빅 데이터 분석을 위한 알고리즘 활용

빅 데이터분석을 위한 알고리즘 소개 및 Python을 활용한 실습

SUNGHEE KUK

DATA Analysis

2025
프로그래밍

CONTENTS

4.1 선형 회귀 모형

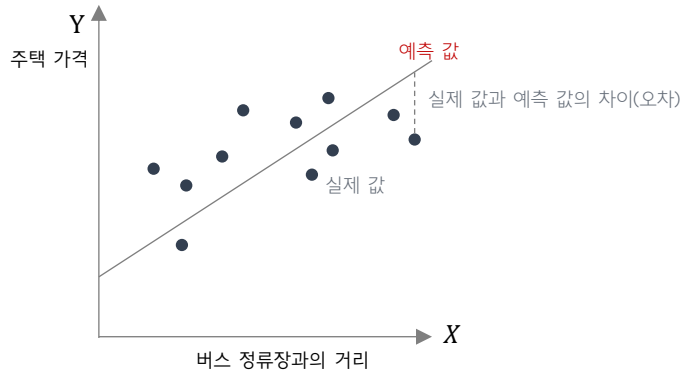
- 선형 회귀 모형(linear regression model)은 연속형 타겟 변수(Y)와 설명 변수(X)를 다음과 같은 **선형적 함수 관계로 모형화 하는 방법**으로 통계학 및 데이터 분석에서는 가장 기본으로 사용되는 주요 알고리즘 중 하나입니다.
- 머신 러닝 회귀 예측의 핵심은 데이터 기반으로 학습을 통한 최적의 **회귀 계수**를 찾아내는 것입니다.

선형 회귀 모형(linear regression model)

- 설명 변수(X)가 주어졌을 때 타겟 변수(Y)가 어떤 관계를 나타내는지를 다음과 같은 형태의 함수로 모델링하고 예측하는 통계적 방법

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$

- 일반적인 선형 회귀 모형(linear regression model)은 타겟 변수(Y)가 연속형일 때 사용됨
  ex. 주택 가격, 상품 가격, 상품 판매 수, …



선형 회귀 모형의 종류

- 선형 회귀 모형은 설명 변수의 개수에 따라 **단순 선형 회귀 모형(Simple linear regression)**과 **다중 선형 회귀 모형(Multiple linear regression)**으로 구분 됨

단순 선형 회귀(Simple linear regression)

- 설명 변수(X)와 타겟 변수(Y)가 하나 씩인 선형 회귀 모형을 단순 선형 회귀 모형이라고 함

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \qquad i = 1, \ldots, n$$

다중 선형 회귀(Multiple linear regression)

- 설명 변수(X)가 둘 이상인 선형 회귀 모형을 다중 선형 회귀 모형이라고 함

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip} \qquad i = 1, \ldots, n$$

4.1 선형 회귀 모형

- 선형 회귀 모형(linear regression model)은 해석이 직관적이라는 장점을 가지고 있으나, 데이터의 수와 이상치에 영향을 많이 받고 실제 데이터에서 만족하기 어려운 가정들을 가지고 있는 단점을 가지고 있습니다.
- 이상치는 데이터에서 발생하는 일반적인 결과와 극단적으로 다른 데이터 값을 의미합니다.

장점과 단점

장점

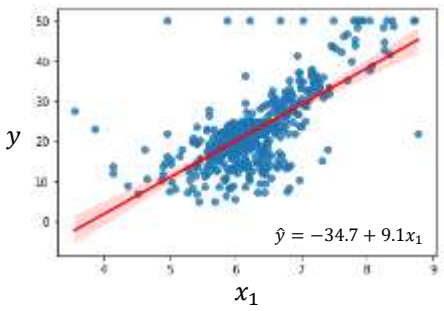
- 타겟 변수(Y)의 예측 값이 왜 이런 값을 갖게 되었는지에 대한 해석이 직관적임
- 설명 변수(X)들로부터 Y로의 영향력의 크기 비교가 가능하고 의미 있는 인자들을 선택할 수 있음

* 인자 : 설명 변수로 사용한 각각의 데이터 특성을 의미함
ex. 버스 정류장과의 거리, 성별, ..

단점

- 데이터 수가 많아질수록 연산 속도가 기하급수적으로 느려짐 (데이터가 적을 때는 가장 빠름)
- 이상치의 영향을 매우 많이 받을 수 있음
- 좋은 모형을 얻기 위한 비현실적인 가정이 많음 (정규성, 독립성, 등분산성, X들 간의 독립성.. 등)
- 다양한 비현실적 가정을 맞추기 위해 수작업과 주관적 판단이 많이 필요됨

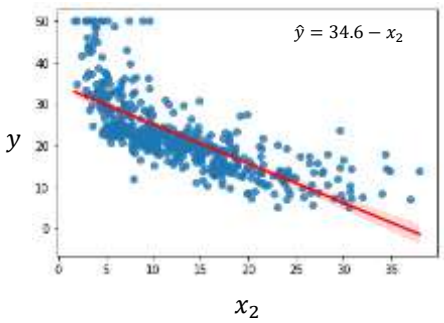
결과 예시 및 해석



y와 x1은 양의 상관 관계(선형 관계)가 있어 보임

단순 회귀 모형으로부터 절편 값은 -34.7, 회귀 계수는 9.1로 추론됨

x1이 한 단위 증가 할 때, y값은 9.1배 증가함



y와 x2는 음의 상관 관계(선형 관계)가 있어 보임

단순 회귀 모형으로부터 절편 값은 34.6, 회귀 계수는 -1로 추론됨

x2이 한 단위 증가 할 때, y값은 -1배 증가함

4.2 로지스틱 회귀 모형

- 타겟 변수(Y)가 정규 분포인 경우인 기존의 선형 회귀 모형에서 다양한 타겟 변수(Y)의 분포들을 고려하여 설명 변수(X)들과의 선형 관계로 확장시킨 모형을 일반화 선형 모형(generalized linear model)이라고 합니다.
- 일반화 선형 모형에는 타겟 변수(Y)의 분포에 따라 다양한 모형이 존재 합니다.

일반화 선형 모형(generalized linear model)

- 타겟 변수(Y)와 설명 변수(X)들의 선형 관계를 고려하기 위하여 선형 예측치와 타겟 변수(Y)를 연결하기 위한 연결 함수(link function)을 사용함
- 예를 들어, 타겟 변수(Y)가 **이항 분포**인 경우 선형 예측치와 E(Y)의 범위가 일치하지 않는 문제가 생기기 때문에 연결 함수(link function)을 고려하여 모델링 함

$$g(E(Y)) = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_p x_{ip}$$

g() : link function

서로 간의 범위가 다르기 때문에 연결될 수가 없기 때문에,
연결 함수 (g())을 통해 일치 시켜 줌

- 타겟 변수(Y)의 분포에 따라 다른 연결 함수를 사용하며, 분포에 따라 사용하는 모형이 다름

일반화 선형 회귀 모형의 종류

- 타겟 변수(Y)의 분포에 따라 선형 회귀 모형의 종류가 분류되며 자주 사용되는 분포는 다음과 같음

■ 이항 분포 / 베르누이 분포

- 성공 또는 실패처럼 두 가지 결과로만 생성된 데이터
  ex. 생산된 제품이 양품 / 불량
- 타겟 변수가 이항 분포에 해당하는 경우 "로지스틱 회귀 모형(logistic regression model)"을 활용할 수 있음

■ 포아송 분포 (포아송 회귀 분석)

- 어떤 사건이 발생하는 건수와 같은 결과로 생성된 데이터
  ex. 한 야구경기에서 실책이 발생한 수
  국내에 발생하는 진도 4이상의 지진의 횟수
  특정 영화관에서 영화를 관람한 사람의 수
- 타겟 변수가 포아송 분포에 해당하는 경우 "포아송 회귀 모형(Poisson regression model)"을 활용함

4.2 로지스틱 회귀 모형

- 성공 또는 실패처럼 두 가지 결과를 분류하기 위한 일반화 선형 모형은 타겟 변수의 분포를 이항 분포로 가정하고 사용 합니다.
- 이 때 사용하는 연결 함수(link function)로 logit을 사용하는 모형을 **로지스틱 회귀 모형(logistic regression model)**이라고 합니다.

장점과 단점

장점

- 이항 분포의 Y에 대한 합리적인 예측 값을 구할 수 있음
- X들로부터 Y로의 영향력의 크기 비교가 가능하고 유의한 X인자들을 선택할 수 있음
- 이진 분류 뿐만 아니라 희소한 영역을 분류 하는 곳에서도 뛰어난 예측 성능을 보임

단점

- 데이터 수가 많아질수록 연산 속도가 기하급수적으로 느려짐 (데이터가 적을 때는 가장 빠름)
- 이상치의 영향을 매우 많이 받을 수 있음
- 좋은 모형을 얻기 위한 비현실적인 가정이 많음 (정규성, 독립성, 등분산성, X들 간의 독립성.. 등)
- 다양한 비현실적 가정을 맞추기 위해 수작업과 주관적 판단이 많이 필요됨
- 회귀 계수에 대한 해석을 비 전공자가 이해하기 어려울 수 있음 ex. Odds ratio

결과 예시 및 해석

```
Optimization terminated successfully.
         Current function value: 0.193036
         Iterations 9
                           Logit Regression Results
Dep. Variable:      DIAG RES          No. Observations:  569
Model:              Logit             Df Residuals:      565
Method:             MLE               Df Model:          3
Date:               Wed, 22 Jun 2022  Pseudo R-squ.:    0.7077
Time:               12:55:11          Log-Likelihood:    -109.84
converged:          True              LL-Null:           -375.72
Covariance Type:    nonrobust         LLR p-value:       6.227e-115
              coef    std err     z     P>|z|  [0.025  0.975]
x1         -2.7594      1.613  -1.710   0.087  -5.922   0.403
x2          1.1749      0.194   6.053   0.000   0.794   1.555
x3          8.2305      1.811   4.545   0.000   4.682  11.780
x4          1.5213      0.215   7.078   0.000   1.100   1.943
```

x1은 한 단위 증가할 수록 y가 1일 확률이 감소하고,
x2~x4은 한 단위 증가할 수록 y가 1일 확률이 증가함

각각의 회귀 계수들은 선형 회귀 계수와는 다르게 "odds ratio" 라는 값으로 해석해야 함

4.3 실습

- 선형 회귀 모형과 로지스틱 회귀 모형을 실습하기 위한 데이터로 "보스턴 주택 데이터"와 "위스콘신 유방암 데이터"를 사용합니다.
- Python을 활용하여 각 데이터에 대한 예측 모형을 생성할 수 있습니다.

보스턴 주택 데이터

데이터 설명

- 보스턴 시의 주택 가격에 대한 데이터로 주택의 여러가지 요건들과 주택의 가격 정보가 포함되어 있음
- 보스턴 인근의 주택 가격의 중앙값을 기준으로 하고 있으며 여러 개의 측정 지표들을 포함하고 있음
- 총 14개의 column과 506개의 row를 가지고 있음

- **CRIM** : 지역별 범죄 발생률
- **ZN** : 25,000평방피트를 초과하는 거주 지역의 비율
- **INDUS** : 비상업 지역 넓이 비율
- **CHAS** : 찰스강의 경계에 위치 여부
- **NOX** : 일산화질소 농도
- **RM** : 거주할 수 있는 방 개수
- **AGE** : 1940년 이전에 건축된 소유 주택의 비율
- **DIS** : 5개 주요 고용센터까지의 가중 거리
- **RAD** : 고속도로 접근 용이도
- **TAX** : 10,000달러당 재산세율
- **PTRATIO** : 지역의 교사와 학생 수 비율
- **B** : 지역의 흑인 거주 비율
- **LSTAT** : 하위 계층의 비율
- **PRICE** : 본인 소유의 주택 가격 (중앙값)

데이터 형태

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

위스콘신 유방암 데이터

데이터 설명

- 위스콘신 유방암 데이터는 종양의 크기, 모양 등의 다양한 속성 값을 기반으로 해당 종양이 악성인지 양성인지를 분류한 데이터임
- 총 31개의 column과 569개의 row를 가지고 있으나 대표적인 4개의 설명변수만 사용

- **mean radius** : 중심에서 외벽까지 거리들의 평균값
- **mean texture** : 평균 질감
- **mean area** : 평균 면적
- **mean symmetry** : 대칭 정도
- **DIAG RES** : 양성 여부 (1: 양성 : 0: 악성)

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0
5	12.45	15.70	477.1	0.2087	0
6	18.25	19.98	1040.0	0.1794	0
7	13.71	20.83	577.9	0.2196	0
8	13.00	21.82	519.8	0.2350	0
9	12.46	24.04	475.9	0.2030	0

4.3 실습 (선형 회귀 모형)

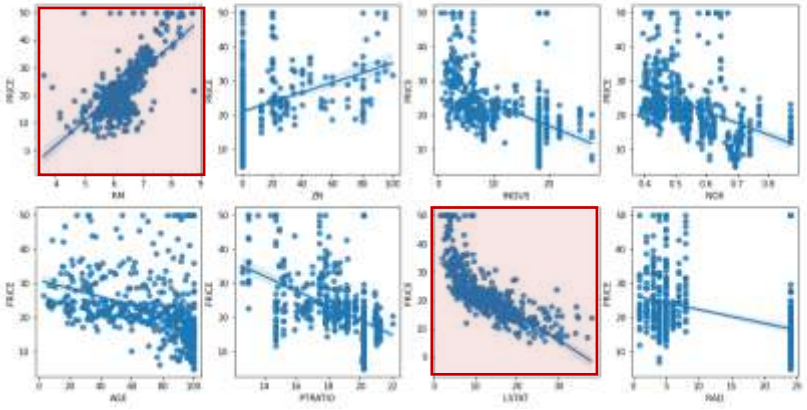
```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from scipy import stats
    from sklearn.datasets import load_boston
    %matplotlib inline
```

```
[2] # boston 데이터 로드
    boston = load_boston()

    # boston 데이터 DataFrame으로 변환
    bostonDF = pd.DataFrame(boston.data, columns = boston.feature_names)
```

```
[3] # boston 데이터의 타겟 변수와 정의
    bostonDF['PRICE'] = boston.target
    bostonDF.shape
    bostonDF.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
[4] # 타겟 변수와 설명 변수간의 상관 관계 확인을 위한 시각화
    # 2행의 합과 4개의 열을 가진 multi-plot
    fig, axs = plt.subplots(figsize = (16,8), ncols = 4, nrows = 2)
    lm_features = ['RM', 'ZN', 'INDUS', 'NOX', 'AGE', 'PTRATIO', 'LSTAT', 'RAD']
    for i, feature in enumerate(lm_features):
      row = int(i/4)
      col = i%4
      sns.regplot(x=feature, y='PRICE', data = bostonDF, ax=axs[row][col])
```



Y

X

```
[5] from sklearn.linear_model import LinearRegression
    from sklearn.metrics import r2_score
    # 타겟 변수와 설명 변수 데이터를 구분
    y_target = bostonDF['PRICE']
    X_data = bostonDF.drop(['PRICE'], axis = 1, inplace = False)

    # 선형 회귀 모델 생성
    LM = LinearRegression()
    LM.fit(X_data, y_target)

    # 예측값 생성
    y_preds = LM.predict(X_data)

    # 예측값 비교
    pd.DataFrame({'y_target' : y_target, 'y_preds' : y_preds})
```

	y_target	y_preds
0	24.0	30.003843
1	21.6	25.025562
2	34.7	30.567597
3	33.4	28.607036
4	36.2	27.943524
...	...	...
501	22.4	23.533341
502	20.6	22.375719
503	23.9	27.627426
504	22.0	26.127967
505	11.9	22.344212

506 rows × 2 columns

```
[6] print('intercept:', np.round(LM.intercept_ ,1))
    print('coefficients:', np.round(LM.coef_ ,1))
```

```
intercept: 36.5
coefficients: [ -0.1   0.    0.    2.7 -17.8   3.8   0.   -1.5   0.3  -0.   -1.    0.
  -0.5]
```

```
coeff = pd.Series(data=np.round(LM.coef_ ,1), index = X_data.columns)
coeff
```

```
CRIM      -0.1
ZN         0.0
INDUS      0.0
CHAS       2.7
NOX      -17.8
RM         3.8
AGE        0.0
DIS       -1.5
RAD        0.3
TAX       -0.0
PTRATIO   -1.0
B          0.0
LSTAT     -0.5
dtype: float64
```

NOX의 회귀 계수 값이
다른 값에 비해 – 방향으로 너무 큼

$$\widehat{PRICE} = 36.5 - 0.1CRIM + 2.7CHAS - 17.8NOX + 3.8RM - 1.5DIS + 0.3RAD - 1PTRATIO - 0.5LSTAT$$

다른 모든 값이 동일한 경우,
RM(거주할 수 있는 방 개수)가 1개 증가할 수록 주택 가격이 3.8만큼 증가함

4.3 실습 (로지스틱 회귀 모형)

```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn.datasets import load_breast_cancer
    %matplotlib inline
```

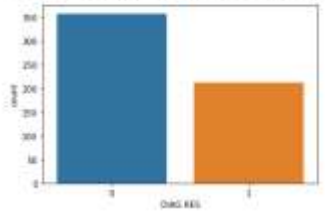
```
[2] # 유방암 데이터 로드
    cancer = load_breast_cancer()

    # 유방암 데이터 DataFrame으로 변환
    cancerDF = pd.DataFrame(cancer.data, columns = cancer.feature_names)
```

```
[3] # 일부 변수만 사용
    cancerDF = cancerDF[['mean radius', 'mean texture', 'mean area', 'mean symmetry']]

    # 유방암 데이터의 타겟 변수와 정의 (1이면 양성 종양, 0이면 악성 종양)
    cancerDF['DIAG RES'] = cancer.target
    # 보통 1을 주요 타겟이 되는 범주로 정의함
    cancerDF['DIAG RES'] = np.where(cancerDF['DIAG RES'] == 0, 1, 0)
    cancerDF.shape
    cancerDF.head(10)
```

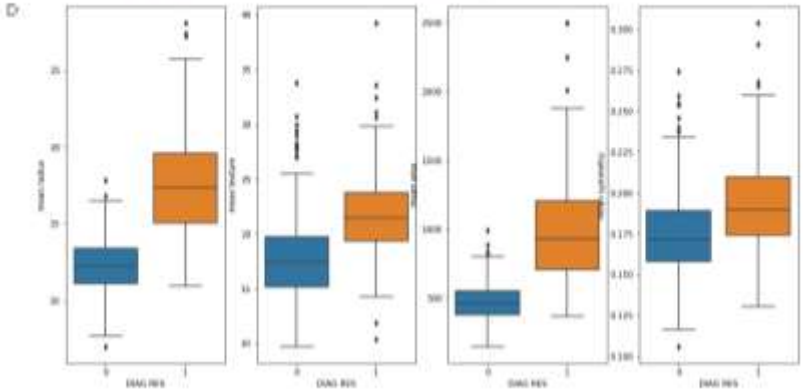
	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	1
1	20.57	17.77	1326.0	0.1812	1
2	19.69	21.25	1203.0	0.2069	1
3	11.42	20.38	386.1	0.2597	1
4	20.29	14.34	1297.0	0.1809	1
5	12.45	15.70	477.1	0.2087	1
6	18.25	19.98	1040.0	0.1794	1
7	13.71	20.83	577.9	0.2196	1
8	13.00	21.82	519.8	0.2350	1
9	12.46	24.04	475.9	0.2030	1

```
[4] # 진단 결과에 대한 분포 확인
    sns.countplot(cancerDF['DIAG RES'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feb36ec5e50>
```



```
# 입력 / 종속변 설명 변수의 분포 확인
fig, axs = plt.subplots(figsize = (16,8), ncols = 4, nrows =1)
for i, feature in enumerate(['mean radius', 'mean texture', 'mean area', 'mean symmetry']):
    sns.boxplot(x='DIAG RES', y = feature, data = cancerDF, ax = axs[i])
```



```
[5] from sklearn.linear_model import LogisticRegression

    # 타겟 변수와 설명 변수 데이터를 구분
    y_target = cancerDF['DIAG RES']
    X_data   = cancerDF.drop(['DIAG RES'], axis = 1, inplace = False)

    # 표준화 작업
    scaler = StandardScaler()
    scaler.fit(X_data)
    X_data = pd.DataFrame(scaler.transform(X_data), columns = X_data.columns)

    # 로지스틱 회귀 모형 생성
    Logistic = LogisticRegression()
    Logistic.fit(X_data, y_target)
```

```
LogisticRegression()
```

```
[7] # 회귀 계수 확인
    colum_name = ['const'] + X_data.columns.tolist()
```

```
[8] # 베타 값 추출
    beta = np.concatenate([Logistic.intercept_, Logistic.coef_.reshape(-1)]).round(2)
    # exp(베타) 값 추출
    odds = np.exp(beta).round(2)
    # beta값 비교
    beta_analysis = pd.DataFrame(np.c_[beta, odds], index = colum_name, columns = ['beta', 'exp(beta)'])
    beta_analysis
```

	beta	exp(beta)
const	-0.57	0.57
mean radius	2.05	7.77
mean texture	1.08	2.94
mean area	2.06	8.00
mean symmetry	1.37	3.94

4.3 실습 (로지스틱 회귀 모형)

[9]
```
# 예측 확률
probs = Logistic.predict_proba(X_data)[:,1]

# 타겟 변수 예측
y_preds = np.where(probs.reshape(-1) >= 0.5, 1, 0)
```
타겟 변수가 1일 확률이 0.5보다 크거나 같을 경우 1로 분류하고, 작은 경우 0으로 분류함
```
# 예측값 비교
pd.DataFrame({'y_target' : y_target,'y_preds' : y_preds})
```

	y_target	y_preds
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
564	1	1
565	1	1
566	1	1
567	1	1
568	0	0

569 rows × 2 columns

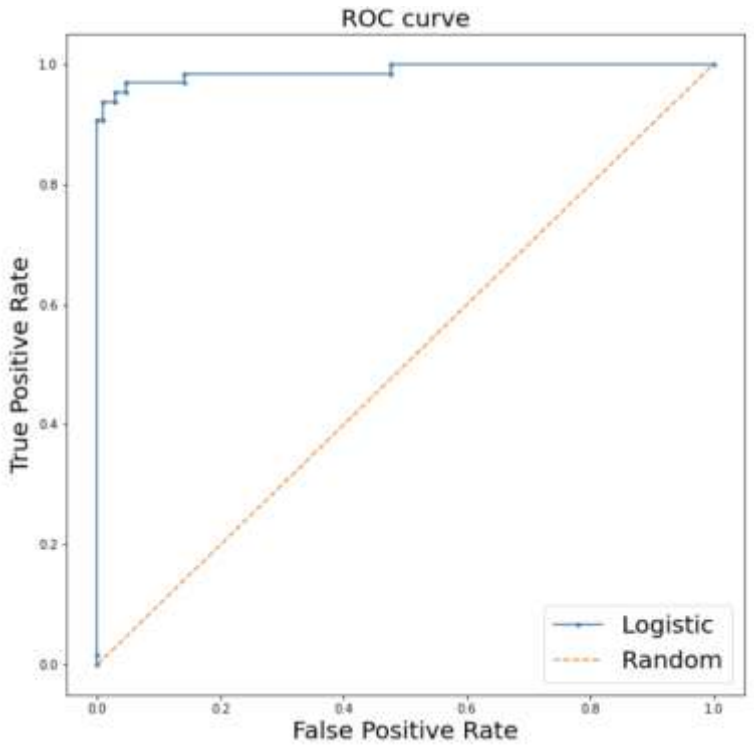
[10]
```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_target, y_preds))
```

실제 값 \ 예측 값	0	1
0	343	14
1	30	182

[11]
```
# 타겟 변수 예측
y_preds = np.where(probs.reshape(-1) >= 0.8, 1, 0)
```
타겟 변수가 1일 확률이 0.8보다 크거나 같을 경우 1로 분류하고, 작은 경우 0으로 분류함
```
# 예측값 비교
pd.DataFrame({'y_target' : y_target,'y_preds' : y_preds})

# confusion matrix
pd.DataFrame(confusion_matrix(y_target, y_preds))
```

실제 값 \ 예측 값	0	1
0	352	5
1	54	158

어떤 결과가 좋은 결과 인가?



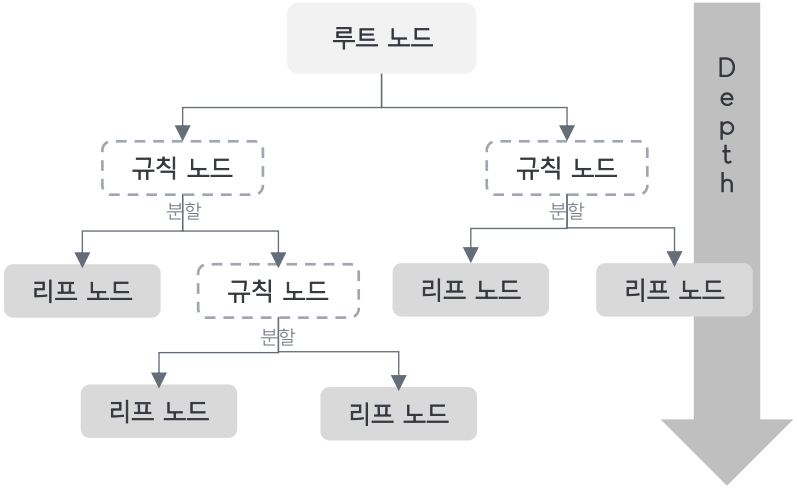
4.5 의사 결정 나무 (Decision Tree)

- 의사 결정 나무(Decision Tree)는 머신 러닝 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘이며, 예측을 위한 알고리즘 뿐만 아니라 정상적인 값들에 비하여 분리되어 있는 이상 값들을 탐지하는 등을 위한 방법으로도 많이 사용되는 알고리즘 입니다.
- 의사 결정 나무는 예측력이 좋은 것으로 알려져 있는 랜덤 포레스트(Random Forest)등에 응용되어 사용되는 기본 알고리즘 중 하나 입니다.

의사 결정 나무(Decision Tree)

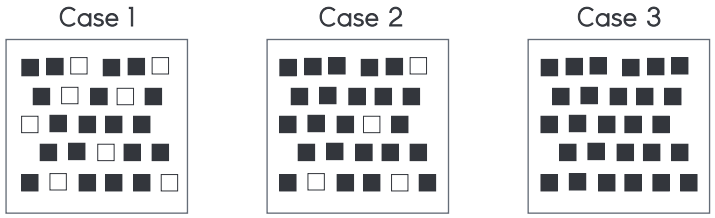
- 의사 결정 나무는 데이터에 있는 규칙을 학습으로 통해 찾아내어 나무(Tree) 기반의 분류 규칙을 만들어내는 알고리즘임
- 의사 결정 나무는 기본적으로 규칙 노드(Decision Node)와 리프 노드(Leaf Node)로 나누어서 생성 됨

* 규칙 노드(Decision Node) : 규칙 조건을 표기하는 노드
리프 노드(Leaf Node) : 규칙 조건에 의해 결정된 결과 값 노드



분할 기준 종류

- 의사 결정 나무는 정보의 불순도가 낮은 데이터로 분할 시킴



어떤 Case의 데이터가 불순도가 낮다고 할 수 있겠는가 ?

- 불순도를 측정하는 대표적인 방법으로는 엔트로피를 이용한 정보 이득(Information Gain)지수, 지니 계수(Gini index)가 있음
- C4.5는 엔트로피를 이용한 의사결정나무 알고리즘이며, CART는 지니 계수를 이용하는 의사결정나무 알고리즘이다.

4.4 의사 결정 나무 (Decision Tree)

- 의사 결정 나무에서 가지를 분할하기 위한 기준으로는 주로 "엔트로피 지수"를 이용한 정보 이득 지수와 "지니 계수"가 대표적입니다.
- 의사 결정 나무는 쉽게 생성이 가능하며 해석이 쉽다는 장점이 있으나 과적합의 문제가 발생할 수 있다는 단점을 가지고 있습니다.

분할 기준 종류

엔트로피 (Entropy)

- 주어진 데이터 집합의 혼잡도를 의미하는 지수로 낮을 수록 불순도가 낮음

$$Entropy(A) = -\sum_{k=1}^{n} p_k \log_2(p_k) \qquad p_k = \frac{\text{주어진 데이터 중 } C_k\text{에 속하는 수}}{\text{주어진 데이터 수}}$$

정보 이득 (Information Gain) 지수

- 이전 엔트로피 지수에서 현재 엔트로피 지수를 뺀 값

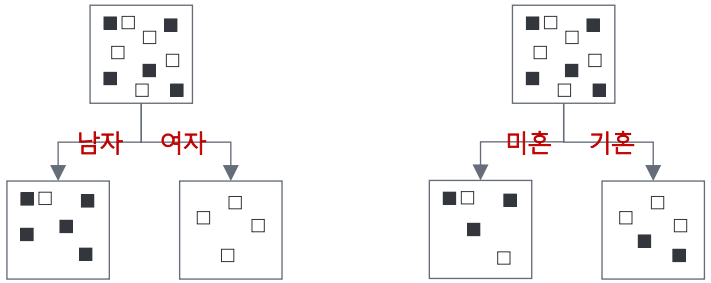
지니 계수 (Gini index)

- 주어진 데이터 집합의 혼잡도를 의미하는 지수로 0에 가까울수록 불순도가 낮고 1에 가까울수록 불순도가 높음

$$G(A) = 1 - \sum_{k=1}^{n} p_k^2 \qquad p_k = \frac{\text{주어진 데이터 중 } C_k\text{에 속하는 수}}{\text{주어진 데이터 수}}$$

분할 기준 예시 및 장단점

분할 기준 지수 예시



- A게임을 탈퇴할 것으로 예상되는 고객을 예측하여 고객 관리를 할 계획일때, 성별과 결혼 유무 중 어떤 조건을 사용해야 하는가

장점

- 구성하기 쉽고, 직관적이기 때문에 해석이 쉬움
- 이상치에 대한 영향이 크지 않음

단점

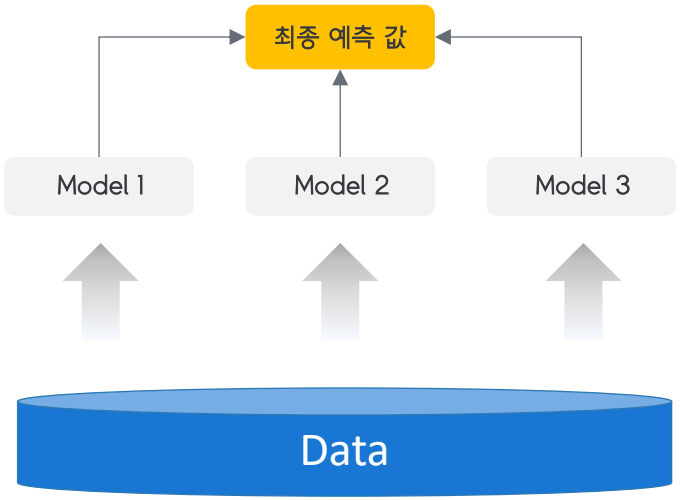
- 과적합의 문제가 발생할 수 있으므로 depth를 잘 조절 해야함

4.5 랜덤 포레스트(Random Forest)

- 랜덤 포레스트(Random Forest)는 1개의 의사 결정 나무(Decision tree)이 아닌 보다 더 좋은 예측 알고리즘을 생성하기 위해 여러 개의 의사 결정 나무를 사용하는 앙상블(Ensemble) 알고리즘 입니다.
- 랜덤 포레스트는 예측 문제에서 매우 높은 활용도를 자랑하는 대표적인 알고리즘 입니다.

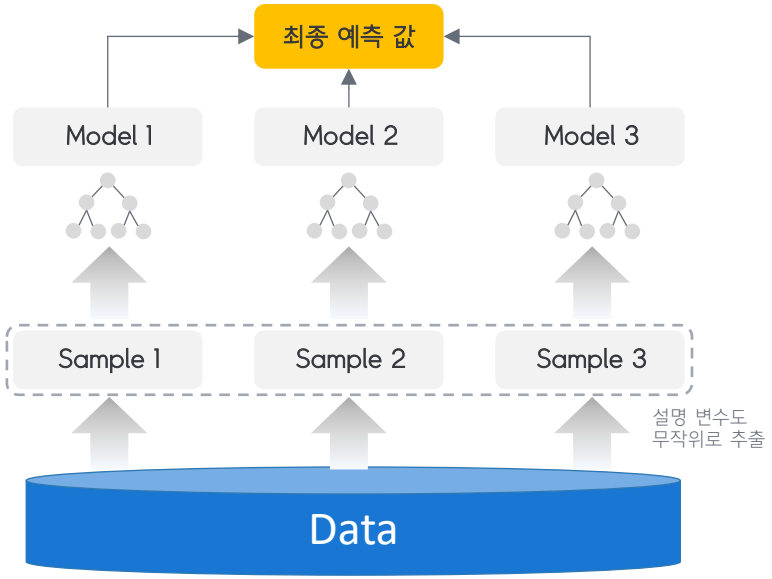
앙상블(Ensemble)

- 앙상블 학습(Ensemble Learning)은 여러 개의 알고리즘을 생성하여 그 예측결과들을 결합함으로써 더 정확한 최종 결과를 도출하는 방법을 의미함
- 앙상블 학습방법은 사용 알고리즘의 종류나, 데이터 샘플링 여부에 따라 구분 될 수 있음



랜덤 포레스트 (Random Forest)

- 랜덤 포레스트(Random Forest)는 의사결정나무(Decision Tree)기반의 알고리즘으로 데이터를 샘플링하여 여러 개의 의사결정나무를 학습한 뒤, 최종적으로 예측 값을 결정하는 알고리즘임



4.5 랜덤 포레스트(Random Forest)

- 랜덤 포레스트(Random Forest)는 의사 결정 나무(Decision Tree)에서의 단점인 과적합(overfitting)문제를 보완하는 방법으로 높은 예측력을 보인다는 장점을 가지고 있습니다.
- 그러나 의사 결정 나무에 비해 여러 개의 나무를 사용하기 때문에 해석의 어려움이 있습니다.

장점과 단점

장점

- 과적합(Overfitting)의 위험이 있는 의사결정 나무에 비해 과적합 문제에 강함
- 기본적으로 분류 및 예측 문제에서 비교적 높은 예측 정확도를 보임
- 이상치에 대한 영향이 크지 않음

단점

- 개별 적인 의사 결정 나무에 대한 분석이 어렵고 복잡한 구조이기 때문에 해석의 어려움이 있음
- 고차원의 데이터나 Sparse한 데이터에는 적절한 학습이 어려움
- 선형 모델에 비해 많은 메모리를 사용하기 때문에 알고리즘을 생성하는데 속도가 느려질 가능성이 있음
- 메모리 사용에 제약에 발생될 수 있음

실습 데이터

데이터 설명

- 위스콘신 유방암 데이터는 종양의 크기, 모양 등의 다양한 속성 값을 기반으로 해당 종양이 악성인지 양성인지를 분류한 데이터임
- 총 32개의 column과 569개의 row를 가지고 있음

- **radius** : 중심에서 외벽까지 거리
- **texture** : 질감
- **area** : 면적
- **perimeter** : 둘레
- **smoothness** : 매끄러운 정도
- **compactness** : 조그만 정도
- **concavity** : 오목함
- **points** : 오목한 점의 수
- **symmetry** : 대칭 정도
- **dimension** : 프랙탈 차원

평균값,
표준 오차값,
제일 큰 3개의 값을 평균낸 값

- **diagnosis** : 양성 여부 (1, 양성 : 0, 악성)

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0

4.6 실습 (Decision Tree)

```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from sklearn.datasets import load_breast_cancer
    from os import system  # Tree 시각화를 위함
    system("pip install graphviz")
    import graphviz
```

```
[2] # 유방암 데이터 로드
    cancer = load_breast_cancer()

    # 유방암 데이터 DataFrame으로 변환
    cancerDF = pd.DataFrame(cancer.data, columns = cancer.feature_names)
```

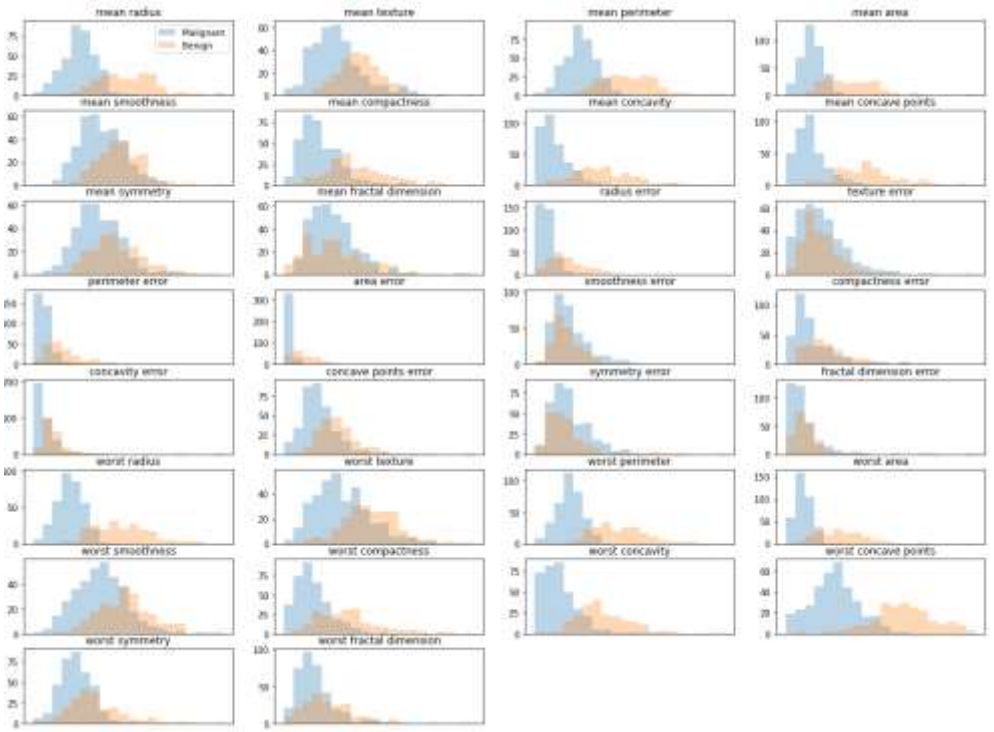
```
[3] # 유방암 데이터의 타겟 변수와 값의 (1이면 양성 종양, 0이면 악성 종양)
    cancerDF['diagnosis'] = cancer.target
    # 보통 1을 주요 타겟이 되는 범주로 정의함
    cancerDF['diagnosis'] = np.where(cancerDF['diagnosis'] == 0, 1, 0)
    cancerDF.shape
    cancerDF.head(10)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864

```
# 진단 결과에 대한 분포 확인
sns.countplot(cancerDF['diagnosis'])
```

```
# 양성 / 음성별 설명 변수의 분포 확인
plt.figure(figsize=[20,15])

malignant = cancerDF[cancerDF.diagnosis==0]
benign = cancerDF[cancerDF.diagnosis==1]

# 설명 변수 데이터 set
X_data = cancerDF.drop("diagnosis", axis = 1, inplace = False)

# 히스토그램을 활용한 모든 설명 변수에 대한 분포 확인
for col in range(30):
    plt.subplot(8,4,col+1)
    _, bins=np.histogram(X_data.iloc[:,col], bins=20)

    plt.hist(malignant.iloc[:,col],bins=bins, alpha=0.3)
    plt.hist(benign.iloc[:,col], bins=bins ,alpha=0.3)
    plt.title(X_data.columns[col])
    if col==0: plt.legend(['Malignant', 'Benign'])
    plt.xticks([])
```

시각화 결과

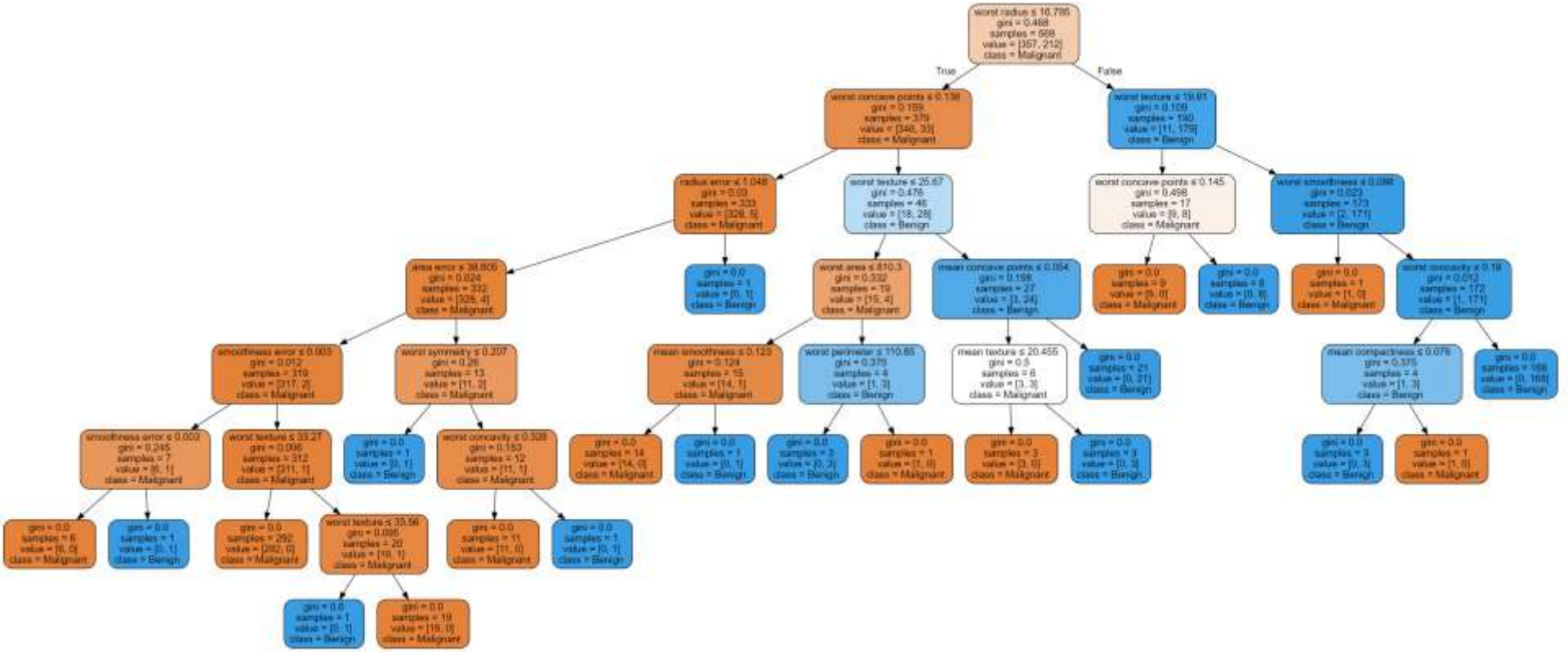


의사결정 모형 생성

```
[6] from sklearn import tree

    # 타겟 변수
    y_target = cancerDF['diagnosis']

    # 의사결정나무(Decision Tree) 모형 생성
    DTree = tree.DecisionTreeClassifier(random_state=300)
    DTree.fit(X_data, y_target)
```

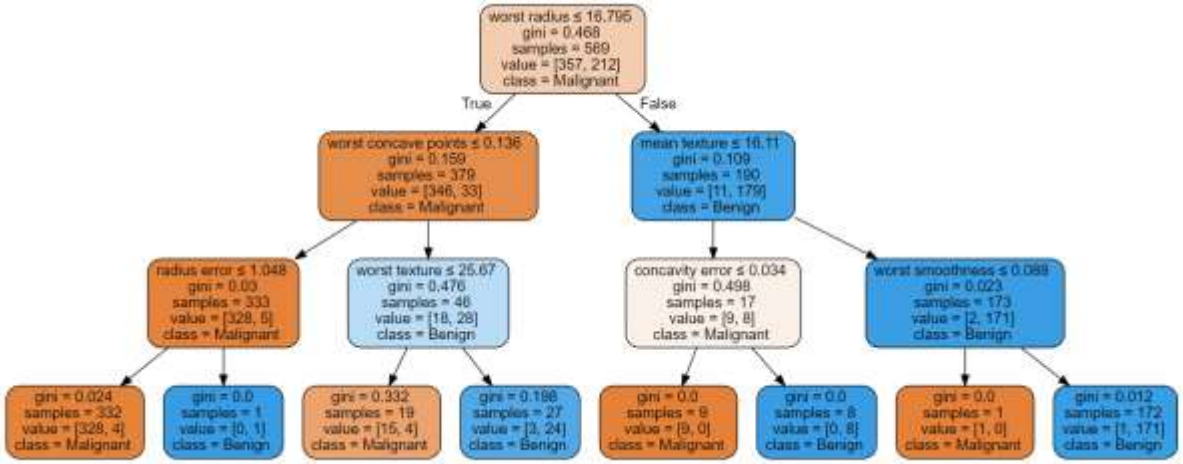
4.6 실습 (Decision Tree)

```
# Decision Tree 그래프로 확인
dot_data = tree.export_graphviz(DTree,
                                out_file = None,                    # file로 변환하지 않음
                                feature_names = X_data.columns,     # 변수명
                                class_names = ['Malignant', 'Benign'], # 타겟 종류
                                filled = True,                      # 색상 채움
                                rounded = True,                     # 반올림 함
                                special_characters = True)          # 특수 문자 사용함

graph = graphviz.Source(dot_data)
graph
```



4.6 실습 (Decision Tree)


```
# 의사결정나무(Decision Tree) 모형 생성 (depth 3으로 조절)
DTree_depth3 = tree.DecisionTreeClassifier(random_state=000, max_depth = 3)
DTree_depth3.fit(X_data, y_target)

# Decision Tree 그래프로 확인 (depth 3으로 조절)
dot_data3 = tree.export_graphviz(DTree_depth3,
                                 out_file = None,                    # file로 변환하지 않음
                                 feature_names = X_data.columns,     # 변수명
                                 class_names = ['Malignant', 'Benign'], # 타겟 종류
                                 filled = True,                      # 색상 채움
                                 rounded = True,                     # 반올림 함
                                 special_characters = True)          # 특수 문자 사용함

graph3 = graphviz.Source(dot_data3)
graph3
```




```
# 타겟 변수 예측 (depth를 주지 않은 것과, 준 것 비교)
y_preds = DTree.predict(X_data)
y_preds3 = DTree_depth3.predict(X_data)

# 예측값 비교
pd.DataFrame({'y_target' : y_target,'y_preds' : y_preds, 'y_preds3' : y_preds3})
```

```
[10] from IPython.core.interactiveshell import InteractiveShell
     InteractiveShell.ast_node_interactivity = "all"
```

```
from sklearn.metrics import confusion_matrix
print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds))
print("\n")
print("Depth를 3으로 제한한 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds3))
```

Depth 조절 별 결과 비교

Depth 제한 없는 의사결정 나무 결과

실제 값 \ 예측 값	0	1
0	357	0
1	0	212

Depth를 3으로 제한한 의사결정 나무 결과

실제 값 \ 예측 값	0	1
0	353	4
1	8	204

4.6 실습 (Random Forest)

랜덤 포레스트(Random Forest) 모형 생성

```
[12] from sklearn.ensemble import RandomForestClassifier

     # 100개의 decision tree를 사용한 랜덤 포레스트 (max_features는 sqrt개수를 사용하는 것이 디폴트임)
     RF = RandomForestClassifier(n_estimators=100, random_state=0)
     RF.fit(X_data, y_target)

     # 예측값 생성
     y_preds_RF = RF.predict(X_data)

     # 예측값 비교
     pd.DataFrame({'y_target' : y_target,'y_preds' : y_preds, 'y_preds3' : y_preds3, 'y_preds_RF' : y_preds_RF})
```

RandomForestClassifier(random_state=0)

	y_target	y_preds	y_preds3	y_preds_RF
0	1	1	1	1
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
...	...	...	...	...
564	1	1	1	1
565	1	1	1	1
566	1	1	1	1
567	1	1	1	1
568	0	0	0	0

569 rows × 4 columns

```
pd.DataFrame(confusion_matrix(y_target, y_preds_RF))
```

	0	1
0	357	0
1	0	212

```
[14] # 의사 결정 나무와 랜덤 포레스트의 결과 비교

     from sklearn.metrics import confusion_matrix
     print("Depth 제한 없는 의사결정 나무 결과")
     pd.DataFrame(confusion_matrix(y_target, y_preds))
     print("₩n")
     print("Depth를 3으로 제한한 의사결정 나무 결과")
     pd.DataFrame(confusion_matrix(y_target, y_preds3))
     print("₩n")
     print("랜덤 포레스트 결과(100 개의 tree 사용)")
     pd.DataFrame(confusion_matrix(y_target, y_preds3))
```

Depth 제한 없는 의사결정 나무 결과

	0	1
0	357	0
1	0	212

Depth를 3으로 제한한 의사결정 나무 결과

	0	1
0	353	4
1	8	204

랜덤 포레스트 결과(100 개의 tree 사용)

	0	1
0	353	4
1	8	204