03 평가의 사본

성능 평가 지표(Evaluation Metric)

회귀(Regression)의 성능 평가 지표

실제값과 예측값의 오차 평균값에 기반함

기본적으로 예측 오차를 가지고 정규화 수준을 재가공하는 방법

분류(Classification)의 성능 평가 지표

- 정확도(Accuracy)
- 오차행렬(Confusion Matrix)
- 정밀도(Precision)
- 재현율(Recall)
- F1 스코어
- ROC AUC

결정 클래스 값 종류의 유형에 따라 나뉘는 분류 유형

- **2개의 결괏값만을 가지는 이진 분류(특히 이진분류에서 더 중요하게 강조하는 지표임)**
- 여러 개의 결정 클래스 값을 가지는 멀티 분류

01 정확도(Accuracy)

정확도는 실제 데이터에서 예측 데이터가 얼마나 같은지를 판단하는 지표이다.

정확도(Accuracy) = 예측 결과가 동일한 데이터 건수 / 전체 예측 데이터 건수

사이킷런의 BaseEstimator 클래스를 상속받아 아무런 학습을 하지 않고 성별에 따라 생존자를 예측하는 단순한 Classifier를 생성한다.

```
from sklearn.base import BaseEstimator

#BaseEstimator는 Customized 형태의 Estimator를 개발자가 생성할 수 있다.
class MyDummyClassifier(BaseEstimator):
    # fit() 메서드는 아무것도 학습하지 않음.
    def fit(self,X,y=None):
        pass
    #predict() 메서드는 단순히 Sex 피처가 1이면 0, 그렇지 않으면 1로 예측함.
    def predict(self, X):
        pred = np.zeros((X.shape[0],1))
        for i in range(X.shape[0]):
            if X['Sex'].iloc[i] ==1:
                pred[i] = 0
            else:
                pred[i] = 1
        return pred
```

질문: Sex 피처는 Male과 Female로 구분이 되는데, 피처가 1과 0을 어떻게 구분할 수 있을까?

원본 데이터 로딩

```
## 생성된 MyDummyClassifier를 이용해 타이타닉 생존자 예측 수행

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

## Null 처리 함수
def fillna(df):
    df['Age'].fillna(df['Age'].mean(), inplace=True)
    df['Cabin'].fillna('N', inplace=True)
    df['Embarked'].fillna('N', inplace=True)
    df['Fare'].fillna(0, inplace=True)
    return df

## 머신러닝에 불필요한 피처 제거
def drop_features(df):
    df.drop(['PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)
    return df

## Label Encoding 수행
def format_features(df):
    df['Cabin'] = df['Cabin'].str[:1]
    features = ['Cabin', 'Sex', 'Embarked']
    for feature in features:
        le = LabelEncoder()
        le.fit(df[feature])
        df[feature] = le.transform(df[feature])
    return df

## 앞에서 실행한 Data Preprocessing 함수 호출
def transform_features(df):
    df = fillna(df)
    df = drop_features(df)
    df = format_features(df)
    return df
```

위에서 생성한 DummyClassifier를 이용해 학습/예측/평가 수행

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#원본 데이터를 재로딩, 데이터 가공, 학습 데이터/테스트 데이터 분할
titanic_df = pd.read_csv('./titanic_train.csv')
y_titanic_df = titanic_df['Survived']
X_titanic_df = titanic_df.drop('Survived', axis = 1)
X_titanic_df = transform_features(X_titanic_df)
X_train, X_test, y_train, y_test = train_test_split(X_titanic_df,y_titanic_df,test_size = 0.2, random_state = 0)

#위에서 생성한 Dummy Classifier를 이용해 학습/예측/평가 수행.
myclf = MyDummyClassifier()
myclf.fit(X_train, y_train)

mypredictions = myclf.predict(X_test)
print('Dummy Classifier의 정확도 : {0:.4f}'.format(accuracy_score(y_test,mypredictions)))
```

```
Dummy Classifier의 정확도 : 0.7877
```

불균형한 레이블 값 평가

불균형한(imbalanced) 레이블 값 분포에서 ML 모델의 성능을 평가할 경우, 적합한 평가 지표가 아니다.

MNIST 데이터 세트를 변환해 multi classification에서 binary classification로 변경

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd

# BaseEstimator 생성
class MyFakeClassifier(BaseEstimator):
    def fit(self,X,y):
        pass

    # 입력값으로 들어오는 X 데이터 세트의 크기만큼 모두 0값으로 만들어서 반환
    def predict(self, X):
        return np.zeros((len(X),1),dtype = bool)

#사이킷런의 내장 데이터 세트인 load_digits()를 이용해 MINST 데이터 로딩
digits = load_digits()

#digits 번호가 7번이면 True이고, 이를 astype(int)로 1로 변환, 7번이 아니면 False이고 0으로 변환
y = (digits.target == 7).astype(int)
X_train, X_test, y_train, y_test = train_test_split(digits.data,y,random_state=11)
```

digit 번호가 7번인 세트를 True 라고 했을 때, 레이블 테스트 세트 크기는 450개이다. 이 때 분포도는 0(7을 제외한 나머지 숫자의 set) 405 , 1(7) 45임

> 근데, 말이 안되는것이, 모든 데이터 세트를 False 즉 0이라고 했을 때, 정확도를 측정하면 90%에 가까운 예측 정확도를 나타낸다. 여기서 정확도는 실제 데이터와 예측 데이터가 얼마나 일치하는 지를 의미하는데, 아무것도 하지않고 무조건 특정한 결과로 찍어도 데이터 분포도가 균일하지 않을 때 높은 수치가 나타날 수 있는 것이 정확도 평가 지표의 맹점이다.
>
> X_test의 결과값을 모두 0으로 했는데 어떻게 y_test의 정확도가 0.9가 나오는가.. 말이 안된다.

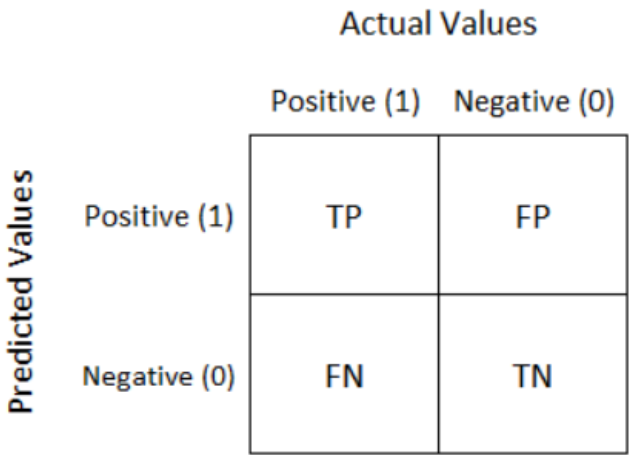
```
#불균형한 레이블 데이터 분포도 확인
print('레이블 테스트 세트 크기 :',y_test.shape)
print('테스트 세트 레이블 0과 1의 분포도')
print(pd.Series(y_test).value_counts())

#Dummy Classifier로 학습/예측/정확도 평가
fakeclf = MyFakeClassifier()
fakeclf.fit(X_train,y_train)
fakepred = fakeclf.predict(X_test)
print('모든 예측을 0으로 하여도 정확도: {0:.3f}'.format(accuracy_score(y_test,fakepred)))
```

따라서 정확도 평가 지표는 **불균형한 레이블 데이터 세트**에서는 성능 수치로 사용되서는 안된다.

정확도가 가지는 분류 평가 지표로서 이러한 한계점을 극복하기 위해 여러 가지 분류 지표와 함께 사용해야 한다.

02 오차 행렬

오차행렬(confusion matrix)은 학습된 분류 모델이 예측을 수행하면서 얼마나 헷갈리고 있는지도 함께 보여주는 지표이다. 즉, 이진 분류의 예측 오류가 얼마인지와 더불어 어떠한 유형의 예측 오류가 발생하고 있는지를 함께 나타내는 지표이다.



Ture/False는 예측값과 실제값이 '같은가/'틀린가'를 의미한다. 뒤 문자 Negative/Positive는 예측 결과 값이 부정(0)/긍정(0)을 의미한다.

- TN은 예측값을 Negative 값 0으로 예측했고 실제 값 역시 Negative 값 0
- FP는 예측값을 Positive 값 1로 예측했는데 실제 값은 Negative 값 0
- FN는 예측값을 Negative 값 0으로 예측했는데 실제 값은 Positive 값 1
- TP는 예측값을 Positive 값 1로 예측했고 실제 값 역시 Positive 값 1

사이킷런은 오차 행렬을 구하기 위해 confusion matrix() API를 제공한다.

다음은 MyFakeClassifier의 예측결과인 fakepred()와 실제 결과인 y_test를 confusion_matrix()의 인자로 입력해 오차 행렬을 confusion_matrix()를 이용해 배열 형태로 출력한다.

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, fakepred)
```

```
array([[405,   0],
       [ 45,   0]], dtype=int64)
```

실제 클래스에서 7인 digit 45 : 7이 아닌 digit 405개 이다.

그러나, 예측 클래스에서는 모두 7이 아닌 digit(값이 전부 0을 가짐)이다.

TN은 array[0,0]으로 405, FP는 array[0,1]로 0, FN은 array[1,0]로 45, TP는 array[1,1]로 0에 해당한다.

정확도 = 예측 결과와 실제 값이 동일한 건수/전체 데이터 수 = (TN+TP)/(TN + FN + FP + TP)

일반적으로 불균형한 레이블 클래스를 가지는 이진 분류 모델에서는 많은 데이터 중 중점적으로 찾아야하는 매우 적은 수의 결괏값에 Positive에 1값을 부여하고, 그렇지 않은 경우는 Negative로 0을 부여한다.

불균형한 이진 분류 데이터 세트에서는 Positive의 데이터 건수가 매우 작기 때문에 데이터에 기반한 ML 알고리즘은 Positive 보다는 Negative로 예측 정확도가 높아지는 경향이 발생한다.

TN만 높아지고, TP, FN, FP는 모두 작아진다.

결과적으로 정확도 지표는 비대칭한 데이터 세트에서 **Positive에 대한 예측 정확도를 판단하지 못한 채 Negative에 대한 예측 정확도만으로도 분류의 정확도가 매우 높게 나타나는 수치적인 판단 오류**를 일으킨다.

03 정밀도와 재현율

정밀도와 재현율은 Positive 데이터 세트의 예측 성능에 좀 더 초점을 맞춘 평가 지표이다.

MyFakeClassifier는 Positive로 예측한 TP 값이 하나도 없기 때문에 정밀도와 재현율 값이 모두 0이다.

정밀도 = TP / (FP + TP)

재현율 = TP / (FN + TP)

정밀도는 예측을 Positive로 한 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율

재현율은 실제 값이 Positive인 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율

- 재현율이 중요 지표인 경우는 **실제 Positive 양성 데이터를 Negative로 잘못 판단하게 되어** 업무상 큰 영향이 발생하는 경우

  ex) 암 판단 모델, 금융 사기 적발 모델
- 정밀도가 중요 지표인 경우는 **실제 Negative 음성 데이터를 Positive로 잘못 판단**하게 되어 업무상 큰 영향이 발생하는 경우

  ex) 스펨 메일

```
from sklearn.metrics import accuracy_score,precision_score, recall_score, confusion_matrix

def get_clf_eval(y_test, pred):
    confusion = confusion_matrix(y_test, pred)
    accuracy = accuracy_score(y_test,pred)
    precision = precision_score(y_test,pred)
    recall = recall_score(y_test, pred)
    print('오차 행렬')
    print(confusion)
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f}'.format(accuracy, precision,recall))
```

```
## 생성된 MyDummyClassifier를 이용해 타이타닉 생존자 예측 수행
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression

## Null 처리 함수
def fillna(df):
    df['Age'].fillna(df['Age'].mean(), inplace=True)
    df['Cabin'].fillna('N', inplace=True)
    df['Embarked'].fillna('N', inplace=True)
    df['Fare'].fillna(0, inplace=True)
    return df

## 머신러닝에 불필요한 피처 제거
def drop_features(df):
    df.drop(['PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)
    return df

## Label Encoding 수행
def format_features(df):
    df['Cabin'] = df['Cabin'].str[:1]
    features = ['Cabin', 'Sex', 'Embarked']
```

```
    for feature in features:
        le = LabelEncoder()
        le.fit(df[feature])
        df[feature] = le.transform(df[feature])
    return df

## 앞에서 실행한 Data Preprocessing 함수 호출
def transform_features(df):
    df = fillna(df)
    df = drop_features(df)
    df = format_features(df)
    return df

# 원본 데이터를 재로딩, 데이터 가공, 학습 데이터/테스트 데이터 부활
titanic_df = pd.read_csv('./titanic_train.csv')
y_titanic_df = titanic_df['Survived']
X_titanic_df = titanic_df.drop('Survived',axis = 1)
X_titanic_df = transform_features(X_titanic_df)

X_train, X_test, y_train, y_test = train_test_split(X_titanic_df, y_titanic_df, test_size = 0.20, random_state = 11)

lr_clf = LogisticRegression()

lr_clf.fit(X_train,y_train)
pred = lr_clf.predict(X_test)
get_clf_eval(y_test,pred)
```

```
오차 행렬
[[104  14]
 [ 13  48]]
정확도: 0.849162, 정밀도: 0.849162, 재현율: 0.7869
```

정밀도/재현율 트레이드 오프

정밀도/재현율의 Trade-off

정밀도와 재현율은 상호 보안적인 평가 지표이기 때문에, 어느 한쪽을 강제로 높이면 다른 하나의 수치는 떨어지기 쉽다.

predict_proba()

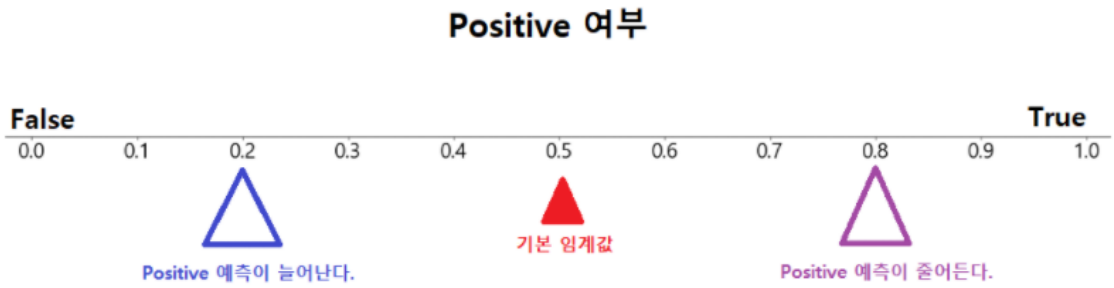
테스트 피처 데이터 세트를 파라미터로 입력해주면 테스트 피처 레코드의 개별 클래스 예측 확률을 반환함

- **입력 파라미터 :** predict() 메서드와 동일하게 보통 테스트 피처 데이터 세트를 입력
- **반환 값 :** 개별 클래스의 예측 확률을 ndarray m x n 형태로 반환

임계값(Threshold) 변경을 통한 재현율, 정밀도 변환

- 임계값 : 모델이 분류의 답을 결정할 때 기준값

임계값 변경에 따른 정밀도와 재현율의 변화관계



임곗값이 증가할수록 많은 수의 음성 예측으로 인해 정밀도 값은 동시에 높아지나 재현율 값은 낮아짐을 알 수 있다.

임곗값이 낮을수록 많은 수의 양성 예측으로 인해 재현율 값이 극도로 높아지고 정밀도 값이 극도로 낮아진다.

정밀도와 재현율의 맹점

정밀도가 100%가 되는 방법

확실한 기준이 되는 경우에만 Positive, 나머지는 모두 Negative

정밀도 = TP / (TP + FP) 이며, FP가 0이 될 수 밖에 없으므로 100%가 된다.

재현율이 100%가 되는 방법

모든 환자를 Positive로 **예측하면 된다.**

재현율 = TP / (TP + FN) 이며, FN은 0이 될 수 밖에 없으므로 100%가 된다.

따라서, 정밀도와 재현율 중 하나만 강조하는 상황이 돼서는 안된다. → 적절한 조합 필요

04 F1 스코어

F1 스코어 : 정밀도와 재현율을 결합한 지표

정밀도와 재현율이 어느 한쪽으로 치우치지 않는 수치를 나타낼 때 상대적으로 높은 수치를 가진다.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

05 ROC 곡선과 AUC

이진 분류 모델 성능 측정에서 중요하게 사용되는 지표

FPR(False Positive Rate) : 실제값 Negative 중 Positive로 잘못 예측한 비율

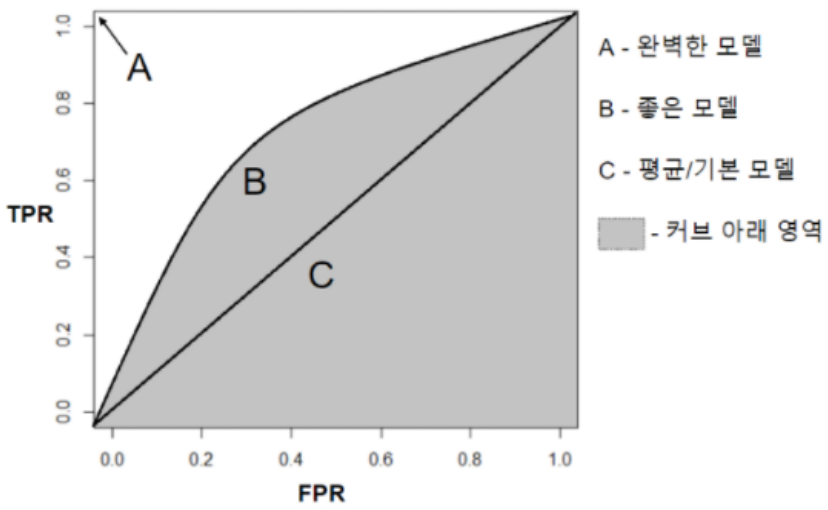
TPR(True Positive Rate, 민감도): 실제값 Positive가 정확히 예측돼야 하는 수준

TNR(True Negative Rate, 특이성): 실제값 Negative가 정확히 예측돼야 하는 수준

FPR = FP / (FP + TN) = 1 - TNR = 1 - 특이성

ROC 곡선

- FPR을 X축, TPR을 Y축으로 놓고 임계값을 변경해서 FPR이 변할 때, TPR이 어떻게 변하는지 나타내는 곡선



FPR을 변경하는 방법

분류 결정 임계값(Positive 예측값을 결정하는 확률의 기준)을 변경한다

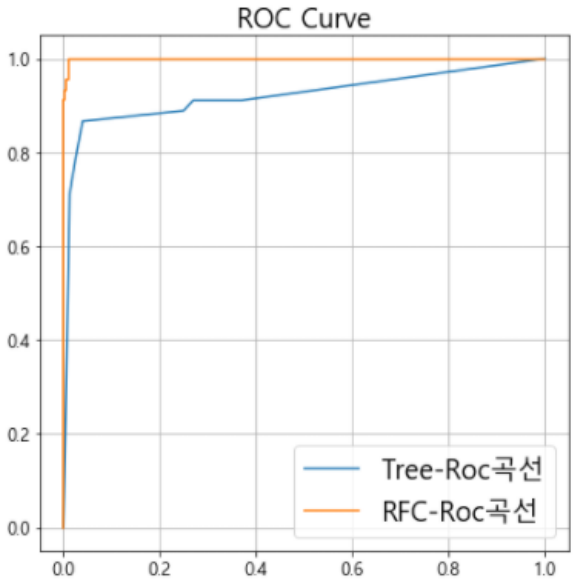
ROC, AUC 점수 확인

AUC(Area Under Curve) : ROC 곡선 밑의 면적 1에 가까울수록 좋은 수치

AUC가 커지려면 **FPR이 작은 상태에서 얼마나 큰 TPR을 얻을 수 있냐**가 관건

roc_curve(y값, 예측확률) : FPR, TPR, 임계치

roc_auc_score(y값, 예측확률) : AUC 점수 반환



타이타닉 생존자 예측

피마 인디언 당뇨병 예측