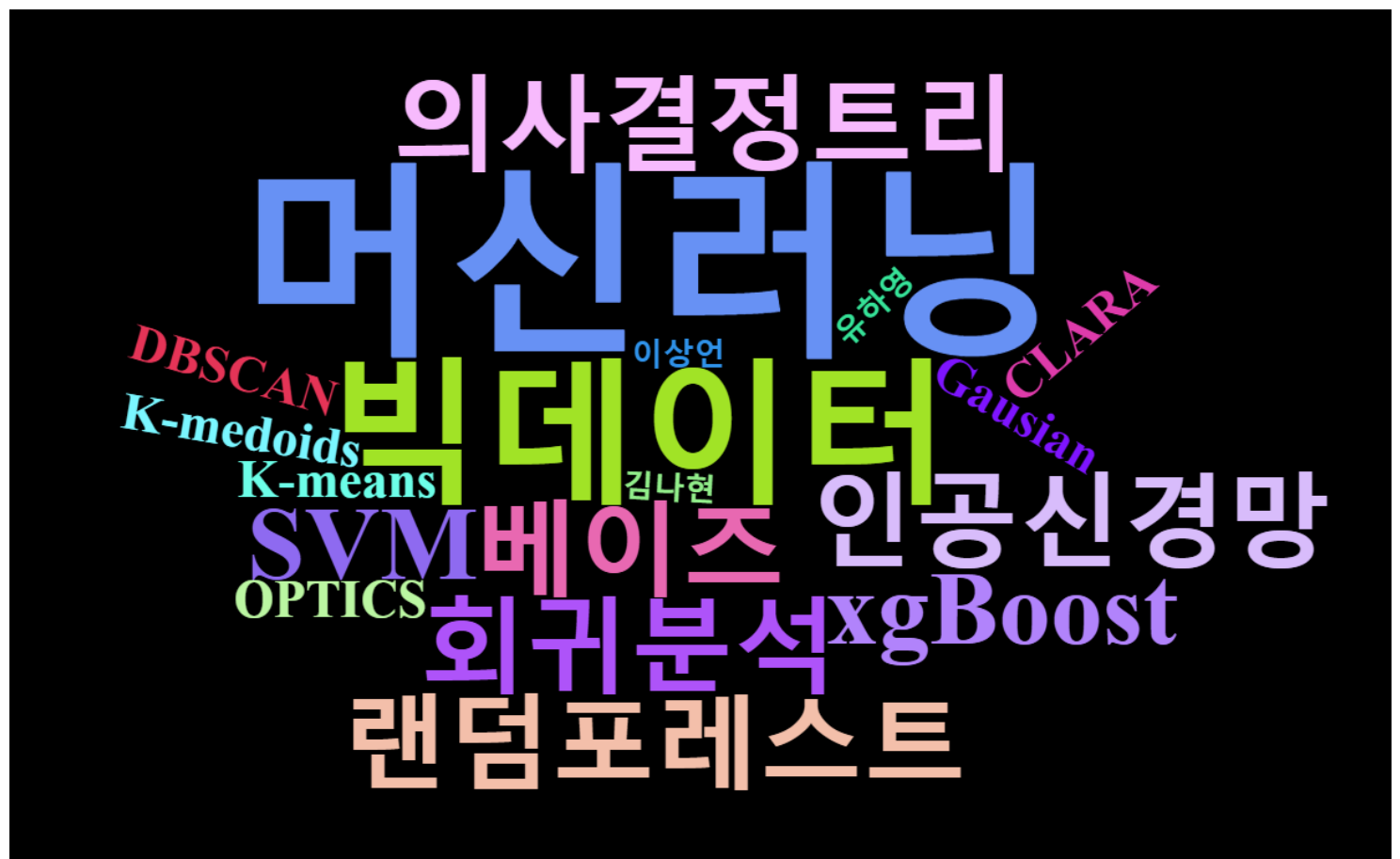

사례연구
#5

A1 팁

머신러닝기반 데이터 분석



목차

Chapter. 1 — 머신러닝 분류기법

- 1) Wisconsin Diagnostic Breast Cancer Data
- 2) 나이브 베이즈
- 3) 로지스틱 회귀분석
- 4) 의사결정트리
- 5) 인공신경망
- 6) SVM
- 7) 앙상블 (랜덤포레스트)
- 8) 앙상블 (xgBoost)
- 9) 결과 비교

Chapter. 2 — 머신러닝 예측기법

- 1) Boston Housing Price Data
- 2) 다중회귀분석
- 3) 의사결정트리
- 4) 인공신경망
- 5) 랜덤포레스트
- 6) 결과 비교

Chapter. 3 — 참고 사항 (사용된 R코드)

Chapter. 4 — 머신러닝 Clustering 기법

- 1) K-means (k-평균값 군집)
- 2) K-medoids (k-중앙값 군집)
- 3) Gaussian mixture models
- 4) DBSCAN
- 5) OPTICS
- 6) STING
- 7) CLIQUE
- 8) CLARA
- 9) CLARAN
- 10) BIRCH
- 11) Canopy
- 12) SUBCLU
- 13) 참고 자료

Chapter. 1 – 머신러닝 분류기법

1) 데이터 소개

Wisconsin Diagnostic Breast Cancer Data

Summary

데이터 내용

위스콘신 유방암 진단 데이터
미세 바늘로 흡입한 세포들을 디지털 이미지화한 후, 각 이미지를 분석할 결과를 예측 변수로 사용하여 종양이 악성(M)인지 양성(B)인지 판별

데이터 크기

1. 레코드수 : 569개
2. 컬럼개수 : 32개(ID, 결과, 30개 실험값)

id	환자 식별 번호
diagnosis	양성 여부 (M = 악성, B = 양성)
각 세포에 대한 정보들	
radius	반경 (중심에서 외벽까지 거리들의 평균값)
texture	질감 (Gray-Scale 값들의 표준편차) #gray-scale 값은 광도의 정보를 전달할 수
perimeter	둘레
area	면적
smoothness	매끄러움(반경길이의 국소적 변화)
compactness	조그만 정도(둘레 ² /면적 - 1)
concavity	오목함(윤곽의 오목한 부분의 정도)
points	오목한 점의 수
symmetry	대칭
dimension	프랙탈 차원(해안선근사 -1)
_mean	3 ~ 12 번까지는 평균값을 의미합니다.
_se	13 ~ 22 번까지는 표준오차(Standard Error) 를 의미합니다.
_worst	23 ~ 32 번까지는 각 세포별 구분들에서 제일 큰 3개의 값을 평균낸 값입니다.

2) 나이브 베이즈

Naïve Bayes 알고리즘

나이브 베이즈는 목표 값을 예측하는 모든 기능이 서로 독립적이라는 가정하에 Bayes의 정리를 기반으로 하는 분류 기술이다. 각 클래스의 확률을 계산 한 다음 확률이 가장 높은 클래스를 선택한다.

Bayes의 정리는 해당사건과 관련이 있을 수 있는 조건에 대한 사전 지식을 기반으로 사건의 확률을 설명한다.

Naïve Bayes는 대상을 예측하는 데 사용하는 기능이 독립적이며 서로 영향을 미치지 않는 다고 가정한다.

독립 가정은 실제 데이터에서는 정확하지 않지만 실제로 작동 수준은 훌륭합니다. 그래서 “Naïve” 라 불린다..

Naive Bayes

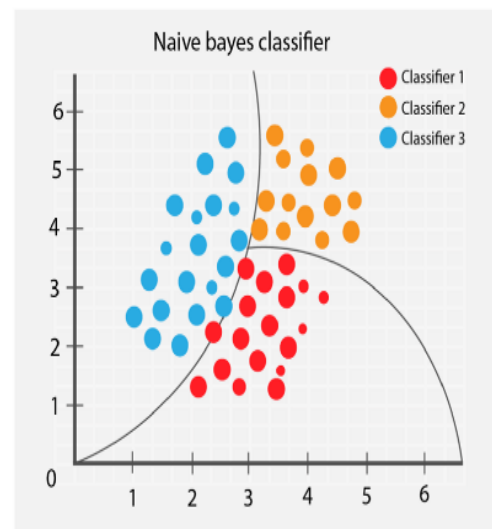


In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$



나이브 베이즈 적용 결과

1. Confussion Matrix

Confusion Matrix and Statistics

Reference
Prediction B M
B 105 2
M 2 61

Accuracy : 0.9765 ————— 정확도
95% CI : (0.9409, 0.9936)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9496

Mcnemar's Test P-Value : 1

Sensitivity : 0.9813

Specificity : 0.9683

Pos Pred Value : 0.9813

Neg Pred Value : 0.9683

Prevalence : 0.6294

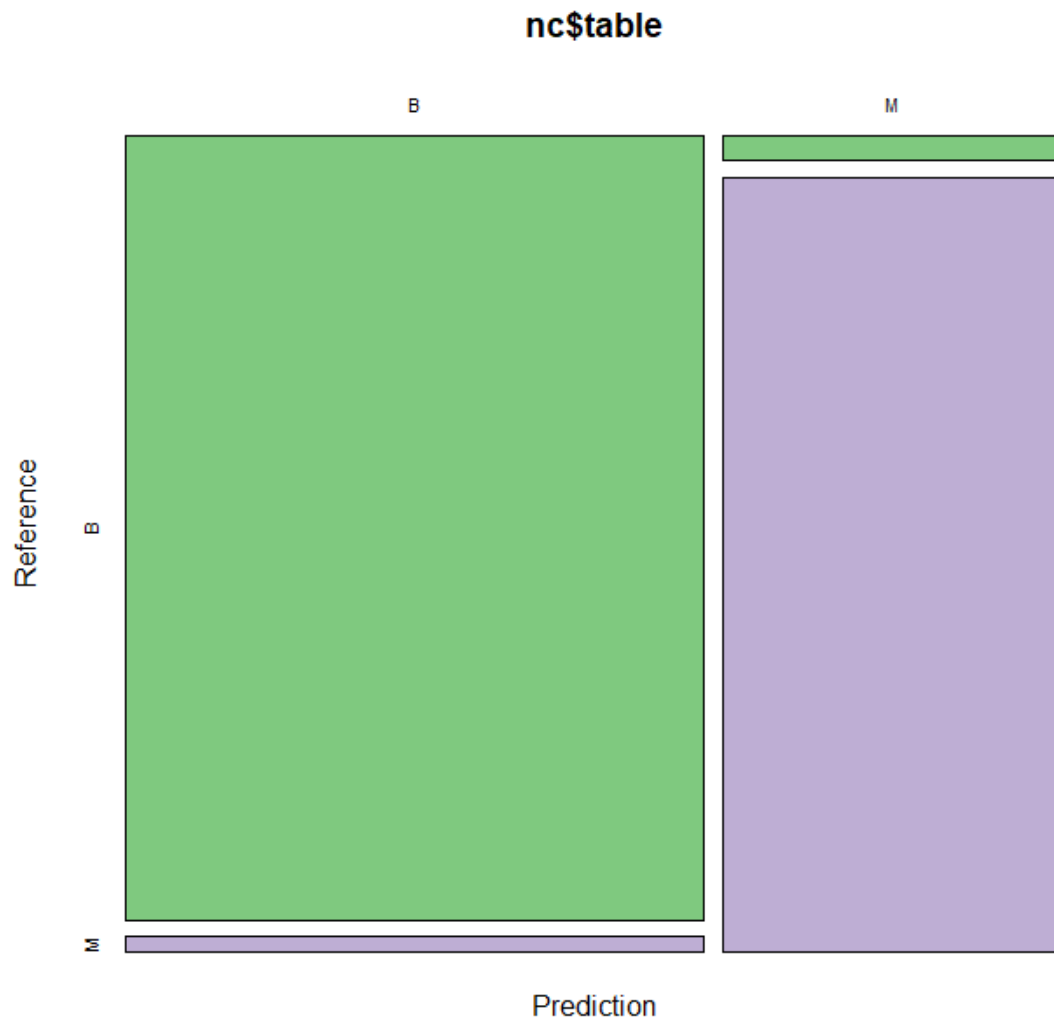
Detection Rate : 0.6176

Detection Prevalence : 0.6294

Balanced Accuracy : 0.9748

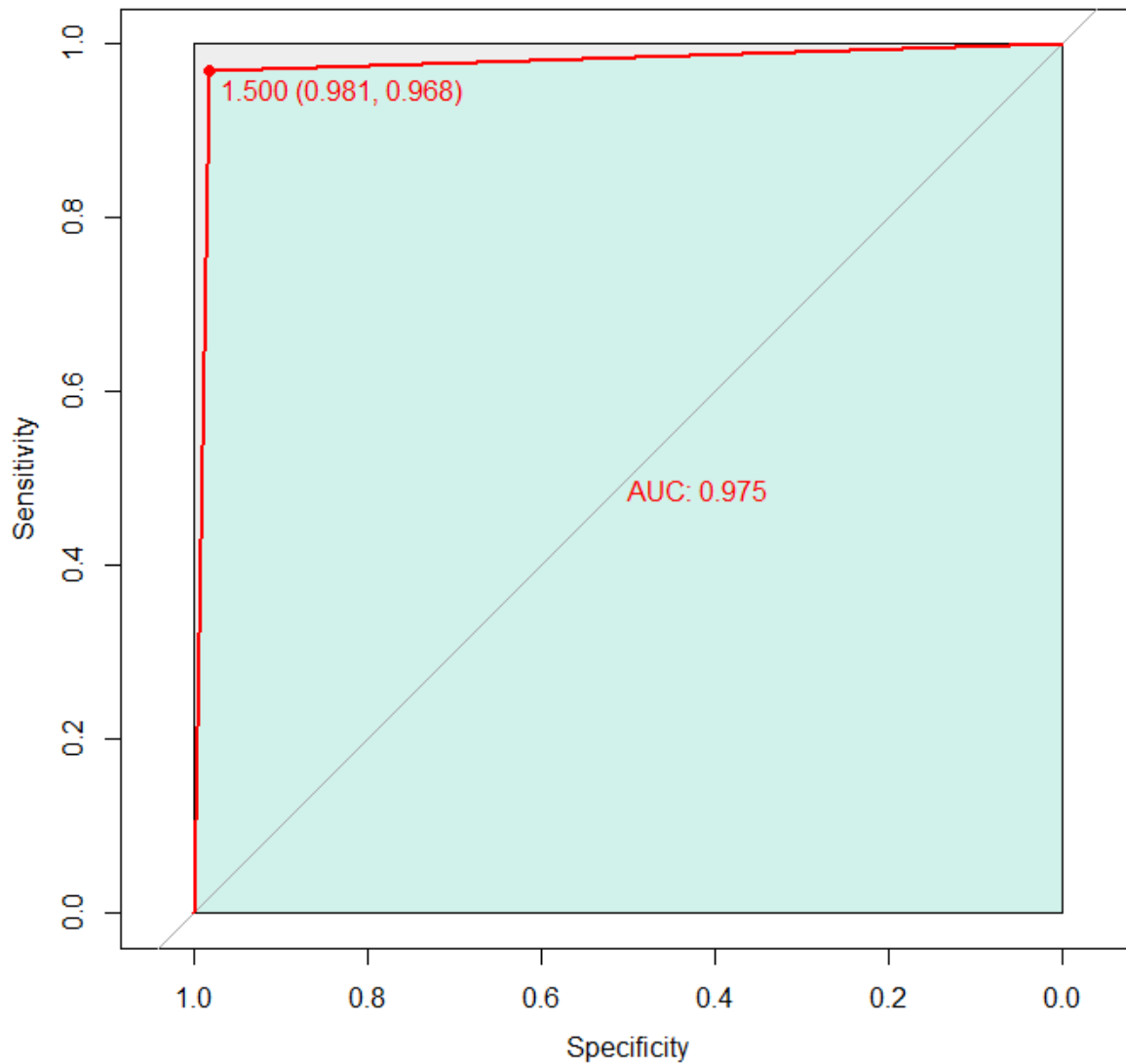
'Positive' Class : B

2. Confusion Matrix Plot



나이브 베이즈 적용 결과

ROC 곡선 (AUC)



민감도 : 0.9813

특이도 : 0.9683

AUC : 0.9765

민감도와 특이도 모두 1에 가까운 수치를 보여 높은 분류 수준을 확인 할 수 있다.
따라서 AUC 값 역시 1에 가까운 높은 수치를 보인다.

3) 로지스틱 회귀분석

로지스틱 회귀분석 알고리즘

선형 회귀는 연속적인 종속 변수와 한 개 이상의 예측 변수 사이의 관계를 모델링하는 접근법이다. 종속 변수가 연속형이 아니라 범주형이라면 선형 회귀는 로짓 연결(logit link) 함수를 이용해 로지스틱 회귀로 변환될 수 있다. 로지스틱 회귀는 단순하고 빠르지만 강력한 분류 알고리즘이다. 로지스틱 회귀에서는 주어진 자료가 각각의 클래스에 속할 때 그 개연성을 비교 예측하기 위해 각기 다른 가설 클래스(hypothesis class)를 사용한다.

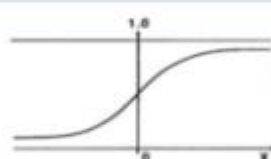
Logistic Regression 이란?

Ex. 고양이 or 강아지?

- 목표: 입력된 data의 '특징'을 이용하여, 0 or 1의 output을 추정함
- Task: Binary Classification에 사용되며, 활성화 함수로 sigmoid가 사용된다.

예측값 $\hat{y} = \text{Sigmoid}(w^T x + b)$, $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$ ($w = \text{weight}, b = \text{bias}$)

Logistic Regression에서 $0 \leq \hat{y} \leq 1$ 으로 만들기 위해 Sigmoid 함수 사용

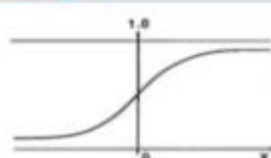
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{Sigmoid } \sigma(x) = \frac{1}{1+e^{-x}}$$

Logistic Regression 이란?

- 목표: 입력된 data의 '특징'을 이용하여, 0 or 1의 output을 추정함
- Task: Binary Classification에 사용되며, 활성화 함수로 sigmoid가 사용된다.

예측값 $\hat{y} = \text{Sigmoid}(w^T x + b)$, $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$ ($w = \text{weight}, b = \text{bias}$)

Logistic Regression에서 $0 \leq \hat{y} \leq 1$ 으로 만들기 위해 Sigmoid 함수 사용

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{Sigmoid } \sigma(x) = \frac{1}{1+e^{-x}}$$

1. Confussion Matrix

Confusion Matrix and Statistics

Reference
Prediction B M
B 105 3
M 2 60

Accuracy : 0.9706 ————— 정확도

95% CI : (0.9327, 0.9904)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9367

Mcnemar's Test P-Value : 1

Sensitivity : 0.9813

Specificity : 0.9524

Pos Pred Value : 0.9722

Neg Pred Value : 0.9677

Prevalence : 0.6294

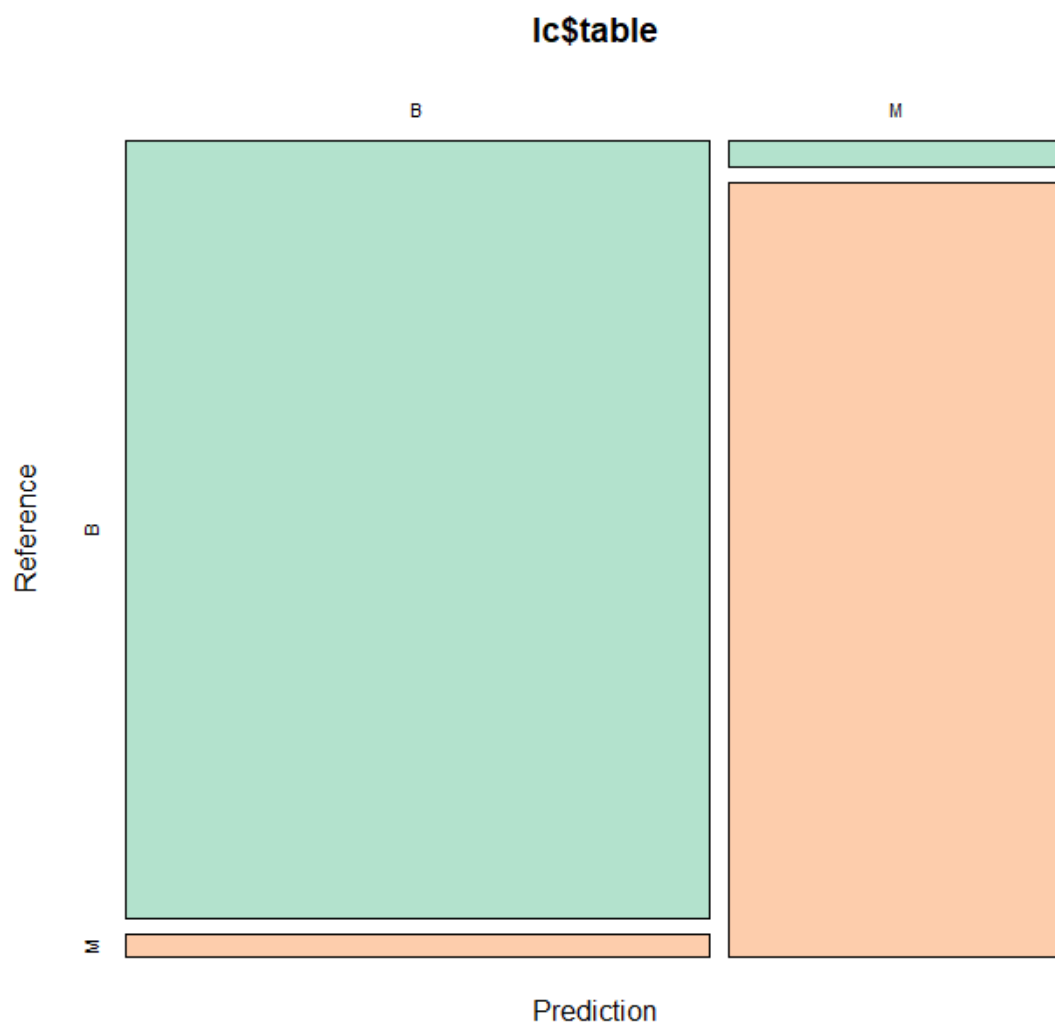
Detection Rate : 0.6176

Detection Prevalence : 0.6353

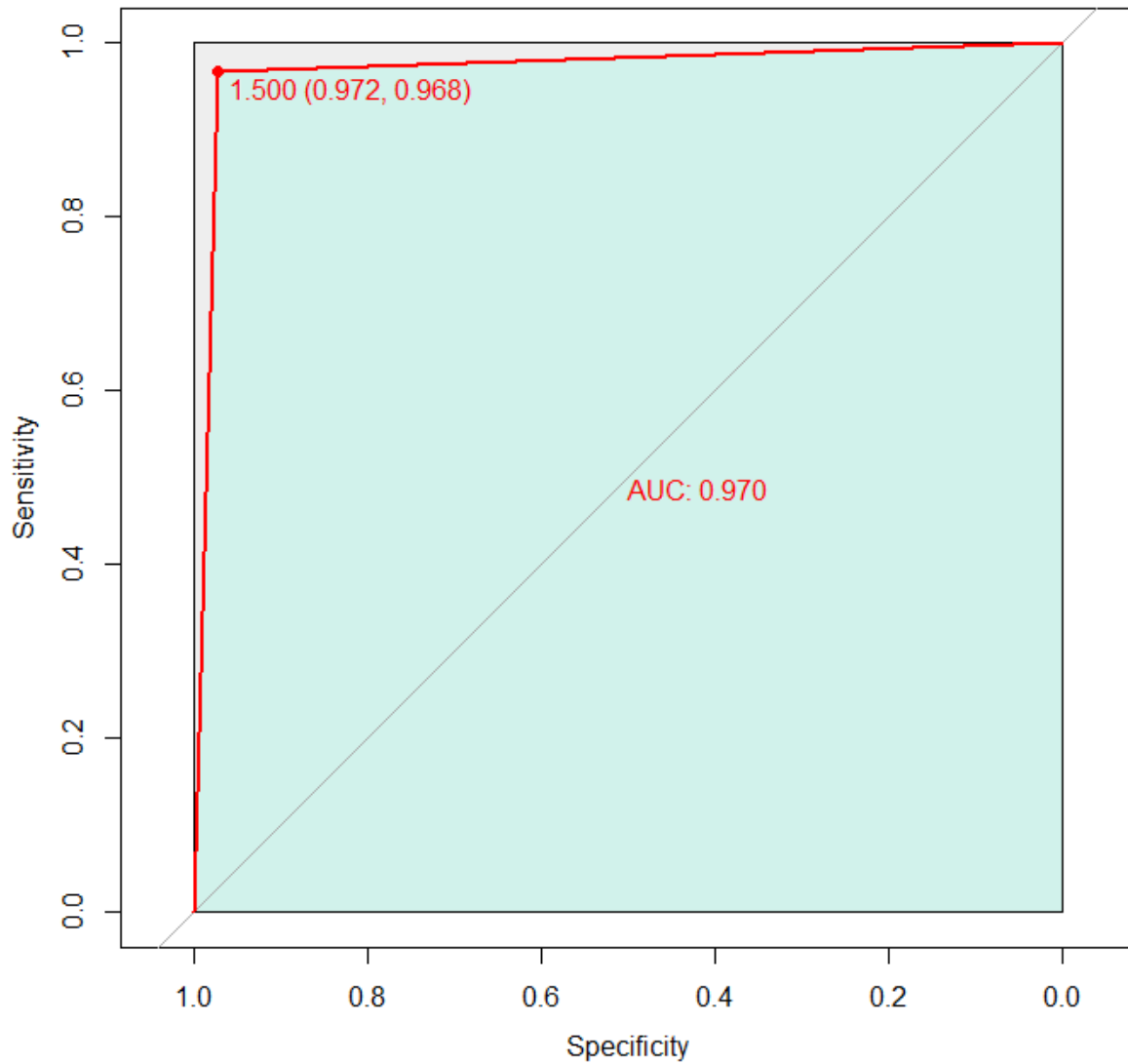
Balanced Accuracy : 0.9668

'Positive' Class : B

2. Confusion Matrix Plot



3. ROC 곡선 (AUC)



민감도 : 0.9813
특이도 : 0.9524
AUC : 0.970

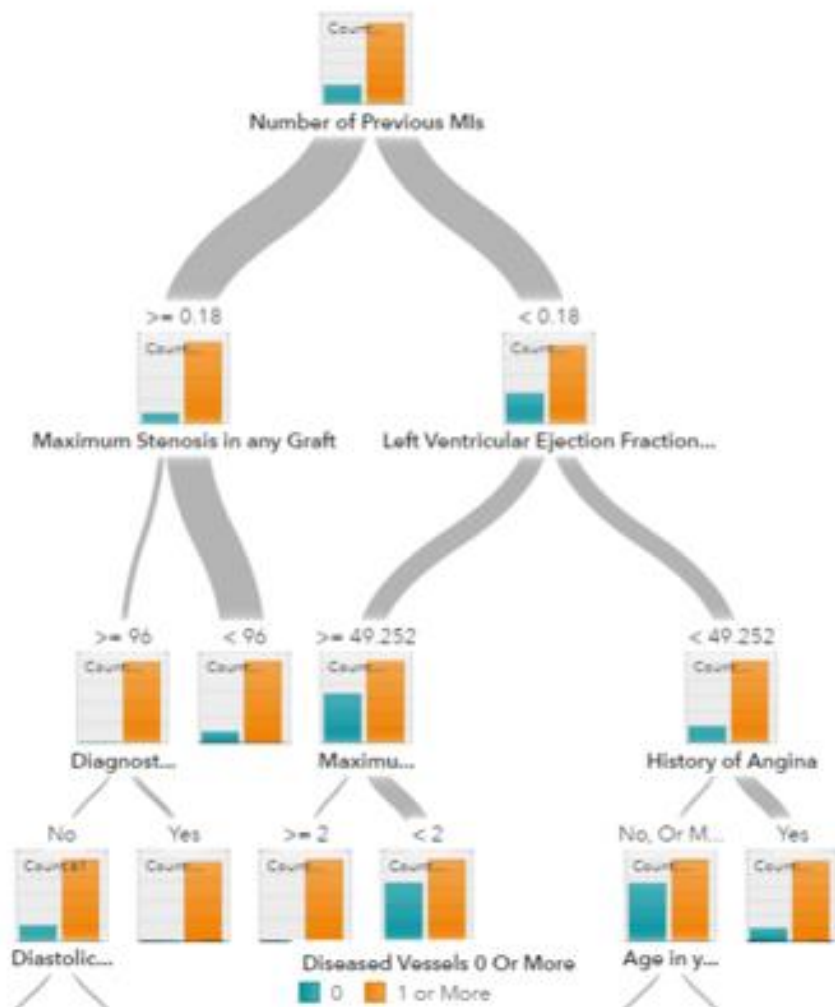
민감도와 특이도 모두 1에 가까운 수치를 보여 높은 분류 수준을 확인 할 수 있다.
따라서 AUC 값 역시 1에 가까운 높은 수치를 보인다.

4) 의사결정트리

의사결정트리 알고리즘

의사결정 트리는 훈련 데이터에서 간단한 의사 결정 규칙을 학습해 예측하는 모델이다.

다양한 종류의 트리가 있지만, 모두 동일한 작업을 수행한다. 즉 특징 공간(feature space)을 거의 같은 레이블로 구별되도록 분리한다. 의사결정 트리는 이해와 구현이 쉽지만 가지를 다 쳐내고 트리의 깊이가 너무 깊어질 경우 데이터를 과적합(overfit)하는 경향이 있다.



의사결정트리 적용 결과

1. Confussion Matrix

Confusion Matrix and Statistics

Reference
Prediction B M
B 101 8
M 6 55

Accuracy : 0.9176 ————— 정확도
95% CI : (0.8657, 0.9542)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8223

Mcnemar's Test P-Value : 0.7893

Sensitivity : 0.9439

Specificity : 0.8730

Pos Pred Value : 0.9266

Neg Pred Value : 0.9016

Prevalence : 0.6294

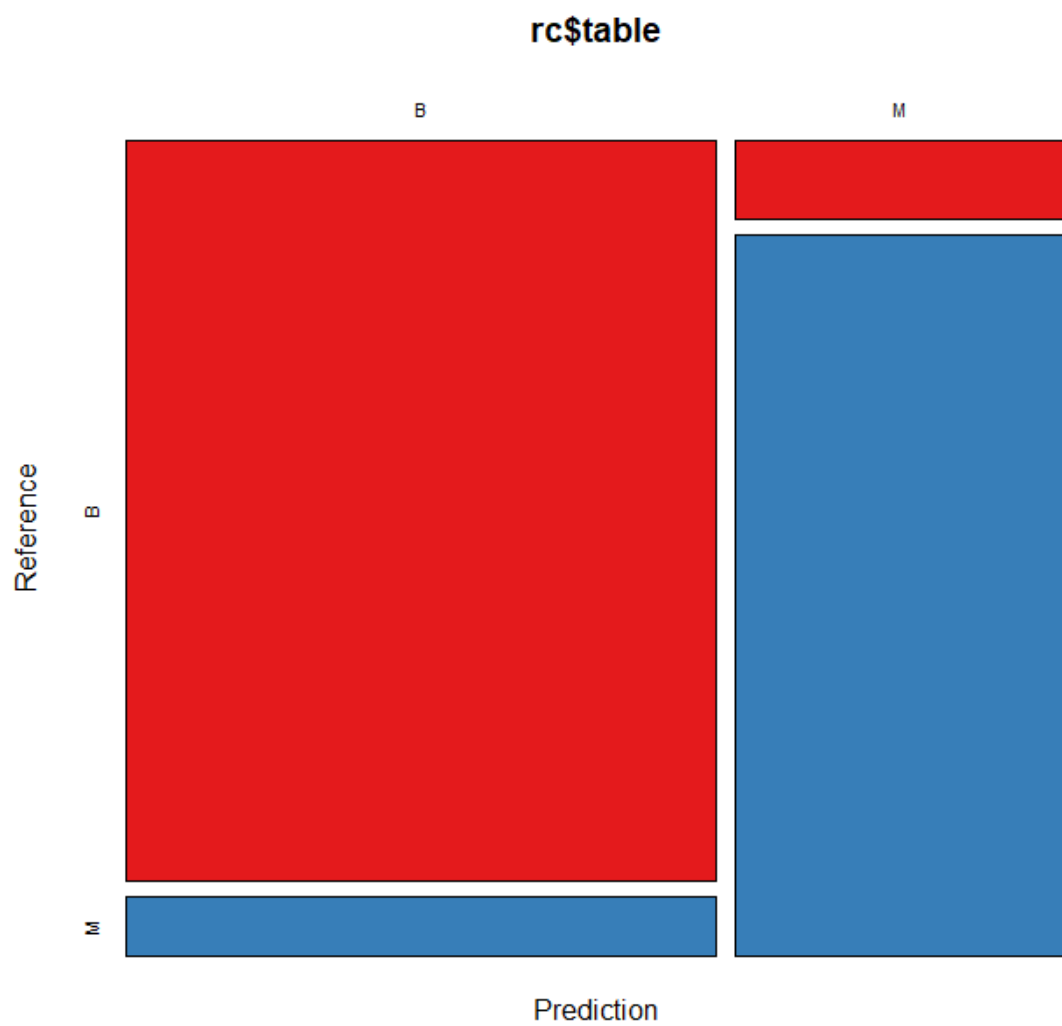
Detection Rate : 0.5941

Detection Prevalence : 0.6412

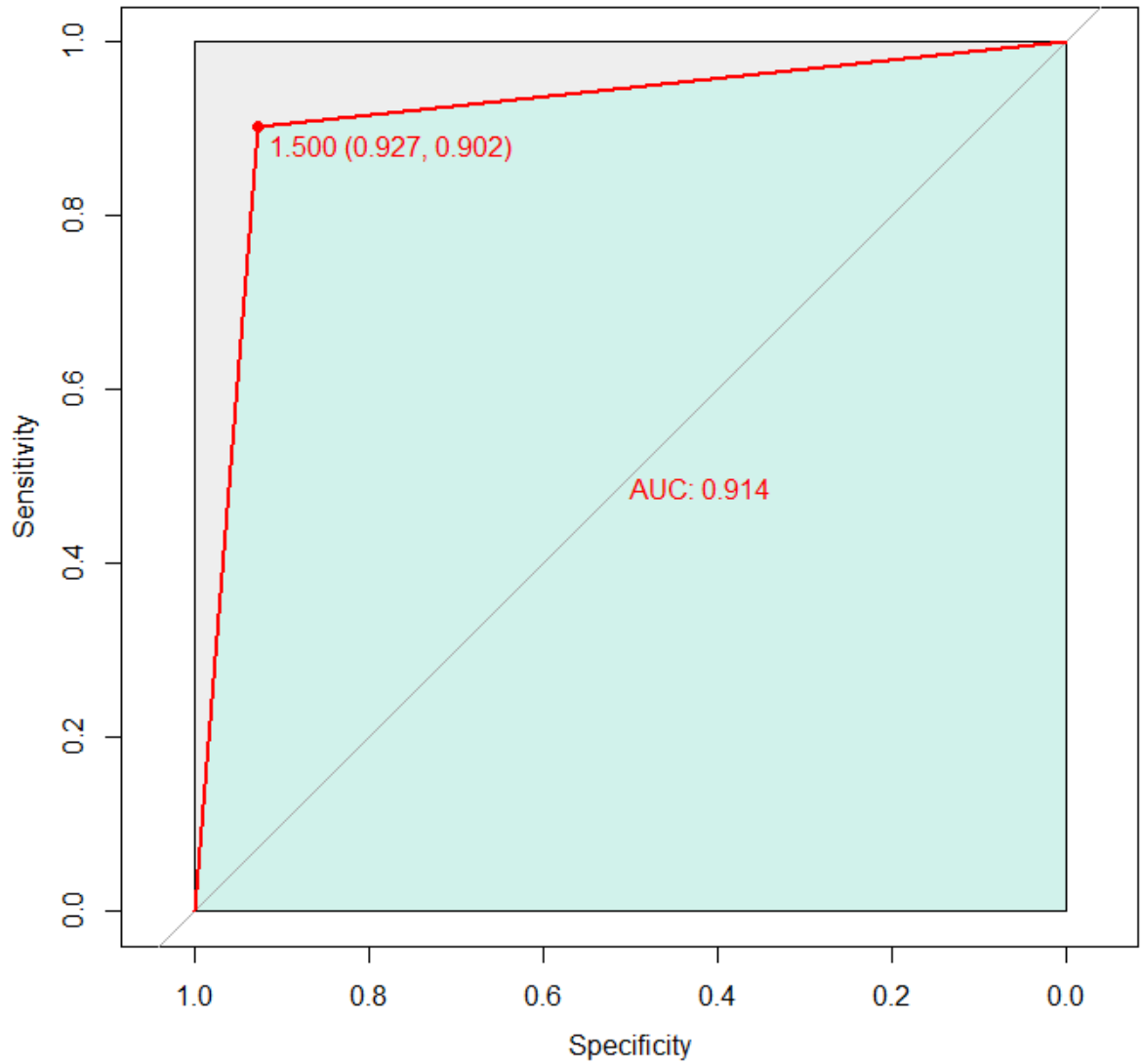
Balanced Accuracy : 0.9085

'Positive' Class : B

2. Confusion Matrix Plot



3. ROC 곡선 (AUC)



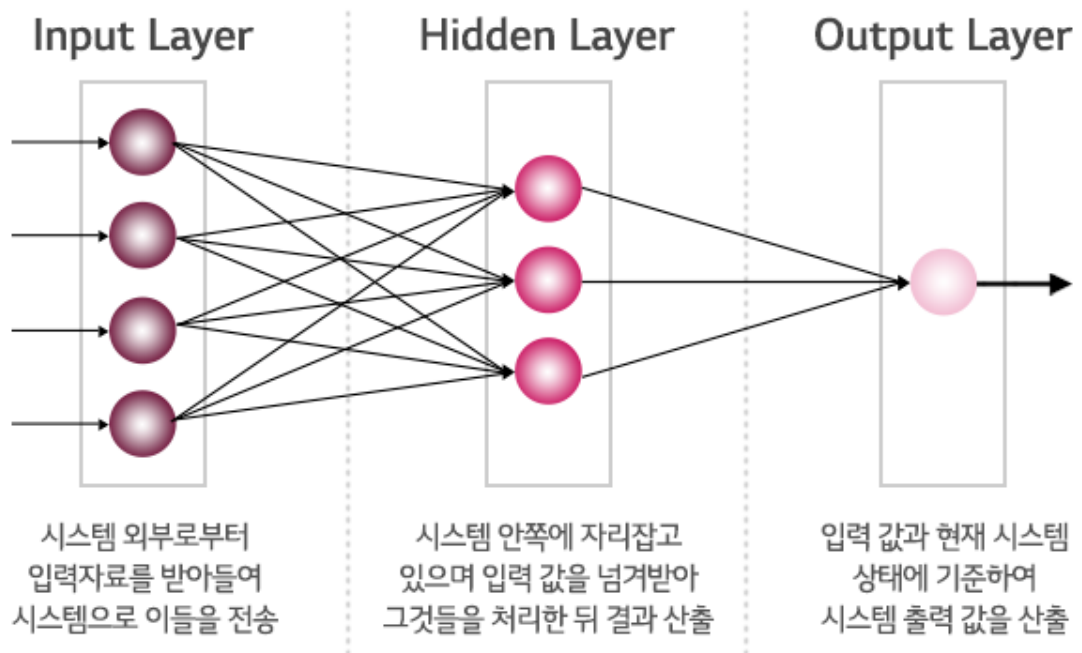
민감도 : 0.9439
특이도 : 0.8730
AUC : 0.914

민감도와 특이도 모두 1에 가까운 수치를 보여 높은 분류 수준을 확인 할 수 있다.
따라서 AUC 값 역시 1에 가까운 높은 수치를 보인다.

5) 인공신경망

인공신경망 알고리즘

신경망은 입력 계층(input layer), 은닉 계층(hidden layers), 출력 계층(output layer)의 세 부분으로 구성된다. 학습 표본(training samples)은 입력 및 출력 계층을 정의한다. 출력 계층이 범주형 변수일 때 신경망은 분류 문제를 해결한다. 출력 계층이 연속 변수일 때 신경망은 회귀 작업을 위해 사용될 수 있다. 또 출력 계층이 입력 계층과 동일할 때 신경망은 고유한 특징을 추출할 수 있다. 이때 은닉 계층의 수는 모델 복잡성과 모델링 수용력(capacity)을 결정한다.



1. Confussion Matrix

Confusion Matrix and Statistics

Reference
Prediction B M
B 106 6
M 1 57

Accuracy : 0.9588

정확도

95% CI : (0.917, 0.9833)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9103

Mcnemar's Test P-Value : 0.1306

Sensitivity : 0.9907

Specificity : 0.9048

Pos Pred Value : 0.9464

Neg Pred Value : 0.9828

Prevalence : 0.6294

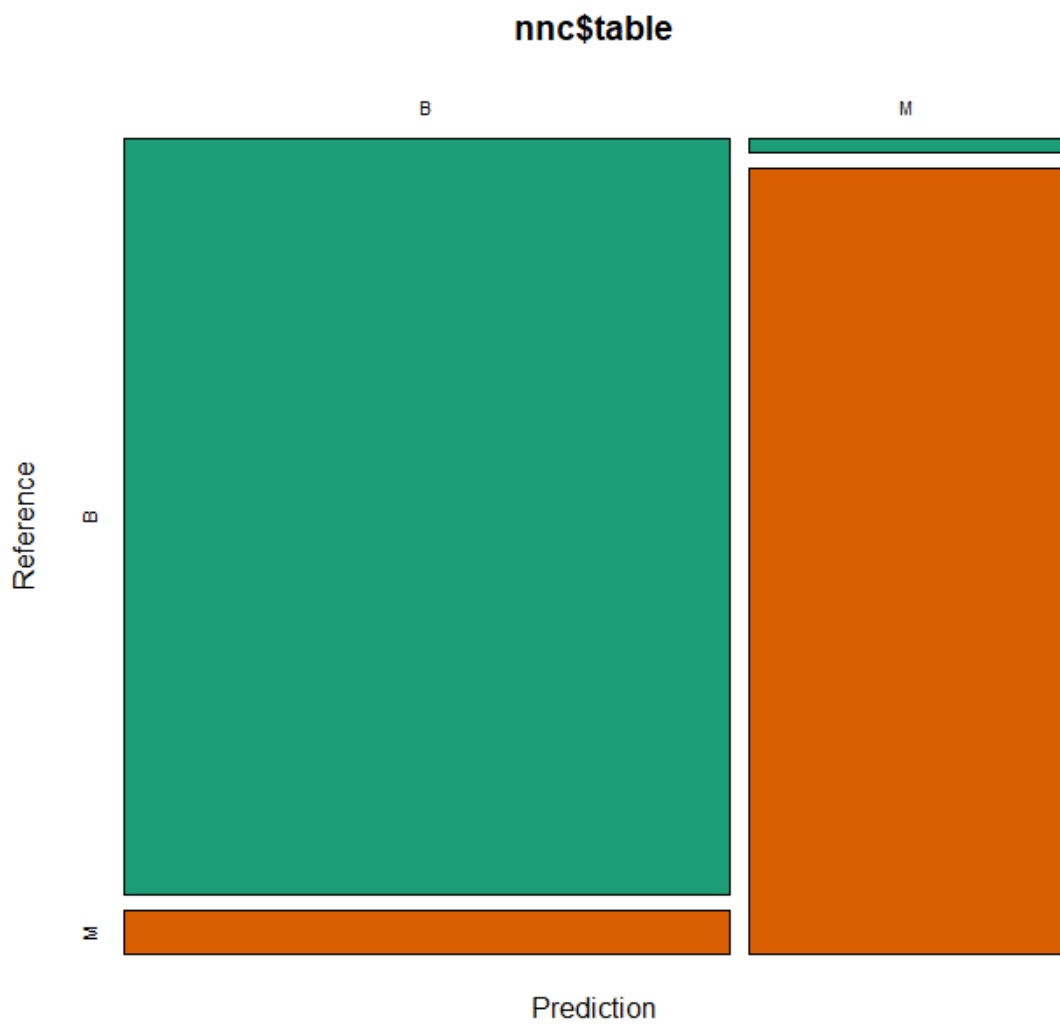
Detection Rate : 0.6235

Detection Prevalence : 0.6588

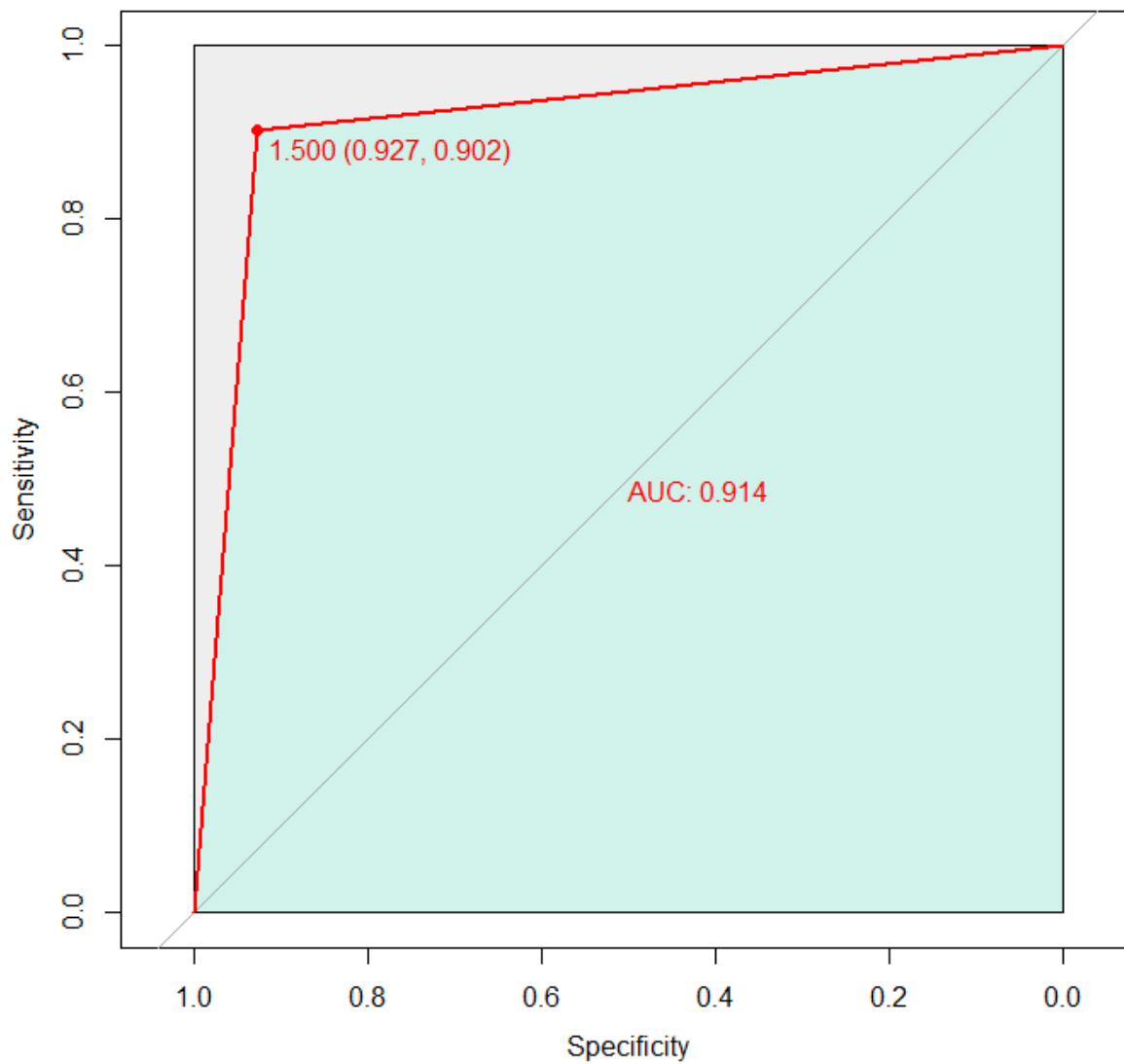
Balanced Accuracy : 0.9477

'Positive' Class : B

2. Confusion Matrix Plot



3. ROC 곡선 (AUC)



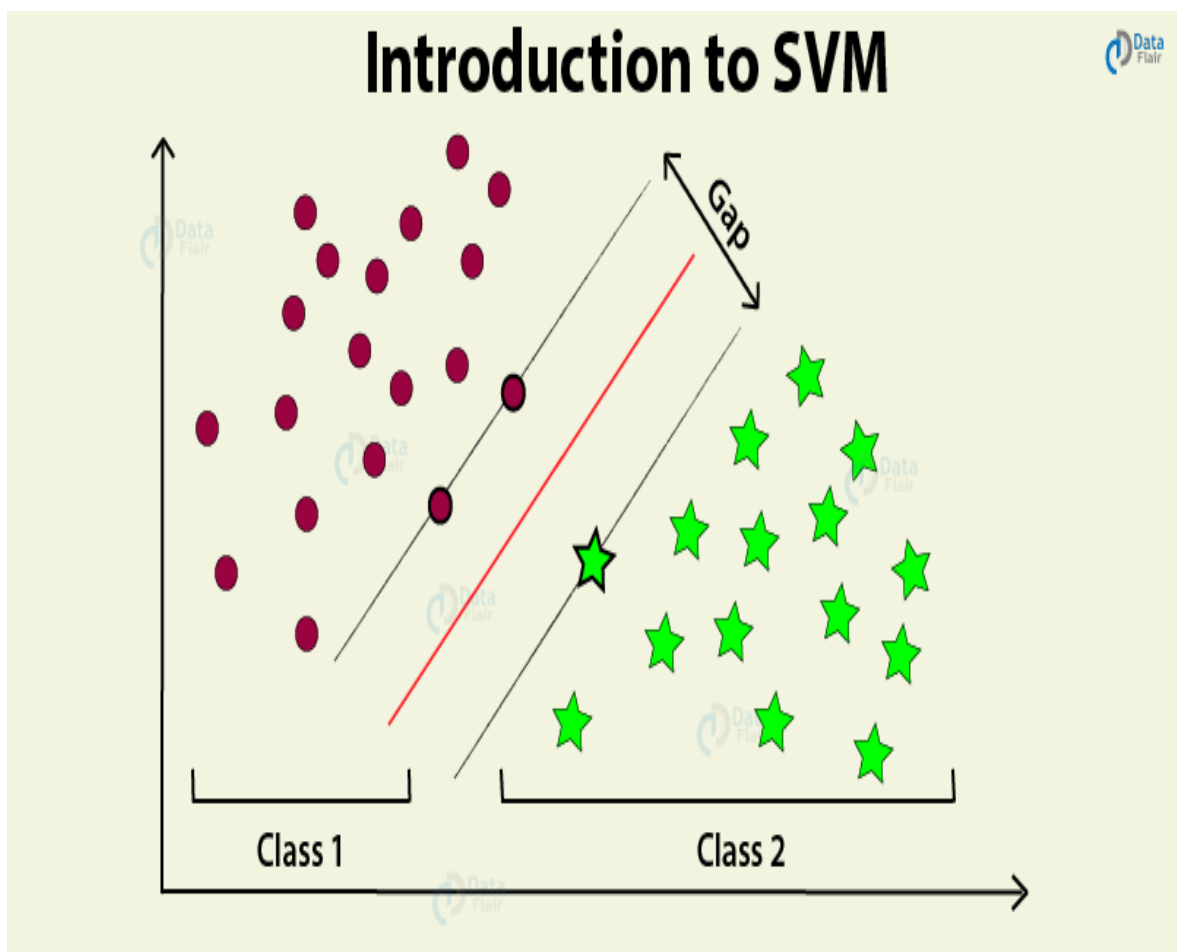
민감도 : 0.9907
특이도 : 0.9048
AUC : 0.914

민감도와 특이도 모두 1에 가까운 수치를 보여 높은 분류 수준을 확인 할 수 있다.
따라서 AUC 값 역시 1에 가까운 높은 수치를 보인다.

6) SVM

SVM 알고리즘

서포트 벡터 머신(support vector machine)은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다. 두 카테고리 중 어느 하나에 속한 데이터의 집합이 주어졌을 때, SVM 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어느 카테고리에 속할지 판단하는 비확률적 이진 선형 분류 모델을 만든다. 만들어진 분류 모델은 데이터가 사상된 공간에서 경계로 표현 되는데 SVM 알고리즘은 그 중 가장 큰 폭을 가진 경계를 찾는 알고리즘이다. SVM은 선형 분류와 더불어 비선형 분류에서도 사용될 수 있다



SVM 적용 결과

1. Confussion Matrix

Confusion Matrix and Statistics

Reference
Prediction B M
B 106 3
M 1 60

Accuracy : 0.9765 ————— 정확도

95% CI : (0.9409, 0.9936)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9492

Mcnemar's Test P-Value : 0.6171

Sensitivity : 0.9907

Specificity : 0.9524

Pos Pred Value : 0.9725

Neg Pred Value : 0.9836

Prevalence : 0.6294

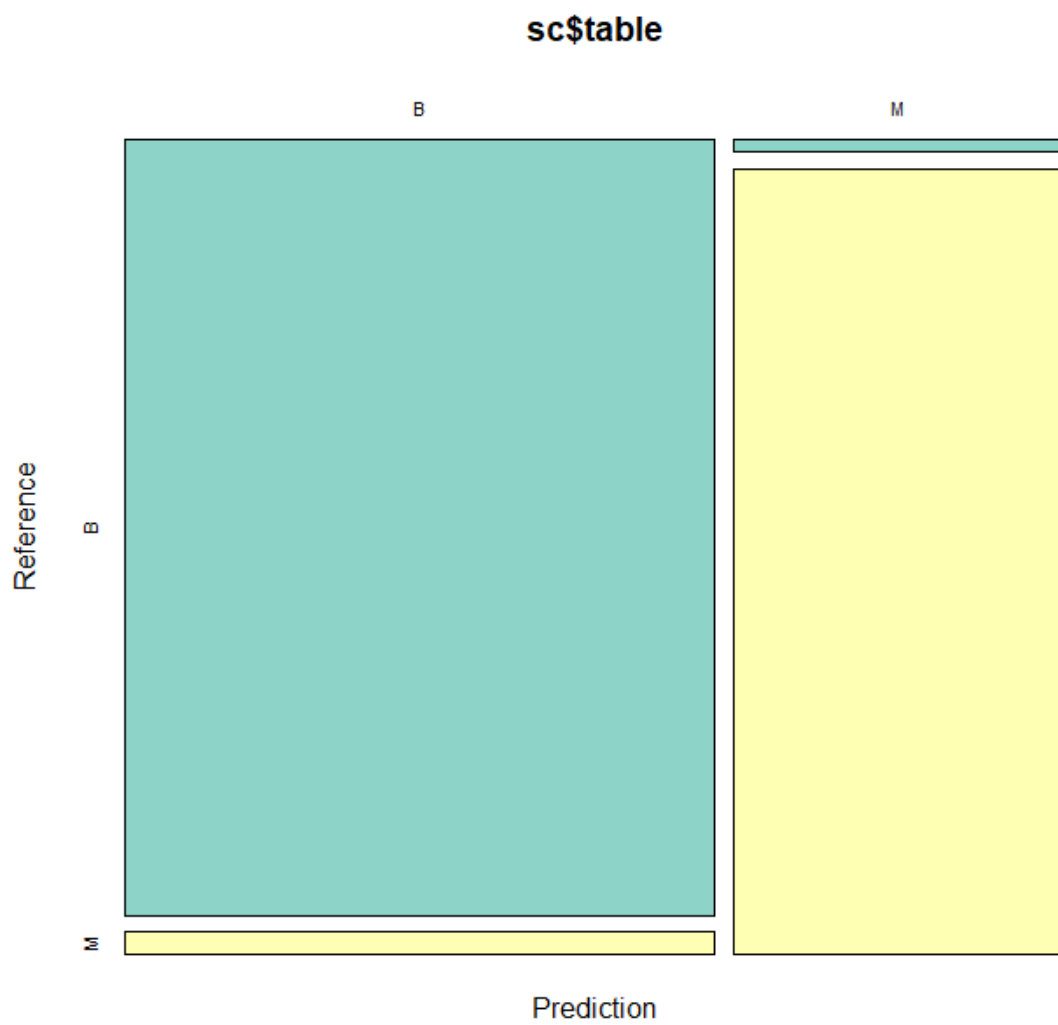
Detection Rate : 0.6235

Detection Prevalence : 0.6412

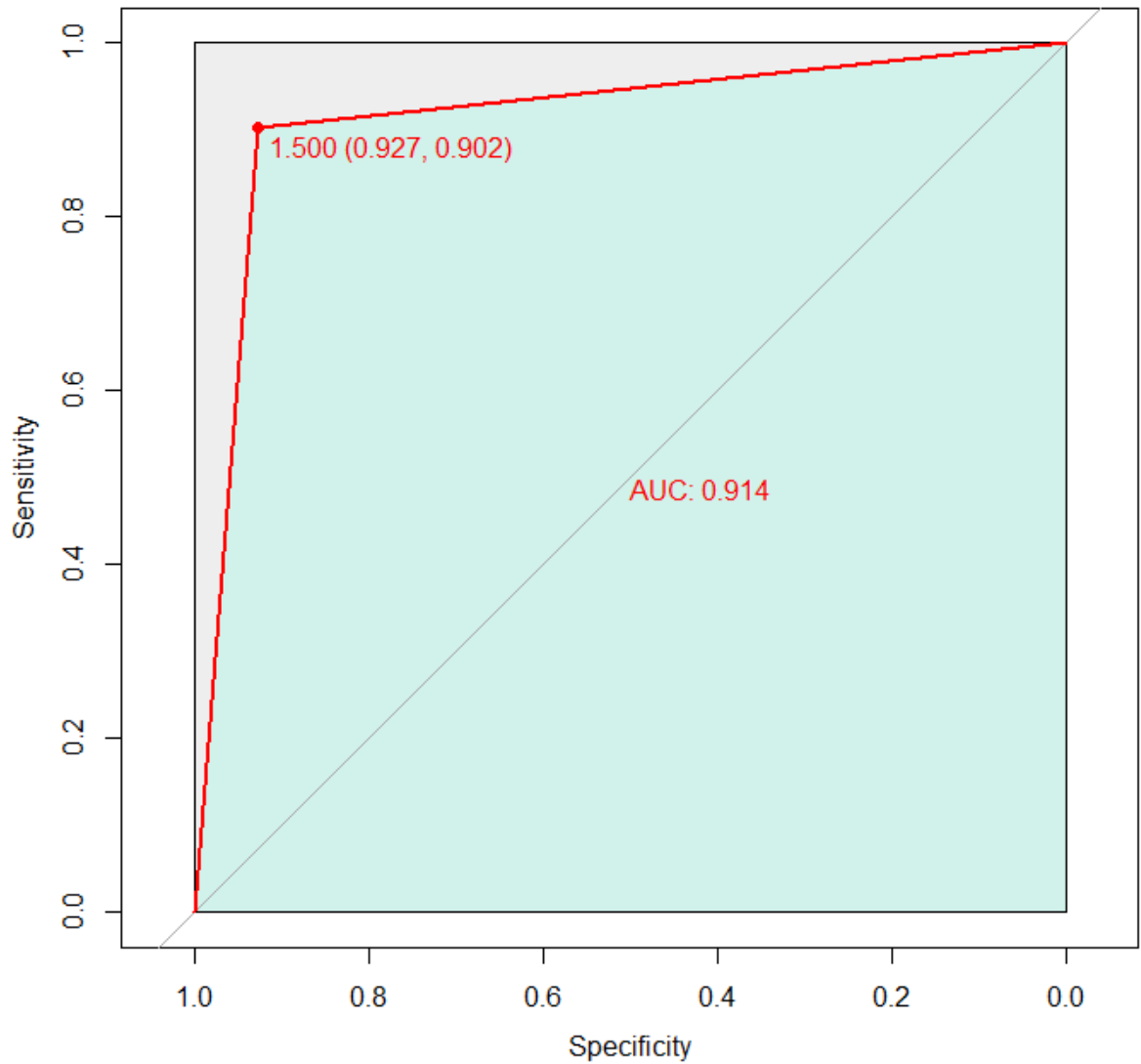
Balanced Accuracy : 0.9715

'Positive' Class : B

2. Confusion Matrix Plot



3. ROC 곡선 (AUC)



민감도 : 0.9907
특이도 : 0.9524
AUC : 0.914

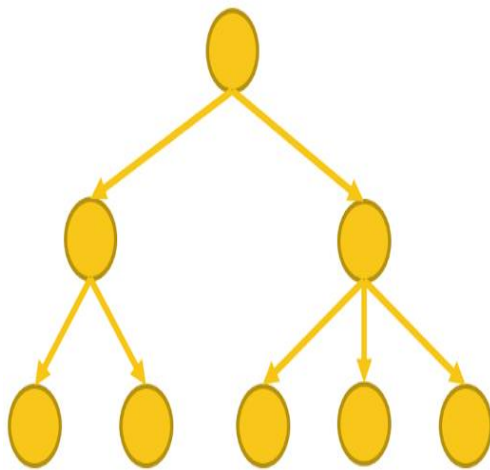
민감도와 특이도 모두 1에 가까운 수치를 보여 높은 분류 수준을 확인 할 수 있다.
따라서 AUC 값 역시 1에 가까운 높은 수치를 보인다.

7) 랜덤포레스트

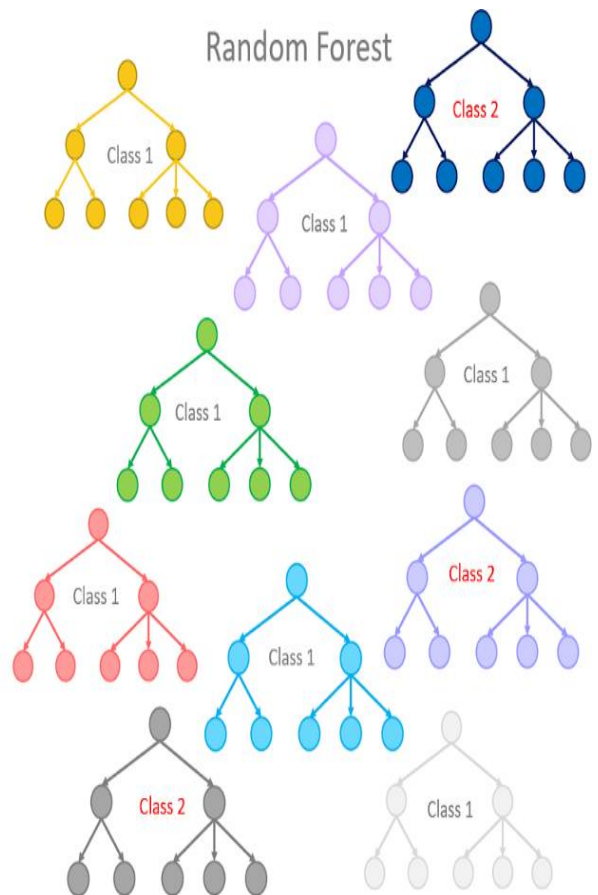
랜덤포레스트 알고리즘

랜덤 포레스트의 가장 큰 특징은 랜덤성(randomness)에 의해 트리들이 서로 조금씩 다른 특성을 갖는다는 점이다. 이 특성은 각 트리들의 예측(prediction)들이 비상관화(decorrelation) 되게하며, 결과적으로 일반화(generalization) 성능을 향상시킨다. 또한, 랜덤화(randomization)는 포레스트가 노이즈가 포함된 데이터에 대해서도 강인하게 만들어 준다. 랜덤화는 각 트리들의 훈련 과정에서 진행되며, 랜덤 학습 데이터 추출 방법을 이용한 앙상블 학습법인 배깅(bagging)과 랜덤 노드 최적화(randomized node optimization)가 자주 사용된다. 이 두 가지 방법은 서로 동시에 사용되어 랜덤화 특성을 더욱 증진 시킬 수 있다.

Single Decision Tree



Random Forest



랜덤포레스트 적용 결과

1. 랜덤포레스트 결과

Call:

```
randomForest(formula = diagnosis ~ ., data = wisc.test,  
importance = T)
```

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 5

OOB estimate of error rate: 4.71% — 오차율

Confusion matrix:

B M class.error

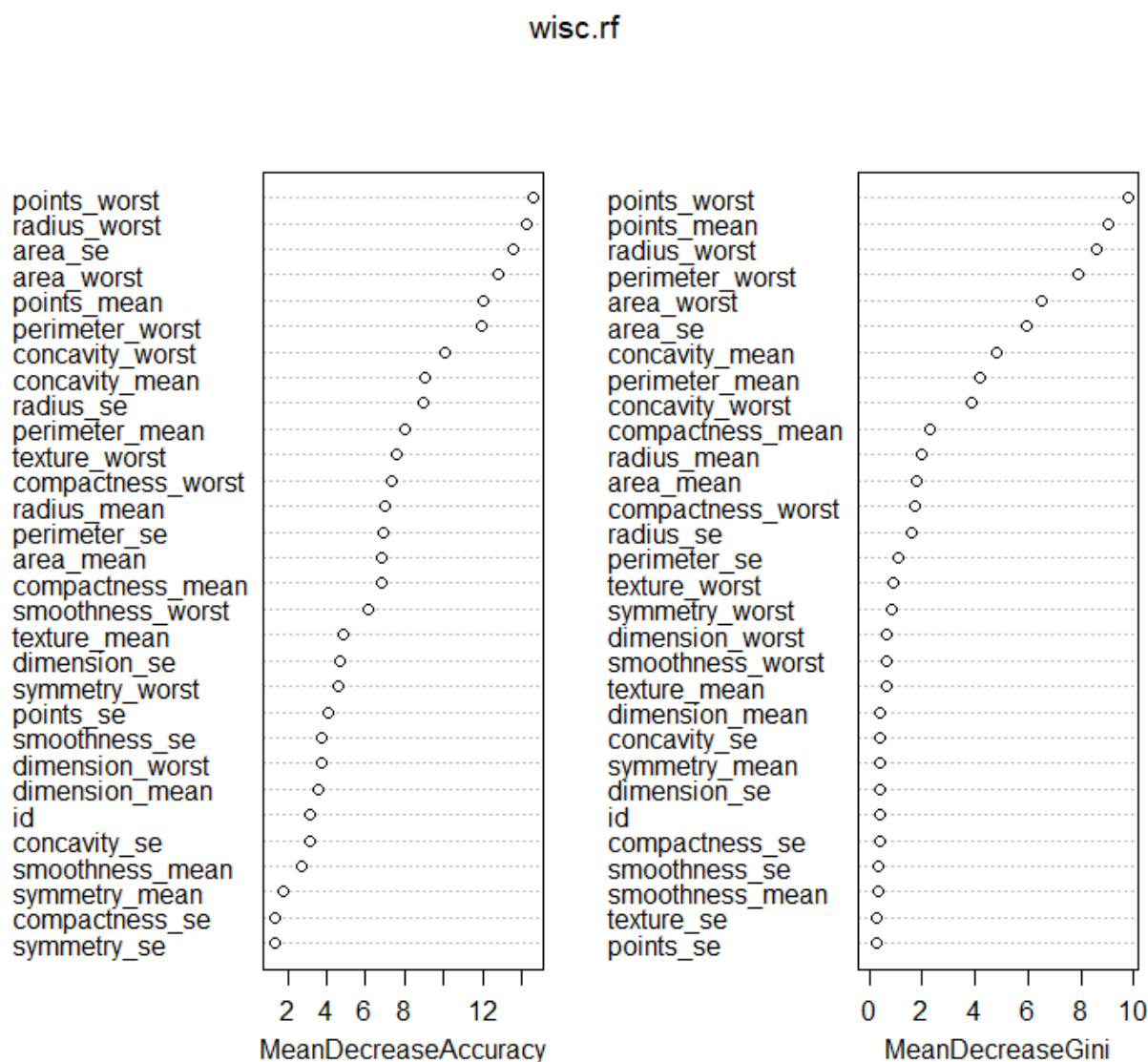
B 104 3 0.02803738

M 5 58 0.07936508

※ 정확도 : $1 - \text{오차율} = 0.9529$

랜덤포레스트 적용 결과

2. 중요 변수 시각화



중요 변수 순서는 다음과 같다.

points_worst (악성의 오목한 점), radius_worst (악성의 반경), area_se (세포 면적), area_worst (악성 세포 면적), points_mean(점의 평균값), perimeter_worst (악성 세포의 둘레) 가 주요 변수임을 알 수 있다.

8) xgBoost

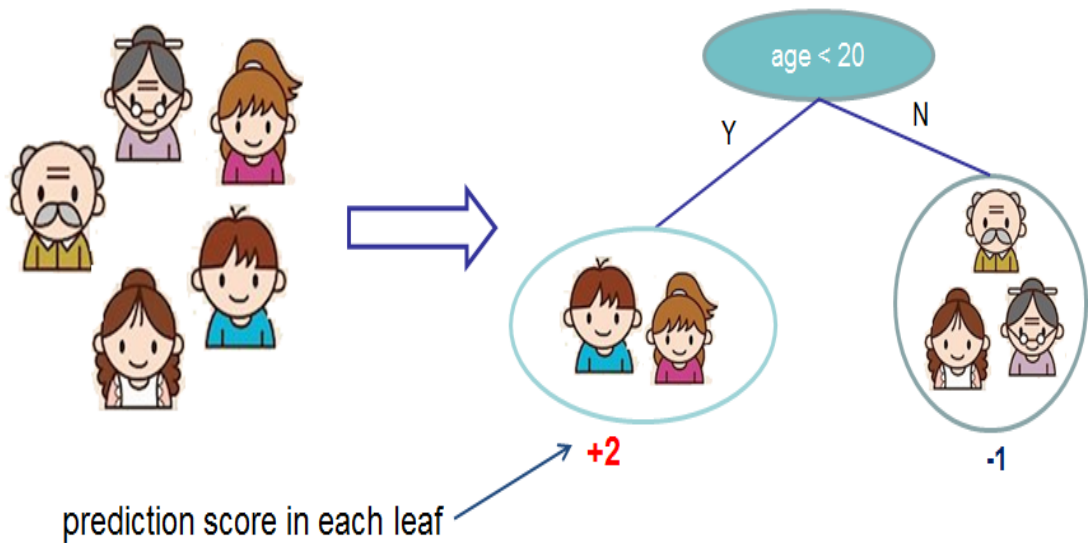
xgBoost 알고리즘

Boosting이란 여러 개의 약한 Decision Tree(의사결정트리)를 조합해서 사용하는 앙상블 기법 중 하나이다. 약한 예측 모형들의 학습 에러에 가중치(weights)를 두고, 순차적으로 다음 학습 모델에 반영하여 결과적으로 강한 예측 모형을 만들어 내는 것이다.

Regression(회귀 분석, 예측), Classification 문제를 모두 지원하며, 성능과 자원 효율이 매우 우수해서 Boosting 알고리즘에서는 xgBoost가 현재 가장 많이 사용된다.

Input: age, gender, occupation, ...

Like the computer game X



1. xgBoost 결과

Confusion Matrix and Statistics

Reference
Prediction 1 2
 1 102 6
 2 5 57

Accuracy : 0.9353 ————— 정확도

95% CI : (0.8872, 0.9673)

No Information Rate : 0.6294

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8608

Mcnemar's Test P-Value : 1

Sensitivity : 0.9533

Specificity : 0.9048

Pos Pred Value : 0.9444

Neg Pred Value : 0.9194

Prevalence : 0.6294

Detection Rate : 0.6000

Detection Prevalence : 0.6353

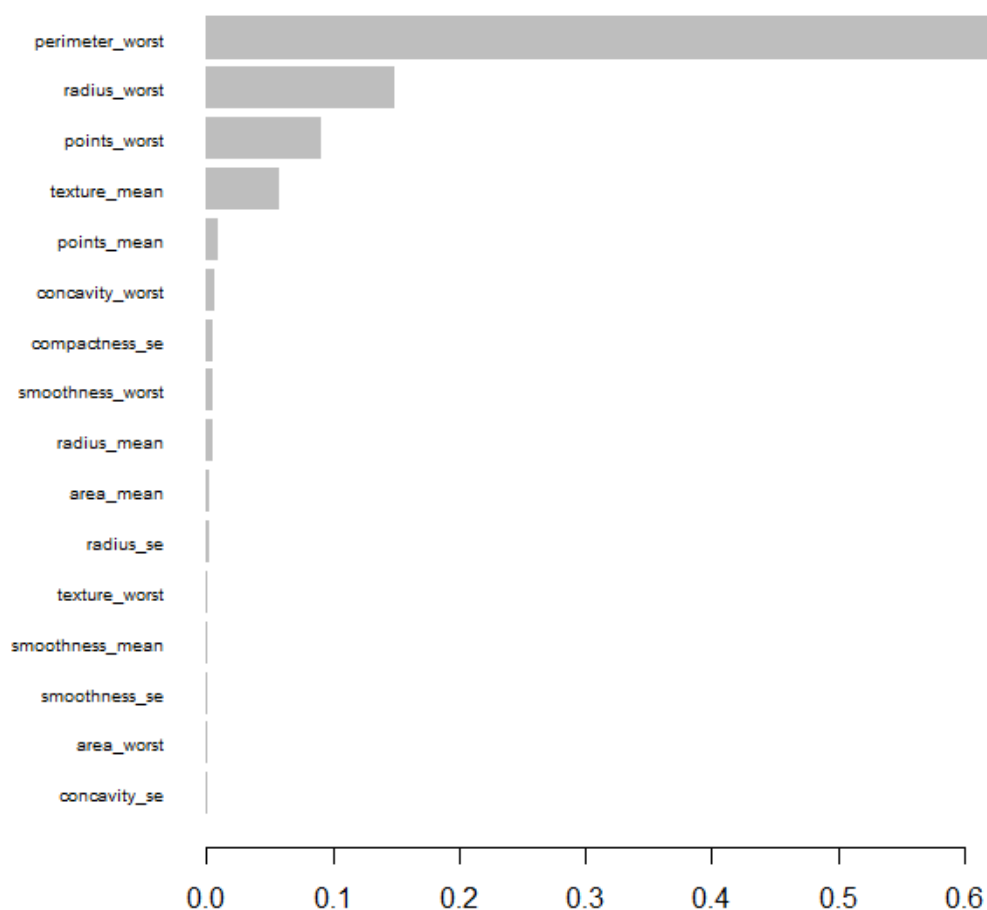
Balanced Accuracy : 0.9290

'Positive' Class : 1

※ xgBoost를 적용하기 위해 변수 레이블이 숫자로 표시 되어
야 하므로 리코딩 되어 있는 변수가 나타난다.

※ 1 = B (양성) // 2 = M (악성)

2. 중요 변수 시각화



xgBoost 분석에서 사용된 중요 변수는 다음과 같다.
perimeter_worst (악성 세포의 둘레)가 우세하게 사용 되었고,
radius_worst (악성의 반경), points_worst (악성의 오목한 점) 순서로 사용 되었다.

9) 분류 결과 비교

나이브 베이즈 // 정확도 : 0.9765

- 일반적인 선형 분류 모델 보다 예측과 훈련 속도가 빠르다.
- 훈련 과정을 이해하기 쉽고 데이터 크기에 영향이 적다.
- 본 데이터는 샘플에 비해 특성이 많은 데이터 이므로 높은 분류 정확도를 보여주었다.

로지스틱 회귀분석 // 정확도 : 0.9706

- 학습속도와 예측이 빠른편이다.
- 훈련 과정 이해가 쉽고 데이터 크기에 영향이 적다.
- 태생적으로 이진 분류만을 지원한다. 하지만 이진분류시 좋은 성능을 보여준다.

의사결정트리 // 정확도 : 0.9176

- 모델을 이해하기 쉽다.
- 훈련 데이터에 과적합 되는 경우가 많다.
- 본 데이터에선 가장 낮은 정확도를 보여주었지만 이는 트리의 깊이 조정으로 발전 가능하다.

인공신경망 // 정확도 : 0.9588

- 복잡한 영역에서도 훌륭한 결과를 도출할 수 있다.
- 범주 및 연속형 변수 모두 다룰 수 있다.
- 본 데이터에선 훈련 셋 분석이 부족했고, 은닉노드의 개수를 여러 번 적합 하지 못하였지만 비교적 높은 정확도를 보여주었다. 이를 개선하면 더 높은 정확도를 기대할 수 있다.

SVM // 정확도 : 0.9765

- 데이터의 특성이 적더라도 높은 정확도를 보여준다.
- 모델 분석이 어렵고 예측의 과정을 파악하기 어렵다. (블랙박스 특성)
- 본 데이터 분석 모델 중 가장 높은 정확도를 보여주었다.

랜덤포레스트 // 정확도 : 0.9529

- 의사결정트리에 과적합 오류를 개선한 모델로 더욱 안정적인 성능을 보인다.
- 많은 메모리와 긴 훈련 시간이 단점이다.
- 본 데이터에선 500개의 의사결정트리를 앙상블한 모형으로 높은 정확도를 보여주었다.

xgBoost // 정확도 : 0.9353

- 랜덤포레스트 보다 배개변수 설정에 더 민감하다.
- 얇은 트리를 많이 연결하여 성능을 개선하는 알고리즘이므로 트리를 추가하면 더 높은 정확도를 기대할 수 있다.
- 본 데이터에선 배개변수 설정을 다수 적합 하지 못하였지만 안정적인 분류 정확도를 보여주었다.

Chapter. 2 – 머신러닝 예측기법

1) 데이터 소개

Boston Housing Price Data

Summary

데이터 내용

보스턴 시의 주택 가격에 대한 데이터

주택의 여러가지 조건들과 주택의 가격 정보가 포함되어 있다. 주택의 가격에 영향을 미치는 요소를 분석하고자 하는 목적으로 사용될 수 있다. 회귀분석 등의 분석에 활용될 수 있다.

보스턴 주택 데이터는 여러 개의 측정지표들 (예를 들어, 범죄율, 학생/교사 비율 등)을 포함한, 보스턴 인근의 주택 가격의 중앙값(median value)이다. 이 데이터 집합은 14개의 변수를 포함하고 있다.

필드의 이해

데이터의 이해를 돕기 위해 포함된 14개의 변수에 대하여 간략하게 설명한다.

위 14개의 필드는 입력 변수로 사용되고, 맨 아래의 MEDV 속성이 목표(종속) 변수로 사용된다.

[01]	CRIM	자치시(town) 별 1인당 범죄율
[02]	ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03]	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04]	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05]	NOX	10ppm 당 농축 일산화질소
[06]	RM	주택 1가구당 평균 방의 개수
[07]	AGE	1940년 이전에 건축된 소유주택의 비율
[08]	DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09]	RAD	방사형 도로까지의 접근성 지수
[10]	TAX	10,000 달러 당 재산세를
[11]	PTRATIO	자치시(town)별 학생/교사 비율
[12]	B	$1000(Bk-0.63)^2$, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13]	LSTAT	모집단의 하위계층의 비율(%)
[14]	MEDV	본인 소유의 주택가격(중앙값) (단위: \$1,000)

2) 다중회귀분석

1. 다중회귀 분석을 하기 위한 전처리 과정

다중회귀분석시 높은 정확도와 과대적합 또는 과소적합을 방지하기 위한 전처리를 한다.
또한 본 데이터가 회귀분석에 적합한지에 대한 초기 검정을 실시한다.
전처리와 초기 검정에 과정은 다음과 같이 진행하였다.

1. 회귀분석의 타당성 검정
2. 변수선택법
3. 잔차 독립성 검정
4. 등분산성 확인

① 회귀 분석의 타당성 검정

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	40.050541	5.851782	6.844	3.60e-11	***
crim	-0.106995	0.038370	-2.789	0.00559	**
zn	0.051474	0.016504	3.119	0.00197	**
indus	0.063534	0.072528	0.876	0.38165	
chas1	2.518270	1.073381	2.346	0.01954	*
nox	-18.389451	4.530458	-4.059	6.12e-05	***
rm	2.973357	0.480871	6.183	1.80e-09	***
age	0.025817	0.016230	1.591	0.11262	
dis	-1.293557	0.246288	-5.252	2.66e-07	***
rad	0.323554	0.077821	4.158	4.07e-05	***
tax	-0.012151	0.004398	-2.763	0.00604	**
ptratio	-0.947835	0.157479	-6.019	4.55e-09	***
b	0.009616	0.003454	2.784	0.00567	**
lstat	-0.649187	0.059235	-10.959	< 2e-16	***

Signif. codes:

0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.755 on 340 degrees of freedom

Multiple R-squared: 0.7317, Adjusted R-squared: 0.7215

F-statistic: 71.33 on 13 and 340 DF, p-value: < 2.2e-16

- ※ 검정 결과 R 값은 0.7215, F 통계량 71.33으로 타당성은 양호하다.
- ※ 하지만 계수들 중 일부 유의하지 않은 것으로 분석 된다. (분석 결과에 빨간 줄)
- ※ 따라서 변수 선택법을 통하여 다시 검정을 실시한다.

② 변수 선택법

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	39.756827	5.840175	6.807	4.49e-11	***
crim	-0.108098	0.038336	-2.820	0.00509	**
zn	0.049728	0.016377	3.036	0.00258	**
nox	-17.348312	4.370288	-3.970	8.78e-05	***
rm	2.943053	0.479462	6.138	2.32e-09	***
dis	-1.333770	0.241889	-5.514	6.94e-08	***
rad	0.304161	0.074580	4.078	5.65e-05	***
tax	-0.010422	0.003929	-2.653	0.00836	**
ptratio	-0.933993	0.156631	-5.963	6.19e-09	***
b	0.009448	0.003448	2.740	0.00646	**
lstat	-0.646629	0.059143	-10.933	< 2e-16	***

Signif. codes:

0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.753 on 341 degrees of freedom

Multiple R-squared: 0.7311, Adjusted R-squared: 0.7216

F-statistic: 77.26 on 12 and 341 DF, p-value: < 2.2e-16

※ 변수 선택 결과 R 값은 0.7216, F 통계량 77.26으로 타당성은 증가 하였다.

※ 모든 계수가 유의하게 나타나므로 예측 분석을 시행한다.

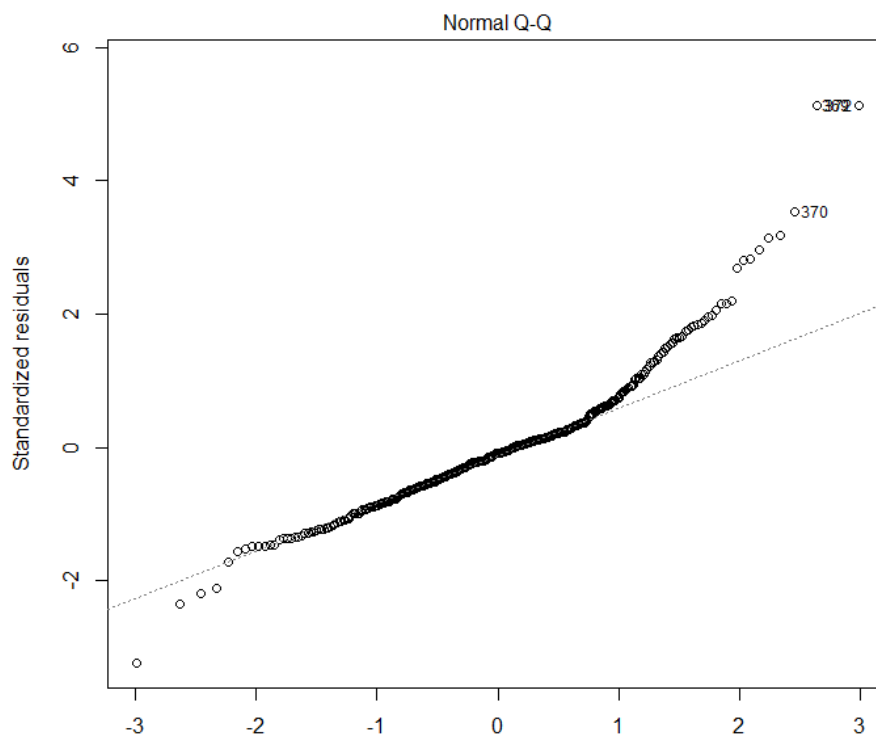
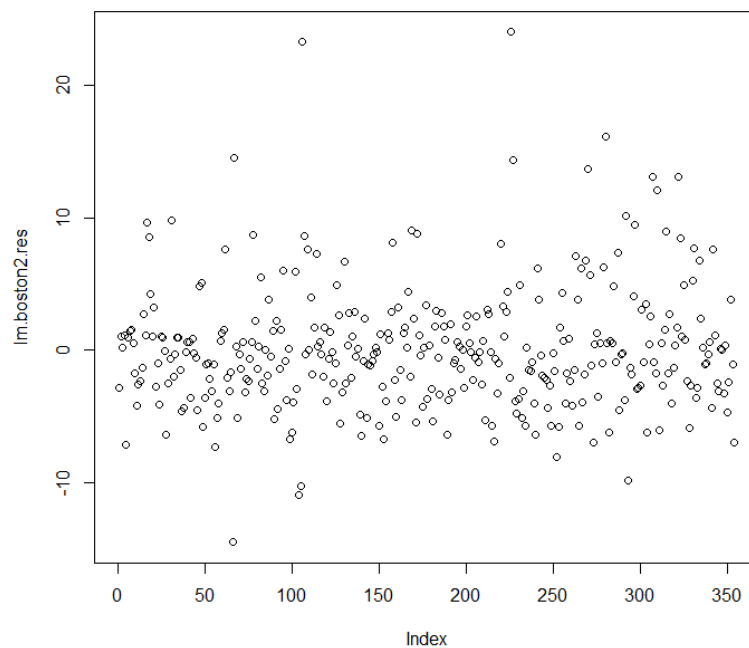
③ 잔차 독립성 검정

DurbinWaton 검정 결과 : 1.9735

※ 독립성 상한인 1.69 보다 크므로 계수들 서로 독립적이다

④ 등분산성 확인

등분산성을 확인하기 위해 산점도와 Qqplot을 이용한다.



※ 잔차의 산점도와 QQ플롯을 확인한 결과 등분산성이 있다고 보기 어렵지만 예측 정확도를 확인 하기 위해 다중회귀분석을 계속 진행 하였다.

다중회귀분석 적용 결과

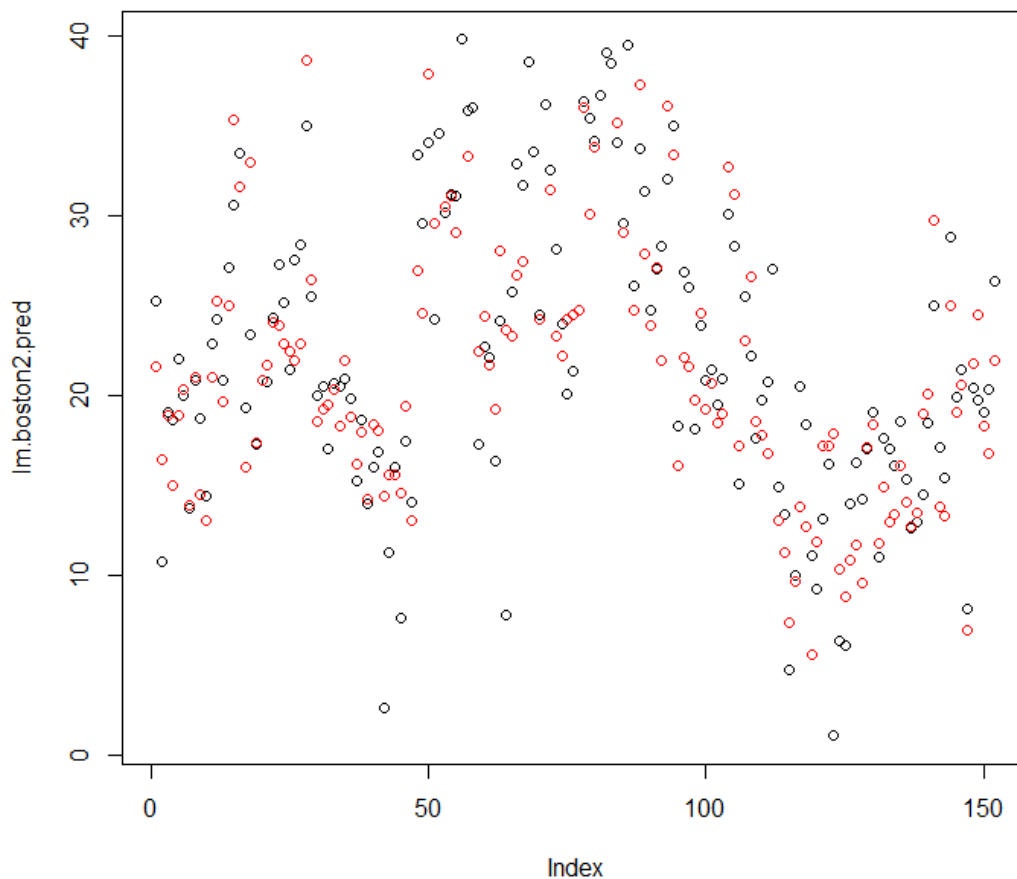
2. 상관계수와 평균제곱오차(MSE)를 통한 예측 결과

정확도 : 0.8647503

MSE : 23.73715 (제공근 : 4.872079)

※ 실제 주택 가격은 \$4,872 이내로 예상 된다.

3. 예측 값과 실제 값의 비교 시각화



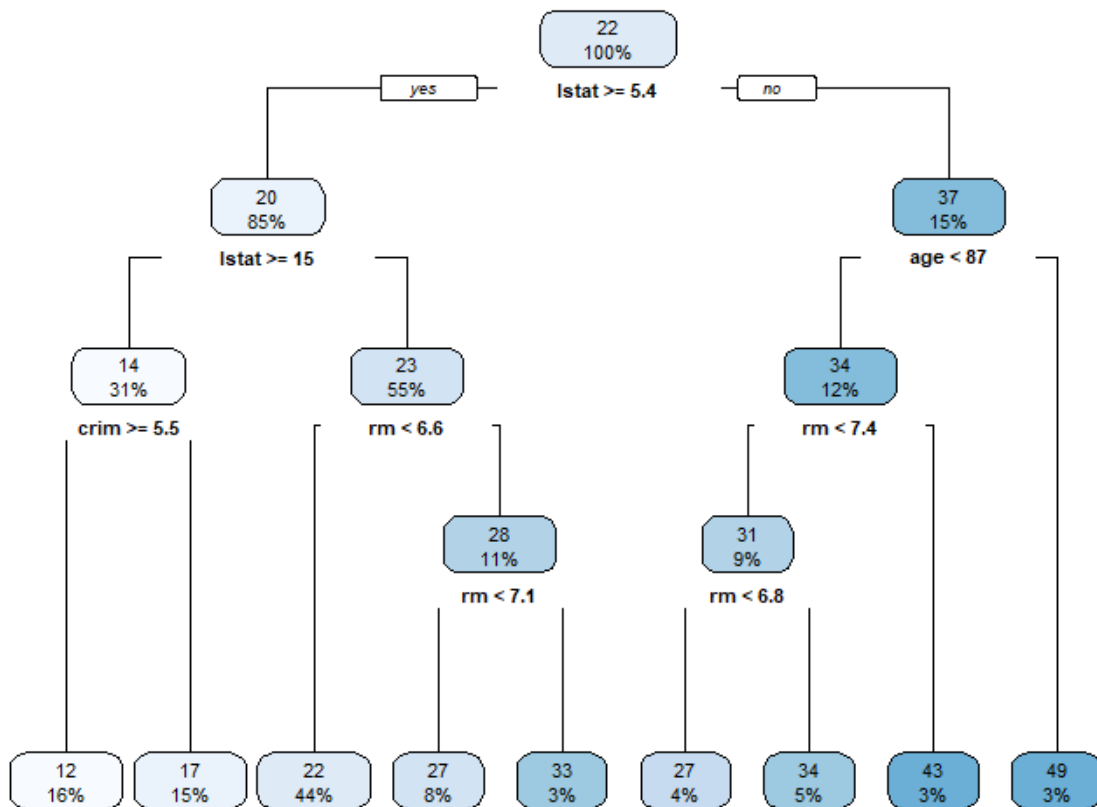
Black points -> 예상 값

Red points -> 실제 값

3) 의사결정트리

1. 의사결정트리 시각화

Rpart.Plot을 활용하여 의사결정트리 시각화 결과



※ 시각화 결과 : lstat(모 집단의 하위 계층 비율) 변수가 처음 가지를 구성했고 이후 중요 변수로는 Age(1940년 이전에 건축된 주택 비율), rm(1가구당 평균 방의 개수), crim(1인당 범죄율)이 분할 변수로 사용 되었다.

의사결정트리 적용 결과

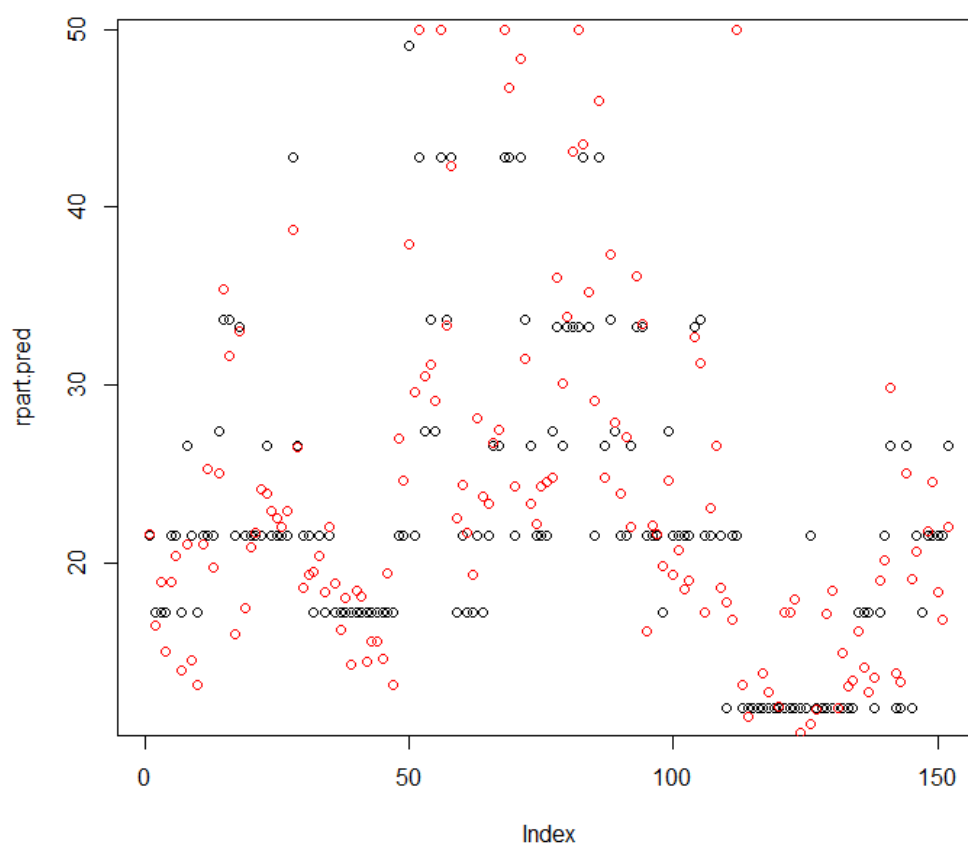
2. 상관계수와 평균제곱오차(MSE)를 통한 예측 결과

정확도 : 0.8825783

MSE : 20.68817 (제곱근 : 4.548425)

※ 실제 주택 가격은 \$4,548 이내로 예상 된다.

3. 예측 값과 실제 값의 비교 시각화



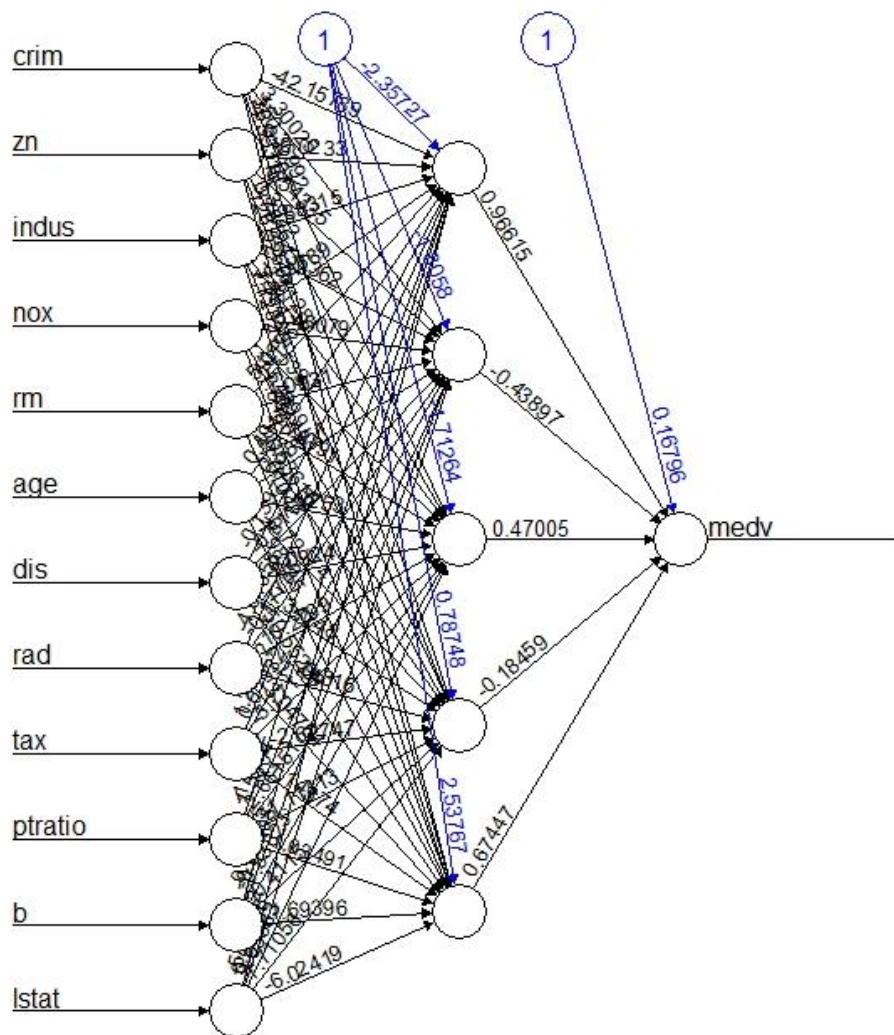
Black points -> 예상 값

Red points -> 실제 값

4) 인공신경망

1. 인공신경망 시각화

Plot을 활용하여 인공신경망 Network 시각화 결과



※ 12개의 각 변수들이 5개의 은닉층을 거쳐서 medv(주택 가격) 종속변수로 출력 되는 Network 알고리즘을 확인 할 수 있다.

인공신경망 적용 결과

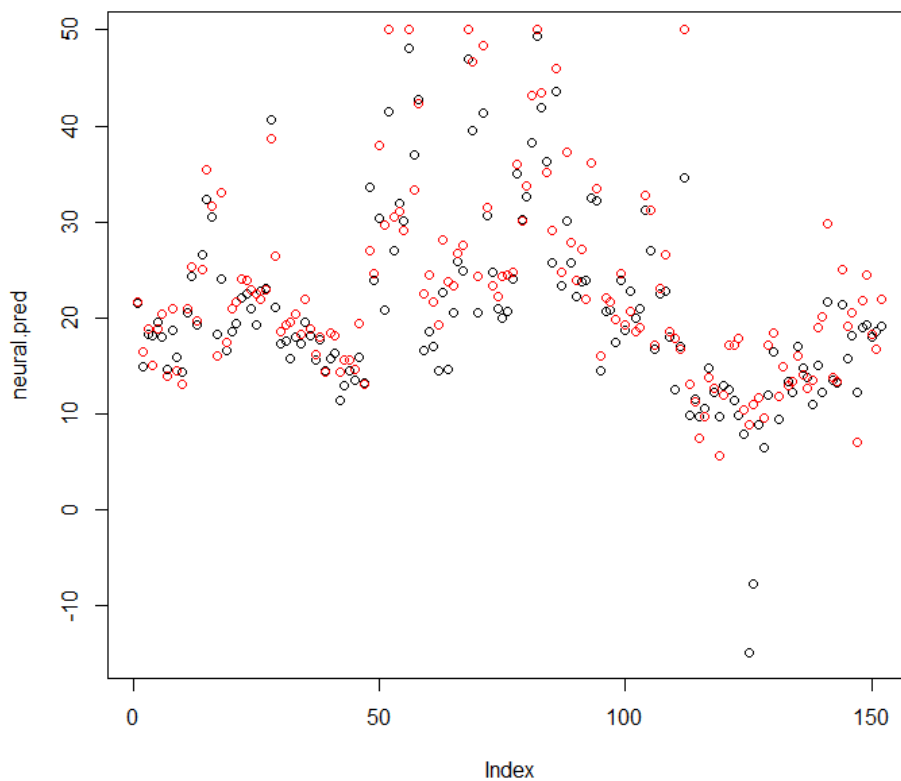
2. 상관계수와 평균제곱오차(MSE)를 통한 예측 결과

정확도 : 0.9244139

MSE : 18.41591 (제공근 : 4.291377) → 역 정규화 계산을 통해 원래 값 복원

※ 실제 주택 가격은 \$4,291 이내로 예상 된다.

3. 예측 값과 실제 값의 비교 시각화



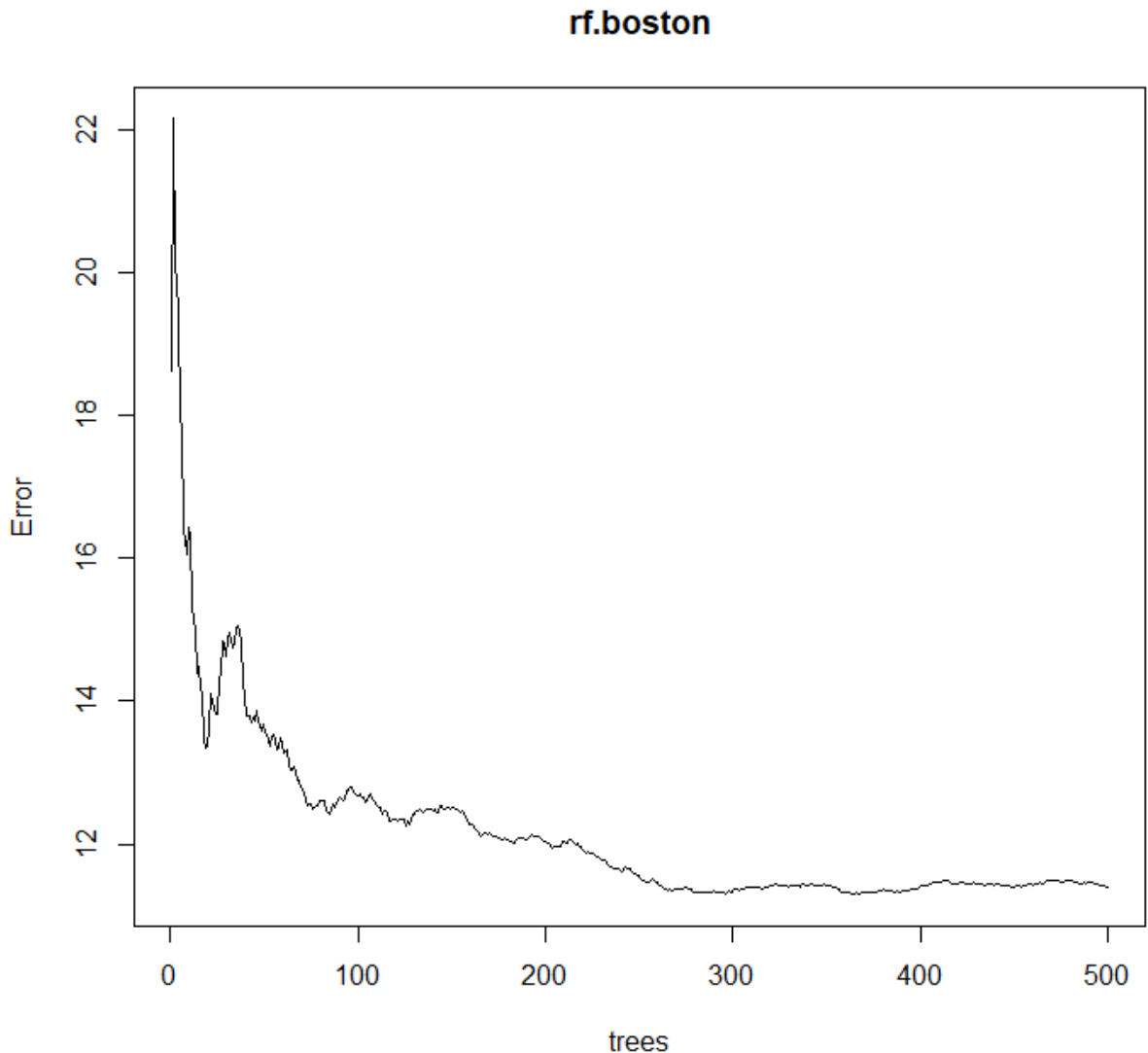
Black points -> 예상 값

Red points -> 실제 값

5) 랜덤포레스트

1. 랜덤포레스트 시각화

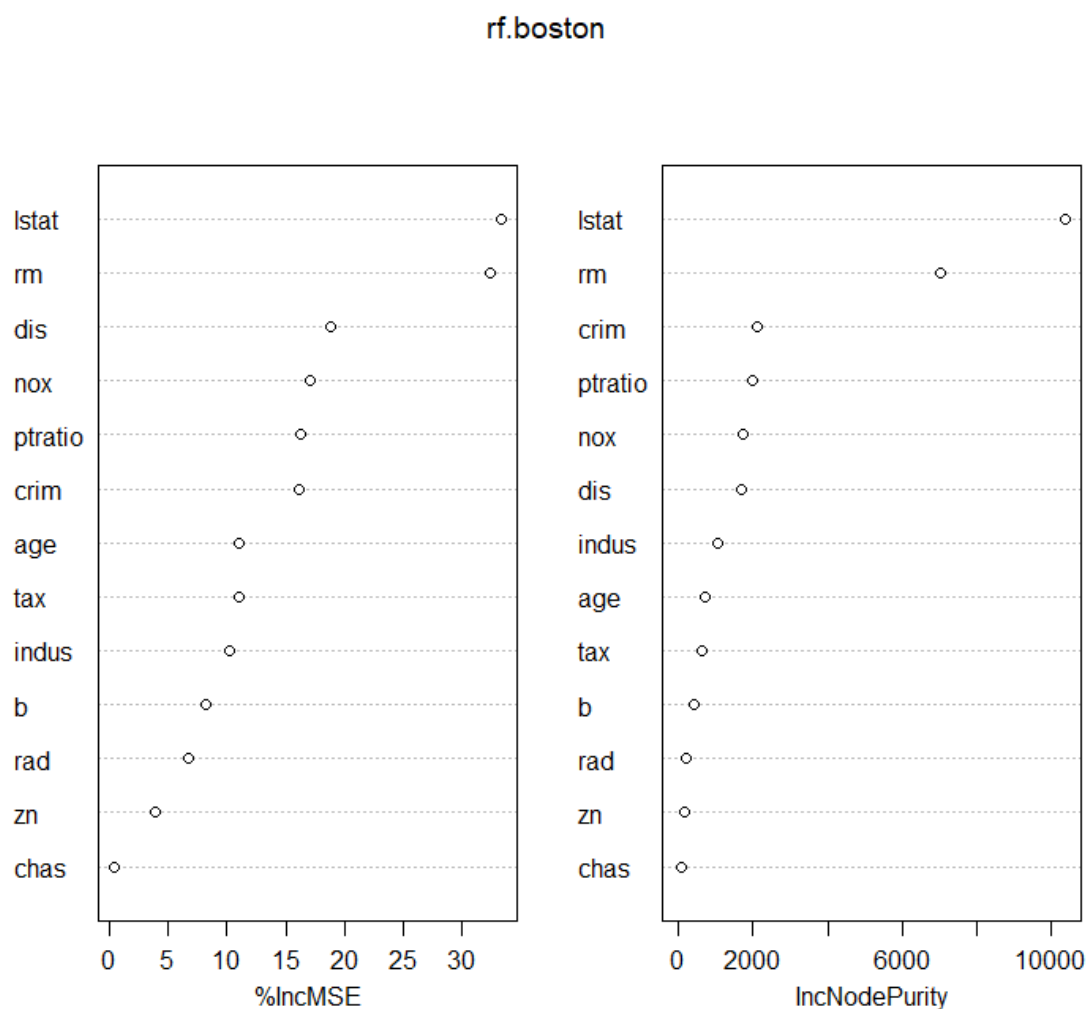
트리 개수 변화에 따른 오류 감소 추이



- ※ 랜덤포레스트는 기존 의사결정트리와 앙상블의 배깅 알고리즘을 개선하여 더 좋은 성능을 내는 머신러닝 분석 방법이다. 하지만 하이퍼파라미터 지정에 따른 성능의 변화가 심하여 파라미터 지정에 신경 써줘야 한다.
- ※ 본 분석에서 사용된 파라미터는 다음과 같다.
 1. Mtry(트리의 노드 결정시 설명변수의 후보 개수) : 5
 - Mtry의 개수는 일반적으로 $m = p/3$ (p는 설명변수 개수) 이다.
 2. 총 연산에 참가할 트리 생성 수 : 500

랜덤포레스트 적용 결과

2. 변수 중요도 시각화



※ 본 랜덤포레스트 분석시 사용된 변수의 중요도는 위와 같다.

lstat(모 집단의 하위 계층 비율) 변수가 가장 높은 비중을 차지하고 있는 변수이고 두 번째 변수는 rm(1가구당 평균 방의 개수)이다.

이 상위 두가지 변수가 분석에 압도적인 중요성을 보여주고 있다.

랜덤포레스트 적용 결과

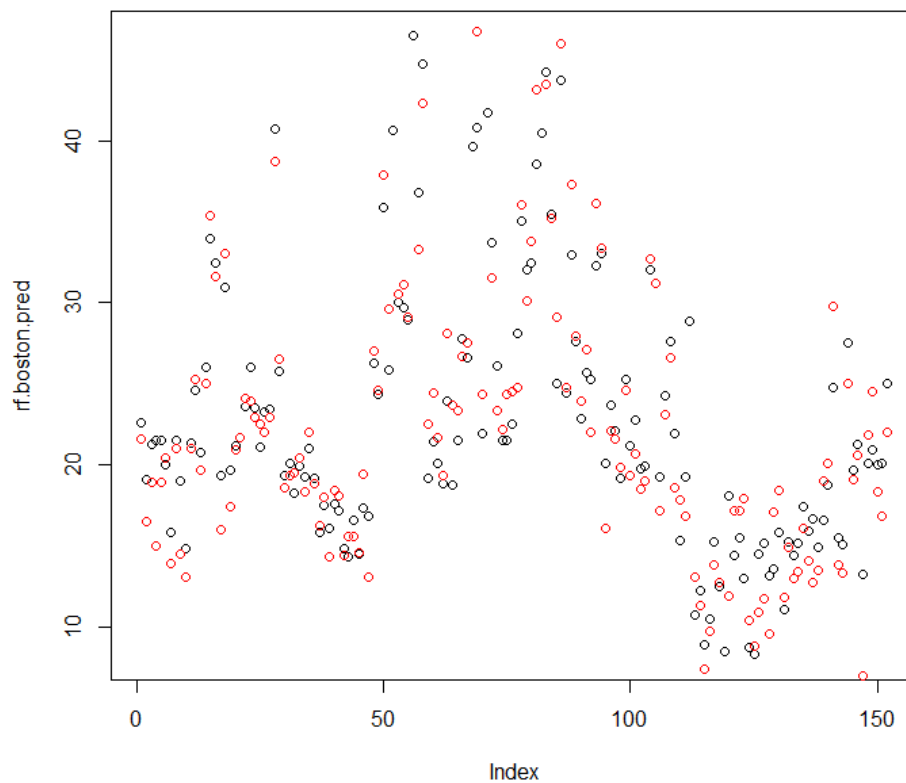
3. 상관계수와 평균제곱오차(MSE)를 통한 예측 결과

정확도 : 0.9474494

MSE : 10.53445 (제곱근 : 3.245682)

※ 실제 주택 가격은 \$3,245 이내로 예상 된다.

4. 예측 값과 실제 값의 비교 시각화



Black points -> 예상 값

Red points -> 실제 값

6) 분류 결과 비교

다중회귀분석

- 정확도 : 0.8647503
- MSE : 23.73715 // MSE제곱근 : 4.872079
- 결과 : 실제 주택 가격의 오차는 \$4,872 이내이다.

의사결정트리

- 정확도 : 0.8825783
- MSE : 20.68817 // MSE제곱근 : 4.548425
- 결과 : 실제 주택 가격의 오차는 \$4,548 이내이다.

인공신경망

- 정확도 : 0.9244139
- MSE : 18.41591 // MSE제곱근 : 4.291377
- 결과 : 실제 주택 가격의 오차는 \$4,291 이내이다.

랜덤포레스트

- 정확도 : 0.9474494
- MSE : 10.41591 // MSE제곱근 : 3.245682
- 결과 : 실제 주택 가격의 오차는 \$3,245 이내이다.

※ 분석 총 평

1. 예측 성능은 내림차순으로 다음과 같다.
랜덤포레스트 → 인공신경망 → 의사결정트리 → 다중회귀분석
2. 랜덤포레스트의 경우 높은 예측 정확도를 보여주었지만 하이퍼파라미터 지정에 예민하다.
하지만 그렇기 때문에 미세한 하이퍼파라미터 조정으로 성능 발전을 기대할 수 있다.
3. 인공신경망 역시 높은 예측 정확도를 보여주었지만 은닉층 재설정과 파라미터 재지정과 반복 학습을 통해 더 나은 성능을 기대해 볼 수 있다.
4. 의사결정트리의 경우 예측 모델로도 사용 가능하지만 연속형 범주의 변수에는 적합하지 않다.
그렇기에 다른 분석 기법들에 비해 좋은 예측률을 보여주진 못했다.
5. 다중회귀분석의 경우 본 데이터에 적합한 방식의 모델은 아닌 것으로 판단 되어진다.
그 이유로는 변수들의 독립성은 보장되어 지나 정규성, 선형성에 있어 유의확률을 만족 시킬 수 없었기 때문이다. 따라서 가장 낮은 예측 정확도를 확인할 수 있다.

Chapter. 3 – 참고사항

1. 분류 기법에 사용된 R코드

```
### 1번. (분류기법 적용)

# 데이터 준비
raw.wisc <- read.csv("/rwork/wisc_bc_data.csv", header = T)
str(raw.wisc)
head(raw.wisc)

# 훈련, 테스트 데이터 준비
library(caret)
library(pROC)
library(RColorBrewer)

idx <- createDataPartition(raw.wisc$diagnosis, p = 0.7, list = F)
wisc.train <- raw.wisc[idx, ]
wisc.test <- raw.wisc[-idx, ]
table(wisc.train$diagnosis)
table(wisc.test$diagnosis)

## 1) 나이브 베이즈 모형
library(e1071)

naive.model <- naiveBayes(wisc.train, wisc.train$diagnosis, laplace = 1) # 모델링
naive.pred <- predict(naive.model, wisc.test, type = "class") # 모델로 예측

nc <- confusionMatrix(naive.pred, wisc.test$diagnosis)
nc
# 혼돈 매트릭스 결과 확인
#! 결과 : Accuracy : 0.9941

# 시각화
roc.naive <- roc(as.numeric(naive.pred), as.numeric(wisc.test$diagnosis))
plot.roc(roc.naive, col = "red", print.auc = T,
         max.auc.polygon = T, print.thres = T,
         print.thres.pch = 19, print.thres.col = "red",
         auc.polygon = T, auc.polygon.col = "#D1F2EB")

plot(nc$table, col = brewer.pal(2, "Accent"))

## 2) 로지스틱 회귀분석
library(nnet) # 다항 로지스틱 회귀를 위한 패키지

logit.model <- multinom(diagnosis ~ ., data = wisc.train) # 훈련 데이터 모형 적합
logit.pred <- predict(logit.model, wisc.test) # 테스트 데이터를 이용한 평가
lc <- confusionMatrix(logit.pred, wisc.test$diagnosis)
lc
# 분류 결과
#! 정확도 : 0.9647

# 시각화
roc.logit <- roc(as.numeric(logit.pred), as.numeric(wisc.test$diagnosis))
plot.roc(roc.logit, col = "red", print.auc = T,
         max.auc.polygon = T, print.thres = T,
         print.thres.pch = 19, print.thres.col = "red",
         auc.polygon = T, auc.polygon.col = "#D1F2EB")

plot(lc$table, col = brewer.pal(2, "Pastel2"))
```

```

## 3) 의사결정트리
library(rpart) # 의사결정트리를 사용하기 위한 패키지
library(rpart.plot)

rpart.model <- rpart(diagnosis ~ ., data = wisc.train) # 훈련 모형 적합
rpart.plot(rpart.model) # 트리 시각화

rpart.pred <- predict(rpart.model, wisc.test, type = "class") # 평가

rc <- confusionMatrix(rpart.pred, wisc.test$diagnosis)
rc
#! 정확도 : 0.9471

# 시각화
roc.rpart <- roc(as.numeric(rpart.pred), as.numeric(wisc.test$diagnosis))
plot.roc(roc.rpart, col = "red", print.auc = T,
         max.auc.polygon = T, print.thres = T,
         print.thres.pch = 19, print.thres.col = "red",
         auc.polygon = T, auc.polygon.col="#D1F2EB")

plot(rc$table, col = brewer.pal(2, "Set1"))

## 4) 인공 신경망 기법 (nnet)
library(nnet)

# 데이터 표준화
wisc.train.scale <- as.data.frame(sapply(wisc.train[, -2], scale))
wisc.test.scale <- as.data.frame(sapply(wisc.test[, -2], scale))

wisc.train.scale$diagnosis <- wisc.train$diagnosis
wisc.test.scale$diagnosis <- wisc.test$diagnosis

# 훈련데이터로 모형 적합
wisc.nnet <- nnet(diagnosis ~., wisc.train.scale, size = 3)
wisc.nnet.pred <- predict(wisc.nnet, wisc.test.scale, type = "class")

nnc <- confusionMatrix(as.factor(wisc.nnet.pred), wisc.test$diagnosis)
nnc

# 시각화
roc.nnc <- roc(as.numeric(as.factor(wisc.nnet.pred)), as.numeric(wisc.test$diagnosis))
plot.roc(roc.nnc, col = "red", print.auc = T,
         max.auc.polygon = T, print.thres = T,
         print.thres.pch = 19, print.thres.col = "red",
         auc.polygon = T, auc.polygon.col="#D1F2EB")

plot(nnc$table, col = brewer.pal(2, "Dark2"))

```

```

## 5)SVM
library(kernlab)

# 분석 실행
wisc.svm <- ksvm(diagnosis ~ ., wisc.train, kernel = "rbfdot")

# 예측 및 결과
wisc.svm.pred <- predict(wisc.svm, wisc.test, type = "response")
sc <- confusionMatrix(wisc.svm.pred, wisc.test$diagnosis)
sc

# 시각화
roc.svm <- roc(as.numeric(wisc.svm.pred), as.numeric(wisc.test$diagnosis))
plot.roc(roc.rpart, col = "red", print.auc = T,
         max.auc.polygon = T, print.thres = T,
         print.thres.pch = 19, print.thres.col = "red",
         auc.polygon = T, auc.polygon.col="#D1F2EB")

plot(sc$table, col = brewer.pal(2, "Set3"))

## 6)양상불 (배깅 - 랜덤 포레스트)
library(randomForest)

# 분석 실행
wisc.rf <- randomForest(diagnosis ~ ., wisc.test,
                       importance = T) # ntree = 500, mtry = 2
wisc.rf # 랜덤포레스트 분석 결과

# 중요 변수 시각화
importance(wisc.rf)
varImpPlot(wisc.rf)

## 7)양상불 (부스팅 -xgboost)
library(xgboost)

# 변수 리코딩 - xgboost는 변수 레이블이 숫자로 표시 되어야 한다.
wisc.train$diagnosis2[wisc.train$diagnosis == 'B'] <- 1
wisc.train$diagnosis2[wisc.train$diagnosis == 'M'] <- 2

wisc.test$diagnosis2[wisc.test$diagnosis == 'B'] <- 1
wisc.test$diagnosis2[wisc.test$diagnosis == 'M'] <- 2

# 매트릭스로 변환
wisc.train.mat <- as.matrix(wisc.train[-c(1, 2, 33)])
wisc.train.mat
train.label <- wisc.train$diagnosis2

wisc.test.mat <- as.matrix(wisc.test[-c(1, 2, 33)])
wisc.test.mat
test.label <- as.factor(wisc.test$diagnosis2)

# 모델링
mat.train <- xgb.DMatrix(data = wisc.train.mat, label = train.label)
mat.train

xgb.model <- xgboost(data = mat.train, max_depth = 6,
                    eta = 0.3, nthread = 2, nrounds = 2,
                    objective = "multi:softmax",
                    num_class = 3, verbose = 0)

# 예측 및 결과
xgb.pred <- predict(xgb.model, wisc.test.mat)
confusionMatrix(as.factor(xgb.pred), test.label)

# 주요 변수 확인 및 시각화
importance.xgb <- xgb.importance(colnames(mat.train),
                                model = xgb.model)

importance.xgb

xgb.plot.importance(importance.xgb)

```

2. 예측 기법에 사용된 R코드

```
# 데이터 준비
library(mlbench)
data("BostonHousing")
head(BostonHousing)

# 훈련, 테스트 데이터 분할
idx <- sample(1 : nrow(BostonHousing), nrow(BostonHousing)
              * 0.7)
boston.train <- BostonHousing[idx, ]
boston.test <- BostonHousing[-idx, ]

dim(boston.train); dim(boston.test)

# 평균제곱오차(MSE) 계산을 위한 함수 정의
mse <- function(x, y){
  cat(mean((x - y) ^ 2), "\n")
  cat(sqrt(mean((x - y) ^ 2)))
}

## 1)다중회귀분석

# 분석 실행
lm.boston <- lm(medv ~ ., data = boston.train)
summary(lm.boston)
#! 다중회귀분석 결과 R값은 0.7089, F통계량 67.14로 양호한 편이다.
#! 그러나 회귀계수를 중 일부 유의하지 않은 것으로 나타난다.
#! 그러므로 변수선택법을 통하여 다시 분석을 실시한다.

# 변수 선택 (STEP)
lm.boston2 <- step(lm.boston, method = "both")
summary(lm.boston2)
#! 변수선택법 결과 R 값은 0.7104, F통계량은 79.7로 증가하였다.
#! 모든 회귀 계수가 유의하게 나타나므로 수치 예측을 시행한다.

# 회귀분석을 위한 잔차 분포 검토

library(car)
lm.boston2.res <- residuals(lm.boston2)
durbinWatsonTest(lm.boston2.res) # 잔차의 독립성 검토
#! 독립성 상한인 1.69보다 큰 1.9735이므로 서로 독립적이다.

plot(lm.boston2) # 등분산성 확인
plot(lm.boston2.res)
#! QQ플롯과 잔차의 산점도를 확인 결과 어느정도 등분산한다고 보기 어렵다.

shapiro.test(lm.boston2.res) # 잔차의 정규성 검토
#! P값이 유의수준 0.05보다 작으므로 잔차의 등분산성과 정규성을 가정할 수 없다.
#! 회귀분석을 시행하기에 등분산성과 정규성을 가정할 수 없지만 수치예측기법
#! 비교를 위해 예측 정확도 확인만 하기위해 예측을 실시 한다.

# 예측 및 결과
lm.boston2.pred <- predict(lm.boston2, newdata = boston.test)

cor(lm.boston2.pred, boston.test$medv)
#! 상관계수를 이용한 분류정확도 검토 결과 : 0.8859837

mse(lm.boston2.pred, boston.test$medv) # MSE 값 계산
#! mse값은 18.93024, 제공된 값 4.3508로 실제 주택 가격은 $4,350 이내로 예상

# 시각화
plot(lm.boston2.pred)
points(boston.test$medv, col = "red")
```

```

## 2)의사결정트리
library(rpart)
library(rpart.plot)

rpart.boston <- rpart(medv ~ ., data = boston.train)
summary(rpart.boston)

rpart.plot(rpart.boston)
#! 시각화 결과 lstat가 영향력이 큰 변수로 첫 번째 분할 변수로 사용되었다.

# 예측 및 결과
rpart.pred <- predict(rpart.boston, newdata = boston.test)

cor(rpart.pred, boston.test$medv)
#! 상관계수를 이용한 분류정확도 검정 결과 : 0.894273

mse(rpart.pred, boston.test$medv)
#! mse값은 17.62551, 제공된 값 4.1982로 실제 주택 가격은 $4,198 이내에 있다고 예상

# 시각화
plot(rpart.pred)
points(boston.test$medv, col = "red")

## 3)인공신경망
library(neuralnet)

# 정규화
normalize <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}
#! BostonHousing 데이터는 각 변수들의 단위와 차이가 크므로 표준화가 아닌 정규화를 실시 한다.

boston.train.norm <- as.data.frame(sapply(boston.train[-4], normalize))
boston.test.norm <- as.data.frame(sapply(boston.test[-4], normalize)) # factor형 변수 생략

# 인공신경망 분석 실행
neural.boston <- neuralnet(medv ~ crim + zn + indus +
                           nox + rm + age +
                           dis + rad + tax + ptratio +
                           b + lstat,
                           data = boston.train.norm,
                           hidden =5)

plot(neural.boston) # 인공신경망 시각화

# 모델 성능평가 및 예측
neural.boston.result <- compute(neural.boston, boston.test.norm[1:12])

neural.yhat <- neural.boston.result$net.result

cor(neural.yhat, boston.test.norm$medv)
#! 상관계수를 이용한 분류정확도 검정 결과 : 0.9237873

# 정규화를 역계산 후 MSE 값 구하기
neural.pred <- neural.yhat * (max(BostonHousing$medv) - min(BostonHousing$medv)) +
min(BostonHousing$medv)

mse(neural.pred, as.matrix(boston.test$medv))

# 시각화
plot(neural.pred)
points(boston.test$medv, col = "red")

```



```
## 4)랜덤포레스트
library(randomForest)

# 랜덤포레스트 분석 실행
rf.boston <- randomForest(medv ~ ., data = boston.train,
                           mtry = 5, importance = T)
#! mtry 인자는 일반적으로  $m=p/3$  (p는 설명변수)를 사용한다. 여기선 13개의 설명변수를 사용하므로 mtry
값을 5로 설정한다.
plot(rf.boston) # 트리 개수 변화에 따른 오류 감소 추이

# 변수 중요도
importance(rf.boston) # 변수 중요도 확인
varImpPlot(rf.boston) # 변수 중요도 플로팅
#! 변수 중요도 확인 결과 rm과 lstat이 다른 설명변수들에 비해 압도적으로 높은 중요도를 보인다.

# 예측 및 결과
rf.boston.pred <- predict(rf.boston, newdata = boston.test)

cor(rf.boston.pred, boston.test$medv)
#! 상관계수를 이용한 분류정확도 검정 결과 : 0.9462929

mse(rf.boston.pred, boston.test$medv)
#! MSE값은 9.237684로 이전 분석을 보다 오차율이 개선 되었다. 제공근은 3.039로 실제 주택가격 값의 약
$3.039 이내에 있음을 알 수 있다.

# 시각화
plot(rf.boston.pred)
points(boston.test$medv, col = "red")
```

Chapter. 4 – clustering 기법

1) K-means (k-평균값 군집)

A. 개념

k-평균 알고리즘(K-means clustering algorithm)은 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작하여 군집을 이루는 것

B. 특징

클러스터 개수 k값을 입력 파라미터로 지정해주어야 한다.

알고리즘의 에러 수렴이 전역 최솟값이 아닌 지역 최솟값으로 수렴할 가능성이 있다.

이상값 (outlier) 에 민감하다.

구형 (spherical) 이 아닌 클러스터를 찾는 데에는 적절하지 않다.

C. 절차

초기 K 평균값은 데이터 오브젝트 중에서 무작위로 뽑는다.

k 각 데이터 오브젝트들은 가장 가까이 있는 평균 값을 기준으로 묶인다. 평균값을 기준으로 분할된 영역은 보로노이 다이어그램으로 표시된다.

k 개의 클러스터의 중심점을 기준으로 평균값이 재조정된다.

수렴할때 까지 단계 2,3 을 반복 실행한다.

D. R 수행

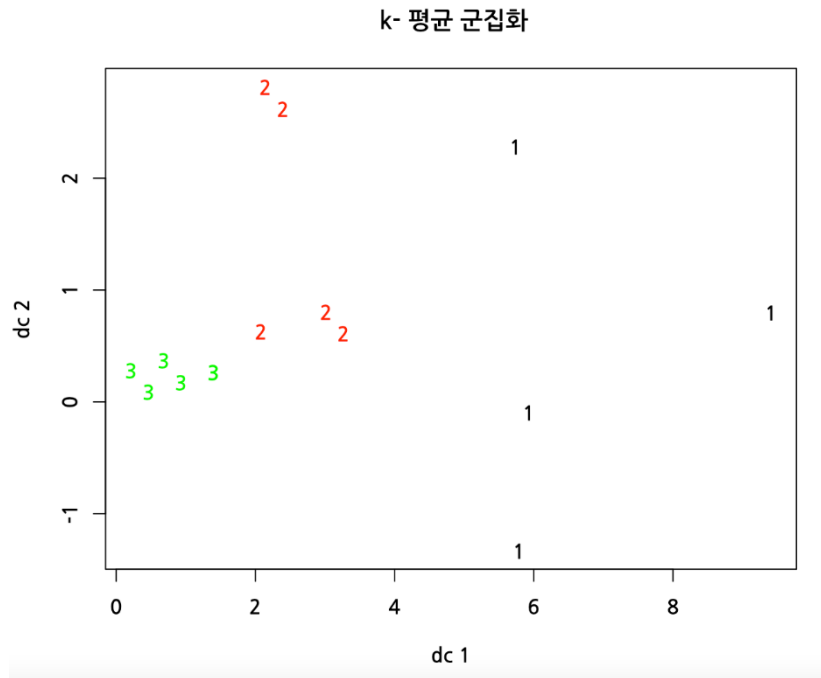
Package 라이브러리도 필요없이 kmeans()함수를 사용 하면 된다.

함수

```
kmeans(x, centers, iter.max = 10, nstart = 1,  
algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",  
"MacQueen"), trace=FALSE)
```

```
> member <- data.frame( spent = c(10,1,1,17,4,6,1,15,22,3,0,3,7,0,2),  
+                         time = c(15,2,10,7,5,7,1,10,18,3,1,3,7,10,2))  
>  
> result <- kmeans(member, 3)  
> result  
>  
> install.packages('fpc')  
> library(fpc)  
> plotcluster(member, result$cluster, color=TRUE, shade=TRUE , main = ' k- 평균 군집  
,family="NanumGothicBold" )  
>
```

시각화 결과



2) K-medoids (k-중앙값 군집)

A. 개념

각 군집이 군집의 중앙(중앙값)에 가장 가깝게 위치해 있는 객체 중 하나로 대표될 때, k-medoids 알고리즘을 적용하여 군집을 이루는 것

k-medoids는 평균을 정의하거나 해석할 수 없는 범주형 데이터를 클러스터링하는 데 유용하게 사용하는 군집화 중 하나이다.

B. 특징

이해하기 쉽고 구현하기 쉽다.

K-Medoid 알고리즘은 빠르고 고정된 수의 단계로 수렴한다.

PAM은 다른 파티셔닝 알고리즘보다 특이치에 덜 민감하다.

K-Medoid 알고리즘의 가장 큰 단점은 비구형 (임의의 모양) 개체 그룹을 클러스터링하는 데 적합하지 않다는 것이다. 그 이유는 비정형 물체와 중형 (클러스터 중심) 사이의 거리를 최소화하는 데 의존하기 때문이다. 간단히 말해서 연결 대신 클러스터링 기준으로 압축을 사용한다.

처음 k 개의 medoid가 무작위로 선택되기 때문에 동일한 데이터 세트에서 다른 실행에 대해 다른 결과를 얻을 수 있다.

C. 절차

n 개의 데이터 포인트 중 k 개의 무작위 포인트를 medoid로 선택

일반적인 거리 측정 방법을 사용하여 각 데이터 포인트를 가장 가까운 medoid에 연결
medoid가 변경되지 않거나 다른 종료 기준이 충족될 때까지 빌드 및 스왑 단계를 반복

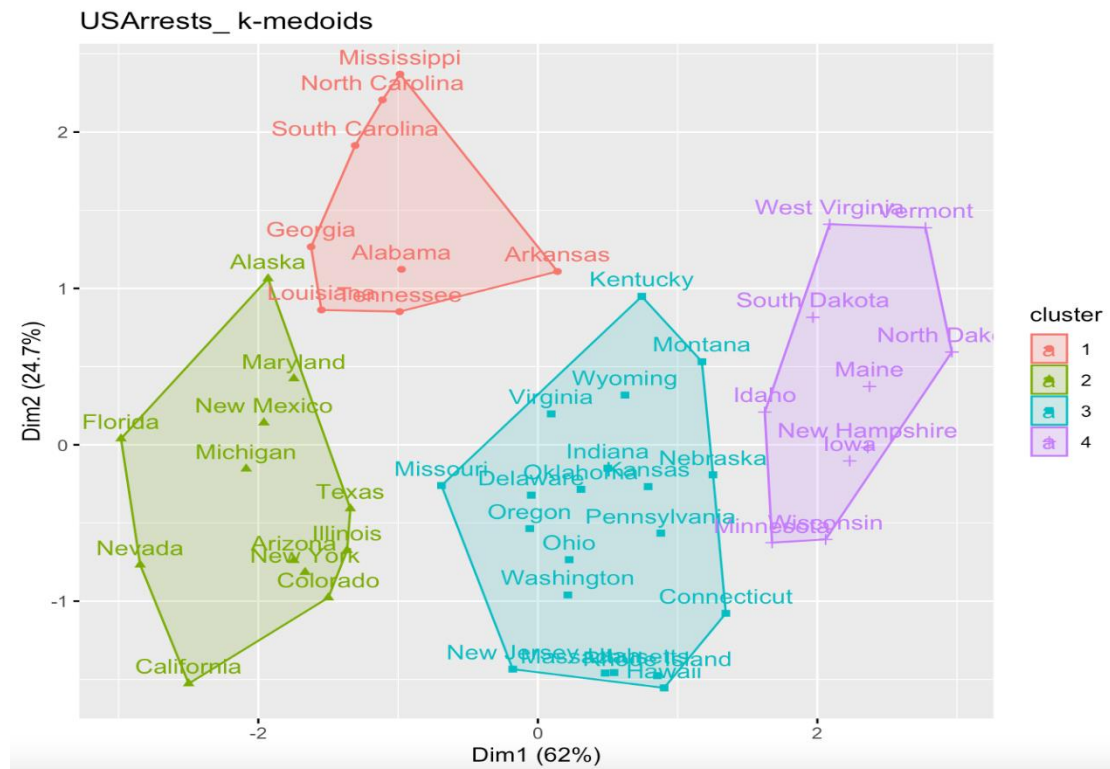
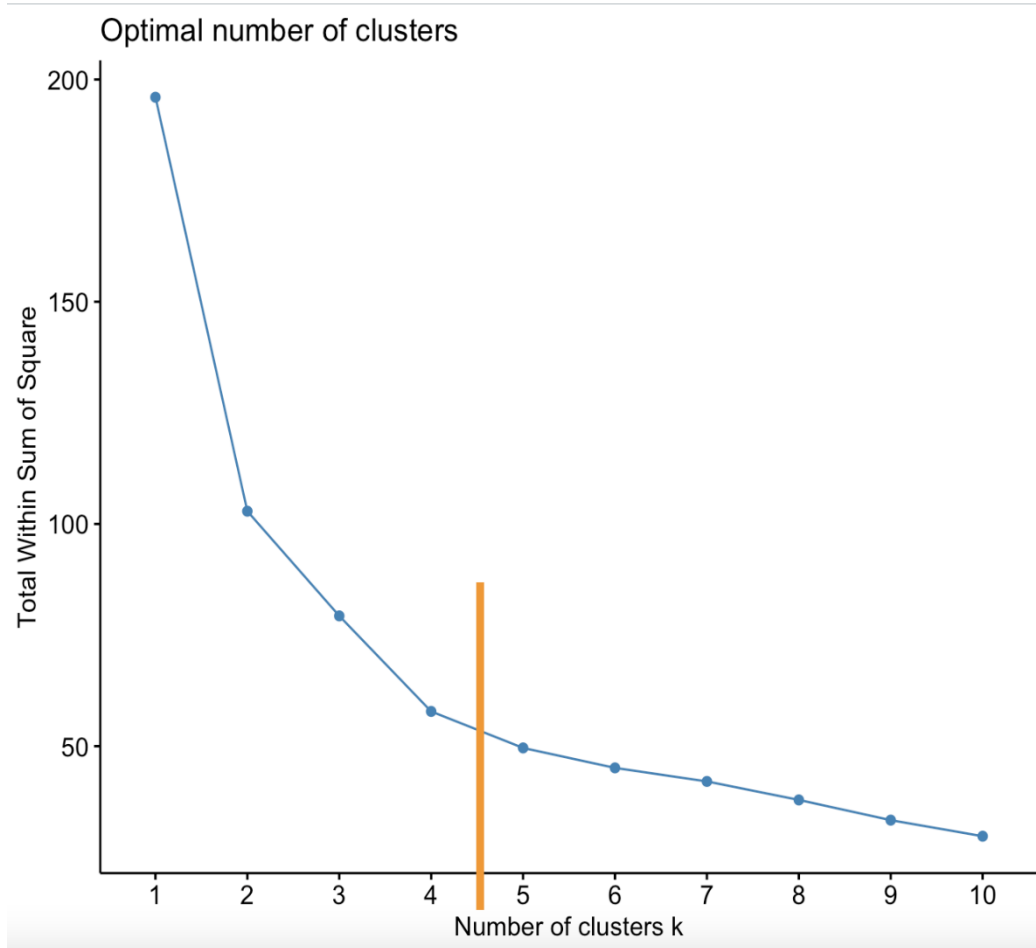
D. R 코드

함수

pam(data, k, metric = "euclidean", stand = FALSE)

```
> # install.packages('factoextra')
> library(factoextra)
> library(cluster)
>
> df <- USArrests
> df <- na.omit(df)
> df <- scale(df)
>
> fviz_nbclust(df, pam, method = "wss") # 군집 개수 4 개
>
>
> set.seed(1)
> kmed <- pam(df, k = 4) # 4개로 pam 알고리즘 실행
> kmed
>
> # 시각화
> fviz_cluster(kmed, data = df, main = 'USArrests_ k-medoids')
>
> # 군집화 결과 보기
> final_data <- cbind(USArrests, cluster = kmed$cluster)
> head(final_data)
```

시각화 결과



3) Gaussian mixture models

A. 개념

가우스 혼합 모델 (GMM)은 다변량 밀도 추정 및 확률 적 클러스터링에 대한 유연한 접근 방식 이다.

B 특징

군집화를 적용하고자 하는 데이터가 여러 개의 가우시안 분포를 가진 데이터 집합들이 섞여서 생성된 것이라는 가정하에 군집화를 수행하는 방법

EM 알고리즘을 통해 로그 우도가 최대가 되는 지점을 찾아 해당 파라미터 $\hat{\theta}$ 가 최적의 가우시안 혼합 모델의 파라미터 집합

C. 절차

$$p(z_j = 1|x) = \frac{p(z_j = 1)p(x|z_j = 1)}{\sum_j^k p(z_j = 1)p(x|z_j = 1)} = \frac{\pi_j N(x|\mu_j, \Sigma_j)}{\sum_j^k \pi_j N(x|\mu_j, \Sigma_j)}$$

각각의 데이터가 어느 클러스터에 속할지에 대한 정보를 z 을 업데이트하여 기대치를 계산한다. 업데이트된 정보들을 사용해 나머지 값들로 가장 log likelihood(로그 우도, 로그 가능도)를 최대화 하는 파라미터를 찾아낸다.

단계 2 를 반복하여 단계가 수행될 때 마다 점차 좋은 값을 찾아준다.

D. R 코드

함수

```
- gmm(g,x,t0=NULL,gradv=NULL, type=c("twoStep","cue","iterative"),  
wmatrix = c("optimal","ident"), vcov=c("HAC","MDS","iid","TrueFixed"),  
kernel=c("Quadratic Spectral","Truncated", "Bartlett", "Parzen", "Tukey-Hanning"),  
crit=10e-7,bw = bwAndrews, prewhite = 1, ar.method = "ols", approx="AR(1)",  
tol = 1e-7, itermax=100,optfct=c("optim","optimize","nllminb", "constrOptim"),  
model=TRUE, weightsMatrix = NULL, tracelter = FALSE, data, eqConst = NULL,  
eqConstFullVcov = FALSE, mustar = NULL, onlyCoefficients=FALSE, ...)
```

```
> library(tidyquant)  
> data(Finance)  
> r <- Finance[1:300, 1:10]  
> rm <- Finance[1:300, "rm"]  
> rf <- Finance[1:300, "rf"]  
>  
> z <- as.matrix(r-rf)  
> t <- nrow(z)  
> zm <- rm-rf  
> h <- matrix(zm, t, 1)  
> res <- gmm(z ~ zm, x = h)  
> summary(res)
```

4) DBSCAN

A. 개념

- DBSCAN은 밀도를 기반으로 하여 군집화하는 매우 유용한 군집 알고리즘이다. k-평균 군집이나 계층적 군집 알고리즘의 경우 데이터 간의 거리를 이용하여 클러스터를 나누는데 반해, DBSCAN 알고리즘은 데이터 포인트가 세밀하게 몰려 있어 밀도가 높은 부분을 군집화하는 방식이다.

B. 특징

DBSCAN은 k-means 와 달리 데이터의 클러스터 수를 사전에 지정할 필요가 없다 .

DBSCAN은 임의의 모양의 클러스터를 찾을 수 있다.(계층적 군집이나 k-평균 군집과는 달리 복잡한 형상도 찾을 수 있다. / 노이즈(동떨어진 데이터)의 처리에 대해 K 평균 대비 높은 성능)

DBSCAN은 두 개의 매개 변수 만 필요하며 대부분 데이터베이스의 포인트 순서에 민감하지 않다.

DBSCAN은 완전히 결정적이지 않다. 둘 이상의 클러스터에서 도달 할 수 있는 경계 지점은 데이터가 처리되는 순서에 따라 두 클러스터의 일부가 될 수 있다.

2차원이나 3차원 등 차원수가 낮은 데이터셋에는 문제가 되지 않지만, 고차원 데이터셋으로 갈수록 필요한 학습 데이터 양이 급증하는 문제점이며, 이 때문에 많은 연산이 필요해진다는 단점이 있다.

C. 절차

데이터셋을 받아온다. 데이터셋의 모든 데이터에 대해 탐색을 수행하는데, 이 탐색의 과정은 '밀도'를 기반으로 '코어', '노이즈', '보더(경계)'를 나누는 곳에서 시작한다.

탐색할 순서가 된 데이터가 방문한 데이터인지 알아보고, 방문하지 않은 데이터라면 탐색할 데이터에서부터 미리 정의한 거리 내에 있는 데이터들의 개수를 파악한다. 그 개수가 기준치 이상이면 이를 코어로 설정한다. (코어로 설정할 때, 군집을 설정한다.)

혹 개수가 기준치 미만이라면, 그 데이터를 노이즈 혹은 보더로 정의한다.(군집에 대해 탐색 중이었다면 보더, 아니라면 데이터)

코어로 설정한다면, 다시 일정 거리 안에 있는 모든 데이터들에 대해 똑같은 탐색을 실시한다. 코어에서 코어로 탐색을 실시하는 경우는 그 코어의 군집을 부여하며, 아니라면 새로운 군집에 소속되게끔 한다.

D. R 코드

함수

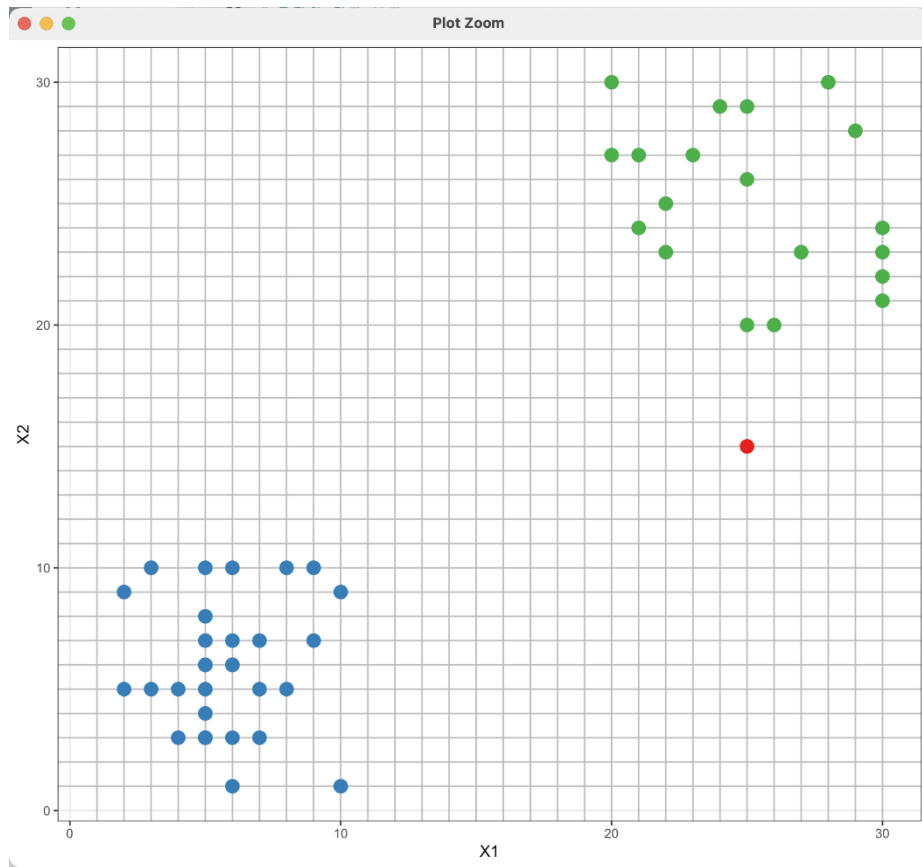
```
dbscan(x, eps, minPts = 5, weights = NULL, borderPoints = TRUE, ...)
```

```

> library(fpc)
> library(ggplot2)
>
> data <- data.frame(cbind(c(sample(1:10, 30, replace = T), sample(20:30, 20,
replace = T)),
+                          c(sample(1:10, 30, replace = T), sample(15:30, 20,
replace = T))))
> names(data) <- c("X1", "X2")
>
> head(data)
>
> ggplot(data) + geom_hline(yintercept = 1:30, colour = "grey") +
+   geom_vline(xintercept = 1:30, colour = "grey") +
+   geom_point(aes(x= X1, y = X2), cex = 2) +
+   theme_bw()
>> db <- dbscan(data, eps = sqrt(10), MinPts = 4)
>> library("RColorBrewer")
> ggplot(data) +
+   geom_hline(yintercept = 1:30, colour = "grey") +
+   geom_vline(xintercept = 1:30, colour = "grey") +
+   geom_point(aes(x= X1, y= X2, col = as.factor(db$cluster)), cex = 4) +
+   scale_color_manual(values = c('black', brewer.pal(n= 9, name = 'Set1')))) +
+   theme_bw() + theme(legend.position = "None")
>
> ggplot( ) +
+   geom_hline(yintercept = 1:30, colour = "grey") +
+   geom_vline(xintercept = 1:30, colour = "grey") +
+   geom_point(data = data, aes(x= X1, y= X2, col = as.factor(db$cluster)),
cex = 4) +
+   scale_color_manual(values = c('black', brewer.pal(n= 9, name = 'Set1')))) +
+   geom_point(data = data[!db$isseed, ], aes(X1, X2), shape = 8, size = 5) +
+   theme_bw() + theme(legend.position = "None")
>
> # noise 없이 전부 클러스터 하기
>
> db <- dbscan(data, eps = sqrt(18), MinPts = 2)
> db
>
> ggplot(data) +
+   geom_hline(yintercept = 1:30, colour = "grey") +
+   geom_vline(xintercept = 1:30, colour = "grey") +
+   geom_point(aes(x= X1, y= X2, col = as.factor(db$cluster)), cex = 4) +
+   scale_color_manual(values = c(brewer.pal(n = 9, name = 'Set1')))) +
+   theme_bw() + theme(legend.position = "None")

```


시각화 결과



5) OPTICS

A. 개념

밀도 기반의 클러스터를 탐색하기 위한 알고리즘이다. 데이터들을 정렬하여 가까운 point들이 이웃이 될 수 있도록 하며, 각 point가 더 밀집된 cluster에 포함될 수 있도록 한다.

B. 특징

밀도 뿐만 아닌 계층까지 유지 하기 때문에 더 효과 적인 군집을 할 수 있다.

OPTICS는 엄격한 파티셔닝을 생성하지 않는다.

OPTICS는 DBSCAN에 비해 비용이 많이 든다

주로 우선 순위 연결 되자만 가장 가까운 이웃 쿼리는 DBSCAN의 반경 쿼리보다 더 복잡 하다. 따라서 속도가 느리다.

C. 절차

DBSCAN 과 동일 하지만 계층을 이룬다.

D. R 코드

함수

`optics()`

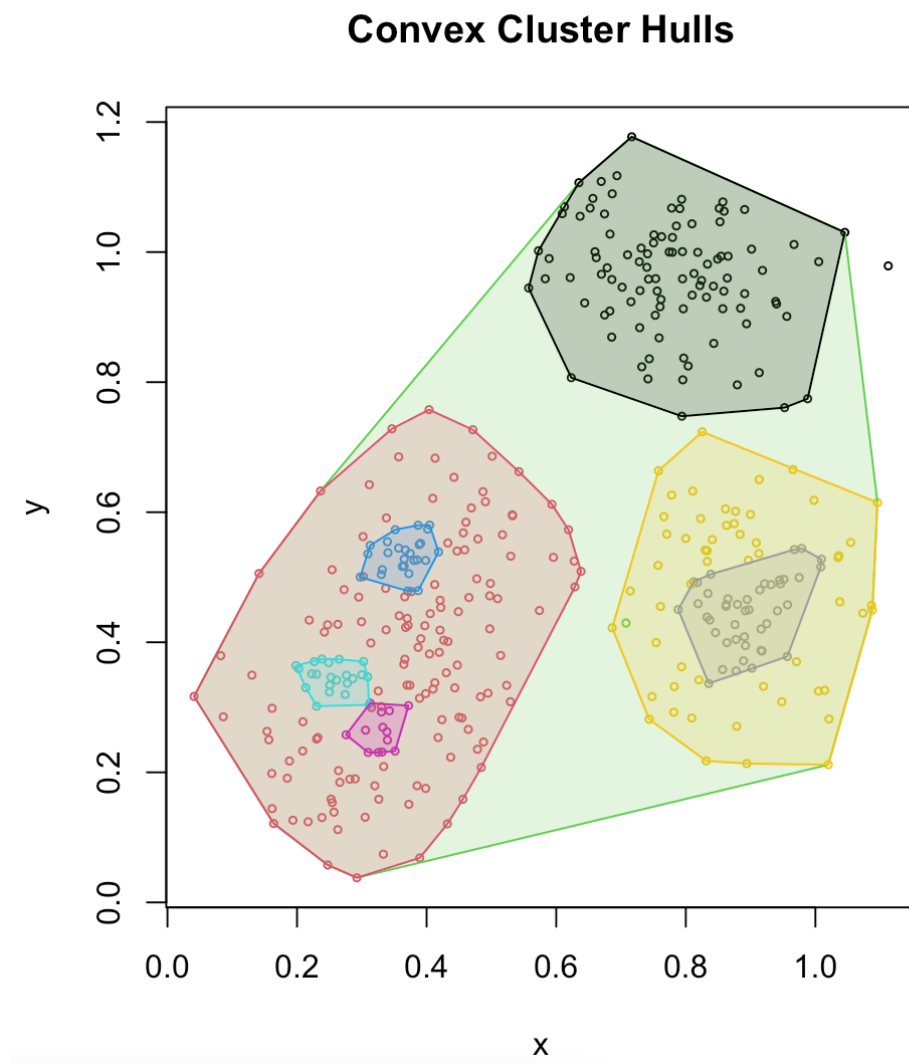
`extractDBSCAN()`

`opticsXi()`

```

> library(dbSCAN)
>
> set.seed(123)
> n <- 400
>
> x <- cbind(
+   x = runif(4, 0, 1) + rnorm(n, sd=0.1),
+   y = runif(4, 0, 1) + rnorm(n, sd=0.1)
+ )
>
> plot(x, col=rep(1:4, time = 100), main = 'OPTICS 분류 전',
family="NanumGothicBold" )
>
> plot(x, col = "grey", main = '모든 산점도 연결',
family="NanumGothicBold" )
> polygon(x[res$order,] )
>
>
> d <- dist(x)
> res <- optics(d, eps = 1, minPts = 10)
> plot(res, main = 'OPTICS 최소 포인트 수 10 ',
family="NanumGothicBold")
>
> res <- extractDBSCAN(res, eps_cl = .065)
> plot(res, main = 'OPTICS_DBSCAN 군집 확인 _ 임계값 .065 (노이즈 제거)', family="NanumGothicBold" )
> hullplot(x, res, main = 'Convex Cluster Hulls _ eps: .065 ')
>
> res <- extractDBSCAN(res, eps_cl = .1)
> plot(res, main = 'OPTICS_DBSCAN 군집확인 _ 임계값 .1 (노이즈 제거)',
family="NanumGothicBold" )
> hullplot(x, res, main = 'Convex Cluster Hulls _ eps: .1 ')
>
> # Xi 방법을 사용하여 계층적으로 표현
> res <- opticsXi(res, xi = 0.05, minimum = F, nocorrect = F)
> plot(x, col = res $ cluster + 1L)
> hullplot(x, res)

```



6) STING

A. 개념

그리드 기반 클러스터링 알고리즘이다. 데이터 세트는 재귀 적으로 계층 구조로 나뉜다. 전체 입력 데이터 세트는 계층 구조에서 루트 노드 역할을 한다. 레이어의 각 셀 유닛은 하위 레이어에 있는 두 개의 셀 유닛으로 구성된다.

B. 특징

그리드 기반 클러스터링의 가장 큰 장점은 특히 매우 큰 데이터 세트를 클러스터링 할 때 계산 복잡성이 크게 줄어든다는 것이다.

C. 절차

그리드 구조 만들기, 즉 데이터 공간을 유한 한 수의 셀로 분할한다.

각 셀의 셀 밀도를 계산한다

밀도에 따른 세포 분류

클러스터 센터 식별

인접 셀 순회

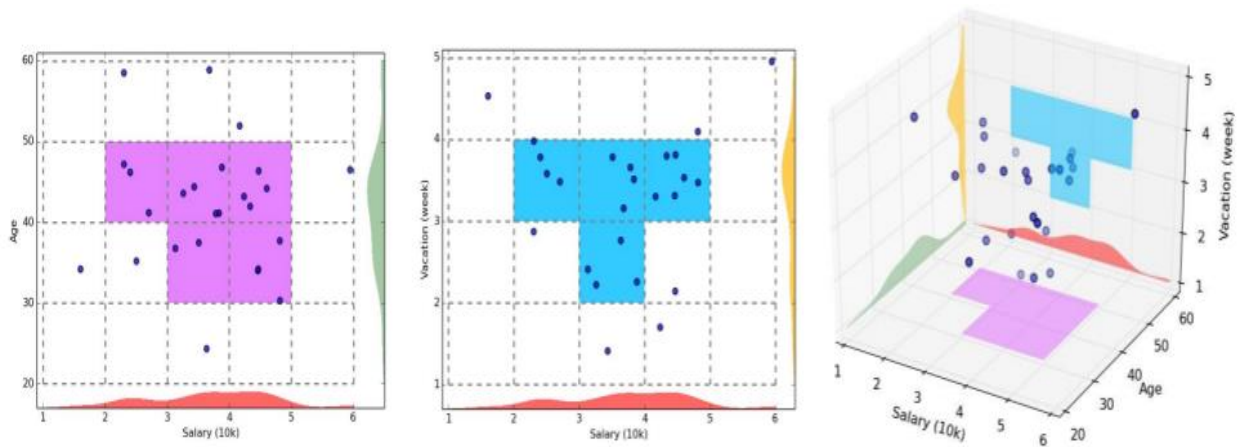
D. 참고 문헌

<https://www.koreascience.or.kr/article/JAKO200311921882775.pdf>

7) CLIQUE (Clustering In QUES)

A. 개념

데이터 공간의 밀도를 간단하게 산정하기 위하여 각 차원을 다수의 일정한 간격으로 나누고 분할된 각 셀(unit)안에 놓여진 점들의 수를 찾는다. 각 셀이 동일한 크기를 가지므로 그안의 점들의 수는 셀의 밀도를 의미한다.



B. 특징

입력 데이터 순서와 무관한 동일한 결과
고밀도 클러스터를 갖는 부분공간 자동식별
많은 부분 공간이 제거될 가능성
클러스터간에 오버랩이 존재할 가능성

C. 절차

데이터 공간을 분할하고 파티션의 각 셀 내부에 있는 점 수를 찾는다. 이후, Apriori 원리를 사용하여 클러스터를 포함하는 하위 공간 식별한다.

모든 관심 하위 공간에서 밀도 단위 결정한다. 이후 연결된 밀도 단위 결정한다.

각 클러스터에 대해 연결된 밀도 단위의 클러스터를 포함하는 최대 영역 결정한다. 이후, 각 클러스터에 대한 최소 적용 범위 결정한다.

D. R 코드

Package

- subspace

함수

- cliques(graph, min = NULL, max = NULL)

```

install.packages('subspace')
install.packages('igraph')

library(subspace)
library(igraph)

# this usually contains cliques of size six
g <- sample_gnp(100, 0.3)
plot(g)

clique_num(g)
cliques(g, min=6)
a <- largest_cliques(g)

# To have a bit less maximal cliques, about 100-200 usually
g <- sample_gnp(100, 0.03)
max_cliques(g)

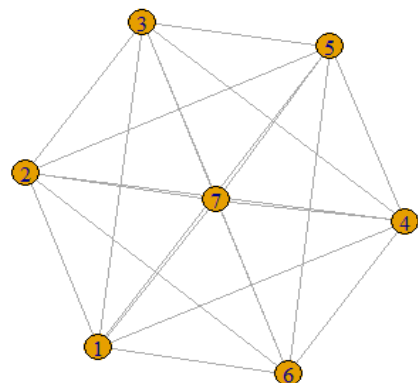
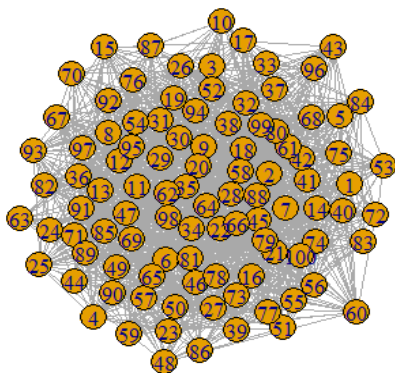
clique1 <- a[[1]]

# subset the original graph by passing the clique vertices
g2 <- induced.subgraph(graph=g, vids=clique1)

# plot the clique
plot(g2)

```

시각화 결과



8) CLARA (Clustering LARge Applications)

A. 개념

전체 데이터 집합에 대한 대표 객체들을 찾는 대신, 데이터 집합에 대한 샘플을 무작위로 추출하고, 그 샘플에 대해서 PAM 을 적용하여 샘플의 medoid들을 찾는다. 이때 샘플 데이터의 신뢰성을 위하여 복수 개의 샘플들을 추출하여 medoid들을 구함으로써 최상의 클러스터를 탐색하도록 한다. (medoid : 군집 내에서 객체들간의 평균 비유사성이 가장 작은 객체)

B. 특징

표본 크기에 의존된다.

추출된 표본에 좋은 medoid가 없다면, 최적의 군집을 찾지 못할 수도 있다.

C. 절차

전체 데이터에서 임의로 표본을 추출하여 그 표본에 PAM 알고리즘을 적용시켜 k 개의 medoid를 찾는다.

단계 1에서 구한 k 개의 medoid에 가까운 객체들로 군집을 형성한다.

전체 데이터 형성된 군집을 이용해 평균 비유사성을 계산 한다. 만약 계산된 값이 현재 값보다 작다면 계산된 값을 현재 값으로 바꾼다.

medoid 가 수렴할 때까지 단계1,2,3을 반복한다.

D. R 코드

함수

- clara(x, k, metric = "euclidean", stand = FALSE, samples = 5, pamLike = FALSE)


```
install.packages(c("cluster", "factoextra"))
library(cluster)
library(factoextra)

set.seed(1234)
# Generate 500 objects, divided into 2 clusters.
df <- rbind(cbind(rnorm(200,0,8), rnorm(200,0,8)),
            cbind(rnorm(300,50,8), rnorm(300,50,8)))

# Specify column and row names
colnames(df) <- c("x", "y")
rownames(df) <- paste0("S", 1:nrow(df))

# Previewing the data
head(df, nrow = 6)

# Compute CLARA
clara.res <- clara(df, 2, samples = 50, pamLike = TRUE)

# Print components of clara.res
print(clara.res)

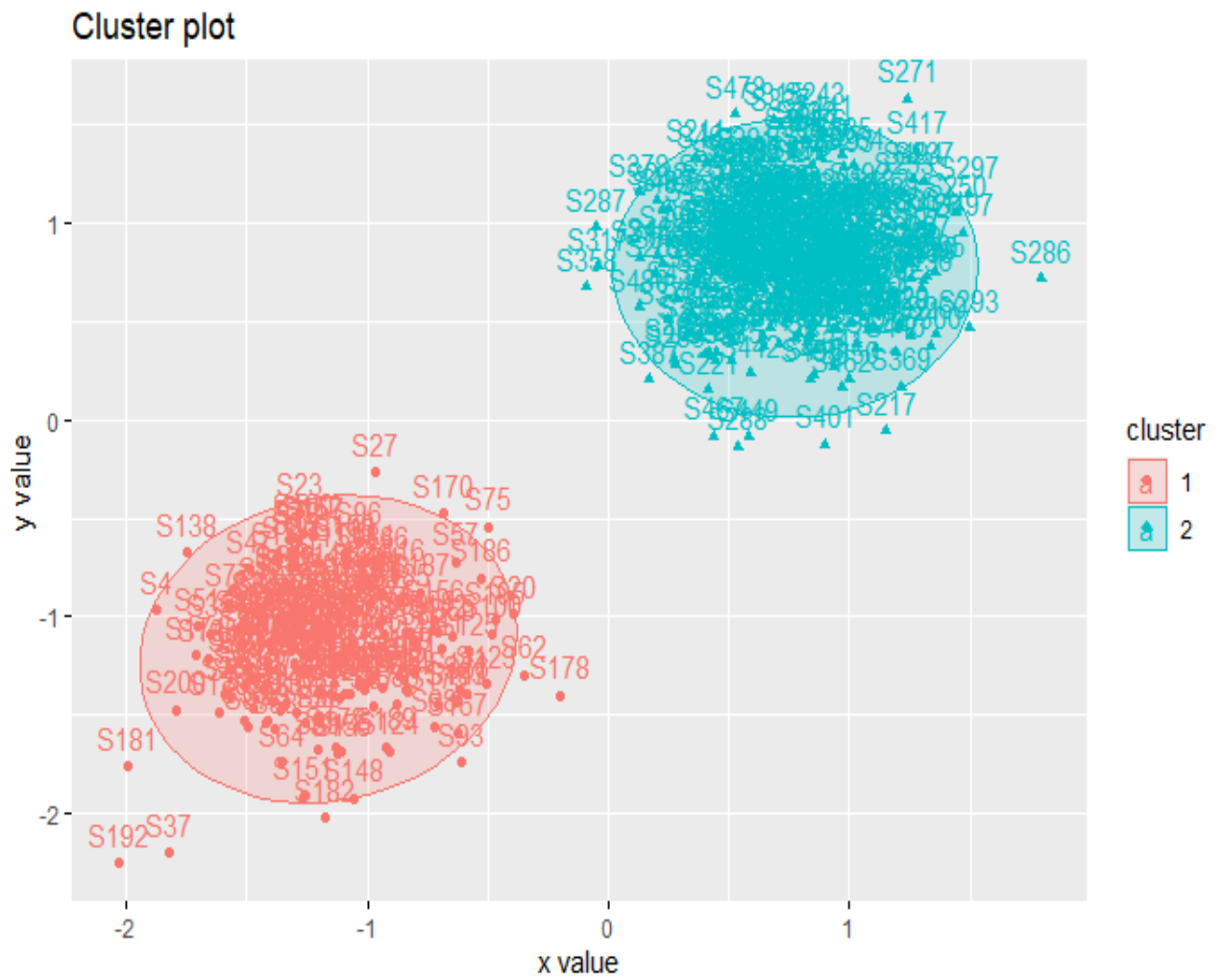
dd <- cbind(df, cluster = clara.res$cluster)
head(dd, n = 4)

# Medoids
clara.res$medoids

# Clustering
head(clara.res$clustering, 10)

#Visualizing CLARA clusters
fviz_cluster(clara.res, df, ellipse.type = "norm")
```

시각화 결과



9) CLARAN

(Clustering Large Applications based on Randomized)

A. 개념

-PAM 과 CLARA 의 장점을 혼합한 알고리즘이다. CLARA 알고리즘은 표본을 이용하여 좀 더 적은 양의 이웃 노드들로 평가하는 것이다. CLARA 알고리즘처럼 CLARSNS 알고리즘은 노드의 모든 이웃들을 평가하지는 않는다. 노드의 이웃의 표본을 추출하여 평가한다.

B. 특징

k-medoid 에서 진일보한 군집화 기법

DB 다중 스캔

무작위 접근 방법의 사용으로, N이 클 경우 결과의 품질을 보장 못함.

객체들의 주기억장치 상주를 전제하므로 대용량 DB 적용 불가능

C. 절차

이용할 총 반복 횟수와 평가할 이웃들의 수를 설정한다.

반복횟수인 i 를 1로 초기화 한다.

그래프에서 현재 사용할 노드를 뽑아 초기 노드로 사용한다.

현재 노드에서 평가할 이웃들이 값의 개수만큼 이웃의 표본을 뽑아 이웃 사이의 총 비용을 계산한다.

만약 표본으로 사용한 이웃들간의 비용이 더 작다면 현재 비용을 작은 비용으로 갱신하고, 단계 3 으

로 돌아가 이웃들 사이의 비용을 다시 계산한다. 이 과정을 표본의 이수들의 수만큼 반복한다.

반복한 후, 최소 비용을 현재 최소 비용으로 저장하고, 수렴할 때까지, 단계3, 4를 반복한다.

i 가 총 반복할 횟수가 될 때까지 전체의 과정을 반복한다.

D. R 코드

Package

- QTCAT/qtcat

함수

- clarans(snp, k, maxNeighbours = 100, nLocal = 10, mc.cores = 1)

```
install.packages("devtools")
#remotes::install_github("QTCAT/qtcat")
devtools::install_github("QTCAT/qtcat")
library(qtcat)

gfile <- system.file("extdata/snpdata.csv", package = "qtcat")
snp <- read.snpData(gfile, sep = ",")

clarans(snp, 3)
```

10) BIRCH

(Balanced Iterative Reducing and Clustering Hierachies)

A. 개념

가능한 한 많은 정보를 보유하는 대규모 데이터 세트의 작고 간결한 요약물 먼저 생성하여 대규모 데이터 세트를 클러스터링 할 수 있는 클러스터링 알고리즘이다. 다른 클러스터링 알고리즘이 현재 사용할 수 있는 데이터 세트의 요약물 생성하여 다른 클러스터링 알고리즘을 보완하는 데 자주 사용된다.

B. 특징

DB의 크기 및 데이터의 방문 횟수 면에 대한 선형적 시간 복잡도
잡음을 고려한 첫 번째 알고리즘
요약정보 기반으로 고차원 공간에서 효과성에 관한 성능 저하
둥근 모양의 클러스터 경우에만 효과적인 탐색
데이터 입력순서에 민감

C. 절차

모든 data를 스캔하고 초기 메모리에 CF Tree를 만든다.
더 작은 CF-Tree 로 만들어 바람직한 길이로 압축한다. 이 단계는 옵션으로 이상치를 제거하고 하위 군집을 더 큰 군집으로 그룹화 하는 것이다.
모든 리프 항목을 클러스터링한다. 사용자가 원하는 클러스터 수, 임계값을 지정할 수 있는 유연성을 제공하는 단계이다. 이 단계 이후 데이터의 주요 분포 패턴을 캡처하는 클러스터 세트를 얻는다.
생성된 클러스터의 시드를 활용하고, 데이터 포인트를 가장 가까운 시드로 분산하여 새로운 클러스터를 얻을 수 있다. 또한 아웃라이어를 처리할 수 있다.

D. R 코드

<https://www.geeksforgeeks.org/ml-birch-clustering/>

BIRCH 패키지가 cran에서 제거되어 확인 불가

11) canopy

A. 개념

캐노피 군집으로도 알려져 있는 캐노피 생성은 속도가 빠른 근사적(Approximate)군집 기술이다. 캐노피 생성은 점으로 구성된 입력 셋을 캐노피라는 중첩 군집으로 나누는 데 사용한다. 캐노피 라는 말 자체는 점의 울타리 혹은 군집을 의미한다. 캐노피 군집은 군집의 센트로이드 혹은 캐노피 중심을 두 거리의 임계값을 사용해서 근사적으로 예측한다.

B. 특징

전체 데이터를 단일 패스로 처리해 매우 빠르게 군집을 생성

정확하지 않은 군집 생성 가능성

구체적인 K 군집수 없이 최적의 군집 수 제시 가능

C. 절차

점으로 구성된 데이터 셋과 비어있는 캐노피 리스트로 시작한다.

두개의 임계값을 사용하여 첫번째 지점까지의 거리가 느슨한 거리보다 작은 경우 T1, 좁은 거리보다 짧은 경우, T2 로 설정 후 원래 세트에서 제거

클러스터에 대한 세트에 더이상 데이터 포인트가 없을 때까지 2단계를 반복

군집한 모든 점이 새로운 캐노피의 중심이 되는 것을 방지

12) SUBCLU (SUBspace CLUstering)

A. 개념

밀도기반 클러스터링 알고리즘인 DBSCAN 을 기반으로 하는 서브스페이스 클러스터링 알고리즘이다. 1차원 클러스터를 DBSCAN으로 생성한 다음, 각 클러스터를 한 차원씩 확장하여 이 클러스터와 한 차원만 다른 클러스터를 갖는 차원으로 확장한다. 이 확장은 클러스터를 생성한 원래 DBSCAN에 사용된 것과 동일한 매개 변수를 가진 DBSCAN을 사용하여 수행된다.

B. 특징

모든 클러스터를 상향식(Bottom-Up)으로 생성하는 프로세스

축 평형적으로 클러스터를 탐색하며 잡음을 스스로 분리한다.

C. 절차

차원 클러스터를 생성한다. 각(k 는 1부터 시작) 1차원의 하위공간에 DBSCAN 을 적용하여 탐지된 각 클러스터에 대해 확인한다. 다른 클러스터는 고차원 하위공간에 존재할 수 있으므로, k 차원의 하위 공간에 대한 서브스페이스를 검색한다.

이전 단계에서 생성한 클러스터(k 차원)로 부터 한 차원 높은 후보 서브스페이스를 생성한다. k 차원 서브스페이스를 가지지 않은 후보들을 제거한다. 관련 없는 후보를 정리하고 나서 후보 하위 공간에 DBSCAN이 적용되어 아직 클러스터가 포함되어 있는지 여부를 확인한다. 포함된 경우, 후보 하위 공간은 다음 하위 공간 조합에 사용된다

k -차원 하위 공간을 $k-1$ 속성을 공유하는 클러스터와 결합하여 $k+1$ 차원 후보 하위 공간을 반복적으로 생성한다. 이후 단계 2를 반복한다.

DBSCAN의 런타임을 개선하기 위해, k 차원 부분 공간의 클러스터에 속한 서브스페이스만 고려한다.

<SUBCLU 절차 참고>

```
SUBCLU(SetOfObjects DB, Real  $\epsilon$ , Integer  $m$ )
/* STEP 1 Generate all 1-D clusters */
 $S_1 := \emptyset$  // set of 1-D subspaces containing clusters
 $C_1 := \emptyset$  // set of all sets of clusters in 1-D subspaces
FOR each  $a_i \in A$  DO
     $C^{\{a_i\}} := DBSCAN(DB, \{a_i\}, \epsilon, m)$  // set of all clusters in subspace  $a_i$ ;
    IF  $C^{\{a_i\}} \neq \emptyset$  THEN // at least one cluster in subspace  $\{a_i\}$  found
         $S_1 := S_1 \cup \{a_i\}$ ;
         $C_1 := C_1 \cup C^{\{a_i\}}$ ;
    END IF
END FOR
/* STEP 2 Generate  $(k+1)$ -D clusters from  $k$ -D clusters */
 $k := 1$ ;
WHILE  $C_k \neq \emptyset$ 
    /* STEP 2.1 Generate  $(k+1)$ -dimensional candidate subspaces */
     $CandS_{k+1} := GenerateCandidateSubspaces(S_k)$ ;
    /* STEP 2.2 Test candidates and generate  $(k+1)$ -dimensional clusters */
    FOR EACH  $cand \in CandS_{k+1}$  DO
        // Search  $k$ -dim subspace of  $cand$  with minimal number of objects in the clusters
         $bestSubspace := \min_{s \in S_k \wedge s \subseteq cand} \sum_{C_i \in C^s} |C_i|$ 
         $C^{cand} := \emptyset$ ;
        FOR EACH cluster  $cl \in C^{bestSubspace}$  DO
             $C^{cand} = C^{cand} \cup DBSCAN(cl, cand, \epsilon, m)$ ;
            IF  $C^{cand} \neq \emptyset$  THEN
                 $S_{k+1} := S_{k+1} \cup cand$ ;
                 $C_{k+1} := C_{k+1} \cup C^{cand}$ ;
            END IF
        END FOR
    END FOR
     $k := k + 1$ 
END WHILE
```

D. R 코드

Package

- subspace

함수

- SubClu(data, epsilon = 4, minSupport = 4)

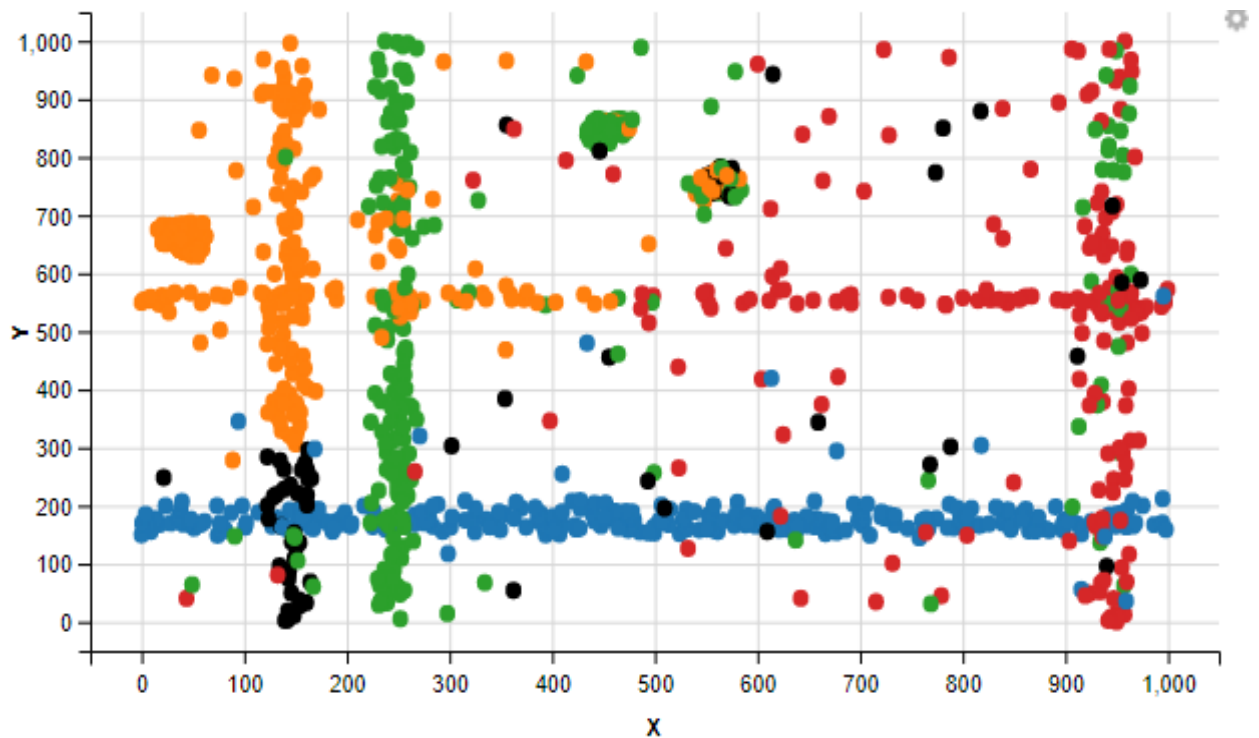
```
install.packages('subspace')
library(subspace)

data("subspace_dataset")
subClustering <- SubClu(subspace_dataset, epsilon=1, minSupport=5)

P3C(subspace_dataset, PoissonThreshold=3)

plot(subClustering,
      subspace_dataset, color_by = "mix",
      standardcolors = c("#1F77B4", "#FF7F0E", "#2CA02C",
                          "#D62728",
                          "#9467BD", "#8C564B", "#E377C2", "#7F7F7F",
                          "#BCBD22", "#17BECF", "#000000"),
      tooltip_on = "hover")
```

시각화 결과



※ 참고 인터넷 자료

https://kcugenii.com/wiki/Cluster_analysis

<http://www.koreascience.kr/article/JAKO200311922676487.pdf>

<https://ko.wikiarabi.org/wiki/SUBCLU>

https://en.wikipedia.org/wiki/Canopy_clustering_algorithm

<https://cran.r-project.org/web/packages/subspace/subspace.pdf>