모델평가

- 권수태 교수

JEONJU UNIVERSITY
The Place for Superstars

1. 모델평가

❖ 모델 평가

- ➢ 머신러닝 모델은 여러가지 방법으로 예측 성능을 평가
- ➢ 성능 평가지표는 일반적으로 모델이 분류나 회귀냐에 따라 달라지게 됨
  - ✓ 회귀: 대부분 실제와 예측값의 차이인 오차 평균값에 기반
  - ✓ 분류: 예측변수(종속변수)가 0 혹은 1, 긍정/부정과 같이 이진 분류에서 정확도 이외에 다른 성능평가를 시도
- ➢ **분류의 성능 평가지표**
  - ✓ **정확도(Accuracy)**
  - ✓ **오차행렬(Confusion Matrix)**
  - ✓ **정밀도(Precision)**
  - ✓ **재현율(Recall)**
  - ✓ **F1 Score**
  - ✓ **ROC와 AUC**

1. 모델평가

❖ 정확도

➢ 정확도는 직관적으로 모델 예측 성능을 나타내는 평가 지표

$$\text{정확도(Accuracy)} = \frac{\text{예측 결과가 동일한 데이터 건수}}{\text{전체 예측 데이터 건수}}$$

➢ 하지만 종속변수가 0과 1로 이루어진 범주형인 경우 데이터 구성에 따른 머신러닝 모델의 성능을 왜곡 우려가 있어, 정확도 하나만으로 성능을 평가하진 않음

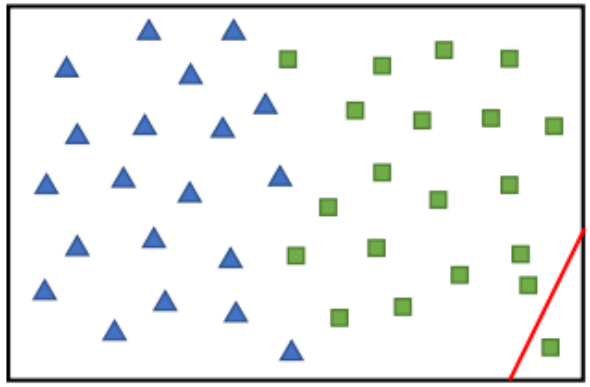
1. 모델평가

❖ 정확도

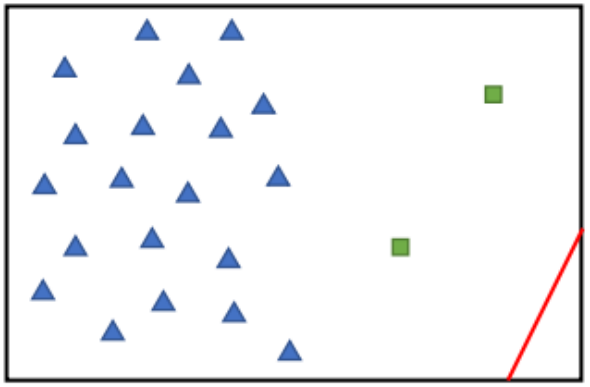
- ➢ 특히 정확도는 불균형한(imbalanced) 레이블 값 분포에서 머신러닝 모델의 성능을 판단 할 경우, 적합한 평가지표가 되지 못함
  - ✓ 신용카드 사기검출 판별: 30만 건 중 29만9천900건 이상에서 정상이고, 100건 정도만 이상이 있다고 했을 때 이미 99%이상이 정상적이고 1%정도만이 사기. 성능을 예측을 할때 그냥 정상이라고만 하면 정확도 수치가 99%
  - ✓ 타이타닉 승객의 생존자 예측: 남성보다 여성이 생존확률이 높았기 때문에 알고리즘을 사용하지 않고도 여성은 모두 생존이라고 판별해버리면 높은 정확도를 받을 수 있음(약 78% 정확도)

1. 모델평가

❖ 정확도



(a) 일반 데이터 세트



(b) 불균형 데이터 세트

Accuracy = (20+1)/40 = 52.5%

Accuracy = 20/22 = 90.9%

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는 지를 함께 나타내는 지표

데이터 총수 = P + N		예측 상황	
		Positive (PP)	Negative (PN)
실제 상황	Positive (P)	True positive (TP)	False negative (FN) [Type II error]
	Negative (N)	False positive (FP) [Type I error]	True negative (TN)

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 정확도(Accuracy)

✓ 예측한 전체 데이터들 중에 정확히 예측한 데이터 수의 비율

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

➢ 에러율(Error rate)

✓ 예측한 전체 데이터들 중에 잘못 예측한 데이터 수의 비율

$$Error\ rate = \frac{FP+FN}{TP+TN+FP+FN}$$

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

- ➢ 정밀도(Precision)
  - ✓ 양성으로 예측한 데이터들 중에 실제로 양성인 데이터의 비율

$$Precision = \frac{TP}{TP+FP}$$

- ➢ 재현율(Recall)
  - ✓ 실제로 양성인 데이터를 양성으로 정확히 예측한 경우의 비율

$$Recall = \frac{TP}{TP+FN}$$

- ➢ 특이도(Specificity)
  - ✓ 실제로 음성인 데이터를 음성으로 정확히 예측한 경우의 비율

$$Specificity = \frac{TN}{FP+TN}$$

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

- ➢ Type I error
  - ✓ 시스템이 실제로 음성인 사람을 양성으로 판정되는 경우
  - ✓ 물론 잘못 판정한 것은 맞는데 이 에러 상황에서 손해 볼 것은 사람이 돈을 더 내서 다시 검사하거나 다른 방식으로 검사 하게 될 것
  - ✓ 이때는 약간의 돈의 손해 있고 사람의 건강에 대해 손해 보는 것이 아니다
- ➢ Type II 에러
  - ✓ 시스템이 실제로 양성인 사람을 음성으로 판정되는 경우
  - ✓ 이 에러 상황은 Type I 에러에 비해 더 거대한 손해가 발생할 것
  - ✓ 질병이 걸린 환자가 원래 빨리 치료를 받아야 되는데 시스템은 음성으로 판정하니까 치료는 받지 못하게 되고 환자에게 생명의 위험이 있을 것

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 머신러닝에서의 손실 함수는 사실 Type I 에러와 Type II 에러로 구성

$$Loss = Loss_{type\ I} + Loss_{type\ II}$$

$$Loss = \alpha Loss_{type\ I} + \beta Loss_{type\ II}$$

1. 모델평가

❖ 오차행렬/혼동행렬 (Confusion Matrix)

- ➢ 사이킷런에서는 정밀도 계산을 위해 precision_score( ) , 재현율 계산을 위해 recall_score( ) 를 제공
- ➢ 업무 특성에 따라서 특정지표가 유용하게 사용
  - ✓ Recall(재현율): 암 판정, 사기 판정
  - ✓ Precision(정밀도) : 스팸메일 분류
- ➢ 재현율과 정밀도 모두 높은 수치를 얻는 것이 가장 좋은 성능평가

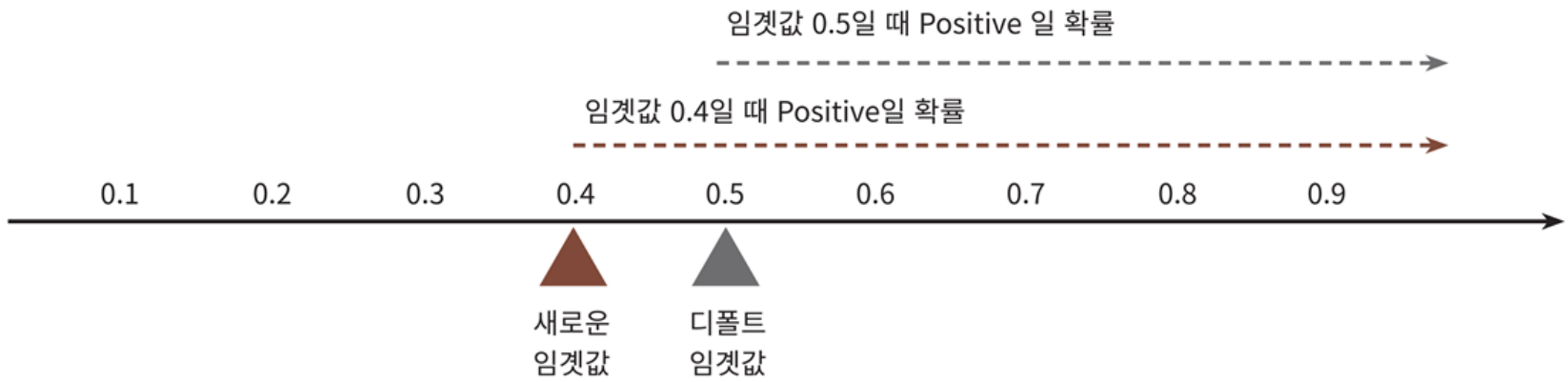
1. 모델평가

❖ 정밀도/재현율 트레이드오프

- ➢ 정밀도와 재현율은 상호보완적인 지표로 한쪽을 높이려고 하다보면 다른 한쪽이 떨어지기 쉬움
- ➢ 사이킷런의 분류 알고리즘은 예측 데이터가 특정 레이블에 속하는지 판단하기 위해 개별 레이블별로 확률을 구하고, 그 확률이 큰 레이블 값으로 예측
  - ✓ 일반적으로는 임계값을 50%로 정하고 이보다 크면 Positive, 작으면 Negative로 결정
  - ✓ predict_proba( ) 를 통하여 개별 레이블별 예측확률을 반환받을 수 있음
  - ✓ 반환 결과인 ndarray는 0과 1에 대한 확률을 나타내므로 첫번째 컬럼과 두번째 컬럼의 합은 1이 됨
  - ✓ 그리고 두 확률 중 큰 값의 레이블 값으로 predict( ) 메서드가 최종 예측

1. 모델평가

❖ 정밀도/재현율 트레이드오프



분류 결정 임곗값이 낮아질수록 Positive로 예측할 확률이 높아짐. 재현율 증가

precision_recall_curve(): 임계값 변화에 따른 평가 지표 값을 반환하는 API

입력 파라미터	y_true: 실제 클래스값 배열 (배열 크기= [데이터 건수]) probas_pred: Positive 칼럼의 예측 확률 배열 (배열 크기= [데이터 건수])
반환 값	**정밀도**: 임곗값별 정밀도 값을 배열로 반환 **재현율**: 임곗값별 재현율 값을 배열로 반환

1. 모델평가

❖ F1 스코어

- 정밀도와 재현율을 결합한 지표로 정밀도와 재현율이 어느 한 쪽으로 치우치지 않을 때 상대적으로 높은 값을 가짐
- F1_score() API 제공

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

1. 모델평가

❖ ROC곡선과 AUC

➢ ROC곡선(Receiver Operation Characteristic Curve)

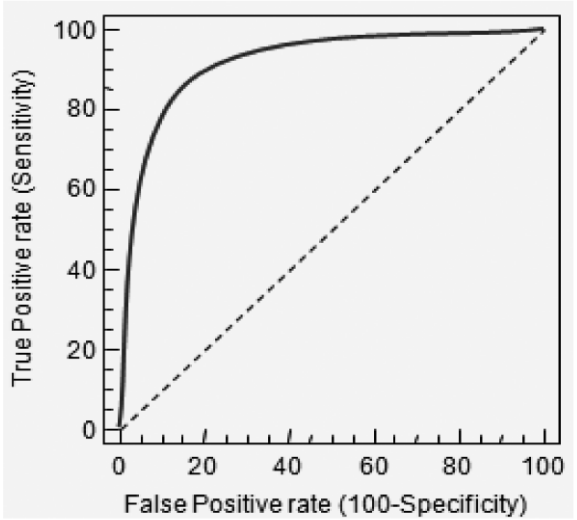
- ✓ FPR(False Positive Rate)이 변할 때 TPR(True Positive Rate)이 어떻게 변하는지를 나타내는 곡선
- ✓ Recall(재현율), Sensitivity(민감도), TPR=TP/(FN+TP) → 실제값 Positive가 정확이 예측되야 하는 수준(질병 보유자를 질병을 보유한 것으로 판정)
- ✓ Specificity(특이성),TrueNegativeRate(TNR)=TN/(TN+FP) → 실제값 Negative가 정확히 예측되야 하는 수준(건강한 사람을 건강하다고 판정)
- ✓ FPR=1−Specificity=FP/(TN+FP) --> 실제값 Negative 중 Positive로 잘못 예측된 비율(건강한 사람을 질병이 있다고 판정)

1. 모델평가

❖ ROC곡선과 AUC

➢ ROC곡선(Receiver Operation Characteristic Curve)

- ✓ 곡선이 가운데 직선에 가까울 수록 성능이 떨어지며, 멀어질수록 성능이 뛰어난 것
- ✓ 분류결정 임계값을 변경함으로써 FPR을 변화시킴('임계값=1'이면 Negative 값을 Positive로 예측하지 않고 FPR이 0이 됨, 반대로 임계값을 낮출수록 FPR이 올라감)



〈 ROC 곡선 예시 〉

1. 모델평가

❖ ROC곡선과 AUC

➢ ROC곡선(Receiver Operation Characteristic Curve)

✓ roc_curve(실제값, 예측 확률 값) : FPR, TPR, 임계값을 반환

입력 파라미터	y_true: 실제 클래스 값 array (array shape = [데이터 건수]) y_score: predict_proba( )의 반환 값 array에서 Positive 칼럼의 예측 확률이 보통 사용됨. array, shape = [n_samples]
반환 값	fpr: fpr 값을 array로 반환 tpr: tpr 값을 array로 반환 thresholds: threshold 값 array

1. 모델평가

❖ ROC곡선과 AUC

➢ AUC (Area Under Curve) 스코어

- ✓ 분류의 성능지표로는 ROC면적에 기반한 AUC 값으로 결정
- ✓ AUC: 곡선 밑의 면적 값으로 1에 가까울 수록 좋은 수치, 대각선 직선일 때 0.5
- ✓ roc_auc_score( ) : AUC 면적을 구하는 사이킷런 API

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

- 목표: 피마 인디언 당뇨병 데이트 세트를 이용하여 당뇨병 여부를 판단하는 머신러닝 예측모델을 수립하고, 여러 평가 지표를 적용
- 피마 당뇨병 데이터 세트 구성(북아메리카 피마지역 원주민의 type-2 당뇨병 결과데이터)
  - ✓ Pregnancies : 임신횟수
  - ✓ Glucose : 포도당 부하 검사 수치
  - ✓ BloodPressure : 혈압
  - ✓ SkinThickness : 팔 삼두근 뒤쪽의 피하지방 측정값
  - ✓ Insulin : 혈청 인슐린
  - ✓ BMI : 체질량 지수
  - ✓ DiabetesPedigreeFunction : 당뇨 내력 가중치 값
  - ✓ Age : 나이
  - ✓ Outcome : 당뇨여부(0 또는 1)

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import f1_score, confusion_matrix, precision_recall_curve, roc_curve
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

diabetes_data = pd.read_csv('diabetes.csv')
print(diabetes_data['Outcome'].value_counts())
diabetes_data.head(3)
```

diabetes_info() : Null 값은 없고, 모두 숫자형 확인 -> 인코딩 X

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
# 수정된 get_clf_eval() 함수
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)
    # ROC-AUC print 추가
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},\F1: {3:.4f}, AUC:{4:.4f}'
                    .format(accuracy, precision, recall, f1, roc_auc))
```

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
def precision_recall_curve_plot(y_test=None, pred_proba_c1=None):
    # threshold ndarray와 이 threshold에 따른 정밀도, 재현율 ndarray 추출.
    precisions, recalls, thresholds = precision_recall_curve( y_test, pred_proba_c1)

    # X축을 threshold값으로, Y축은 정밀도, 재현율 값으로 각각 Plot 수행. 정밀도는 점선으로 표시
    plt.figure(figsize=(8,6))
    threshold_boundary = thresholds.shape[0]
    plt.plot(thresholds, precisions[0:threshold_boundary], linestyle='--', label='precision')
    plt.plot(thresholds, recalls[0:threshold_boundary],label='recall')

    # 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 Recall value')
    plt.legend(); plt.grid()
    plt.show()
```

JEONJU UNIVERSITY

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
# 피처 데이터 세트 X, 레이블 데이터 세트 y를 추출.
# 맨 끝이 Outcome 컬럼으로 레이블 값임. 컬럼 위치 -1을 이용해 추출
X = diabetes_data.iloc[:, :-1]
y = diabetes_data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 156, stratify=y)

# 로지스틱 회귀로 학습,예측 및 평가 수행.
lr_clf = LogisticRegression()
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)

pred_proba_c1 = lr_clf.predict_proba(X_test)[:, 1]
precision_recall_curve_plot(y_test, pred_proba_c1)
```

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

- 최소 값이 0으로 되어 있는 값들이 많이 존재함
- Glucose(당 수치), BloodPressure(혈압), SkinThickness(피하지방), Insulin(인슐린), BMI(체질량 지수) 같은 값이 실제로 0일 수는 없다고 생각되므로 확인이 필요

```
diabetes_data.describe()

feature_list = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

def hist_plot(df):
   for col in feature_list:
      df[col].plot(kind='hist', bins=20).set_title('Histogram of '+col)
      plt.show()

hist_plot(diabetes_data)
plt.hist(diabetes_data['Glucose'], bins=10)
```


2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
# 0값을 검사할 피처명 리스트 객체 설정
zero_features = ['Glucose', 'BloodPressure','SkinThickness','Insulin','BMI']

# 전체 데이터 건수
total_count = diabetes_data['Glucose'].count()

# 피처별로 반복 하면서 데이터 값이 0 인 데이터 건수 추출하고, 퍼센트 계산
for feature in zero_features:
    zero_count = diabetes_data[diabetes_data[feature] == 0][feature].count()
    print('{0} 0 건수는 {1}, 퍼센트는 {2:.2f} %'.format(feature, zero_count,
100*zero_count/total_count))


# zero_features 리스트 내부에 저장된 개별 피처들에 대해서 0값을 평균 값으로 대체
diabetes_data[zero_features]=diabetes_data[zero_features].replace(0,
diabetes_data[zero_features].mean())
```

2. 당뇨병예측

❖피마 인디언 당뇨병 예측

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

# StandardScaler 클래스를 이용해 피처 데이터 세트에 일괄적으로 스케일링 적용
scaler = StandardScaler( )
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2, random_state
= 156, stratify=y)

# 로지스틱 회귀로 학습, 예측 및 평가 수행.
lr_clf = LogisticRegression()
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)
```


2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
from sklearn.preprocessing import Binarizer

def get_eval_by_threshold(y_test , pred_proba_c1, thresholds):
    # thresholds 리스트 객체내의 값을 차례로 iteration하면서 Evaluation 수행.
    for custom_threshold in thresholds:
        binarizer = Binarizer(threshold=custom_threshold).fit(pred_proba_c1)
        custom_predict = binarizer.transform(pred_proba_c1)
        print('임곗값:',custom_threshold)
        get_clf_eval(y_test , custom_predict, pred_proba_c1)

thresholds = [0.3 , 0.33 ,0.36,0.39, 0.42 , 0.45 ,0.48, 0.50]
pred_proba = lr_clf.predict_proba(X_test)
get_eval_by_threshold(y_test, pred_proba[:,1].reshape(-1,1), thresholds )
```


2. 당뇨병예측

❖피마 인디언 당뇨병 예측

```
# 임곗값를 0.48로 설정한 Binarizer 생성
binarizer = Binarizer(threshold=0.48)

# 위에서 구한 lr_clf의 predict_proba() 예측 확률 array에서 1에 해당하는 컬럼값을
Binarizer변환.
pred_th_048 = binarizer.fit_transform(pred_proba[:, 1].reshape(-1,1))

get_clf_eval(y_test , pred_th_048, pred_proba[:, 1])
```

Predict() 메서드는 임계값을 마음대로 변환할 수 없으므로 별도로 구현

Thank You !

JEONJU UNIVERSITY
The Place for Superstars