데이터마이닝 프로젝트

AI소프트웨어과 - 윤주미

데이터마이닝

- 정의 : 대용량 데이터에서 의미있는 패턴을 파악하거나 예측하여 의사결정에 활용하는 방법
- 통계분석과 차이점 : 가설이나 가정에 따른 분석, 검증을 하는 통계분석과 달리 데이터마이닝은 다양한 수리 알고리즘을 이용해 **데이터베이스의 데이터로부터 의미있는 정보 추출**

데이터마이닝의 활용 분야

1. 데이터베이스 마케팅 (Database Marketing)
    - 데이터를 분석하고 획득한 정보를 이용하여 마케팅 전략 구축
      (예: 고객 세분화, 이탈고객 분석, 상품 추천 등)
2. 신용평가 (Credit Scoring)
    - 특정인의 신용상태를 점수화하는 과정
    - 신용거래 대출한도를 결정하는 것이 주요 목표
    - 이를 통하여 불량채권과 대손을 추정하여 최소화함
      (예: 신용카드, 소비자/상업 대출, 주택할부금융)
3. 생물정보학 (Bioinformatics)
    - 게놈(Genome) 프로젝트로부터 얻은 방대한 양의 유전자 정보로부터 가치 있는 정보의 추출
      (응용분야: 신약개발, 조기진단, 유전자 치료)
4. 텍스트 마이닝 (Text Mining)
    - 디지털화된 자료 (예: 전자우편, 신문기사 등)로 부터 유용한 정보를 획득
      (응용분야: 자동응답시스템, 소셜미디어 분석, 상품평 분석 등)
5. 부정행위 적발 (Fraud Detection)
    - 고도의 사기행위를 발견할 수 있는 패턴을 자료로부터 획득
      (응용분야: 신용카드 거래사기 탐지, 부정수표 적발, 부당/과다 보험료 청구 탐지)

데이터마이닝 적용 사례

사례1 - 고객 세분화 및 마케팅 전략 수립

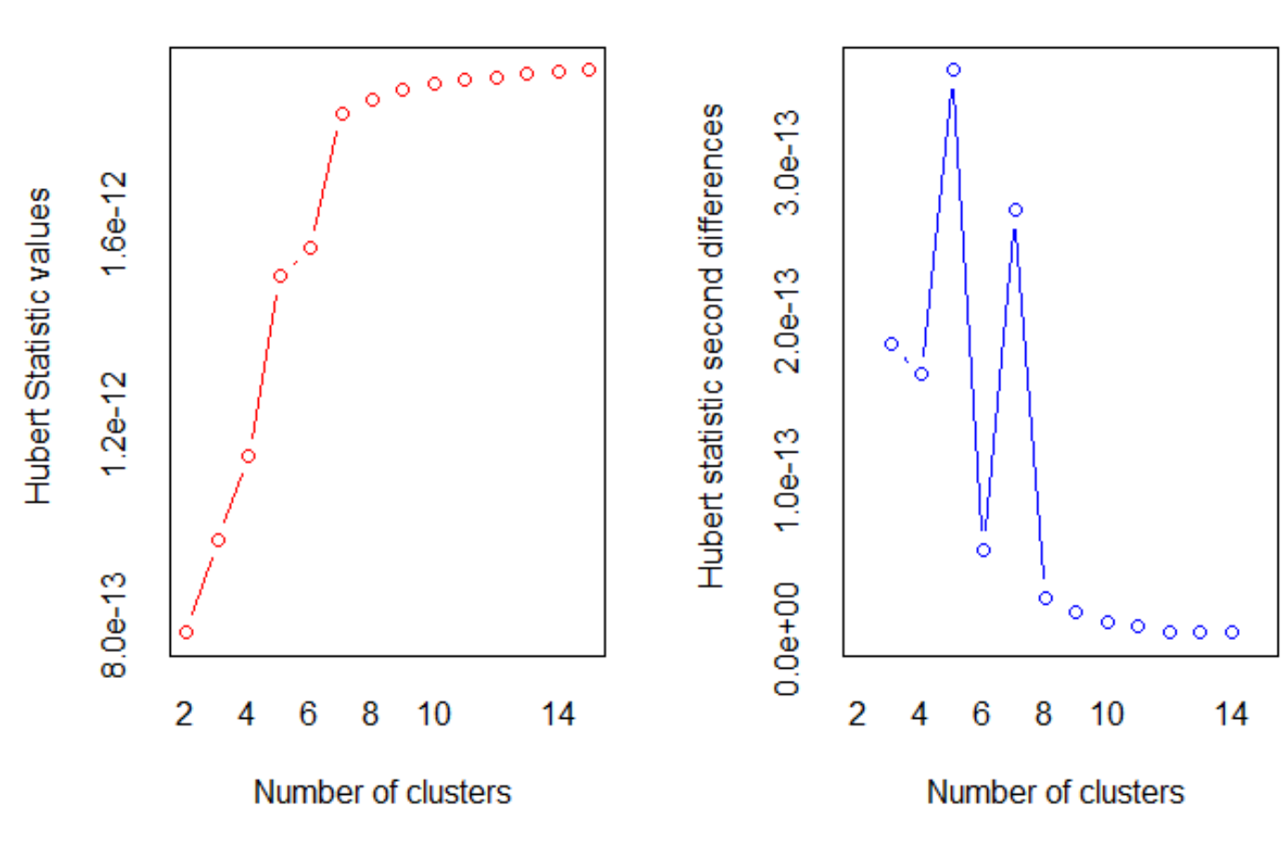
상황 설명:
'언니의 파우치'라는 국내 뷰티 앱을 운영하는 기업이 다양한 고객 데이터를 활용하여 K-means 군집분석을 진행하였다.
분석 결과를 통해 고객을 5개 그룹으로 세분화하였고, 각 그룹의 특성에 따라 구매를 결정하는 주요 요인을 파악하였다.
이러한 분석을 기반으로 마케팅 전략을 수립하였는데, 10대 그룹은 앱 내의 활동 활성화에 초점을 맞추었고, 30대 이상 그룹에게는 차별화된 이벤트를 제공하는 방향으로 전략을 구성하였다.
더 나아가, 로지스틱 회귀분석과 신경망 모델 등의 데이터 마이닝 방법을 활용하여 개인화된 추천 시스템을 구현하였고, 기업은 수익성을 향상시켰다.

과제 목표:
고객 데이터를 활용하여 고객 세분화를 진행한다. K-means 군집분석을 이용하여 해당 과제를 진행하였다.

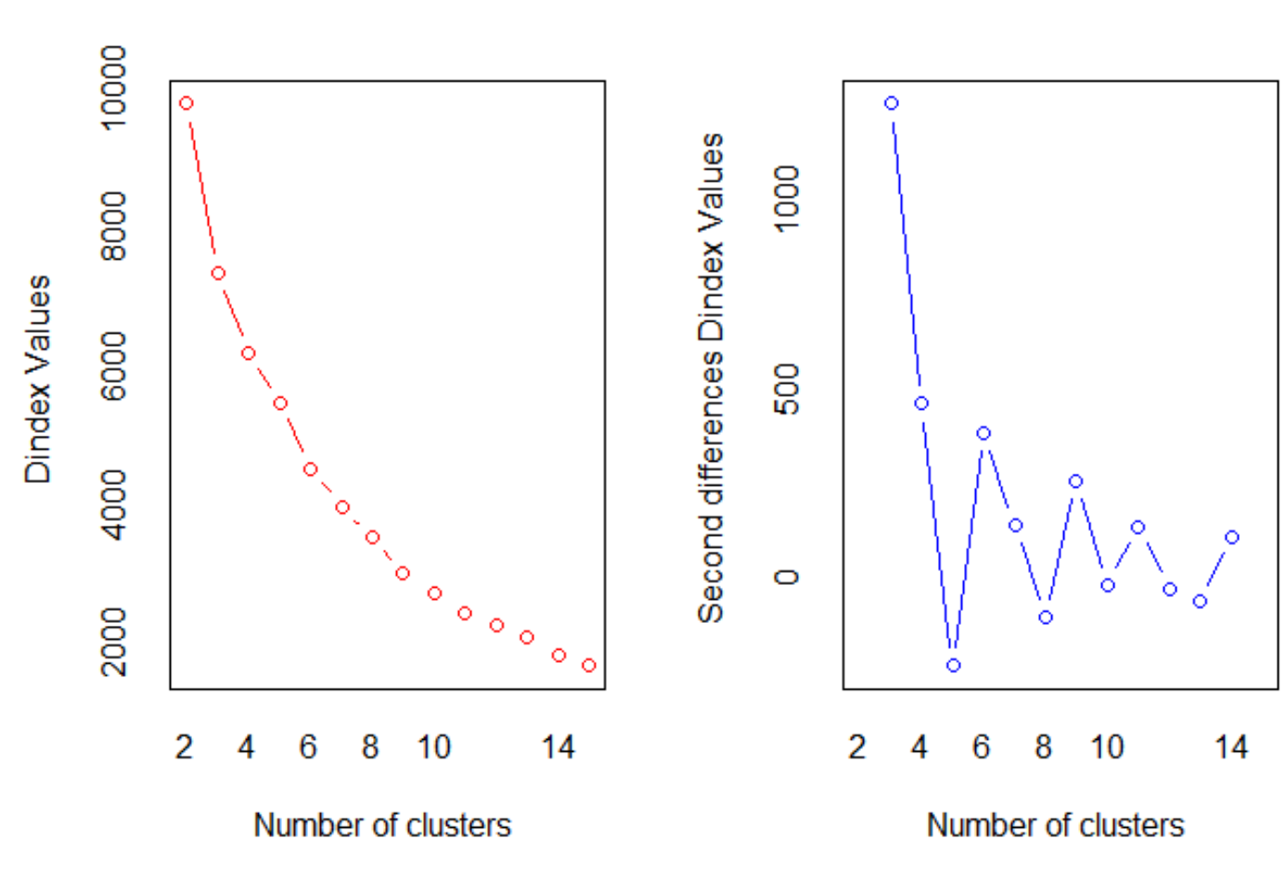
```
# R 스크립트:
# 필요한 라이브러리 불러오기
library(cluster)
library(caret)
```

```
library(NbClust)
```

```
# 고객 데이터 불러오기
data_1 <- read.csv("C:/Users/wnaely/Documents/marketing_campaign.csv")

# 데이터 전처리
# 결측치가 있는 행을 제거
data_1 <- na.omit(data_1)

# 학습, 테스트 데이터로 분할 (train 70%, test 30%)
set.seed(2023)    # 난수를 동일하게 추출되도록 고정시키는 함수
idx1 <- sample(1:nrow(data_1),nrow(data_1)*0.7,replace=FALSE)
train1 <- data_1[idx1,]
test1 <- data_1[-idx1,]

# NbClust 함수로 최적의 군집 수 찾기 (최적의 k = 7)
nc <- NbClust(train1, min.nc = 2, max.nc = 15, method = "kmeans")
```



```
## *** : The Hubert index is a graphical method of determining the number of clu
##                 In the plot of Hubert index, we seek a significant knee that 
##                 significant increase of the value of the measure i.e the sign
##                 index second differences plot.
## 
```



```
## *** : The D index is a graphical method of determining the number of clusters
##                 In the plot of D index, we seek a significant knee (the signi
##                 second differences plot) that corresponds to a significant in
##                 the measure.
## 
## *******************************************************************
## * Among all indices:                                                
## * 5 proposed 2 as the best number of clusters 
## * 5 proposed 3 as the best number of clusters 
## * 2 proposed 6 as the best number of clusters 
## * 7 proposed 7 as the best number of clusters 
## * 2 proposed 13 as the best number of clusters 
## * 2 proposed 15 as the best number of clusters 
## 
##                    ***** Conclusion *****                            
##  
## * According to the majority rule, the best number of clusters is  7 
##  
##  
## *******************************************************************
```

```
# kmeans 함수를 활용하여 kmeans 군집분석 실시
result <- kmeans(train1, centers = 7) # 최적의 k를 사용하여 k-means 수행
result
```

```
## K-means clustering with 7 clusters of sizes 324, 337, 374, 282, 226, 7, 1
## 
## Cluster means:
##   Year_Birth    Income    Kidhome  Teenhome  MntWines MntFruits MntMeatProduc
## 1   1965.738  51100.41 0.43827160 0.8641975 272.96914 13.237654        80.271
## 2   1965.866  65978.93 0.13353116 0.6528190 507.57567 43.771513       241.373
## 3   1971.572  36061.73 0.81016043 0.4679144  57.46791  5.890374        30.954
## 4   1967.433  81314.52 0.07801418 0.2624113 674.59574 66.312057       467.921
## 5   1975.204  20543.01 0.74778761 0.1592920  13.19027  6.238938        17.424
## 6   1971.857 158024.29 0.28571429 0.2857143  29.00000  3.142857       708.142
## 7   1977.000 666666.00 1.00000000 0.0000000   9.00000 14.000000        18.000
##   MntFishProducts MntSweetProducts MntGoldProds NumDealsPurchases
## 1       20.160494        13.981481    41.280864          3.185185
## 2       56.715134        42.629080    67.287834          2.382789
## 3       10.096257         5.967914    21.074866          2.243316
## 4       96.432624        67.382979    69.198582          1.283688
## 5        8.716814         6.716814    18.486726          2.163717
## 6        3.714286         1.285714     2.571429          4.285714
## 7        8.000000         1.000000    12.000000          4.000000
##   NumWebPurchases NumCatalogPurchases NumStorePurchases
## 1       4.5895062           2.1512346         5.7962963
## 2       5.7596439           4.0356083         8.4807122
## 3       2.4598930           0.6604278         3.3957219
## 4       5.2269504           5.9042553         8.4609929
## 5       1.9601770           0.4115044         2.8584071
## 6       0.1428571          11.1428571         0.4285714
## 7       3.0000000           1.0000000         3.0000000
## 
## Clustering vector:
## 1926 1505 1402 2009  879 1582 1934 1170 2143  972  495 1428  218   48 1813 16
##    5    1    2    2    4    4    3    4    3    3    1    3    4    3    4 
 ...
## 1080  401  789   75 1748   16  140 1029  812 1344  932 1614 2189  519   70 
##    3    3    1    3    2    4    3    3    5    5    1    3    2    2    4 
## 
## Within cluster sum of squares by cluster:
## [1] 6165478886 6072680687 6677453928 8444516133 8126642205   50939736
##  (between_SS / total_SS =  96.8 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss
## [6] "betweenss"    "size"         "iter"         "ifault"      
```

```
# 성과분석
print(paste("Total within-cluster sum of square:", result$tot.withinss))
```

```
## [1] "Total within-cluster sum of square: 35537711575.8206"
```

군집의 수를 7개로 분할하여 kmeans 군집분석을 실시한 결과, 324, 337, 374, 282, 226, 7, 1의 개체가 모인 군집으로 나누어졌다. 그리고 전체 변동에서 군집 간 변동이 차지하는 비율인 (between_SS/total_SS)이 1에 가까울수록 군집이 잘 분류되었다고 판단할 수 있으므로, 96.8 %로 좋은 모델이라고 할 수 있다.

사례 2 - 제조업에서의 생산 불량 예측

상황 설명:
반도체 회사에서는 제조 과정에서 발생하는 불량품을 자동으로 검색하는 장치 개발을 위해 데이터 마이닝 기법을 도입하였다.
이 과정에서 연관성 분석과 군집 분석 알고리즘을 활용하여, 정상 제품의 특성을 기준으로 몇 가지 군집으로 나누었다.
이후, 새로 생산되는 제품이 정상 제품 군집의 범위를 벗어나는 경우, 해당 제품을 불량품으로 분류하였다.
이렇게 진행된 분석은 불량품 패턴의 발견에도 도움이 되었으며, 결과적으로 불량품을 감소시켜 회사의 이익을 증가시켰다고 한다.

과제 목표:
해당 사례를 참고하여, 제조 데이터를 활용해 '불량/정상' 값을 예측하는 작업을 진행한다. 이를 위해 RandomForest 분석을 실시하였다.

```
# R 스크립트
# 제조데이터 불러오기
data_2 <- read.csv("C:/Users/wnaely/Documents/semiconductor_data.csv")
class(data_2$Passorfail)
```

```
## [1] "integer"
```

```
# 데이터 전처리
# 결측치가 있는 행을 제거
data_2 <- na.omit(data_2)

# 'Passorfail' 종속변수를 factor 형태로 변환
data_2$Passorfail <- factor(data_2$Passorfail)

# 학습, 테스트 데이터로 분할 (train 70%, test 30%)
set.seed(2023)    # 난수를 동일하게 추출되도록 고정시키는 함수
idx2 <- sample(1:nrow(data_2),nrow(data_2)*0.7,replace=FALSE)
train2 <- data_2[idx2,]
test2 <- data_2[-idx2,]

# 필요한 라이브러리 불러오기
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
library(caret)
library(ROCR)
```

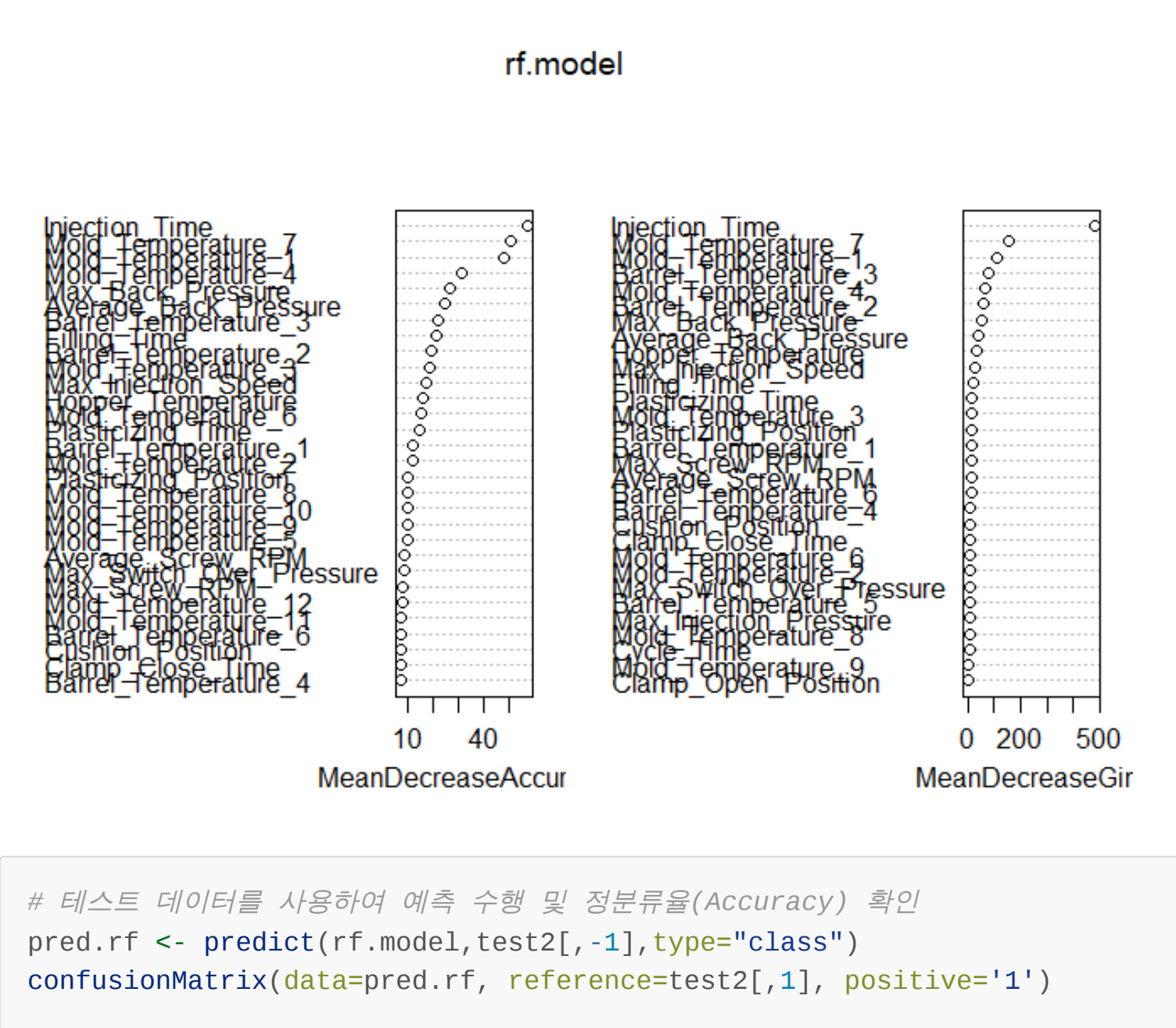
```
# randomForest 함수를 사용하여 RandomForest분석 실시
rf.model <- randomForest(Passorfail ~ .,
                       data = train2,
                       ntree = 500,          # 트리의 개수를 500개로 설정
                       mtry = sqrt(34),      # 사용할 변수의 개수(mtry 값을 변수의
                       importance=T)         # 변수중요도 결과를 확인
rf.model
```

```
## 
## Call:
##  randomForest(formula = Passorfail ~ ., data = train2, ntree = 500,      mtry
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
## 
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##     0    1  class.error
## 0 708   14 0.0193905817
## 1   1 7172 0.0001394117
```

```
names(rf.model)
```

```
##  [1] "call"            "type"            "predicted"       "err.rate"       
##  [5] "confusion"       "votes"           "oob.times"       "classes"        
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"      
## [13] "ntree"           "mtry"            "forest"          "y"              
## [17] "test"            "inbag"           "terms"          
```

```
varImpPlot(rf.model)                              # 변수 중요도를 그래프로 표시합니다.
```

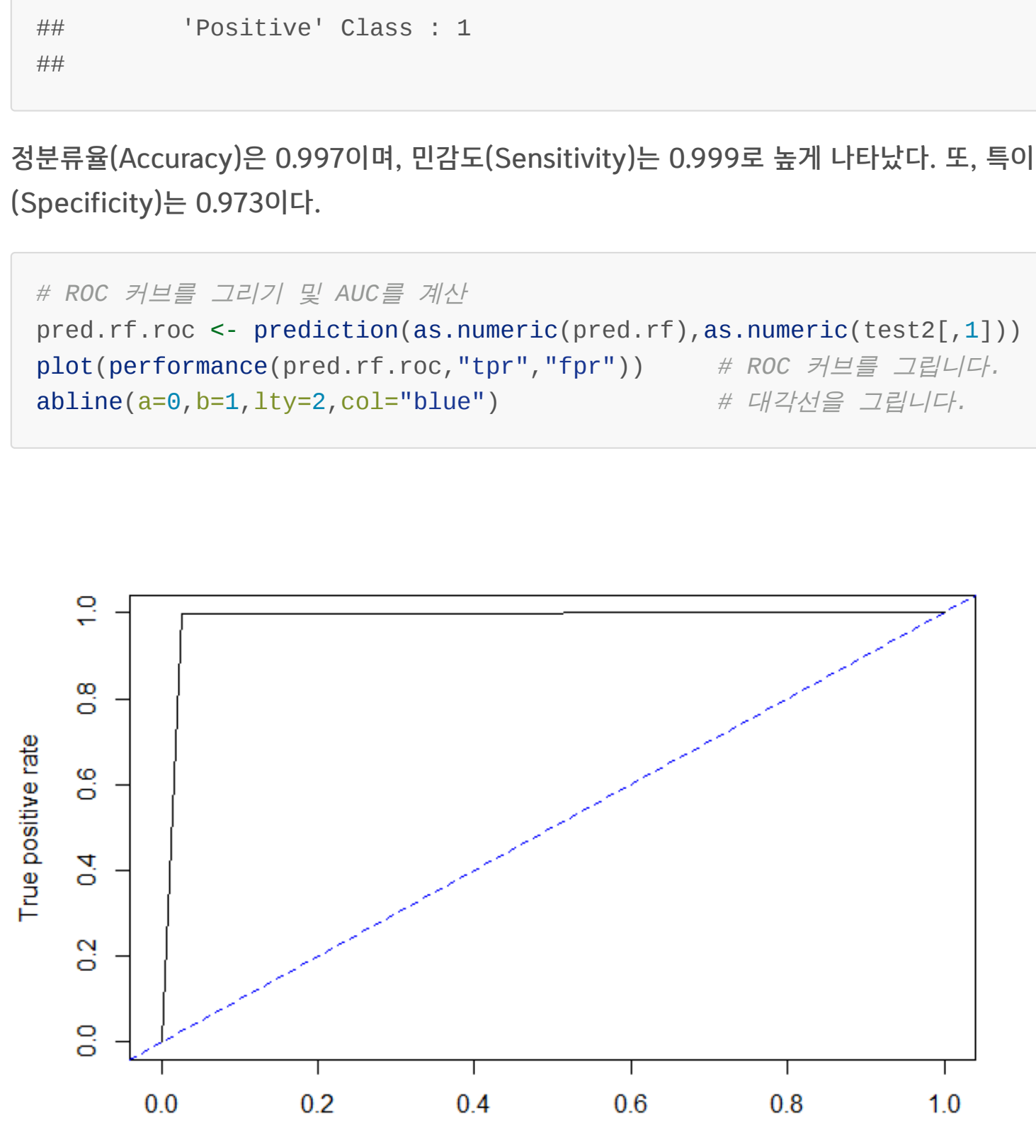


```
# 테스트 데이터를 사용하여 예측 수행 및 정분류율(Accuracy) 확인
pred.rf <- predict(rf.model,test2[,-1],type="class")
confusionMatrix(data=pred.rf, reference=test2[,1], positive='1')
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0  297    1
##          1    8 3079
##                                          
##                Accuracy : 0.9973         
##                  95% CI : (0.995, 0.9988)
##     No Information Rate : 0.9099         
##     P-Value [Acc > NIR] : <2e-16         
##                                          
##                   Kappa : 0.9836         
##                                          
##  Mcnemar's Test P-Value : 0.0455         
##                                          
##             Sensitivity : 0.9997         
##             Specificity : 0.9738         
##          Pos Pred Value : 0.9974         
##          Neg Pred Value : 0.9966         
##              Prevalence : 0.9099         
##          Detection Rate : 0.9096         
##    Detection Prevalence : 0.9120         
##       Balanced Accuracy : 0.9867         
##                                          
##        'Positive' Class : 1              
## 
```

정분류율(Accuracy)은 0.997이며, 민감도(Sensitivity)는 0.999로 높게 나타났다. 또, 특이도(Specificity)는 0.973이다.

```
# ROC 커브를 그리기 및 AUC를 계산
pred.rf.roc <- prediction(as.numeric(pred.rf),as.numeric(test2[,1]))
plot(performance(pred.rf.roc,"tpr","fpr"))      # ROC 커브를 그립니다.
abline(a=0,b=1,lty=2,col="blue")                # 대각선을 그립니다.
```



```
performance(pred.rf.roc,"auc")@y.values[[1]]    # AUC를 계산합니다.
```

```
## [1] 0.9867229
```

prediction 함수와 performance 함수로 값을 구하여 plot 함수로 ROC 커브를 그렸으며, AUC 값은 y.values 값으로 확인한 결과 0.986로 나타났다.