유방암 진단 데이터
분석 리포트

(Breast Cancer Wisconsin Diagnostic Dataset)

[데이터마이닝] 개인과제_이현지_2020380501

목 차

Ⅰ. 문제정의

위스콘신 대학에서 제공한 자료인 유방암 진단 데이터의 특정한 값을 이용하여 유방 종양이 양성(benign)인지 악성(malignant)인지 판별하고자 한다. 특정 값은 유방 종양의 미세침 흡인물 디지털 이미지에서 측정한 값이며, 디지털 이미지에 나타난 세포핵의 특징이다. 종양을 판별하는데 있어 분류 모델을 사용하고, 이에 대한 성능을 평가하고자 한다.

Ⅱ. 데이터 수집

- 분석 데이터

scikit-learn 유방암 진단 데이터 (Breast cancer wisconsin diagonostic dataset)
(https://scikit-learn.org/stable/datasets/toy_dataset.html#breast-cancer-wisconsin-diagnostic-dataset)

```
import pandas as pd
from sklearn import datasets
load_df = datasets.load_breast_cancer()
```

데이터프레임을 다루기 위한 pandas와 데이터를 불러오기 위한 sklearn 라이브러리를 작성하고, load_df로 유방암 진단 데이터를 불러온다.

```
# array형식의 load_df를 DataFrame형태로 변경
data = pd.DataFrame(load_df.data)

# DataFrame 형태로 바뀐 data의 column을 지정
feature = pd.DataFrame(load_df.feature_names)
data.columns = feature[0]

# target 변수의 column 또한 지정
target = pd.DataFrame(load_df.target)
target.columns = ['target']

# data (columns 포함), target 컬럼으로 데이터 합침

# datad와 target을 열 방향으로 합친다.
# 최종 df 생성
df = pd.concat([data, target], axis=1)
df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

5 rows × 31 columns

- 데이터 속성

Number of Instances : 569

Number of Attributes : 30 numeric

- 변수 설명

・특성변수

radius (mean of distances from center to points on the perimeter)	반지름 (중심점에서 외벽까지 거리들의 평균)
texture (standard deviation of gray-scale values)	질감 (Gray-Sclae 값들의 표준편차)
perimeter	둘레
area	면적
smoothness (local variation in radius lengths)	매끄러움 (반지름 길이의 국소적 변화)
compactness (perimeter^2 / area - 1.0)	조밀성 (둘러^2 / 면적-1.0)
concavity (severity of concave portions of the contour)	오목함 (윤곽의 오목한 부분의 정도)
concave points (number of concave portions of the contour)	오목한 점 (윤곽의 오목한 부분 수)
symmetry	대칭
fractal dimension	프랙탈 차원

특정 변수는 mean(평균), standard error(표준오차), worst(가장 큰값 3개의 평균)로 계산되어 각각 총 30개가 생성됨.

・목적변수

target	목적 변수 (악성(malignant) : 0, 양성(benign) : 1)

```
import numpy as np

np.unique(df.target, return_counts=True)
```

```
(array([0, 1]), array([212, 357], dtype=int64))
```

target 변수는 악성 212개, 양성 357개로 구성되어 있다.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int32  
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

info()를 이용하여 데이터의 형식을 확인해보면 종양의 크기와 모양과 관련된 속성들은 숫자 형태인 float, target 변수는 integer로 구성되어 있음을 알 수 있다.

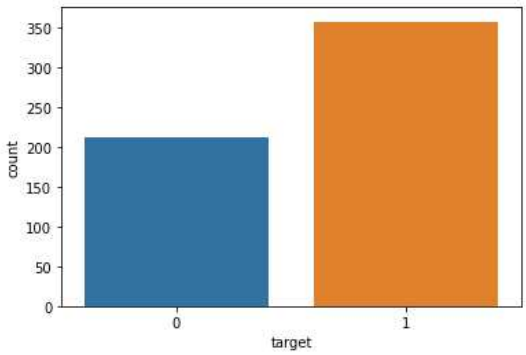
Ⅲ. 데이터 탐색

- 유방암의 발생 빈도 확인

: target 변수는 악성 212개, 양성 357개 구성

```
import seaborn as sns
sns.countplot(df['target'],label="Count")
```

```
<AxesSubplot:xlabel='target', ylabel='count'>
```



- 요약 통계량 확인 (column별 총 데이터수, 평균, 표준편차, 최소 최댓값, 사분위수)

```
df.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...

8 rows × 31 columns

- 결측치 확인

```
df.isnull().sum()
```

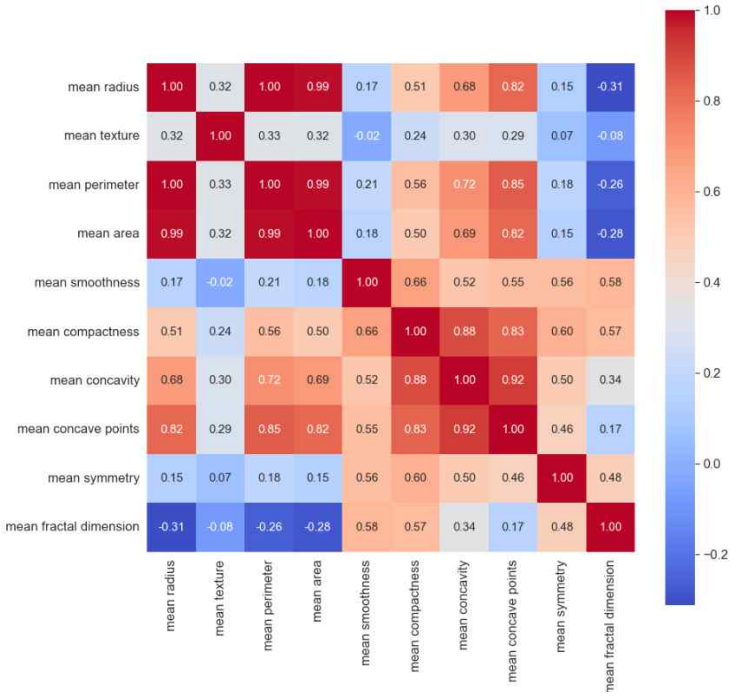
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

- 히트맵 (변수 간 상관관계 확인)

· mean

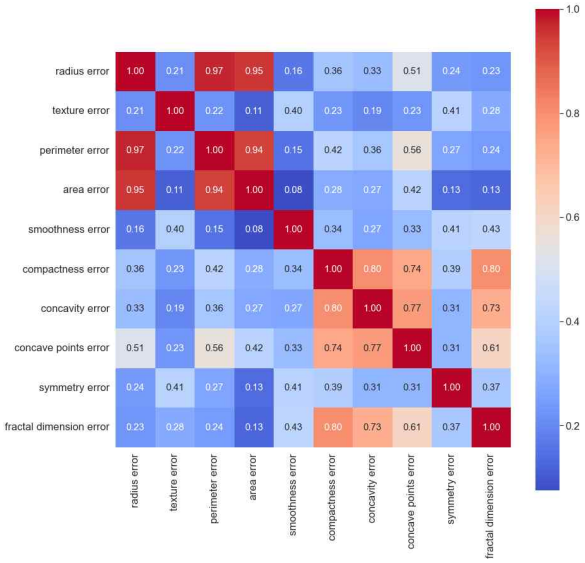
```
features_mean=['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness', 'mean compactness',
               'mean concavity', 'mean concave points', 'mean symmetry', 'mean fractal dimension']
```

```
corr = df[features_mean].corr()
plt.figure(figsize=(14,14))
sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size': 15},
           xticklabels= features_mean, yticklabels= features_mean,
           cmap= 'coolwarm')
```

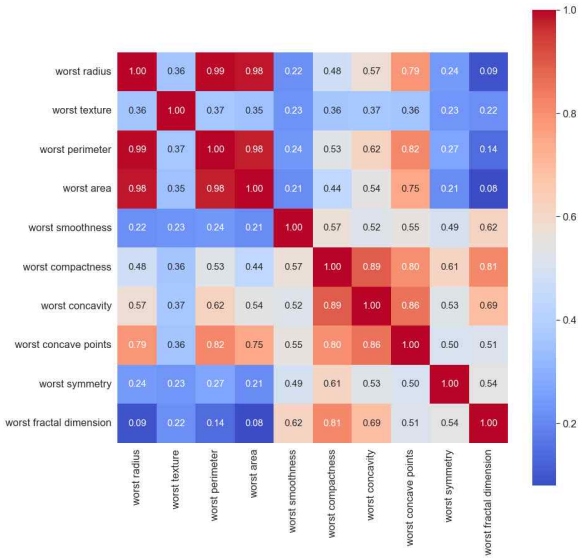


위와 동일한 방식으로 standard error, worst 값의 히트맵 생성

· standard error



· worst



상관관계가 높게 나타나는 변수)

mean : mean compactness, mean concavity, mean concave points

standard error : radius error, perimeter error, area error /
compactness error, concavity error, concave points error

worst : worst radius, worst perimeter, worst area /
worst compactness, worst concavity, worst concave points

위 변수 중 mean concavity, area error, concavity error, worst area, worst concavity 사용

```
drop_list = ['mean compactness', 'mean concave points', 'radius error', 'perimeter error', 'compactness error', 'concave points error',
             'worst radius', 'worst perimeter', 'worst compactness', 'worst concave points']
df_1 = df.drop(drop_list, axis = 1 )
df_1.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean concavity	mean symmetry	mean fractal dimension	texture error	area error	...	concavity error	symmetry error	fractal dimension error	worst texture	worst area	worst smoothness	worst concavity	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.3001	0.2419	0.07871	0.9053	153.40	...	0.05373	0.03003	0.006193	17.33	2019.0	0.1622	0.7119	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.0869	0.1812	0.05667	0.7339	74.08	...	0.01860	0.01389	0.003532	23.41	1956.0	0.1238	0.2416	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.1974	0.2069	0.05999	0.7869	94.03	...	0.03832	0.02250	0.004571	25.53	1709.0	0.1444	0.4504	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.2414	0.2597	0.09744	1.1560	27.23	...	0.05661	0.05963	0.009208	26.50	567.7	0.2098	0.6869	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.1980	0.1809	0.05883	0.7813	94.44	...	0.05688	0.01756	0.005115	16.67	1575.0	0.1374	0.4000	0.2364	0.07678	0

5 rows × 21 columns

전처리 진행 후, 변수의 개수는 target 변수를 제외한 총 20개이다.

Ⅴ. 모델링

분류 기법)

Ⅴ-Ⅰ. K-Nearest Neighbor (KNN)

```
X = df_1.drop('target', axis=1)
y = df['target']
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=30, stratify=y)


KNN_MODEL = KNeighborsClassifier()
KNN_MODEL.fit(x_train, y_train)
prediction = KNN_MODEL.predict(x_test)
```

```
print("Training error       : {0:.3f}".format(KNN_MODEL.score(x_train, y_train)))
print("Testing error        : {0:.3f}".format((prediction==y_test).mean()))
print("cross_val_score      : {0:.3f}".format(cross_val_score(KNN_MODEL, X, y, cv=10).mean()))
```

```
Training error       : 0.948
Testing error        : 0.923
cross_val_score      : 0.924
```

모델링을 진행하기 앞서 목적변수 'target'을 제외한 나머지 열을 input X로 정의, 'target' 열을 label y로 정의한다. 이후, train data set은 75%, test data set은 25%로 데이터의 비율을 유지하며 train set과 test set을 구분한다.

Train set의 예측 정확도는 94.8%, test set의 예측 정확도는 92.3%이다. 이때, 10-fold cross-validation로 모델링을 진행하였을 경우 예측 정확도는 92.4%로 나타났다.

```
train_acc = []
test_acc = []

for n in range(1,15):
    KNN_MODEL = KNeighborsClassifier(n_neighbors=n)
    KNN_MODEL.fit(x_train, y_train)
    prediction = KNN_MODEL.predict(x_test)
    train_acc.append(KNN_MODEL.score(x_train, y_train))
    test_acc.append((prediction==y_test).mean())
```

```
import numpy as np
plt.figure(figsize=(12, 9))
plt.plot(range(1, 15), train_acc, label='Training Dataset')
plt.plot(range(1, 15), test_acc, label='Test Dataset')
plt.xlabel("k")
plt.ylabel("accuracy")
plt.xticks(np.arange(0, 16, step=1))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x2d4468e1070>
```

위 예시는 최적의 k를 찾기 위한 방법이다. K에 따른 training dataset과 test dataset의 accuracy를 가시화하여 그래프로 표현하였다. 이때 test dataset이 가장 높아지는 지점인 6 혹은 8로 k를 지정하여 모델링을 진행할 수 있을 것이다.

```
KNN_MODEL = KNeighborsClassifier(6)
KNN_MODEL.fit(x_train, y_train)
prediction = KNN_MODEL.predict(x_test)

print("Training error       : {0:.3f}".format(KNN_MODEL.score(x_train, y_train)))
print("Testing error        : {0:.3f}".format((prediction==y_test).mean()))
```

```
Training error       : 0.939
Testing error        : 0.937
```

```
KNN_MODEL = KNeighborsClassifier(8)
KNN_MODEL.fit(x_train, y_train)
prediction = KNN_MODEL.predict(x_test)

print("Training error       : {0:.3f}".format(KNN_MODEL.score(x_train, y_train)))
print("Testing error        : {0:.3f}".format((prediction==y_test).mean()))
```

```
Training error       : 0.941
Testing error        : 0.937
```

K=6, K=8일 때의 train data의 예측 정확도는 각각 93.9%, 94.1%이며, test data의 예측 정확도는 93.7%로 동일하다. KNeighborsClassifier() 모델의 default값인 K=5일 때보다 정확도가 높게 나왔음을 확인할 수 있다.

Ⅴ-Ⅱ. Decision Tree (의사결정나무)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=30)

from sklearn.tree import DecisionTreeClassifier
DT_MODEL= DecisionTreeClassifier(random_state=0)
DT_MODEL.fit(X_train, y_train)

prediction=DT_MODEL.predict(X_test)
```

```
# Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix
CM=confusion_matrix(y_test, prediction)
CM_report=classification_report(y_test, prediction)
print('-'*15, 'Confusion Matrix','-'*15)
print(CM)

print('-'*15, 'Confusion Matrix2','-'*15)
import pandas as pd
CM_rename=pd.DataFrame(CM).rename(index={0:'실제값(0)',1:'실제값(1)',2:'실제값(2)'},columns={0:'예측값(0)',1:'예측값(1)',2:'예측값(2)'})
print(CM_rename)
```

```
--------------- Confusion Matrix ---------------
[[47  5]
 [ 5 86]]
--------------- Confusion Matrix2 ---------------
        예측값(0)  예측값(1)
실제값(0)      47       5
실제값(1)       5      86
```

의사결정나무 모델링을 진행한 후, 오차행렬을 통해 분류 결과를 확인해보자면 악성(malignant) - 0과 양성(benign) - 1은 모두 5개가 오분류 되었음을 알 수 있다.

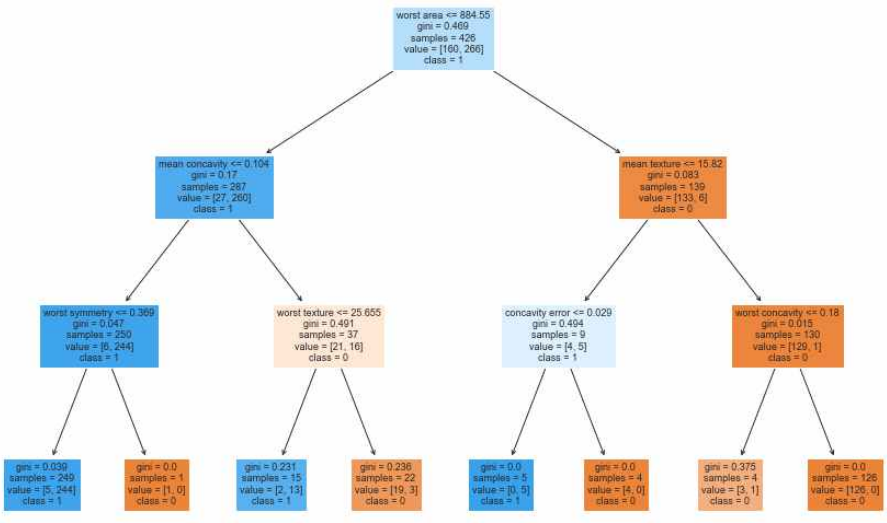
```
print('-'*20, '성능평가','-'*20)
print(CM_report)
```

```
-------------------- 성능평가 --------------------
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        52
           1       0.95      0.95      0.95        91

    accuracy                           0.93       143
   macro avg       0.92      0.92      0.92       143
weighted avg       0.93      0.93      0.93       143
```

test data의 정확도는 93%로 나타났으며, 레이블의 불균형을 고려하지 않은 단순 평균값인 macro avg는 92%, 관측치 개수에 가중치를 더한 가중평균값인 weighted avg는 93%이다.

```
DT_MODEL_DEP3= DecisionTreeClassifier(max_depth=3, random_state=0)
DT_MODEL_DEP3.fit(X_train, y_train)

prediction_DEP3=DT_MODEL_DEP3.predict(X_test)

CM_DEP3=confusion_matrix(y_test, prediction_DEP3)
CM_report_DEP3=classification_report(y_test, prediction_DEP3)

print('-'*15, 'Confusion Matrix','-'*15)
print(CM_DEP3)
print('-'*20, '성능평가','-'*20)
print(CM_report_DEP3)

plt.figure(figsize=(15,10))
plot_tree(DT_MODEL_DEP3, feature_names=df_1.columns, class_names=df1_target_names, filled=True, fontsize=9)
plt.show()
```

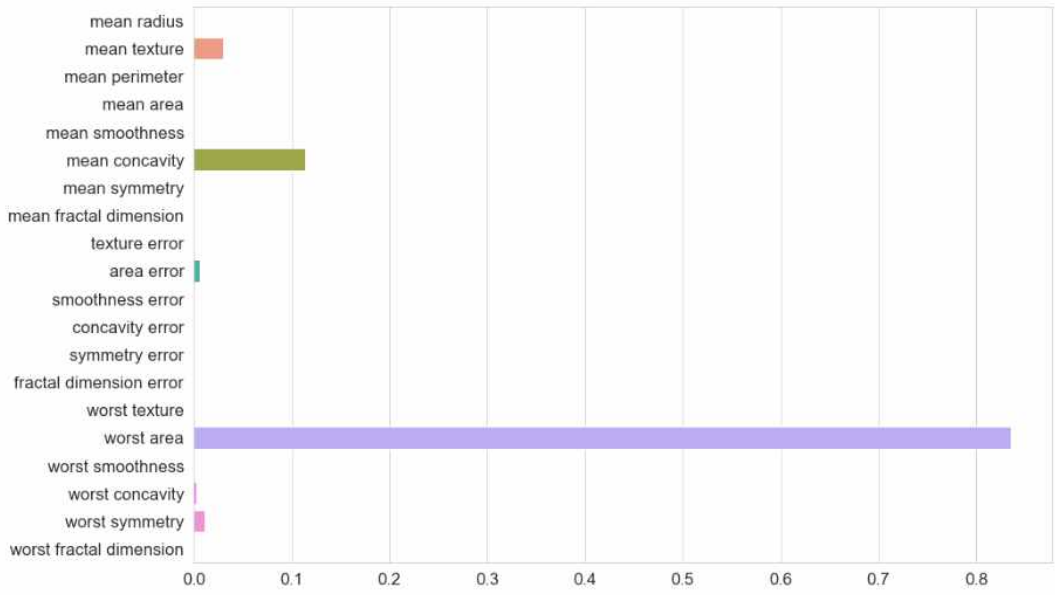
```
--------------- Confusion Matrix ---------------
[[46  6]
 [ 4 87]]
-------------------- 성능평가 --------------------
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        52
           1       0.94      0.96      0.95        91

    accuracy                           0.93       143
   macro avg       0.93      0.92      0.92       143
weighted avg       0.93      0.93      0.93       143
```



종양의 악성, 혹은 양성을 분류함에 있어 기준이 되는 의사결정 나무의 결과는 다음과 같으며, 모델의 정확도는 93%로 나타났다.
오차 행렬을 통해 악성(malignant) 값이 6개, 양성(benign) 4개의 값이 오분류 되었음을 확인할 수 있다.

```
for name, value in zip(X.columns, DT_MODEL_DEP3.feature_importances_):
    print('{0} : {1:.3f}'.format(name, value))

plt.figure(figsize=(15,10))
sns.barplot(x=DT_MODEL_DEP3.feature_importances_ , y=X.columns)
```

```
mean radius : 0.000
mean texture : 0.030
mean perimeter : 0.000
mean area : 0.000
mean smoothness : 0.000
mean concavity : 0.113
mean symmetry : 0.000
mean fractal dimension : 0.000
texture error : 0.000
area error : 0.006
smoothness error : 0.000
concavity error : 0.000
symmetry error : 0.000
fractal dimension error : 0.000
worst texture : 0.001
worst area : 0.836
worst smoothness : 0.000
worst concavity : 0.003
worst symmetry : 0.011
worst fractal dimension : 0.000
<AxesSubplot:>
```



이는 변수의 중요도를 나타낸 수치 및 그래프이다. worst area(면적)의 중요도 값은 0.836으로 가장 높게 나타났고, 그 다음으로 mean concavity(오목함)은 0.113, mean texture(질감)은 0.030 값으로 나타났다. 분류를 진행하는데 있어 어떠한 변수가 결과에 가장 영향을 미치는지에 대해 다음과 같이 확인할 수 있다.

Ⅴ-Ⅲ. PCA (차원축소)

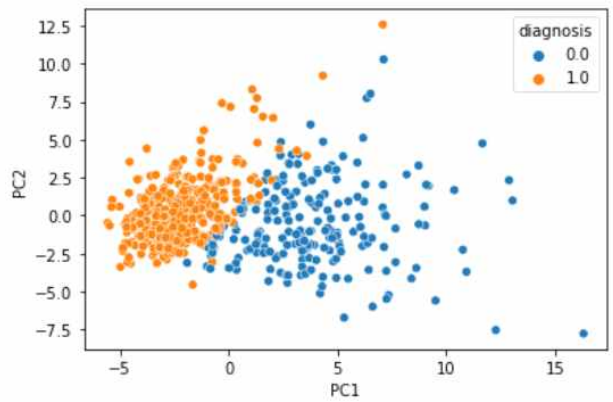
```
# Data Normalizing (Mean Centering)

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

```
X_ = StandardScaler().fit_transform(X)
pca = PCA(n_components=2)
pc = pca.fit_transform(X_)
```

```
# 카테고리정보 (음성 양성)

pc_y = np.c_[pc,y]
df = pd.DataFrame(pc_y,columns=['PC1','PC2','diagnosis'])
```

```
sns.scatterplot(data=df,x='PC1',y='PC2',hue='diagnosis')
```

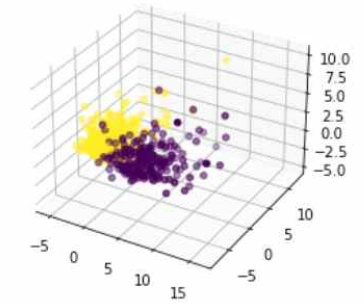
```
<AxesSubplot:xlabel='PC1', ylabel='PC2'>
```



표준화를 진행한 후, 2개의 주성분으로 PCA를 수행한다. 유방암 진단 데이터의 분산을 최대한 보존하는 새로운 축을 찾고, 두 축에 사영을 시킨 모습이다. 양성일 때와 음성일 때가 어느정도 뚜렷하게 구별된 모습을 확인할 수 있다.

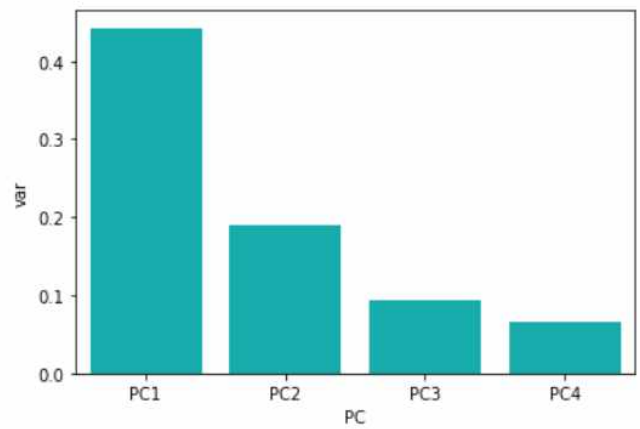
```
pca = PCA(n_components=3)
pc = pca.fit_transform(X_)

pc_y = np.c_[pc,y]
df = pd.DataFrame(pc_y,columns=['PC1','PC2','PC3','diagnosis'])

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['PC1'], df['PC2'], df['PC3'], c=df['diagnosis'],  s=20)
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x27e6b985f70>
```



이는 주성분 3개의 PCA 수행 결과이다.

```
pca = PCA(n_components=4)
pc = pca.fit_transform(X_)

df_var = pd.DataFrame({'var':pca.explained_variance_ratio_,
             'PC':['PC1','PC2','PC3','PC4']})
sns.barplot(x='PC',y="var",
           data=df_var, color="c");
```



주성분 4개의 PCA를 수행했을 때의 barplot이다. scikit-learn의 유방암 진단 데이터의 전체 변수의 수는 30개이지만, feature변수 4개로 전체의 70%정도 (대략 40%+ 20% + 10% + 8%) 설명이 가능하다는 것을 알 수 있다.

Ⅵ. 평가

유방암 진단 데이터를 분석함에 있어 K-Nearest Neighbor(KNN)의 테스트 결과, 최적의 k값이 6일 경우 모델의 정확도는 93.7%이다.

```
print(confusion_matrix(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
[[45  8]
 [ 3 87]]
              precision    recall  f1-score   support

           0       0.94      0.85      0.89        53
           1       0.92      0.97      0.94        90

    accuracy                           0.92       143
   macro avg       0.93      0.91      0.92       143
weighted avg       0.92      0.92      0.92       143
```

평가척도를 진행한 결과, 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율을 나타내는 정밀도(precision)의 경우, 0이라고 예측한 데이터의 94%만 실제로 0, 1이라고 예측한 데이터 중 92%가 실제로 1이었음을 알 수 있다.
재현율(recall)인 경우 실제 0인 데이터 중의 85%만 0으로 판별되었으며, 실제로 1인 데이터 중의 97%가 1로 판별되었음을 알 수 있다.
또한, precision과 recall의 조화평균인 f1-score은 89%, 94%로 나타났다.

의사결정나무 방법에서는 추가적으로 불순도를 기준으로 Gini Index와 Cross Entropy를 각각 적용하고, 최대 나무 깊이를 2부터 7까지 증가시킴으로써 정확도, 오차 행렬, 의사결정나무 그래프를 추가적으로 진행하였다. 이때, 불순도 기준과 최대 나무 깊이에 따라 정확도의 변화를 확인하였다.

- 불순도 기준 : Gini Index의 train set과 test set의 정확도

Max_Depth / Data	Train set	Test set
2	93.7	89.5
3	97.4	93.0
4	98.8	93.7
5	99.1	93.7
6	99.8	93.0
7	1.0	93.0

-불순도 기준 : Cross Entropy의 train set과 test set의 정확도

Max_Depth / Data	Train set	Test set
2	92.3	91.6
3	96.9	95.1
4	98.1	94.4
5	99.3	94.4
6	99.5	95.1
7	1.0	95.1

Y train set의 정확도는 gini index와 cross entropy 모두 최대 나무 깊이가 증가할수록 계속해서 증가한다.
Y test set의 정확도는 불순도 기준이 gini index인 경우, max_depth=2를 3으로 변경하였을 때 정확도가 89.5%에서 93.0%로, 이후 93.7%까지 증가하는 것을 확인할 수 있다. 하지만, 계속해서 나무의 깊이를 증가시킨 후 max_depth=6이 된 어느 시점부터 정확도가 감소하는 것을 확인할 수 있다. 불순도 기준이 cross entropy 기준인 경우, max_depth=2에서 3으로 증가할 때 정확도가 증가한다. 이후, max_depth=4일 때 다시 값이 감소하게 된다.
이처럼 max_depth를 증가시키면 오히려 정확도가 떨어지거나, 일정하게 지속되는 모습을 확인할 수 있다. 이는 과적합(over_fitting)이 되었다는 것을 의미한다. 따라서, 과적합의 형태를 확인하였을 경우 더 이상 노드를 확장시키지 않는 것이 가장 좋은 설정값이라 할 수 있다.

정확도가 높게 나온 두 예시를 살펴보고자 한다.
첫 번째로, 불순도 기준이 gini index, 최대 나무 깊이가 4일 때의 모델 정확도는 93.7%이다.

```
[[47  5]
 [ 4 87]]
              precision    recall  f1-score   support

           0       0.92      0.90      0.91        52
           1       0.95      0.96      0.95        91

    accuracy                           0.94       143
   macro avg       0.93      0.93      0.93       143
weighted avg       0.94      0.94      0.94       143
```

오차행렬을 확인한 결과, 0을 1로 오분류한 개수는 5개, 1을 0으로 오분류한 개수는 4개이다.

또한, 평가척도를 진행하였을 때 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율을 나타내는 정밀도(precision)의 경우, 0이라고 예측한 데이터의 92%만 실제로 0, 1이라고 예측한 데이터 중 95%가 실제로 1이었음을 알 수 있다.
재현율(recall)인 경우 실제 0인 데이터 중의 90%만 0으로 판별되었으며, 실제로 1인 데이터 중의 96%가 1로 판별되었음을 알 수 있다. 또한, precision과 recall의 조화평균인 f1-score은 91%, 95%로 나타났다.

두 번째로, 불순도 기준이 cross entropy, 최대 나무 깊이가 3일 때의 모델 정확도는 95.1%이다.

```
[[48  4]
 [ 3 88]]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        52
           1       0.96      0.97      0.96        91

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

오차행렬을 확인한 결과, 0을 1로 오분류한 개수는 4, 1을 0으로 오분류한 개수는 3이다.
또한, 평가척도를 진행하였을 때 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율을 나타내는 정밀도(precision)의 경우, 0이라고 예측한 데이터의 94%만 실제로 0, 1이라고 예측한 데이터 중 96%가 실제로 1이었음을 알 수 있다.
재현율(recall)인 경우 실제 0인 데이터 중의 92%만 0으로 판별되었으며, 실제로 1인 데이터 중의 97%가 1로 판별되었음을 알 수 있다. 또한, precision과 recall의 조화평균인 f1-score은 93%, 96%로 나타났다.

유방 종양을 분류하고자 대표적으로 K-Nearest Neighbor 방법과 Decision tree 방법을 사용하였다. 위의 결과와 같이 의사결정나무 모델이 KNN 모델에 비해 정확도 및 평가척도 부분에서 높게 평가되었으며, 특히 불순도 기준을 엔트로피 지수로 설정하였을 때 더욱 높은 성능을 보임을 알 수 있다.