**Project 8**

주제: 통계/ML 기반 데이터 분석 비교

부제: 내장 패키지의 Train 모델과 caret 패키지의 Train 모델 비교

Team) 오준서(Leader) 유준 천성한 이순규

목차

[**분석** 3](#_Toc94750260)

[１ Spam DATA SET 3](#_Toc94750261)

[１-１ 나이브 베이즈 4](#_Toc94750262)

[１-２ SVM(서포트 벡터 머신) 7](#_Toc94750263)

[１-３ 로지스틱회귀분석 10](#_Toc94750264)

[１-４ KNN(최근접 이웃 모델) 12](#_Toc94750265)

[１-５ 인공신경망 14](#_Toc94750266)

[２ Abalone DATA SET 16](#_Toc94750267)

[２-１ 의사결정나무 17](#_Toc94750268)

[２-２ 다중회귀분석 19](#_Toc94750269)

[３ Titanic DATA SET 22](#_Toc94750270)

[３-１ two sample T-test 24](#_Toc94750271)

[３-２ 랜덤포레스트 28](#_Toc94750272)

[４ Women DATA SET 31](#_Toc94750273)

[４-１ 단순선형회귀 32](#_Toc94750274)

[５ economics Data Set 35](#_Toc94750275)

[５-１ 시계열 36](#_Toc94750276)

[６ Pima.te DATA SET 38](#_Toc94750277)

[６-１ Roc커브 39](#_Toc94750278)

[７ Iris DATA SET 42](#_Toc94750279)

[７-１ Xgboost 43](#_Toc94750280)

[７-２ 배깅 46](#_Toc94750281)

[７-３ 부스팅 48](#_Toc94750282)

[참고자료 50](#_Toc94750283)

# **분석**

## Spam DATA SET

사용 데이터(spam)

# spam데이터 설명

# 불러온 spam 데이터는 4601개의 이메일에서 등장하는 단어의 종류와 관련된 58개의 변수로 구성되어 있다.

# 58개의 변수 중 처음 48개 변수(A.1~A.48)은 총 단어 수 대비 해당 단어의 출현비율을 나타내며,

# 6개 변수(A.49~A.54)는 총 문자 수 대비 특정 문자의 출현비율을 나타내며,

# 3개 변수(A.55~A.57)은 연속되는 대문자 철자의 평균길이, 최대길이, 대문자의 총수를 나타낸다.

# 마지막 변수(spam)스팸 메일의 여부를 나타냅니다.

(표1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A.1** | **A.2** | **A.3** | **A.4** | **A.5** | **A.6** | **(중략)** | **A.56** | **A.57** | **spam** |
| 0 | 0.64 | 0.64 | 0 | 0.32 | 0 | (중략) | 61 | 278 | spam |
| 0.21 | 0.28 | 0.5 | 0 | 0.14 | 0.28 | (중략) | 101 | 1028 | spam |
| 0.06 | 0 | 0.71 | 0 | 1.23 | 0.19 | (중략) | 485 | 2259 | spam |
| 0 | 0 | 0 | 0 | 0.63 | 0 | (중략) | 40 | 191 | spam |
| 0 | 0 | 0 | 0 | 0.63 | 0 | (중략) | 40 | 191 | spam |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | (중략) | …(생략) | …(생략) | …(생략) |

### 나이브 베이즈

#(1)나이브베이즈 학습 모델 생성  
 spam\_nb\_doby <- naiveBayes(spam ~ .,  
 data = spam\_train\_doBy,  
 laplace = 1)

#(2)예측 분류 결과 생성  
 spam\_nb\_pred\_doby <- predict(spam\_nb\_doby, newdata = spam\_test\_doBy, type = 'class')

#(3)나이브베이즈 적용 분류 결과 도출  
 table(spam\_nb\_pred\_doby, spam\_test\_doBy$spam)

##   
## spam\_nb\_pred\_doby email spam  
## spam 358 515  
## email 478 29

#(4)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_nb\_pred\_doby, as.factor(spam\_test\_doBy$spam))

## Warning in confusionMatrix.default(spam\_nb\_pred\_doby,  
## as.factor(spam\_test\_doBy$spam)): Levels are not in the same order for reference  
## and data. Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 478 29  
## spam 358 515  
##   
## Accuracy : 0.7196   
## 95% CI : (0.6951, 0.7431)  
## No Information Rate : 0.6058   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4689   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5718   
## Specificity : 0.9467   
## Pos Pred Value : 0.9428   
## Neg Pred Value : 0.5899   
## Prevalence : 0.6058   
## Detection Rate : 0.3464   
## Detection Prevalence : 0.3674   
## Balanced Accuracy : 0.7592   
##   
## 'Positive' Class : email

# caret패키지

#(1)모델 생성  
 ctrl <- trainControl(method="cv", 10)  
 spam\_nb\_caret <- train(spam ~ ., data = spam\_train\_caret,   
 method = 'naive\_bayes',  
 trControl = ctrl)

#(2)예측 분류 결과 생성  
 spam\_nb\_pred\_caret <- predict(spam\_nb\_caret, newdata = spam\_test\_caret)

#(3)나이브베이즈 적용 분류 결과 도출  
 table(spam\_nb\_pred\_caret, spam\_test\_caret$spam)

##   
## spam\_nb\_pred\_caret email spam  
## email 449 27  
## spam 387 516

#(4)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_nb\_pred\_caret, as.factor(spam\_test\_caret$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 449 27  
## spam 387 516  
##   
## Accuracy : 0.6998   
## 95% CI : (0.6748, 0.7239)  
## No Information Rate : 0.6062   
## P-Value [Acc > NIR] : 2.897e-13   
##   
## Kappa : 0.4366   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5371   
## Specificity : 0.9503   
## Pos Pred Value : 0.9433   
## Neg Pred Value : 0.5714   
## Prevalence : 0.6062   
## Detection Rate : 0.3256   
## Detection Prevalence : 0.3452   
## Balanced Accuracy : 0.7437   
##   
## 'Positive' Class : email   
##

결과)

나이브 베이즈는 신뢰수준 95%에서 신뢰하한: 0.67 신뢰상한: 0.72로 나왔다. 정확도는 약 70% 수준이었다.

**내장 패키지**에서는 laplace 옵션을 통해 **라플라스 변환 여부**를 조절할 수 있었다.

- 라플라스 추정량은 빈도표의 각 합계에 작은 숫자를 더하는데, 특징이 각 클래스에 대해 발생할 확률이 0이 되지 않게 보장한다. 보통 라플라스 추정량은 1로 설정해서 데이터에 각 클래스 특징 조합이 최소 한번은 나타나게 보장한다.

**caret 패키지**에서는 trControl의 ‘cv’옵션으로 **교차검정 횟수**를 조절할 수 있었다.

### SVM(서포트 벡터 머신)

#(1)SVM 학습 모델 생성  
 spam\_svm\_doBy <- svm(factor(spam) ~ .,  
 data = spam\_train\_doBy,  
 gamma = 0.5,  
 cost = 4)

#(2)예측 분류 결과 생성  
 spam\_svm\_pred\_doBy <- predict(spam\_svm\_doBy, newdata = spam\_test\_doBy)

#(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_svm\_pred\_doBy, factor(spam\_test\_doBy$spam))

## Warning in confusionMatrix.default(spam\_svm\_pred\_doBy,  
## factor(spam\_test\_doBy$spam)): Levels are not in the same order for reference and  
## data. Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 830 256  
## spam 6 288  
##   
## Accuracy : 0.8101   
## 95% CI : (0.7884, 0.8305)  
## No Information Rate : 0.6058   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5678   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9928   
## Specificity : 0.5294   
## Pos Pred Value : 0.7643   
## Neg Pred Value : 0.9796   
## Prevalence : 0.6058   
## Detection Rate : 0.6014   
## Detection Prevalence : 0.7870   
## Balanced Accuracy : 0.7611   
##   
## 'Positive' Class : email   
##

# caret패키지  
 #(1)caret패키지 SVM훈련 모델 생성  
 ctrl <- trainControl(method="cv", 10)  
 spam\_svm\_caret <- train(spam ~ .,  
 data = spam\_train\_caret,  
 method = 'svmRadial',  
 trControl = ctrl,  
 tuneGrid = expand.grid(sigma= 0.5 , C = 4))

#(2)caret패키지 SVM학습모델 예측 분류 결과 생성  
 spam\_svm\_pred\_caret<- predict(spam\_svm\_caret, newdata = spam\_test\_caret)

#(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_svm\_pred\_caret, factor(spam\_test\_caret$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 824 224  
## spam 12 319  
##   
## Accuracy : 0.8289   
## 95% CI : (0.8079, 0.8484)  
## No Information Rate : 0.6062   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6152   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9856   
## Specificity : 0.5875   
## Pos Pred Value : 0.7863   
## Neg Pred Value : 0.9637   
## Prevalence : 0.6062   
## Detection Rate : 0.5975   
## Detection Prevalence : 0.7600   
## Balanced Accuracy : 0.7866   
##   
## 'Positive' Class : email   
##

결과)

SVM의 정확도는 약 82%였고, 각각 gamma(sigma), cost 옵션을 조절하여 **SVM 경계의 기울기**를 설정할 수 있었다.

SVM의 gamma(sigma)와, Cost 옵션은 아래와 같은 과정으로 도출되었다.

|  |
| --- |
| # tune.svm(factor(spam) ~ ., data = spam, gamma = 2^(-1:1), cost = 2^(1:4))  # 시간 오래 걸림, 결과 밑에 작성. |
| # Parameter tuning of ‘svm’: |
| # - sampling method: 10-fold cross validation |
| # - best parameters: |
| # gamma cost |
| # 0.5 4 |
| # - best performance: 0.16714 |

### 로지스틱회귀분석

#(1)glm함수 로지스틱 회귀분석 훈련모델 생성  
 spam\_glm\_doBy <- glm(factor(spam) ~ ., data = spam\_train\_doBy, family = 'binomial')

#(2)예측 분류 결과 생성  
 spam\_glm\_pred\_doBy <- predict(spam\_glm\_doBy, newdata = spam\_test\_doBy, type = 'response')

#(3)모델 성능 평가 지표(정확도 확인)  
 spam\_glm\_pred\_doBy2 <- ifelse(spam\_glm\_pred\_doBy < 0.5, 'spam', 'email') # 컷오프 0.5로 설정하여 사후확률이 0.5초과이면 spam, 0.5이하이면 email로 예측한다.  
 confusionMatrix(factor(spam\_glm\_pred\_doBy2), factor(spam\_test\_doBy$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 797 66  
## spam 39 478  
##   
## Accuracy : 0.9239   
## 95% CI : (0.9086, 0.9373)  
## No Information Rate : 0.6058   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.8393   
##   
## Mcnemar's Test P-Value : 0.01117   
##   
## Sensitivity : 0.9533   
## Specificity : 0.8787   
## Pos Pred Value : 0.9235   
## Neg Pred Value : 0.9246   
## Prevalence : 0.6058   
## Detection Rate : 0.5775   
## Detection Prevalence : 0.6254   
## Balanced Accuracy : 0.9160   
##   
## 'Positive' Class : email   
##

# caret 패키지  
 #(1)caret 패키지 로지스틱 회귀 훈련 모델 생성  
 ctrl <- trainControl(method="cv", 10)  
 spam\_glm\_caret <- train(factor(spam) ~ .,  
 data = spam\_train\_caret,  
 method = 'glm',  
 trControl = ctrl)

#(2)caret패키지 로지스틱 회귀 예측 분류 결과 생성  
 spam\_glm\_pred\_caret <- predict(spam\_glm\_caret, newdata =spam\_test\_caret, type = 'raw')

#(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(factor(spam\_glm\_pred\_caret), factor(spam\_test\_caret$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 792 87  
## spam 44 456  
##   
## Accuracy : 0.905   
## 95% CI : (0.8883, 0.92)  
## No Information Rate : 0.6062   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7982   
##   
## Mcnemar's Test P-Value : 0.000243   
##   
## Sensitivity : 0.9474   
## Specificity : 0.8398   
## Pos Pred Value : 0.9010   
## Neg Pred Value : 0.9120   
## Prevalence : 0.6062   
## Detection Rate : 0.5743   
## Detection Prevalence : 0.6374   
## Balanced Accuracy : 0.8936   
##   
## 'Positive' Class : email   
##

결과)

정확도는 약 90%가 나왔다. stats패키지의 glm함수는 이분형 로지스틱 분석을 위해서 **별도로 binomial을 지정**해야 했지만, caret 패키지는 **자동**으로 이항분류를 해주었다.

### KNN(최근접 이웃 모델)

# class 패키지  
 #(1)doBy 데이터 셋 라벨링  
 spam\_train\_label\_doBy <- spam$spam[nurow]   
 spam\_test\_label\_doBy <- spam$spam[-nurow]

#(2)class패키지 knn학습모델 생성(k = 57)  
 spam\_knn\_doBy <- knn(train = spam\_train\_doBy[,-58],  
 test = spam\_test\_doBy[,-58],  
 cl = spam\_train\_label\_doBy,  
 k = 57)

#(3)class패키지 knn학습모델 분류 결과 도출  
 tt <- table(spam\_test\_label\_doBy, spam\_knn\_doBy)

#(4)모델 성능 평가 지표(정확도 확인)  
 sum(tt[row(tt) == col(tt)])/sum(tt) # 정분류율

## [1] 0.884058

1-sum(tt[row(tt) == col(tt)])/sum(tt) # 오분류율

## [1] 0.115942

# caret 패키지  
 #(1)caret패키지 knn학습모델 생성(k = 57)  
 tune <- trainControl(method = 'cv', number = 10)  
 spam\_knn\_caret <- train(spam ~ ., data = spam\_train\_caret,  
 method = 'knn',  
 tuneGrid = expand.grid(k=57),  
 trControl = tune)

#(2)caret패키지 knn학습모델 분류 결과 도출  
 spam\_knn\_pred\_caret<- predict(spam\_knn\_caret, newdata = spam\_test\_caret)

#(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_knn\_pred\_caret, as.factor(spam\_test\_caret$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 745 80  
## spam 91 463  
##   
## Accuracy : 0.876   
## 95% CI : (0.8574, 0.8929)  
## No Information Rate : 0.6062   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7412   
##   
## Mcnemar's Test P-Value : 0.4444   
##   
## Sensitivity : 0.8911   
## Specificity : 0.8527   
## Pos Pred Value : 0.9030   
## Neg Pred Value : 0.8357   
## Prevalence : 0.6062   
## Detection Rate : 0.5402   
## Detection Prevalence : 0.5983   
## Balanced Accuracy : 0.8719   
##   
## 'Positive' Class : email   
##

결과)

KNN 분석을 위해서는 별도의 라벨링이 필요하다. 정확도는 약 88%가 나왔다.

**옵션 설명**

# k=57 설정 이유, k의 제곱이 nrow(data)값과 비슷하게 나오는 k를 설정

# k는 홀수로 설정, k를 짝수로 설정하지 않는 이유는

# 만일 근처 데이터에 상반된 유형이 반으로 나뉘게 된다면,

# 해당 데이터를 무엇으로 분류해야 할지에 대한 교착 상태가 일어나기 때문

### 인공신경망

#nnet 패키지  
 #(1)nnet패키지 인공신경망 학습모델 생성  
 spam\_nnet\_doBy <- nnet(factor(spam) ~ .,  
 data = spam\_train\_doBy,  
 size = 4,  
 decay = 5e-04) # 가장 정확하다는 옵션 선택.

#(2)nnet패키지 인공신경망 학습모델 분류 결과 도출  
 spam\_nnet\_pred\_doBy <- predict(spam\_nnet\_doBy, newdata = spam\_test\_doBy, type = 'class')  
   
 #(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(factor(spam\_nnet\_pred\_doBy), factor(spam\_test\_doBy$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 796 56  
## spam 40 488  
##   
## Accuracy : 0.9304   
## 95% CI : (0.9157, 0.9433)  
## No Information Rate : 0.6058   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8536   
##   
## Mcnemar's Test P-Value : 0.1258   
##   
## Sensitivity : 0.9522   
## Specificity : 0.8971   
## Pos Pred Value : 0.9343   
## Neg Pred Value : 0.9242   
## Prevalence : 0.6058   
## Detection Rate : 0.5768   
## Detection Prevalence : 0.6174   
## Balanced Accuracy : 0.9246   
##   
## 'Positive' Class : email   
##

#caret패키지  
 #(1)caret패키지 인공 신경망 학습모델 생성  
 spma\_nnet\_caret <- train(spam ~ .,  
 data = spam\_train\_caret,  
 method = 'nnet',  
 trace = F,  
 tuneGrid = expand.grid(.size= 4, .decay = 5e-04))  
   
 #(2)caret패키지 인공 신경망학습모델 분류 결과 도출  
 spam\_nnet\_pred\_caret<- predict(spma\_nnet\_caret, newdata = spam\_test\_caret)  
   
 #(3)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(spam\_nnet\_pred\_caret, as.factor(spam\_test\_caret$spam))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction email spam  
## email 803 40  
## spam 33 503  
##   
## Accuracy : 0.9471   
## 95% CI : (0.9339, 0.9583)  
## No Information Rate : 0.6062   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8889   
##   
## Mcnemar's Test P-Value : 0.4825   
##   
## Sensitivity : 0.9605   
## Specificity : 0.9263   
## Pos Pred Value : 0.9526   
## Neg Pred Value : 0.9384   
## Prevalence : 0.6062   
## Detection Rate : 0.5823   
## Detection Prevalence : 0.6113   
## Balanced Accuracy : 0.9434   
##   
## 'Positive' Class : email   
##

결과)

정확도는 약 94%가 나왔다.

**은닉층의 개수는 4개로 설정**했을 때 가장 좋은 결과값을 보였다.

## Abalone DATA SET

사용 데이터(Abalone)

# Abalone데이터 설명

# 불러온 Abalone 데이터는 4177개의 전복에서 관측된 9개의 변수로 구성되어 있다.

# 9개의 변수 중 첫번째 컬럼은 범주형 변수에 속하며, 8개의 컬럼이 연속형 변수이다.

# 각 변수에 대한 설명

Sex 성별 Length 길이 Diameter 직경 Height 높이 Whole weight 전체 무게 Shucked weight 축소된 무게 Viscera weight 내장 무게 Shell weight 껍질 무게 Rings +1.5를 한 전복의 나이

(표2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sex | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | Shell weight | Rings |
| M | 0.455 | 0.365 | 0.095 | 0.514 | 0.2245 | 0.101 | 0.15 | 15 |
| M | 0.35 | 0.265 | 0.09 | 0.2255 | 0.0995 | 0.0485 | 0.07 | 7 |
| F | 0.53 | 0.42 | 0.135 | 0.677 | 0.2565 | 0.1415 | 0.21 | 9 |
| M | 0.44 | 0.365 | 0.125 | 0.516 | 0.2155 | 0.114 | 0.155 | 10 |
| I | 0.33 | 0.255 | 0.08 | 0.205 | 0.0895 | 0.0395 | 0.055 | 7 |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) |

**# 종속변수가 범주형이고 설명변수가 연속형의 자료이므로,**

**# 통계기반 분석기법에서는 다중회귀분석을 실시할 수 있고,**

**# ML기반 분석기법은 의사결정나무를 활용했다.**

### 의사결정나무

#귀무가설: 각종 전복의 수치로 성별을 오차범위 5%내로 구분할 수 없다.

#대립가설: 각종 전복의 수치로 성별을 오차범위 5%내로 구분할 수 있다.

# party 패키지  
 #(1)학습모델 생성  
 treeOption1 <- ctree\_control(maxdepth = 10)  
 abalone\_tree1 <- ctree(Sex~.,  
 data = abalone\_doBy\_train,  
 controls = treeOption1)  
   
 #(2)예측치 생성  
 table(abalone\_doBy\_train$Sex, predict(abalone\_tree1,data=abalone\_doBy\_train),dnn=c('Actual','Predicted'))

## Predicted  
## Actual M F  
## M 1070 0  
## F 915 0

#(3)모형의 정확성 검정  
 confusionMatrix(data=abalone\_doBy\_test$Sex,predict(abalone\_tree1,abalone\_doBy\_test))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction M F  
## M 458 0  
## F 392 0  
##   
## Accuracy : 0.5388   
## 95% CI : (0.5046, 0.5727)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.5388   
## Specificity : NA   
## Pos Pred Value : NA   
## Neg Pred Value : NA   
## Prevalence : 1.0000   
## Detection Rate : 0.5388   
## Detection Prevalence : 0.5388   
## Balanced Accuracy : NA   
##   
## 'Positive' Class : M   
##

# caret 패키지  
 #(1)모델 생성 및 시각화  
 treemod <- train(Sex ~., method = "ctree", data=abalone\_caret\_train)  
   
 #(2)예측 및 모델 평가  
 pred = predict(newdata=abalone\_caret\_test,treemod)  
   
 #(3)모델 평가  
 confusionMatrix(pred, abalone\_caret\_test$Sex)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction F M  
## F 0 0  
## M 392 458  
##   
## Accuracy : 0.5388   
## 95% CI : (0.5046, 0.5727)  
## No Information Rate : 0.5388   
## P-Value [Acc > NIR] : 0.5141   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.5388   
## Prevalence : 0.4612   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : F   
##

결과)

정확도는 약 54%가 나왔다. 여기서 p-value값이 유의수준 95%하에서 0.05보다 크므로, **대립가설을 기각**한다. 따라서, 전복의 성별은 전복의 각종 수치로는 구별이 불가능하다.

### 다중회귀분석

#stats 패키지  
 #(1)학습모델 생성  
 abalone\_lm\_model <- lm(Rings ~., data=abalone\_doBy\_train)  
 summary(abalone\_lm\_model) #p-값 확인: 0.05이하이므로 독립변수들 간의 모형은 유의하다.

##   
## Call:  
## lm(formula = Rings ~ ., data = abalone\_doBy\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.1433 -1.5427 -0.3353 1.0634 14.1013   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.39722 0.57106 11.202 < 2e-16 \*\*\*  
## SexF -0.04994 0.10706 -0.466 0.640970   
## Length -4.80674 2.82705 -1.700 0.089237 .   
## Diameter 10.17192 3.36076 3.027 0.002505 \*\*   
## Height 6.27662 1.87442 3.349 0.000828 \*\*\*  
## WholeWeight 9.66846 1.01999 9.479 < 2e-16 \*\*\*  
## ShuckedWeight -19.70785 1.14724 -17.179 < 2e-16 \*\*\*  
## VisceraWeight -8.07370 1.73508 -4.653 3.48e-06 \*\*\*  
## ShellWeight 8.05941 1.56909 5.136 3.08e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.35 on 1976 degrees of freedom  
## Multiple R-squared: 0.388, Adjusted R-squared: 0.3855   
## F-statistic: 156.6 on 8 and 1976 DF, p-value: < 2.2e-16

#(2)다중공선성 확인

vif(abalone\_lm\_model)

## Sex Length Diameter Height WholeWeight   
## 1.024155 25.601069 24.450132 1.926609 74.467523   
## ShuckedWeight VisceraWeight ShellWeight   
## 20.856707 11.018083 14.174415

#(3)다중공선성 문제가 가장 심각한 변수를 제외한다.  
 abalone\_lm\_model2 <- lm(Rings ~ ., data = abalone\_doBy\_train[,-5])  
 summary(abalone\_lm\_model2)

##   
## Call:  
## lm(formula = Rings ~ ., data = abalone\_doBy\_train[, -5])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.8495 -1.5804 -0.3865 1.0531 15.9758   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.98245 0.58203 10.279 < 2e-16 \*\*\*  
## SexF -0.03332 0.10943 -0.304 0.760820   
## Length -4.98590 2.88981 -1.725 0.084623 .   
## Diameter 11.75263 3.43122 3.425 0.000627 \*\*\*  
## Height 6.07713 1.91596 3.172 0.001538 \*\*   
## ShuckedWeight -10.62379 0.64468 -16.479 < 2e-16 \*\*\*  
## VisceraWeight 1.89333 1.41085 1.342 0.179757   
## ShellWeight 19.89890 0.97082 20.497 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.402 on 1977 degrees of freedom  
## Multiple R-squared: 0.3601, Adjusted R-squared: 0.3579   
## F-statistic: 159 on 7 and 1977 DF, p-value: < 2.2e-16

#결과 해석  
 # 성별의 p-값이 다소 높지만 큰 영향을 주는 변수는 아니기 때문에 무시하기로 한다.  
 # 그 외에 Rings에 가장 큰영향을 미치는 변수는 Shellweight, Diameter, shuckedWeight  
 # 순서로 영향을 미쳤다.

#caret 패키지  
 #(1) 학습모델 생성  
 ctrl <- trainControl(method="cv", 10)  
 abalone\_lm\_caret <- train(Rings ~ .,  
 data = abalone\_caret\_train[,-5],  
 na.action = na.omit,  
 method = 'lm',  
 trControl = ctrl)  
 summary(abalone\_lm\_caret)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.0582 -1.5746 -0.3984 1.0633 16.0481   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.09354 0.61048 8.343 < 2e-16 \*\*\*  
## SexM -0.07135 0.11187 -0.638 0.5237   
## Length -1.38573 3.01179 -0.460 0.6455   
## Diameter 8.00091 3.59880 2.223 0.0263 \*   
## Height 15.11430 3.21720 4.698 2.81e-06 \*\*\*  
## ShuckedWeight -10.64442 0.66911 -15.908 < 2e-16 \*\*\*  
## VisceraWeight -1.19942 1.48258 -0.809 0.4186   
## ShellWeight 19.46719 1.00007 19.466 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.459 on 1977 degrees of freedom  
## Multiple R-squared: 0.3714, Adjusted R-squared: 0.3692   
## F-statistic: 166.9 on 7 and 1977 DF, p-value: < 2.2e-16

#결과 해석  
 # 그 외에 Rings에 가장 큰영향을 미치는 변수는 Shellweight, Height, shuckedWeight  
 # 순으로 파악됐다.

## Titanic DATA SET

사용 데이터(Titanic)

# Titanic 데이터 설명

# 불러온 Titanic 데이터는 타이타닉 호 사고 사건에 관련한 1309명 승객 정보와 14개의 변수로 구성 되어있다.

# 14개의 변수 구성

# 범주형 변수 : pclass, survived, name, sex, embarked

# 연속형 변수 : age, sibsp, parch, fare

# 제외 변수 : ticket, cabin, boat, body, home.dest

# 각 변수에 대한 설명

|  |  |
| --- | --- |
| **pclass** | 1, 2, 3등석 정보를 각각 1, 2, 3으로 저장 |
| **survived** | 생존 여부. survived(생존), dead(사망) |
| **name** | 이름 |
| **sex** | 성별. female(여성), male(남성) |
| **age** | 나이 |
| **sibsp** | 함께 탑승한 형제 또는 배우자의 수 |
| **parch** | 함께 탑승한 부모 또는 자녀의 수 |
| **ticket** | 티켓 번호(제외) |
| **fare** | 티켓 요금 |
| **cabin** | 선실 번호(제외) |
| **embarked** | 탑승한 곳. C(Cherbourg), Q(Queenstown), S(Southampton) |

(표3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pclass | survived | name | sex | age | sibsp | parch | fare | embarked |
| 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29 | 0 | 0 | 211.3375 | S |
| 1 | 1 | Allison, Master. Hudson Trevor | male | 0.92 | 1 | 2 | 151.55 | S |
| 1 | 0 | Allison, Miss. Helen Loraine | female | 2 | 1 | 2 | 151.55 | S |
| 1 | 0 | Allison, Mr. Hudson Joshua Creighton | male | 30 | 1 | 2 | 151.55 | S |
| 1 | 0 | Allison, Mrs. Hudson J C (Bessie Waldo Daniels) | female | 25 | 1 | 2 | 151.55 | S |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) |

**# 종속변수가 범주형이고 설명변수가 혼합형의 자료이므로,**

**# 통계기반 분석기법에서는 Two sample T-test 기법을 적용했고,**

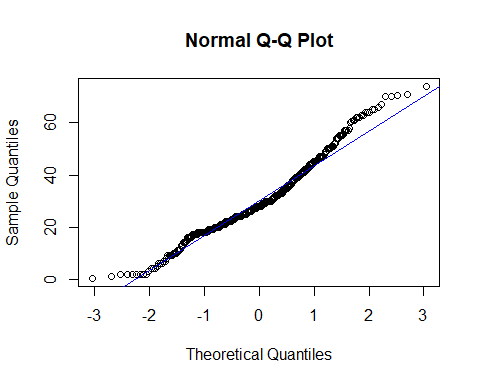
**# ML기반 분석기법에서는 randomForest 기법을 적용했다.**

### two sample T-test

# stats 패키지  
 #(1)대응하는 두 집단 생성  
 dead <- subset(titanic\_doBy\_train,titanic\_doBy\_train$survived == 0)  
   
 #(2)양측 검정 - titanic객체의 기존 모집단의 평균 28.92세 비교  
 t.test(dead$age, mu = 28.92)

##   
## One Sample t-test  
##   
## data: dead$age  
## t = 2.0755, df = 427, p-value = 0.03854  
## alternative hypothesis: true mean is not equal to 28.92  
## 95 percent confidence interval:  
## 28.99576 31.70438  
## sample estimates:  
## mean of x   
## 30.35007

qqnorm(dead$age)  
 qqline(dead$age, lty = 1, col = "blue")



t.test(dead$age, mu = 28.92, alter = "two.side", conf.level = 0.95)

##   
## One Sample t-test  
##   
## data: dead$age  
## t = 2.0755, df = 427, p-value = 0.03854  
## alternative hypothesis: true mean is not equal to 28.92  
## 95 percent confidence interval:  
## 28.99576 31.70438  
## sample estimates:  
## mean of x   
## 30.35007

#p-값이 유의수준 0.05보다 낮기 때문에 평균 수명에 차이가 있다고 볼 수 있다.  
   
 #(3)단측 검정 - 방향성을 가짐  
 t.test(dead$age, mu = 28.92, alter= "greater", conf.level = 0.95)

##   
## One Sample t-test  
##   
## data: dead$age  
## t = 2.0755, df = 427, p-value = 0.01927  
## alternative hypothesis: true mean is greater than 28.92  
## 95 percent confidence interval:  
## 29.21425 Inf  
## sample estimates:  
## mean of x   
## 30.35007

#(4)귀무가설의 임계값 계산  
 qt(0.05,427,lower.tail = F)

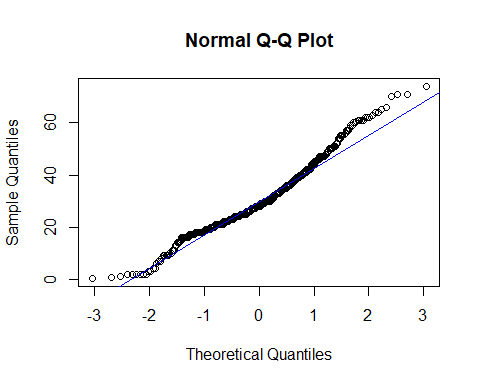
## [1] 1.64843

#귀무가설을 기각할 수 있는 임계값 = 1.64843#검정통계량 t=2.0755, 유의확률P=0.01927

# caret 패키지  
 #(1)대응하는 두 집단 생성  
 dead2 <- subset(titanic\_caret\_train,titanic\_caret\_train$survived == 0)  
   
 #(2)양측 검정 - titanic객체의 기존 모집단의 평균 28.92세 비교  
 t.test(dead2$age, mu = 28.92)

##   
## One Sample t-test  
##   
## data: dead2$age  
## t = 2.4023, df = 434, p-value = 0.01671  
## alternative hypothesis: true mean is not equal to 28.92  
## 95 percent confidence interval:  
## 29.20715 31.79091  
## sample estimates:  
## mean of x   
## 30.49903

qqnorm(dead2$age)  
 qqline(dead2$age, lty = 1, col = "blue")



t.test(dead2$age, mu = 28.92, alter = "two.side", conf.level = 0.95)

##   
## One Sample t-test  
##   
## data: dead2$age  
## t = 2.4023, df = 434, p-value = 0.01671  
## alternative hypothesis: true mean is not equal to 28.92  
## 95 percent confidence interval:  
## 29.20715 31.79091  
## sample estimates:  
## mean of x   
## 30.49903

#(3)단측 검정 - 방향성을 가짐  
 t.test(dead2$age, mu = 28.92, alter= "greater", conf.level = 0.95)

##   
## One Sample t-test  
##   
## data: dead2$age  
## t = 2.4023, df = 434, p-value = 0.008355  
## alternative hypothesis: true mean is greater than 28.92  
## 95 percent confidence interval:  
## 29.41557 Inf  
## sample estimates:  
## mean of x   
## 30.49903

#(4)귀무가설의 임계값 계산  
 qt(0.05,434,lower.tail = F)

## [1] 1.648372

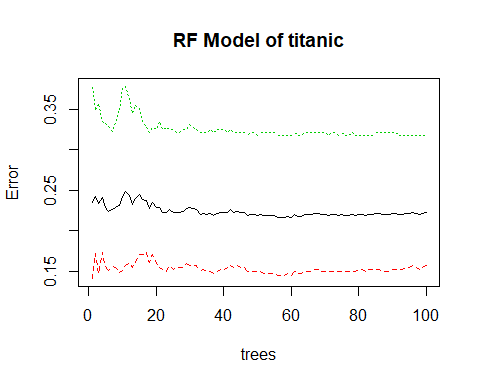
#귀무가설을 기각할 수 있는 임계값 = 1.64837#검정통계량 t=2.4023, 유의확률P=0.008355# 결론: 유의수준 0.05에서 귀무가설이 기각되므로,# 성별은 평균생존률에 차이가 있다.

### 랜덤포레스트

#(1)randomForest 패키지  
 # 모델 생성  
 titanic\_RF\_RF <- randomForest(survived~age+sex+embarked,  
 data=titanic\_doBy\_train,  
 na.action = na.omit,  
 ntree=100,  
 proximity=T)  
 table(titanic\_doBy\_train$survived)

##   
## 0 1   
## 428 299

plot(titanic\_RF\_RF,main="RF Model of titanic")



importance(titanic\_RF\_RF) # 노드 불순도 개선에 기여한 변수: sex > age > embarked

## MeanDecreaseGini  
## age 18.14739  
## sex 74.84015  
## embarked 16.11502

# 예측치 생성  
 titan\_pred\_doBy<- predict(titanic\_RF\_RF,  
 newdata = titanic\_doBy\_test)  
 confusionMatrix(titan\_pred\_doBy, factor(titanic\_doBy\_test$survived))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 160 40  
## 1 28 94  
##   
## Accuracy : 0.7888   
## 95% CI : (0.7401, 0.8321)  
## No Information Rate : 0.5839   
## P-Value [Acc > NIR] : 5.987e-15   
##   
## Kappa : 0.5598   
##   
## Mcnemar's Test P-Value : 0.1822   
##   
## Sensitivity : 0.8511   
## Specificity : 0.7015   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.7705   
## Prevalence : 0.5839   
## Detection Rate : 0.4969   
## Detection Prevalence : 0.6211   
## Balanced Accuracy : 0.7763   
##   
## 'Positive' Class : 0   
##

# 정확도 약79%

#(2)caret 패키지  
 ctrl <- trainControl(method="cv", 3)  
 titanic\_RF\_caret <- train(survived ~ age+sex+embarked,  
 data = titanic\_caret\_train,  
 na.action = na.omit,  
 method = 'cforest',  
 trControl = ctrl)  
 titan\_pred\_caret<- predict(titanic\_RF\_caret,  
 newdata = titanic\_caret\_test)  
   
 titan\_caret\_test <- subset(titanic\_caret\_test,!is.na(titanic\_caret\_test$age))  
 confusionMatrix(titan\_pred\_caret, factor(titan\_caret\_test$survived))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 156 42  
## 1 28 82  
##   
## Accuracy : 0.7727   
## 95% CI : (0.7218, 0.8183)  
## No Information Rate : 0.5974   
## P-Value [Acc > NIR] : 5.767e-11   
##   
## Kappa : 0.5187   
##   
## Mcnemar's Test P-Value : 0.1202   
##   
## Sensitivity : 0.8478   
## Specificity : 0.6613   
## Pos Pred Value : 0.7879   
## Neg Pred Value : 0.7455   
## Prevalence : 0.5974   
## Detection Rate : 0.5065   
## Detection Prevalence : 0.6429   
## Balanced Accuracy : 0.7546   
##   
## 'Positive' Class : 0   
##

# 정확도 약77%

결과)

정확도는 약 78%가 나왔다.

노드 불순도 개선에 가장 많이 기여한 변수는 Sex > age > 탑승위치 순이였다.

doBy에서는 아래와 같은 옵션을 사용하였다.

ntree=100 : 나무의 가지 개수 100개

proximity=T : 개체들 간의 근접도 행렬을 제공: 동일한 최종노드에 포함되는 빈도에 기초한다.

## Women DATA SET

# Women 데이터 설명

# 불러온 Women 데이터는 15명의 임의 정보와 2개의 변수로 구성 되어있다.

# 2개의 변수 구성

# 연속형 변수 : height, weight

# 각 변수에 대한 설명

# height : 미국 여성의 몸무게(lb)

# weight : 미국 여성의 키(in)

(표5)

|  |  |  |
| --- | --- | --- |
| index | height | weight |
| 1 | 58 | 115 |
| 2 | 59 | 117 |
| 3 | 60 | 120 |
| 4 | 61 | 123 |
| 5 | 62 | 126 |
| …(생략) | …(생략) | …(생략) |

**# 종속변수와 설명변수 모두 수치형의 자료이다.**

**# 통계기반의 단순선형회귀를 활용할 수 있다.**

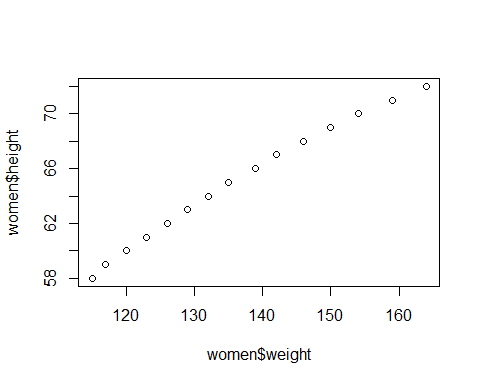
### 단순선형회귀

# stats 패키지  
 #(1)모델 생성  
 lm\_women <- lm(weight ~ height, data = women\_doBy\_train)  
 summary(lm\_women)

##   
## Call:  
## lm(formula = weight ~ height, data = women\_doBy\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.7333 -1.1333 -0.3833 0.7417 3.1167   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -87.51667 5.93694 -14.74 1.71e-09 \*\*\*  
## height 3.45000 0.09114 37.85 1.09e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.525 on 13 degrees of freedom  
## Multiple R-squared: 0.991, Adjusted R-squared: 0.9903   
## F-statistic: 1433 on 1 and 13 DF, p-value: 1.091e-14

# 키가 몸무게에 미치는 영향  
 # 위 결과 키가 몸무게에 대한 설명을 Adjusted R-squared: 0.9903 = 99%만큼 할 수 있다.   
 # p값 : 1.091e-14 / 0.05(유의수준) 작으므로 키는 몸무게에 영향을 미친다.

# 선형회귀 시각화  
 plot(women$weight, women$height)



# 잔차 정규성 검정  
 shapiro.test(lm\_women$residuals)

##   
## Shapiro-Wilk normality test  
##   
## data: lm\_women$residuals  
## W = 0.91909, p-value = 0.1866

# Caret Package  
 #(1)모델 생성  
 basic\_model\_lm <-train(weight ~.,  
 data=women\_caret\_train,  
 method="lm")  
 summary(basic\_model\_lm) # 모델의 전체 요약값

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.4621 -0.9546 -0.2046 0.5987 3.1437   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -90.056 6.734 -13.37 1.05e-07 \*\*\*  
## height 3.485 0.103 33.83 1.20e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.421 on 10 degrees of freedom  
## Multiple R-squared: 0.9913, Adjusted R-squared: 0.9905   
## F-statistic: 1145 on 1 and 10 DF, p-value: 1.204e-11

regressControl <- trainControl(method="repeatedcv", number = 4, repeats = 5)   
   
 regress <- train(weight ~ height,  
 data = women\_caret\_train,  
 method = "lm",  
 trControl = regressControl,   
 tuneGrid = expand.grid(intercept = FALSE))  
   
 summary(regress)

##   
## Call:  
## lm(formula = .outcome ~ 0 + ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.481 -5.033 -1.195 3.003 12.091   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## height 2.110 0.026 81.16 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.887 on 11 degrees of freedom  
## Multiple R-squared: 0.9983, Adjusted R-squared: 0.9982   
## F-statistic: 6586 on 1 and 11 DF, p-value: < 2.2e-16

결과)

caret 패키지에서는 tuneGrid의 intercept = F 옵션을 통해 절편을 x값에 포함하여 계산할 수 있다.

## economics Data Set

사용 데이터(economics)

# economics 데이터 설명

# 불러온 economics 데이터는 574개의 미국 연도별 경제의 관측치에 관한 6개의 변수로 구성 되어있다.

# 6개의 변수 구성

# 범주형 변수 : 없음

# 연속형 변수 : date, pce, pop, psavert, uempmed, unemploy

# 각 변수에 대한 설명

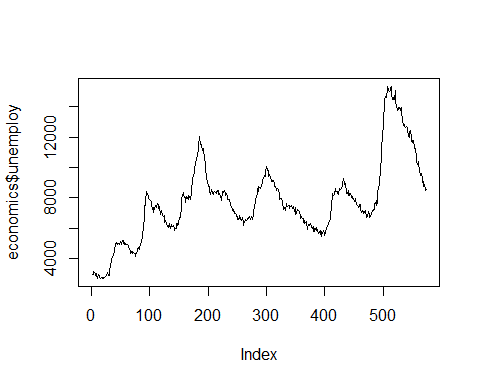
|  |  |
| --- | --- |
| Date | 데이터 수집 월 |
| pce | 개인 소비 지출 |
| Pop | 인구수 |
| Psavert | 개인 저축률 |
| uempmed | 실업 기간 |
| unemploy | 실업자 수(단위: 1,000명) |

(표6)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| index | date | pce | pop | psavert | uempmed | unemploy |
| 1 | 1967-07-01 | 506.7 | 198712 | 12.6 | 4.5 | 2944 |
| 2 | 1967-08-01 | 509.8 | 198911 | 12.6 | 4.7 | 2945 |
| 3 | 1967-09-01 | 515.6 | 199113 | 11.9 | 4.6 | 2958 |
| 4 | 1967-10-01 | 512.2 | 199311 | 12.9 | 4.9 | 3143 |
| 5 | 1967-11-01 | 517.4 | 199498 | 12.8 | 4.7 | 3066 |
| 6 | 1967-12-01 | 525.1 | 199657 | 11.8 | 4.8 | 3018 |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) |

* 1. 시계열

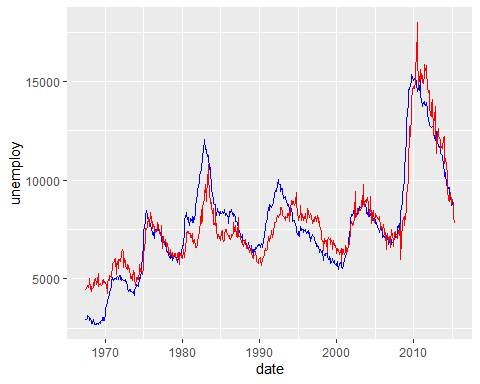
# caret 패키지  
plot(economics$unemploy,type = "l")



myTimeControl <- trainControl(method = "timeslice",  
 initialWindow = 36,  
 horizon = 12,  
 fixedWindow = TRUE)  
  
plsFitTime <- train(unemploy ~ .,  
 data = economics,  
 method = "pls",  
 preProc = c("center", "scale"),  
 trControl = myTimeControl)  
plsFitTime

## Partial Least Squares   
##   
## 574 samples  
## 5 predictor  
##   
## Pre-processing: centered (5), scaled (5)   
## Resampling: Rolling Forecasting Origin Resampling (12 held-out with a fixed window)   
## Summary of sample sizes: 36, 36, 36, 36, 36, 36, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 1091.9049 0.4250687 1008.481  
## 2 919.0868 0.3845386 823.527  
## 3 861.3382 0.3847531 769.253  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 3.

# 3단계 : 예측  
pred <- predict(plsFitTime, economics)  
  
asdf <- cbind(pred, economics[,c(1,6)])  
ggplot(asdf, aes(x = date, y = unemploy)) +  
 geom\_line(color = 'blue') + geom\_line(aes(x = date, y = pred), color = 'red')



## Pima.te DATA SET

사용 데이터(Pima.te)

# Pima.te 데이터 설명

# 불러온 Pima.te 데이터는 332명의 인디언 여성의 당뇨병 관측치에 관한 8개의 변수로 구성 되어있다.

# 8개의 변수 구성

# 범주형 변수 : type

# 연속형 변수 : npreg, glu, bp, skin, bmi, ped, age

# 각 변수에 대한 설명

|  |  |
| --- | --- |
| Npreg | 임신 수 |
| Glu | 혈장 포도당 농도 |
| Bp | 이완기 혈압 |
| Skin | 삼두근 피부 주름 두께 |
| Bmi | 체질량 지수(체중(kg)/(높이(m))\(^2\)) |
| Ped | 당뇨별 혈통 기능 |
| Age | 나이 |
| type | 당뇨병 여부(yes or no) |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | npreg | glu | bp | skin | bmi | ped | age | type |
| 1 | 6 | 148 | 72 | 35 | 33.6 | 0.627 | 50 | Yes |
| 2 | 1 | 85 | 66 | 29 | 26.6 | 0.351 | 31 | No |
| 3 | 1 | 89 | 66 | 23 | 28.1 | 0.167 | 21 | No |
| 4 | 3 | 78 | 50 | 32 | 31 | 0.248 | 26 | Yes |
| 5 | 2 | 197 | 70 | 45 | 30.5 | 0.158 | 53 | Yes |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) |

* 1. Roc커브

# randomforest 패키지  
#(1)모델 생성  
Diag\_DF <- data.frame(Attribute=c(colnames(Pima.te)[1:7]), AUC=NA) # AUC 계산을 위한 데이터 프레임을 생성합니다.   
  
for(i in 1:nrow(Diag\_DF)){  
 roc\_result <- roc(Pima.te$type, Pima.te[,as.character(Diag\_DF$Attribute[i])]) # 확진 결과에 대한 데이터(type)와 진단 방법에 대한 후보 변수를 입력하여 AUC를 계산합니다.   
 Diag\_DF[i,'AUC'] <- roc\_result$auc} # AUC 값을 입력합니다.

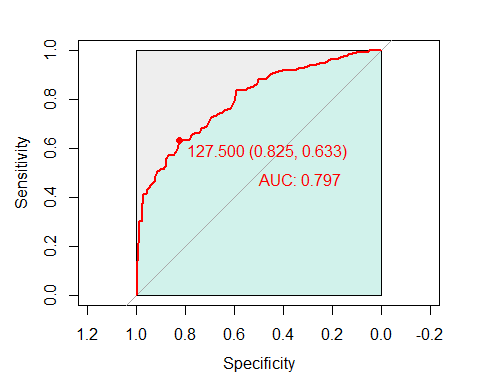
Diag\_DF <- Diag\_DF[order(-Diag\_DF$AUC),] # AUC 값을 오름차순 정렬합니다.  
Diag\_DF # 결과를 확인해보면 "glu" 변수가 가장 좋은 성능임을 확인 할 수 있습니다.

## Attribute AUC  
## 2 glu 0.7970543  
## 7 age 0.7210886  
## 5 bmi 0.6839799  
## 4 skin 0.6656313  
## 6 ped 0.6563541  
## 1 npreg 0.6201094  
## 3 bp 0.6097626

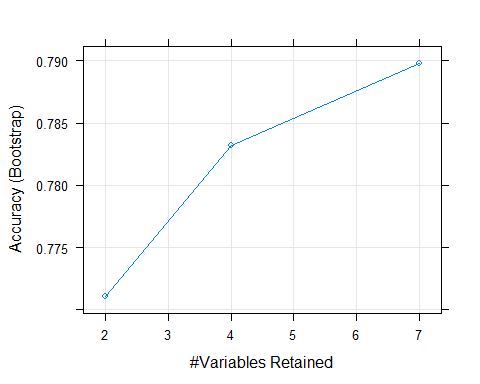
# AUC가 가장 높은 "glu" 변수를 사용하여 ROC curve를 그리기  
glu\_roc <- roc(Pima.te$type, Pima.te$glu) # "glu" 변수에 대한 ROC를 계산하여 value로 저장합니다.

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

plot.roc(glu\_roc, # roc를 계산한 value를 입력합니다.  
 col="red", # 선의 색상을 설정합니다.  
 print.auc=TRUE, # auc 값을 출력하도록 설정합니다.  
 max.auc.polygon=TRUE, # auc의 최대 면적을 출력하도록 설정합니다.  
 print.thres=TRUE, print.thres.pch=19, print.thres.col = "red", # 기준치(cut-off value)에 대한 출력, 포인트, 색상을 설정합니다.  
 auc.polygon=TRUE, auc.polygon.col="#D1F2EB") # 선 아래 면적에 대한 출력, 색상을 설정합니다.



# caret 패키지  
#(1)모델 생성  
a <- train(type ~ .,  
 data = Pima.te,  
 method='rocc')  
plot(a)



summary(a)

## Length Class Mode   
## AUCs 3 data.frame list   
## genes 7 -none- character  
## positiv 0 -none- character  
## negativ 7 -none- character  
## metagene.expression 332 -none- numeric   
## metagene.expression.ranked 331 -none- numeric   
## cutoffvalue 1 -none- numeric   
## method 1 -none- character  
## xNames 7 -none- character  
## problemType 1 -none- character  
## tuneValue 1 data.frame list   
## obsLevels 2 -none- character  
## param 0 -none- list

결과)

pROC 패키지로 ROC Curve를 그리면 데이터의 특이도와 민감도를 구해준다.

## Iris DATA SET

# Iris 데이터 설명

# 불러온 Iris 데이터는 150개의 붓꽃데이터의 관측치에 관한 5개의 변수로 구성 되어있다.

# 14개의 변수 구성

# 범주형 변수 species

# 연속형 변수 : speal.length, sepal,width, petal.length, petal.width

# 각 변수에 대한 설명

Species : iris의 종류. setosa, versicolor, virginica로 이루어져 있음

Sepal.Width : 꽃받침 너비

Sepal.Length : 꽃받침 길이

Petal.Width : 꽃잎 너비

Petal.Length : 꽃잎 길이

(표4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| index | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| …(생략) | …(생략) | …(생략) | …(생략) | …(생략) | …(생략) |

**# 종속변수가 범주형이고 설명변수가 수치형의 자료이므로,**

**# ML기반의 앙상블(배깅, 부스팅), xgboost 기법을 활용했다.**

* 1. Xgboost

#### 1)xgboost ####  
 #xgBoost 패키지  
 # 학습모델 생성  
 iris\_doBy\_dtrain <- xgb.DMatrix(data = iris\_doBy\_train\_mat,  
 label = iris\_doBy\_train\_lab)  
   
 iris\_doBy\_xgb\_model <- xgboost(data = iris\_doBy\_dtrain, max\_depth = 2, eta = 1,  
 nthread = 2, nrounds = 2,  
 objective = "multi:softmax",  
 num\_class = 3,  
 verbose = 0)

## [01:15:45] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.

# 모델 평가  
 iris\_doBy\_test\_mat <- as.matrix(iris\_doBy\_test[-c(5:6)])  
 iris\_doBy\_test\_lab <- iris\_doBy\_test$label  
   
 doBy\_pred\_iris <- predict(iris\_doBy\_xgb\_model, iris\_doBy\_test\_mat)  
   
 table(doBy\_pred\_iris, iris\_doBy\_test\_lab)

## iris\_doBy\_test\_lab  
## doBy\_pred\_iris 0 1 2  
## 0 15 0 0  
## 1 0 15 0  
## 2 0 0 15

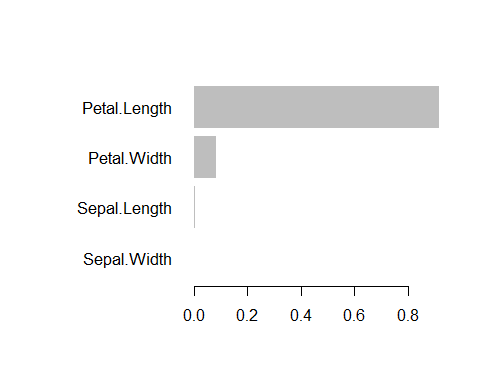
(15+15+15) / length(iris\_doBy\_test\_lab)

## [1] 1

# 정확도 100%  
   
 # 주요변수 확인  
 importance\_matrix <- xgb.importance(colnames(iris\_doBy\_train\_mat),  
 model = iris\_doBy\_xgb\_model)  
 importance\_matrix

## Feature Gain Cover Frequency  
## 1: Petal.Length 0.9168339798 0.66123244 0.53846154  
## 2: Petal.Width 0.0806030105 0.28623924 0.30769231  
## 3: Sepal.Length 0.0020540392 0.01990803 0.07692308  
## 4: Sepal.Width 0.0005089704 0.03262029 0.07692308

xgb.plot.importance(importance\_matrix)



#caret 패키지  
 # 데이터 분할  
 set.seed(123)  
 idx = createDataPartition(data3$Species, list=F, p=0.7)  
 Train = data3[ idx,]  
 Test = data3[-idx,]  
   
 train.data = as.matrix(Train[, names(data3)!="Species"])  
 test.data = as.matrix(Test[ , names(data3)!="Species"])  
 train.label = as.integer(Train$Species) - 1 # 0기반  
 test.label = as.integer(Test$Species) - 1 # 0기반  
   
 # 모델 생성  
 dtrain = xgb.DMatrix(data=train.data, label=train.label)  
 dtest = xgb.DMatrix(data=test.data , label=test.label )  
 watchlist = list(train=dtrain, eval=dtest)  
 param = list(max\_depth=2, eta=1, verbose=0, nthread=2,  
 objective="multi:softprob", eval\_metric="mlogloss", num\_class=3)  
 model = xgb.train(param, dtrain, nrounds=2, watchlist)

## [01:15:45] WARNING: amalgamation/../src/learner.cc:573:   
## Parameters: { "verbose" } might not be used.  
##   
## This may not be accurate due to some parameters are only used in language bindings but  
## passed down to XGBoost core. Or some parameters are not used but slip through this  
## verification. Please open an issue if you find above cases.  
##   
##   
## [1] train-mlogloss:0.213489 eval-mlogloss:0.213489   
## [2] train-mlogloss:0.078791 eval-mlogloss:0.078791

# 테스트  
 pred = as.data.frame(predict(model,test.data,reshape=T))  
 names = levels(data3$Species)  
 colnames(pred) = names  
 pred$prediction = apply(pred,1,function(x) names[which.max(x)])  
 pred$class = Test$Species  
 table(pred$prediction, pred$class)

##   
## setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 15 0  
## virginica 0 0 15

#정분류율  
 sum(pred$prediction==pred$class)/nrow(pred)

## [1] 1

#100%

결과)

xgboost를 통해 주요 변수를 확인한 결과 꽃의 종류를 구분할 때, 꽃잎의 길이가 가장 중요한 변수로 파악됐다.

옵션 설명

caret 패키지의 prams 값 중에서,

verbose 상세한 로깅 logging 을 출력할지 말지를 조정하는 parameter이다.

* 예를 들어, 아래 4번 cell 에서 verbose 값이 2일 때는 “WARNING: ~~ “하고 문구가 있는데, 밑에 5번 cell에서는 verbose 값이 0이기 때문에 코드 실행 시 “WARNING” 이라고 되어있던 부분이 출력되지 않은 것을 확인할 수 있다.
* verbose 값이 0보다 크면 로그를 출력하게 되기 때문에 실행 속도에 영향을 줄 수 있다.
  1. 배깅

# adabag 패키지  
 #(1)Bagging model 생성  
 iris.bagging <- bagging(Species~., data=iris\_doBy\_train[1:5], mfinal=10)  
 iris.bagging$importance

## Petal.Length Petal.Width Sepal.Length Sepal.Width   
## 85.72557 14.27443 0.00000 0.00000

#(2)예측값  
 baggingpred <- predict(iris.bagging, newdata=iris)  
   
 #(3)정오분류표  
 baggingtb <- table(baggingpred$class, iris[,5])  
 sum(baggingtb[row(baggingtb) == col(baggingtb)])/sum(baggingtb) # 정분류율

## [1] 0.9666667

1-sum(baggingtb[row(baggingtb) == col(baggingtb)])/sum(baggingtb) # 오분류율

## [1] 0.03333333

# Caret Package  
 #(1)Caret Package 배깅 모델 생성  
 ctrl <- trainControl(method = 'cv',  
 number = 10)  
 gyu <- train(Species ~ . ,  
 data = iris\_caret\_train,  
 method = 'treebag',  
 trControl = ctrl)  
 gyu

## Bagged CART   
##   
## 105 samples  
## 4 predictor  
## 3 classes: 'setosa', 'versicolor', 'virginica'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 95, 94, 94, 95, 95, 95, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.9423737 0.9132564

#(2)예측 분류 결과 생성  
 gyu\_pred <- predict(gyu, newdata = iris\_caret\_test)  
   
 #(3)적용 분류 결과 도출  
 table(gyu\_pred, iris\_caret\_test$Species)

##   
## gyu\_pred setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 15 1  
## virginica 0 0 14

#(4)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(gyu\_pred, iris\_caret\_test$Species)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 15 1  
## virginica 0 0 14  
##   
## Overall Statistics  
##   
## Accuracy : 0.9778   
## 95% CI : (0.8823, 0.9994)  
## No Information Rate : 0.3333   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9667   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: setosa Class: versicolor Class: virginica  
## Sensitivity 1.0000 1.0000 0.9333  
## Specificity 1.0000 0.9667 1.0000  
## Pos Pred Value 1.0000 0.9375 1.0000  
## Neg Pred Value 1.0000 1.0000 0.9677  
## Prevalence 0.3333 0.3333 0.3333  
## Detection Rate 0.3333 0.3333 0.3111  
## Detection Prevalence 0.3333 0.3556 0.3111  
## Balanced Accuracy 1.0000 0.9833 0.9667

# 98% 신뢰수준을 확인 할 수 있다.

* 1. 부스팅

# adabag 패키지  
 #(1)boosting model 생성  
 boo.adabag <- boosting(Species~., data = iris\_doBy\_train,  
 boos = TRUE,  
 mfinal = 10)  
 boo.adabag$importance

## label Petal.Length Petal.Width Sepal.Length Sepal.Width   
## 49.52823 50.47177 0.00000 0.00000 0.00000

#(2)예측값  
 pred <- predict(boo.adabag, newdata = iris\_doBy\_test)  
   
 #(3)정오분류표  
 tb <- table(pred$class, iris\_doBy\_test[,5])  
 sum(tb[row(tb) == col(tb)])/sum(tb) # 정분류율

## [1] 1

1-sum(tb[row(tb) == col(tb)])/sum(tb) # 오분류율

## [1] 0

# Caret Package  
 #(1)Caret Package 부스팅 학습 모델 설정  
 ctrl <- trainControl(method = 'cv', number = 3) ## method : 샘플링을 하는 방법을 결정  
 m1 <- train(Species ~ . , data = iris\_caret\_train,  
 method = 'AdaBoost.M1',  
 trControl = ctrl)  
   
 #(2)예측 분류 결과 생성  
 m1\_pred <- predict(m1, newdata = iris\_caret\_test)  
   
 #(3)적용 분류 결과 도출  
 table(m1\_pred, iris\_caret\_test$Species)

##   
## m1\_pred setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 15 0  
## virginica 0 0 15

#(4)모델 성능 평가 지표(정확도 확인)  
 confusionMatrix(m1\_pred, iris\_caret\_test$Species)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 15 0  
## virginica 0 0 15  
##   
## Overall Statistics  
##   
## Accuracy : 1   
## 95% CI : (0.9213, 1)  
## No Information Rate : 0.3333   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: setosa Class: versicolor Class: virginica  
## Sensitivity 1.0000 1.0000 1.0000  
## Specificity 1.0000 1.0000 1.0000  
## Pos Pred Value 1.0000 1.0000 1.0000  
## Neg Pred Value 1.0000 1.0000 1.0000  
## Prevalence 0.3333 0.3333 0.3333  
## Detection Rate 0.3333 0.3333 0.3333  
## Detection Prevalence 0.3333 0.3333 0.3333  
## Balanced Accuracy 1.0000 1.0000 1.0000

# 100% 정확도를 확인할 수 있다.

# 참고자료

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=parksehoon1971&logNo=221003622145>

<https://m.blog.naver.com/nife0719/220993392408>

<https://jeonzzang.tistory.com/85>

<https://weeklyclip.com/entry/%ED%85%8D%EC%8A%A4%ED%8A%B8%EB%A7%88%EC%9D%B4%EB%8B%9D-%EC%98%A4%ED%94%BC%EB%8B%88%EC%96%B8%EB%A7%88%EC%9D%B4%EB%8B%9D-%EC%9B%B9%EB%A7%88%EC%9D%B4%EB%8B%9D>

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=tjdudwo93&logNo=221041168345>

<https://zetawiki.com/wiki/R_XGBoost>

<https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/>

<https://kjy042386.tistory.com/105>

<https://topepo.github.io/caret/index.html>

<https://dschloe.github.io/r/machine_learning/3_6_xgboost/>

https://xgboost.readthedocs.io/en/stable/parameter.html