1 빅데이터 기반 AI 응용 솔루션 개발자 전문 과정

1.1 교과목명 : 모델성능평가

- 평가일 : 22.9.19
- 성명 : 양주희
- 점수 : 70

Q1. iris data를 불러와서 붓꽃의 종류를 분류하는 모델링을 수행한 후 오차행렬과 정확도를 평가하세요.

- test_size = 0.2, 분류기는 DecisionTreeClassifier를 이용

In [1]:

```
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
```

In [3]:

```
iris_data = iris.data


iris_label = iris.target

idf = pd.DataFrame(data = iris_data, columns=iris.feature_names)
idf['label'] = iris.target
idf
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [5]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris_data, iri
                                                    test_size = 0.2
```

In [6]:

```
#DecisionTreeClassifier 객체 생성
df_clf = DecisionTreeClassifier(random_state=11)
```

In [7]:

```
#학습 수행 ( 학습용 피터 데이터 속성, 결정값 데이터 세트 )
df_clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=11)
```

In [8]:

```
#학습 완료된 객체에서 테스트 데이터 세트로 예측 수행
pred = df_clf.predict(X_test)
```

In [9]:

```
from sklearn.metrics import accuracy_score,confusion_matrix

def get_clf_eval(y_test , pred):  #y실제값, 예측값
    confusion = confusion_matrix( y_test, pred)
    accuracy = accuracy_score(y_test , pred)

    print('오차 행렬')
    print(confusion)
    print('정확도: {0:.4f}'.format(accuracy))
get_clf_eval(y_test , pred)
```

```
오차 행렬
[[ 9  0  0]
 [ 0 10  0]
 [ 0  2  9]]
정확도: 0.9333
```

Q2. 타이타닉 분석용 데이터세트인 tdf1.pkl를 불러와서 생존자 예측 모델을 만든 후 오차행렬, 정확도, 재현율, f1, AUC를 포함하는 사용자 함수를 활용하여 평가하세요.

- test_size = 0.2, 분류기는 RandomForestClassifier 이용

In [33]:

```
import pandas as pd
tdf = pd.read_pickle('tdf1.pkl')
tdf.head()
```

	Survived	Sex	Town_0	Town_1	Town_2	Family
0	0	1	0	0	1	0
1	1	0	1	0	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	0
4	0	1	0	0	1	0

In [46]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

y_tdf = tdf['Survived']
X_tdf = tdf.drop('Survived', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X_tdf, y_tdf,
                                                    test_size = 0.2

df_clf = RandomForestClassifier(random_state=11)
df_clf.fit(X_train, y_train)
pred = df_clf.predict(X_test)
```

In [53]:

```
def get_clf_eval(y_test, pred=None, pred_proba=None):
    confusion = confusion_matrix( y_test, pred)
    accuracy = accuracy_score(y_test , pred)
    precision = precision_score(y_test , pred)
    recall = recall_score(y_test , pred)
    f1 = f1_score(y_test,pred)
    # ROC-AUC 추가
    roc_auc = roc_auc_score(y_test, pred_proba)
    print('오차 행렬')
    print(confusion)
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},W
    F1: {3:.4f}, AUC:{4:.4f}'.format(accuracy, precision, recall,

pred_proba = df_clf.predict_proba(X_test)[:, 1]

get_clf_eval(y_test , pred, pred_proba)
```

```
오차 행렬
[[99 19]
 [13 48]]
정확도: 0.8212, 정밀도: 0.7164, 재현율: 0.7869,
F1: 0.7500, AUC:0.8598
```

Q3. Q2에서 생성한 모델로 교차검증(cv=5)을 수행하고 평균 정확도를 출력하세요.

In [55]:

```
import numpy as np
from sklearn.model_selection import cross_val_score
scores = cross_val_score(df_clf, X, y, scoring='accuracy',cv=5)
print('정확도:',np.round(scores,4))
print('평균 정확도:',round(np.mean(scores),4))
```

```
정확도: [0.7597 0.7208 0.7792 0.8235 0.7712]
평균 정확도: 0.7709
```

Q4. Q2에서 생성한 예측모델에 대하여 교차 검증 및 성능 개선을 수행하세요. (GridSearchCV 활용)

In [ ]:

Q5 ~ Q7. 'dataset/diabetes.csv'을 불러와서 아래사항을 수행하세요.

- 피마 인디언 당뇨병 예측을 로지스틱 회귀를 이용하여 수행하고 사용자 함수를 작성하여 평가(오차행렬, 정확도, 정밀도, 재현율, F1, ROC_AUC)
- 임곗값을 0.3에서 0.5까지 변화시키면서 정밀도와 재현율이 조정되는 과정을 시각화

- 재현율 기준의 성능을 개선하기 위하여 그 값이 0이 될 수 없는 각 칼럼을 탐색하여 적절한 처리를 한 후 로지스틱 회귀로 예측 및 평가 수행(오차행렬, 정확도, 정밀도, 재현율, F1, ROC_AUC)

In [10]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recal
from sklearn.metrics import f1_score, confusion_matrix, precision_
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

diabetes_data = pd.read_csv('diabetes.csv')
diabetes_data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickn
0	6	148	72	35
1	1	85	66	29
2	8	183	64	0
3	1	89	66	23
4	0	137	40	35

In [16]:

```
def get_clf_eval(y_test, pred=None, pred_proba=None):
    confusion = confusion_matrix( y_test, pred)
    accuracy = accuracy_score(y_test , pred)
    precision = precision_score(y_test , pred)
    recall = recall_score(y_test , pred)
    f1 = f1_score(y_test,pred)
    # ROC-AUC 추가
    roc_auc = roc_auc_score(y_test, pred_proba)
    print('오차 행렬')
    print(confusion)
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},W
    F1: {3:.4f}, AUC:{4:.4f}'.format(accuracy, precision, recall,
```

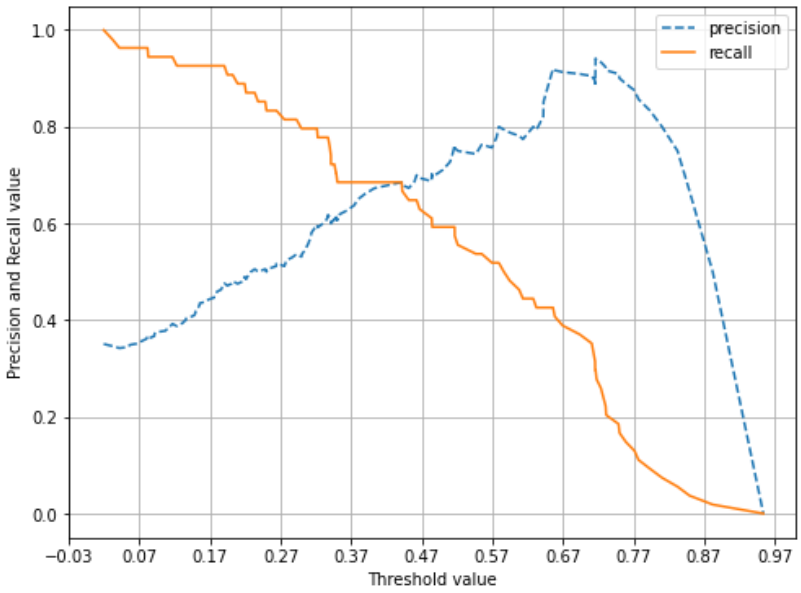
In [37]:

```
X = diabetes_data.iloc[:, :-1]
y = diabetes_data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz

# 로지스틱 회귀로 학습,예측 및 평가 수행.
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train , y_train)
pred = lr_clf.predict(X_test)
pred_proba = lr_clf.predict_proba(X_test)[:, 1]

get_clf_eval(y_test , pred, pred_proba)
```

```
오차 행렬
[[87 13]
 [22 32]]
정확도: 0.7727, 정밀도: 0.7111, 재현율: 0.5926,
F1: 0.6465, AUC:0.8083
```

In [45]:

```
thresholds = [0.3 , 0.33 ,0.36,0.39, 0.42 , 0.45 ,0.48, 0.50]

def precision_recall_curve_plot(y_test , pred_proba_c1):
    # threshold ndarray와 이 threshold에 따른 정밀도, 재현율 ndarr
    precisions, recalls, thresholds = precision_recall_curve( y_te

    # X축을 threshold값으로, Y축은 정밀도, 재현율 값으로 각각 Plot
    plt.figure(figsize=(8,6))
    threshold_boundary = thresholds.shape[0]
    plt.plot(thresholds, precisions[0:threshold_boundary], linesty
    plt.plot(thresholds, recalls[0:threshold_boundary],label='reca

    # threshold 값 X 축의 Scale을 0.1 단위로 변경
    start, end = plt.xlim()
    plt.xticks(np.round(np.arange(start, end, 0.1),2))

    # x축, y축 label과 legend, 그리고 grid 설정
    plt.xlabel('Threshold value'); plt.ylabel('Precision and Recal
    plt.legend(); plt.grid()
    plt.show()


precision_recall_curve_plot(y_test, lr_clf.predict_proba(X_test)[:
```



Q8. "dataset/auto-mpg.xlsx"을 불러와서 회귀 모델을 생성하고 MSE, RMSE, R2로 평가를 수행하세요.

In [11]:

```
df = pd.read_excel('auto-mpg.xlsx')
df.head()
```

	mpg	cylinders	displacement	horsepower	weig
0	18.0	8	307.0	130	3504
1	15.0	8	350.0	165	3693
2	18.0	8	318.0	150	3436
3	16.0	8	304.0	150	3433
4	17.0	8	302.0	140	3449

In [19]:

```
X = df[['weight']] #독립변수
y = df[['mpg']] #종속변수

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz


lr = LinearRegression()
lr.fit(X_train, y_train)
y_preds = lr.predict(X_test)

mse = mean_squared_error(y_test, y_preds)
rmse = np.sqrt(mse)
r2_score = r2_score(y_test, y_preds)

print(f'MSE:{round(mse, 4)}, RMSE:{round(rmse, 4)}, R2_Score:{roun
```

```
MSE:19.1351, RMSE:4.3744, R2_Score:0.6737
```

Q9. 'load_boston' 을 불러와서 cross_val_score를 이용한 cv=5인 교차검증을 수행 후 MSE, RMSE를 출력하세요.(LineaRegression)

In [29]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston

house = load_boston()
bostondf = pd.DataFrame(house.data, columns = house.feature_names)

bostondf['PRICE'] = house.target


bostondf.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.

In [30]:

```
from sklearn.model_selection import cross_val_score

y_target = bostondf['PRICE']
X_data = bostondf.drop(['PRICE'], axis = 1, inplace=False)
lr = LinearRegression()

#cross_val_score()로 5 폴드 세트로 MSE를 구한 뒤 이를 기반으로 다시
neg_mse_scores = cross_val_score(lr, X_data, y_target, scoring='ne
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

#cross_val_score(scoring='neg_mean_squared_error')로 반환된 값은 모
print(np.round(neg_mse_scores, 2))
print(np.round(rmse_scores, 2))
```

```
[-12.46 -26.05 -33.07 -80.76 -33.31]
[3.53 5.1  5.75 8.99 5.77]
```

Q10. 'Q9에 대하여 R2 Score를 구하세요.(k=5)

In [31]:

```
print(avg_rmse)
```

```
5.828658946215837
```