]

# data pre-processing
# logistic regression takes in factor type variables
train$diabetes <- as.factor(train$diabetes)
```

Modeling

```r
log_mod <- glm(diabetes ~., data = train, family = "binomial")
summary(log_mod)
```

```
##
## Call:
## glm(formula = diabetes ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4160  -0.7396  -0.4437   0.7655   2.7580
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.9513478  0.8219838  -9.673  < 2e-16 ***
## pregnant     0.1365115  0.0372059   3.669 0.000243 ***
## glucose      0.0313336  0.0041092   7.625 2.44e-14 ***
## pressure    -0.0127133  0.0061062  -2.082 0.037339 *
## triceps      0.0002463  0.0080616   0.031 0.975628
## insulin     -0.0010894  0.0010684  -1.020 0.307890
## mass         0.0894903  0.0177557   5.040 4.65e-07 ***
## pedigree     0.7669301  0.3433048   2.234 0.025486 *
## age          0.0152128  0.0108031   1.408 0.159073
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 696.28  on 537  degrees of freedom
## Residual deviance: 520.90  on 529  degrees of freedom
## AIC: 538.9
##
## Number of Fisher Scoring iterations: 5
```

Prediction

```
pred <- predict(log_mod, test, type = "response")

# confusion matrix to check prediction accuracy
table(Actual_value = test$diabetes, Predicted_value = pred > 0.5)
```

```
##              Predicted_value
## Actual_value FALSE TRUE
##          neg   137   13
##          pos    30   50
```

Decision Tree

1. Decision tree is a tree shape algorithm that is used to determine a course of actions, with each branch on the tree representing a possible decision
2. Decision tree can be used to solve classification problems
3. Decision tree can also be used to solve continuous predictions such as regression

Entropy describes the messiness of the problem being classified. The messier the problem is, the larger the entropy. In decision tree problems, we can use the change in entropy to determine what the decision node is. **The optimum decision node is where the information gain is the largest (i.e. reduces the most entropy)**.

The entropy of decision problem can be calculated as:

$$-\Sigma_{x=1}^{i} p(value_x) log_2(p(value_x))$$

Where value is the proportion of occurrence of one group $value_x = \frac{counts \quad in \quad one \quad group}{total \quad counts}$

Whichever category can reduce the entropy the greatest will be the node for classification.

Data

```
path <- 'https://raw.githubusercontent.com/guru99-edu/R-Programming/master/titanic_data.csv'
titanic <-read.csv(path)

# data cleaning
titanic <- select(titanic, survived, pclass, sex, age)
titanic <- mutate(titanic, survived = factor(survived), age = as.numeric(age))
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
# data splitting
set.seed(123)
sample = sample.split(titanic$survived, SplitRatio = 0.7)
train <- titanic[sample, ]
test <- titanic[!sample, ]
```

Modeling

```r
tree <- rpart(survived ~., data = train)
```

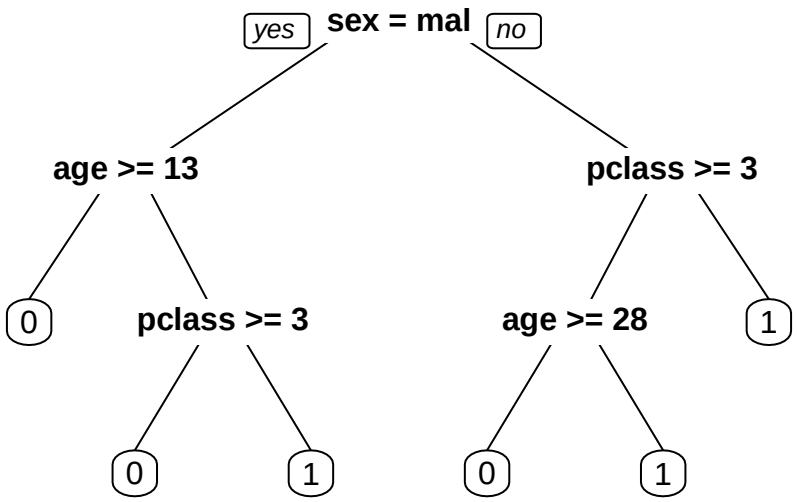
Prediction

```r
tree.survived.pred <- predict(tree, test, type = "class")

# evaluate model accuracy
confusionMatrix(tree.survived.pred, test$survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 218  47
##          1  25 103
##
##                Accuracy : 0.8168
##                  95% CI : (0.7749, 0.8538)
##     No Information Rate : 0.6183
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6006
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 0.8971
##             Specificity : 0.6867
##          Pos Pred Value : 0.8226
##          Neg Pred Value : 0.8047
##              Prevalence : 0.6183
##          Detection Rate : 0.5547
##    Detection Prevalence : 0.6743
##       Balanced Accuracy : 0.7919
##
##        'Positive' Class : 0
##
```

```r
# visualize decision tree
prp(tree)
```


yes
sex = mal
no
age >= 13
pclass >= 3
0
pclass >= 3
age >= 28
1
0
1
0
1