

# **빅데이터예측분석**

**서강대학교 A70071 유지연**

# 목차

## 01. IRIS 코딩활용

SVM  
KNN  
인공신경망  
랜덤포레스트  
나이프베이즈

## 02. 데이터 전처리

100만건 건강 데이터 전처리  
이분형 로지스틱 회귀분석

## (1) R 을 통한 예측 및 분석

```
## SVM ##
library(e1071)
# svm과 나이브 베이즈를 포함하는 패키지

data(iris)

set.seed(123)
# 난수 생성 과정의 시작점을 고정 (=시드 설정)
# 이렇게 하면, 코드를 실행할 때마다 동일한 랜덤 추출 결과를 얻을 수 있다.

tr.idx <- sample(1:nrow(iris), 0.7 * nrow(iris))
# 훈련 데이터 인덱스 추출
# nrow(iris)는 iris 데이터셋의 행 수를 반환한다.
# 즉, 1부터 iris 데이터 전체 행 수 중 70%를 랜덤 추출한 인덱스를 tr.idx에 넣는다는 의미이다.

x.tr <- iris[tr.idx, -5] # 훈련 데이터
x.te <- iris[-tr.idx, -5] # 테스트 데이터
y.tr <- iris[tr.idx, 5] # 훈련 데이터의 타겟
y.te <- iris[-tr.idx, 5] # 테스트 데이터의 타겟

model_svm <- svm(Species ~ ., data = iris, subset = tr.idx)
# svm 알고리즘을 사용하여 모델을 훈련한다. data=iris로 사용하며,
# subset=tr.idx : tr.idx에 해당하는 행만을 사용하여 모델을 훈련시키겠다.

pred_svm <- predict(model_svm, x.te)
# 훈련된 모델(model_svm)을 사용하여 테스트 데이터(x.te)의 species를 예측한다.

table(y.te, pred_svm)
# 실제 타겟 값(y.te)과 예측된 값(pred_svm)을 비교하는 혼동 행렬을 생성한다.
```

실제 품종 y.te와 예측 품종pred\_svm의 Confusion Matrix

	pred_svm		
y.te	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	1
virginica	0	0	13

## (2) 혼동 행렬을 통한 모델 분석

y.te : 실제 붓꽃 품종      Pred\_svm : 예측한 붓꽃 품종

y.te	pred_svm		
	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	1
virginica	0	0	13

14개의 샘플 모두 정확하게 Setosa로 예측, Setosa가 Versicolor나 Virginica로 잘못 분류된 경우는 없다.

실제 versicolor 중 1개가 virginica 로 잘못 예측 되었다.

13개의 샘플 모두 정확하게 virginica 로 예측, virginica 가 Setosa나 Virgicolor 로 잘못 분류된 경우는 없다.

### Model Accuracy (정확도)

모델이 얼마나 잘 예측되었는 지를 나타내는 척도, 전체 샘플 중에서 정확하게 예측된 샘플의 비율을 말한다.

$Accuracy = (14 + 17 + 13) / 45 = 0.9778$

### Class Precision (클래스 별 정밀도)

정밀도란, 특정 클래스로 예측된 것 중 실제로 그 클래스인 비율을 말한다.

Setosa : Precision =  $14 / 14 = 1$

Versicolor : Precision =  $17 / 17 = 1$

Viginicia : Precision =  $13 / (1 + 13) = 0.9286$

### Class Recall (클래스 별 재현율)

재현율이란, 실제 클래스 중 모델이 그 클래스로 올바르게 예측한 비율을 말한다.

Setosa : Recall =  $14 / 14 = 1$

Versicolor : Recall =  $17 / (17 + 1) = 0.9444$

Viginicia : Recall =  $13 / 13 = 1$

### (1) R 을 통한 예측 및 분석

```
## KNN ##
library(class) # class 클래스에는 KNN 알고리즘을 구현하는 knn() 함수가 있다.
data(iris)

y <- iris[,5]
# 5번째 열(품종)을 추출하여 y 변수에 저장한다. 즉 예측하고자 하는 타겟은 '품종'이다.

tr.idx <- sample(length(y), 75)
# y의 길이(150)에서 무작위로 75개의 인덱스를 선택하여 tr.idx 변수에 저장한다.
# 이는 훈련 데이터를 선택하기 위함이다.

x.tr <- iris[tr.idx, -5]
# tr.idx에 해당하는 행을 선택 + 5번째 열(타겟 변수)을 제외한 나머지를 훈련 데이터로 사용한다.

x.te <- iris[-tr.idx, -5]
# tr.idx에 해당하지 않는 행을 선택 + 5번째 열(타겟 변수)은 제외한다.

m <- knn(x.tr, x.te, y[tr.idx], k=3)
# knn() 함수를 사용하여 KNN 모델을 훈련하고 예측을 수행한다.
# 가장 가까운 3개의 이웃을 사용하여 분류를 수행한다.
# 이를 통해 m 에는 각 샘플에 대한 예측된 품종이 들어간다.

yy <- y[-tr.idx]
# 테스트 데이터의 타겟 변수를 yy 에 저장한다.
table(yy,m)
# 예측된 품종(m)과 실제 품종(yy)을 비교하여 혼동 행렬(confusion matrix)을 생성한다,
# 이 결과를 통해 모델의 성능을 평가할 수 있다.
```

타겟 변수 y

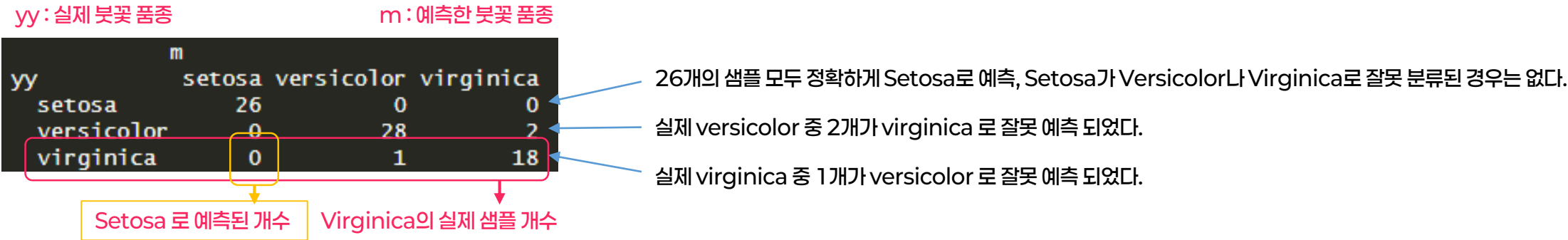
```
> print(iris)
```

	Sepal.Length	Sepal.width	Petal.Length	Petal.width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	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
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa

### 실제 품종 yy와 예측 품종m 의 Confusion Matrix

	m		
yy	setosa	versicolor	virginica
setosa	26	0	0
versicolor	0	28	2
virginica	0	1	18

## (2) 혼동 행렬을 통한 모델 분석



### Model Accuracy (정확도)

모델이 얼마나 잘 예측되었는 지를 나타내는 척도, 전체 샘플 중에서 정확하게 예측된 샘플의 비율을 말한다.

$Accuracy = (26 + 28 + 18) / 75 = 0.96$

### Class Precision (클래스 별 정밀도)

정밀도란, 특정 클래스로 예측된 것 중 실제로 그 클래스인 비율을 말한다.

Setosa : Precision =  $26 / 26 = 1$

Versicolor : Precision =  $28 / (28 + 1) = 0.9655$

Virginica : Precision =  $18 / (2 + 18) = 0.9$

### Class Recall (클래스 별 재현율)

재현율이란, 실제 클래스 중 모델이 그 클래스로 올바르게 예측한 비율을 말한다.

Setosa : Recall =  $26 / 26 = 1$

Versicolor : Recall =  $28 / (28 + 2) = 0.9333$

Virginica : Recall =  $18 / (18 + 1) = 0.9474$

### (1) R 을 통한 예측 및 분석

```
# 라이브러리 로드
library(neuralnet)
data(iris)

# iris 데이터의 타겟 변수를 one-hot Encoding
iris$setosa <- ifelse(iris$species == "setosa", 1, 0)
iris$versicolor <- ifelse(iris$species == "versicolor", 1, 0)
iris$virginica <- ifelse(iris$species == "virginica", 1, 0)
# 각 품종에 대해 별도의 열을 생성하고, 해당 품종에 맞는 경우 1로 설정하여
# 범주형 변수를 one hot 인코딩 하는 것이다.

# 훈련 및 테스트 데이터 분리
set.seed(123)
train_indices <- sample(1:nrow(iris), 0.7 * nrow(iris)) # 데이터의 70%를 훈련 데이터로 사용
train_data <- iris[train_indices,]
test_data <- iris[-train_indices,]

# 인공신경망 모델 구축
# 타겟 변수는 one-hot Encoding된 setosa, versicolor, virginica 3개 컬럼
nn_model <- neuralnet(setosa + versicolor + virginica ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                      data = train_data, hidden = c(5), linear.output = FALSE)

# 신경망 구조 시각화
plot(nn_model)

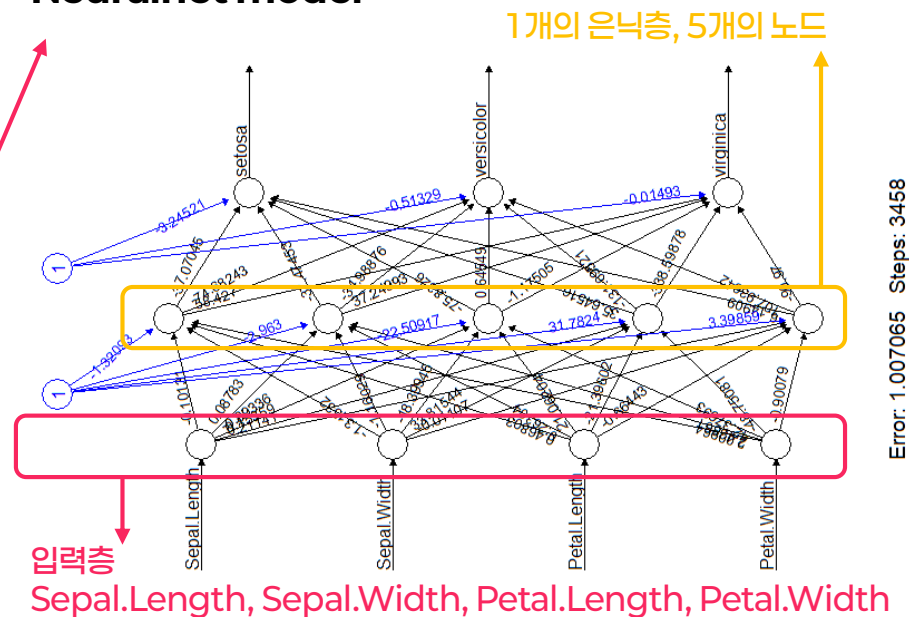
# 테스트 데이터 준비
test_features <- subset(test_data, select = -c(Species, setosa, versicolor, virginica))

# 모델을 통한 예측 수행
nn_results <- compute(nn_model, test_features)
predictions <- nn_results$net.result

# 예측 결과를 통해 가장 높은 값을 가진 인덱스로 품종 예측
predicted_species <- apply(predictions, 1, which.max)
predicted_species <- factor(predicted_species, levels = c(1, 2, 3), labels = c("setosa", "versicolor", "virginica"))

# 실제 값과 예측 값 비교
confusion_matrix <- table(predicted_species, test_data$Species)
print(confusion_matrix)
```

### Neuralnet model



### 실제 품종과 예측 품종의 Confusion Matrix

predicted_species	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	0
virginica	0	1	13

### (2) 혼동 행렬을 통한 모델 분석

실제 붓꽃 품종      Predicted\_species : 예측한 붓꽃 품종

predicted_species	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	0
virginica	0	1	13

14개의 샘플 모두 정확하게 Setosa로 예측, Setosa가 Versicolor나 Virginica로 잘못 분류된 경우는 없다.

17개의 샘플 모두 정확하게 Versicolor로 예측, Setosa나 Virginica로 잘못 분류된 경우는 없다.

14개의 샘플 중 1개가 versicolor 로 잘못 분류 되었다.

### Model Accuracy (정확도)

모델이 얼마나 잘 예측되었는 지를 나타내는 척도, 전체 샘플 중에서 정확하게 예측된 샘플의 비율을 말한다.

$Accuracy = (14 + 17 + 13) / 45 = 0.9778$

### Class Precision (클래스 별 정밀도)

정밀도란, 특정 클래스로 예측된 것 중 실제로 그 클래스인 비율을 말한다.

Setosa : Precision =  $14 / 14 = 1$

Versicolor : Precision =  $17 / (17 + 1) = 0.9444$

Virginica : Precision =  $13 / 13 = 1$

### Class Recall (클래스 별 재현율)

재현율이란, 실제 클래스 중 모델이 그 클래스로 올바르게 예측한 비율을 말한다.

Setosa : Recall =  $14 / 14 = 1$

Versicolor : Recall =  $17 / 17 = 1$

Virginica : Recall =  $13 / (1 + 13) = 0.9286$



### (1) R 을 통한 예측 및 분석

```
library(randomForest)
data(iris) # iris 데이터셋 로드

# 훈련 및 테스트 데이터 분리
set.seed(123)
train_indices <- sample(1:nrow(iris), 0.7 * nrow(iris)) # 70%를 훈련 데이터로 사용
train_data <- iris[train_indices,]
test_data <- iris[-train_indices,]

# 랜덤 포레스트 모델 구축
rf_model <- randomForest(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                          data = train_data, ntree = 100) # 100개의 트리 사용

# 모델 정보 출력
print(rf_model)

# 중요 변수 확인
importance(rf_model)
varImpPlot(rf_model)

# 테스트 데이터에 대한 예측 수행
rf_pred <- predict(rf_model, test_data)

# 실제 값과 예측 값 비교
confusion_matrix <- table(rf_pred, test_data$Species)
print(confusion_matrix)
```

```
> # 중요 변수 확인
> importance(rf_model)

              MeanDecreaseGini
Sepal.Length      8.873711
Sepal.Width       1.909605
Petal.Length     26.801921
Petal.Width      31.583096
```

### Model Info

```
> print(rf_model)

Call:
randomForest(formula = Species ~ Sepal.Length + Sepal.Width +
              Type of random forest: classification
              Number of trees: 100
              No. of variables tried at each split: 2

              OOB estimate of error rate: 6.67%
Confusion matrix:
      setosa versicolor virginica class.error
setosa      36         0         0  0.0000000
versicolor   0        29         3  0.0937500
virginica     0         4        33  0.1081081
```

### Mean Decrease in Gini

: Gini 지수란?

결정 트리에서 주로 사용하는, 불순도를 측정하는 지표로,  
노드를 나눌 때 데이터를 얼마나 잘 나누는지를 평가하는 척도이다.  
이 값이 클수록 해당 변수가 예측에 중요한 역할을 했음을 의미한다.

### 실제 품종과 예측 품종의 Confusion Matrix

```
> confusion_matrix <- table(rf_pred, test_data$Species)
> print(confusion_matrix)

rf_pred      setosa versicolor virginica
setosa       14         0         0
versicolor    0        17         0
virginica     0         1        13
```

Test data 를 통해 얻은 혼동 행렬

OOB(Out-Of-Bag) 샘플 기반 혼동 행렬  
: OOB sample 이란?  
각 트리가 학습하는 과정에서 샘플되지 않은 데이터

## (2) 혼동 행렬을 통한 모델 분석

실제 붓꽃 품종

예측한 붓꽃 품종

rf_pred	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	0
virginica	0	1	13

14개의 샘플 모두 정확하게 Setosa로 예측, Setosa가 Versicolor나 Virginica로 잘못 분류된 경우는 없다.

17개의 샘플 모두 정확하게 Versicolor로 예측, Setosa나 Virginica로 잘못 분류된 경우는 없다.

실제 virginica 중 1개가 versicolor 로 잘못 예측 되었다.

### Model Accuracy (정확도)

모델이 얼마나 잘 예측되었는 지를 나타내는 척도, 전체 샘플 중에서 정확하게 예측된 샘플의 비율을 말한다.

$Accuracy = (14 + 17 + 13) / 45 = 0.9333$

### Class Precision (클래스 별 정밀도)

정밀도란, 특정 클래스로 예측된 것 중 실제로 그 클래스인 비율을 말한다.

Setosa : Precision =  $14 / 14 = 1$

Versicolor : Precision =  $17 / (17 + 1) = 0.9444$

Virginica : Precision =  $13 / 13 = 1$

### Class Recall (클래스 별 재현율)

재현율이란, 실제 클래스 중 모델이 그 클래스로 올바르게 예측한 비율을 말한다.

Setosa : Recall =  $14 / 14 = 1$

Versicolor : Recall =  $17 / 17 = 1$

Virginica : Recall =  $13 / (1 + 13) = 0.9286$

# Naive Bayes

## 01. IRIS 코딩활용

### (1) R 을 통한 예측 및 분석

```
## Naive Bayes ##
library(e1071)
# 다양한 머신러닝 알고리즘을 포함하는 패키지

data(iris)

samp <- c(sample(1:50,35), sample(51:100,35), sample(101:150,35))
# 각 품종(setosa, versicolor, virginica)에서 35개씩 총 105개의 데이터를 랜덤 추출하기 위한 인덱스 추출
# samp 에는 인덱스가 들어간다.

iris.tr <- iris[samp,]
# 선택된 index 를 기반으로 train set 구성

iris.te <- iris[-samp,]
# 선택된 index 를 제외하고 test set 구성

m <- naiveBayes (Species ~ ., data = iris.tr)
# 붓꽃의 품종 (Species)을 목표 변수(종속 변수) 사용하여 나이브 베이즈 모델을 훈련한다.

pred <- predict(m, iris.te[,5])
# test data set 에 대해 모델 m 을 사용하여 품종을 예측한다.

table(iris.te[,5], pred)
# 실제 품종(iris.te[,5])과 예측된 결과(pred)를 비교하여 혼동 행렬을 작성한다.
```

```
> print(samp)
[1] 41 42 47 1
[39] 51 91 88 6
[77] 144 135 138 13
```

생략

인덱스가 들어간 samp

```
1 20 7 2 43 78 90 100
0 54 109 148 143 137 145 140
```

### 실제 품종과 예측 품종 pred의 Confusion Matrix

	pred		
	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	13	2
virginica	0	1	14

## (2) 혼동 행렬을 통한 모델 분석

실제 붓꽃 품종

pred : 예측한 붓꽃 품종

	pred		
	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	13	2
virginica	0	1	14

15개의 샘플 모두 정확하게 Setosa로 예측, Setosa가 Versicolor나 Virginica로 잘못 분류된 경우는 없다.

실제 versicolor 중 2개가 virginica 로 잘못 예측 되었다.

실제 virginica 중 1개가 versicolor 로 잘못 예측 되었다.

### Model Accuracy (정확도)

모델이 얼마나 잘 예측되었는 지를 나타내는 척도, 전체 샘플 중에서 정확하게 예측된 샘플의 비율을 말한다.

$$\text{Accuracy} = (15 + 13 + 14) / 45 = 0.9333$$

### Class Precision (클래스 별 정밀도)

정밀도란, 특정 클래스로 예측된 것 중 실제로 그 클래스인 비율을 말한다.

$$\text{Setosa} : \text{Precision} = 15 / 15 = 1$$

$$\text{Versicolor} : \text{Precision} = 13 / (13 + 1) = 0.9286$$

$$\text{Virginica} : \text{Precision} = 14 / (2 + 14) = 0.875$$

### Class Recall (클래스 별 재현율)

재현율이란, 실제 클래스 중 모델이 그 클래스로 올바르게 예측한 비율을 말한다.

$$\text{Setosa} : \text{Recall} = 15 / 15 = 1$$

$$\text{Versicolor} : \text{Recall} = 13 / (13 + 2) = 0.8667$$

$$\text{Virginica} : \text{Recall} = 14 / (1 + 14) = 0.9333$$

# (1) 전처리 기준

## 1-1. 결측치 제거

```
# 결측치가 있는 행 모두 제거
clean_data <- na.omit(data)
```

결측치 데이터 양이 많아서 평균치를 내어 대체 하는 방법도 있지만,  
2002-2007년 기준으로 비어있는 열들이 많아 전부 제거하기로 결정함

```
### 청력 전처리 ###
# 청력(좌)와 청력(우)에 대해 3을 포함한 행 제거
clean_data <- clean_data[!(clean_data$청력.좌. == 3 | clean_data$청력.우. == 3), ]
```

# 청력 전처리  
1:정상, 2:비정상 이므로 이 외의 값(3)을 가진 row 는 제거.  
... 항목별 코드표에 따름

```
### 치석 데이터 전처리 ###
# 치석 값이 0 또는 1이 아닌 값(즉, 2)을 가진 행 제거
clean_data <- clean_data[clean_data$치석 %in% c(0, 1), ]
```

# 치석 전처리  
0:없음, 1:있음 이므로 이 외의 값(2)을 가진 row 는 제거.  
... 항목별 코드표에 따름

## 1-2. 범주형 변수 one hot 인코딩

정상 0, 그 외 1 로 변경

```
# 청력(좌)와 청력(우)에 대해 1은 0으로, 2는 1로 변환
clean_data$청력.좌. <- ifelse(clean_data$청력.좌. == 1, 0, 1)
clean_data$청력.우. <- ifelse(clean_data$청력.우. == 1, 0, 1)
```

# 청력 전처리  
1 정상, 2 비정상을 0정상 1비정상으로 변경  
... 항목별 코드표에 따름

```
### 식전 혈당 전처리 ###
# 값이 126 이상인 경우 1, 그 외는 0으로 대체
clean_data$식전혈당.공복혈당. <- ifelse(clean_data$식전혈당.공복혈당. >= 126, 1, 0)
```

# 청력 전처리  
126이상을 1비정상, 나머지를 0 정상으로 변경  
... 항목별 코드표에 따름


## (2) 회귀분석

02.100만 건 건강 데이터 전처리

2-1. 회귀 분석을 위한 독립 변수 6개 선택  
- 상관계수 분석을 통해 다중 공선성 확인


2-2. 이분형 로지스틱 회귀 분석

왜 결정 트리에서는 데이터를 보고, 분할하고 knn에서는 전체 랜덤 분할하는 가에 대한 질문


 decision tree 를 했을 때는  
# install.packages("party") # 결정 트리(decision tree) 관련  
library(nrtv)

table(D1\_pred, y)

이런식으로 y 의 데이터를 보고 분류를 해서 진행했으면서 왜  
knn 에서는 이렇게 안하고 랜덤을 하는건가요?

 질문 주신 내용에 대한 해석과 이유를 설명드리겠습니다.

결정 트리와 KNN에서 데이터 분할 방식의 차이

결정 트리와 KNN 모두 지도학습 알고리즘으로, 훈련 데이터를 기반으로 모델을 학습시키고, 테스트 데이터를 사용하여 모델의 예측 성능을 평가합니다. 그러나 훈련 데이터와 테스트 데이터를 구성하는 방식은 상황에 따라 달라질 수 있습니다. 

1. 결정 트리에서의 데이터 분할 방식:
- 결정 트리에서는 종종 클래스 간의 균형을 맞추기 위해 각 클래스에서 동일한 수의 샘플을 선택하는 전략을 사용합니다. 이는 모델이 한 클래스에 편향되지 않도록 하기 위함입니다.
  - 예시에서는 각 붓꽃 품종별로 25개씩 샘플을 선택했습니다. 이 방법은 각 클래스가 훈련 데이터에서 동등하게 대표되도록 하여, 모든 클래스에 대해 학습할 수 있도록 합니다.
2. KNN에서의 랜덤 데이터 분할 방식:
- KNN에서는 데이터의 랜덤 샘플링을 통해 훈련 데이터와 테스트 데이터를 생성하는 것이 일반적입니다. 이는 데이터셋의 다양성을 보장하고, 모델이 데이터의 무작위성에 강하게 만들어 일반화 성능을 높입니다.
  - KNN은 근접한 이웃 데이터 포인트를 기반으로 예측을 수행하기 때문에, 무작위로 데이터를 섞어 다양한 시나리오에서 모델의 robustness(강인성)를 테스트하는 것이 유용합니다.
  - 또한, KNN은 데이터 분포에 매우 민감하므로, 다양한 데이터 포인트가 훈련과 테스트에 포함되도록 하는 것이 중요합니다.

왜 다르게 데이터를 분할하는가?

- 데이터의 특성:** 데이터셋의 특성과 클래스 분포에 따라 적합한 데이터 분할 방식이 달라질 수 있습니다. 클래스 불균형이 심한 경우, 각 클래스에서 동등한 수의 샘플을 추출하는 방식이 더 적합할 수 있습니다.
- 모델의 특성:** KNN과 결정 트리는 작동 방식이 다릅니다. KNN은 모든 데이터 포인트 간의 거리를 계산하여 결정을 내리므로, 데이터의 분포가 예측 성능에 큰 영향을 미칩니다. 반면 결정 트리는 특성 공간을 분할하여 예측을 수행하므로, 각 클래스를 고르게 표본 추출하는 것이 더 중요할 수 있습니다.

결국, 데이터 분할 방법은 사용하는 모델, 데이터셋의 특성, 그리고 특정 문제에 대한 요구 사항에 따라 최적화되어야 합니다. 이러한 결정은 모델의 성능을 최대화하고, 더 일반화된 예측을 가능하게 하는 데 중요한 역할을 합니다.

# ChatGPT – 4o 사용 자료 첨부

## 인공신경망 R script 작성 시, 목표 변수가 Species 가 아닌 각각의 열인 이유

왜 목표변수가 Species 가 아니라 setosa, versicolor, virginica 인가요?

좋은 질문입니다! 목표 변수를 Species 대신 setosa, versicolor, virginica 로 설정한 이유는 One-hot Encoding 때문입니다.

### One-hot Encoding이란?

머신러닝과 딥러닝 모델에서 범주형 데이터(예: Species)를 처리할 때는 숫자로 변환해야 합니다. 그러나 단순히 각 범주를 1, 2, 3 등의 숫자로 변환하면 모델이 이 값을 연속적인 숫자로 오인할 수 있습니다. 즉, setosa = 1, versicolor = 2, virginica = 3 으로 변환하면 모델이 '3(virginica)'가 '1(setosa)'보다 크다는 의미로 해석할 수 있습니다. 하지만 이런 의미는 범주형 데이터에는 적합하지 않죠. 그래서 범주형 변수를 처리할 때 One-hot Encoding을 사용하는 것이 일반적입니다.

### One-hot Encoding 적용 방식

One-hot Encoding을 하면 각 범주에 대해 새로운 이진 변수를 만듭니다. 예를 들어, Species 는 세 가지 품종이 있으므로 세 개의 새로운 변수를 생성합니다:

- setosa: 품종이 setosa 이면 1, 아니면 0
- versicolor: 품종이 versicolor 이면 1, 아니면 0
- virginica: 품종이 virginica 이면 1, 아니면 0

이 방식은 각 품종을 별도의 이진 변수로 나누어 줌으로써, 모델이 각 품종을 독립적으로 예측할 수 있게 도와줍니다.

### 왜 이 방법이 필요한가?

- 분류 문제에서 신경망은 여러 범주를 예측해야 하는데, 이때 범주형 변수를 원활하게 처리할 수 있는 방법이 필요합니다. One-hot Encoding을 통해 모델이 각 범주에 대한 독립적인 확률을 계산하도록 만들 수 있습니다.
- 신경망의 출력층은 각 범주(여기서는 세 가지 품종)에 대해 확률 값을 반환하게 됩니다. 각 범주가 독립적인 출력 노드로 표현되면, 각 노드가 출력층에서 0 또는 1로 예측되는 구조가 되어, 해당 노드의 값이 높을수록 해당 품종에 속할 확률이 높은 것으로 판단합니다.

### 예시

만약 입력 데이터를 신경망에 넣었을 때 예측 값이 setosa = 0.2, versicolor = 0.7, virginica = 0.1 이라고 나왔다면, versicolor 가 예측된 결과입니다. 각 품종에 대한 예측 결과는 신경망의 출력층에서 계산됩니다.

따라서 species 가 아닌 setosa, versicolor, virginica 로 나눈 이유는 각 품종에 대한 예측을 독립적으로 수행하게 하기 위해서입니다. 이를 통해 모델이 정확하게 각 품종에 대해 확률을 예측할 수 있고, 그 결과로 가장 높은 확률을 가지는 품종을 선택할 수 있습니다.

이해가 되셨나요? 더 궁금한 점이 있으면 말씀해 주세요!