# 분류 모델의 성능 평가 지표

## 1. Accuracy(정확도)

- 실제 데이터와 예측 데이터가 얼마나 같은지 판단하는 지표
- $ 정확도(Accuracy) = ₩frac{예측 결과가 동일한 데이터 건수}{전체 예측 데이터 건수} $
- 모델 예측 성능을 나타내는 직관적인 평가 지표
- 불균형한 레이블 값 분포에서 ML 모델의 성능을 판단할 경우 모델의 성능을 왜곡할 수 있어 다른 지표를 함께 사용

## 2. Confusion Matrix(오차 행렬)

- 이진 분류 모델의 예측 결과를 행렬 형태로 표현한 지표
- 오류가 얼마인지, 어떠한 유형의 예측 오류가 발생하고 있는지를 나타냄
- 학습된 분류 모델이 예측을 수행하면서 얼마나 **헷갈리고(confused) 있는지** 보여주는 지표(혼동행렬)

### 이진 분류에 대한 오차행렬

- 4분면 (2×2)행렬에서 실제 클래스 값과 예측 클래스 값이 어떤 유형을 가지고 맵핑되는지 나타냄
- TN, FP, FN, TP 형태
- 분류 모델 예측 성능의 오류가 어떤 모습으로 발생하는지 알 수 있음

| | | 예측(Predicted) | |
|---|---|---|---|
| | | 0 (Negative) | 1 (Positive) |
| 정답 (Actual) | 0 (Negative) | $T_n$ | $F_p$ |
| | 1 (Positive) | $F_n$ | $T_p$ |

### 오차행렬에서 정확도

- 정확도(Accuracy) = 예측 결과와 실제 값이 동일한 건수 / 전체 데이터 수

  $ = ₩frac{TN + TP}{ TN + FP + FN + TP }$

### 불균형한 이진 분류 모델 사례

- 예1: 사기 행위 예측 모델
  - 사기 행위 : Positive 양성으로 1
  - 정상 행위 : Negative 음성으로 0
- 예2 : 암 검진 예측 모델
  - 양성 : Positive 양성으로 1
  - 음성 : Negative 음성으로 0

### 불균형한 이진 분류에서 정확도의 맹점

- 예. 10,000 건의 데이터 세트에서 9,900 건이 Negative이고 100건이 Positive라면
  - Negative로 예측하는 경향이 더 강해져서 TN은 매우 커지고 TP는 매우 작아지게 됨
  - Negative로 예측할 때 정확도가 높기 때문에 FN이 매우 작고
  - Positive로 예측하는 경우가 작기 때문에 FP 역시 매우 작아짐
- Positive 데이터 건수가 매우 작아 Negative에 대한 예측 정확도 만으로 분류의 정확도기 매우 높게 나타나는 수치적인 판단 오류를 일으키게 됨
- **정밀도(Predision)와 재현율(Recall)** 평가지표 이용

## 3. Precision(정밀도)와 Recall(재현율)

- Positive 데이터 세트의 예측 성능에 좀 더 초점을 맞춘 평가 지표

### 1) 정밀도와 재현율 공식

| | | 예측(Predicted) 0 (Negative) | 예측(Predicted) 1 (Positive) |
|---|---|---|---|
| 정답 (Actual) | 0 (Negative) | $T_n$ | $F_p$ |
| | 1 (Positive) | $F_n$ | $T_p$ |

$$\text{Precision} = \frac{T_p}{T_p + F_p}$$

$$\text{Recall} = \frac{T_p}{T_p + F_n}$$

#### precision(정밀도) : TP / (FP + TP)

- Positive로 예측한 대상 중 예측과 실제 값이 Positive로 일치한 데이터의 비율
  - 예측한 양성 vs 맞춘 양성
  - (FP + TP) : Positive로 예측한 모든 데이터 건수 (예측한 양성)
  - TP : 예측과 실제 값이 Positive로 일치한 데이터 건수 (맞춘 양성)
- Positive 예측 성능을 더욱 정밀하게 측정하기 위한 평가 지표로 **양성 예측도**라고도 불림

#### recall(재현율) : TP / (FN + TP)

- 실제값이 Positive인 대상 중 예측과 실제 값이 Positive로 일치한 데이터의 비율
  - 실제 양성 vs 맞춘 양성 비율
  - (FN + TP) : 실제값이 Positive인 모든 데이터 건수 (실제 양성)
  - TP : 예측과 실제 값이 Positive로 일치한 데이터 건수 (맞춘 양성)
- **민감도(Sensitivity) 또는 TPR(True Positive Rate)** 이라고도 불림

### 2) Precision/Recall Trade-off

- 재현율과 정밀도 모두 TP를 높이는 데 동일하게 초점을 맞춤
  - 재현율은 FN(실제 Positive, 예측 Negative)를 낮추는데 초점을 맞추고
  - 정밀도는 FP를 낮추는데 초점을 맞춤

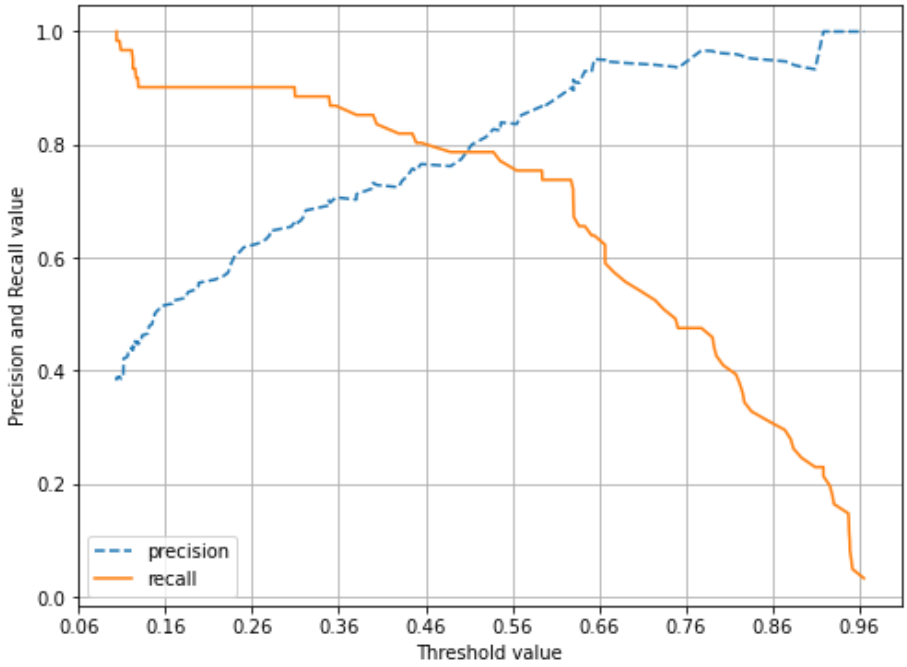
- 업무에 따라 정밀도/재현율 중요도 다름
  - 재현율이 상대적으로 더 중요한 지표인 경우

        - 실제 Positive인 데이터 예측을 Negative로 잘못 판단하게 되면 큰 영향이 발생하는 경우
        - 예. 암 진단 모델
    - 정밀도가 상대적으로 더 중요한 지표인 경우
        - 실제 Negative인 데이터 예측을 Positive로 잘못 판단하게 되면 큰 영향이 발생하는 경우
        - 예. 스팸메일 여부 판단 모델

- 가장 좋은 성능 평가는 재현율과 정밀도 모두 높은 수치를 얻는 것
    - 반면에 둘 중 어느 한 평가 지표만 매우 높고, 다른 지표는 매우 낮은 결과를 나타내는 경우는 바람직하지 않음

- 분류의 결정 임계값(Threshold)을 조정해서 정밀도 또는 재현율의 수치를 높일 수 있음

- 정밀도와 재현율은 상호 보완적인 평가 지표이기 때문에 어느 한쪽을 강제로 높이면 다른 하나의 수치는 떨어지는데 이를 **정밀도/재현율의 트레이드 오프(trade-off)** 라고 함




## 4. F1 Score

- 정밀도와 재현율의 조화평균
- 정밀도와 재현율이 어느 한 쪽으로 치우치지 않는 수치를 나타낼 때 상대적으로 높은 값을 가짐

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

### 예 : 두 예측 모델 비교

- A 예측 모델
    - precision : 0.9

    - recall : 0.1 (극단적 차이)
    - F1 score : 0.18
- B 예측 모델
    - precision : 0.5
    - recall : 0.5 (큰 차이 없음)
    - F1 score : 0.5

=> B모델의 FI score가 A모델에 비해 매우 우수

## 5. G measure

- 정밀도와 재현율의 기하평균
- $ G = ₩sqrt{Precision × Recall}$

## 6. ROC(Reciver Operating Characteristic) curve

- 수신자 판단 곡선
- 2차대전 때 통신장비 성능평가를 위해 고안된 척도
- 의학분야에서 많이 사용
- 이진분류 모델의 예측 성능 평가지표

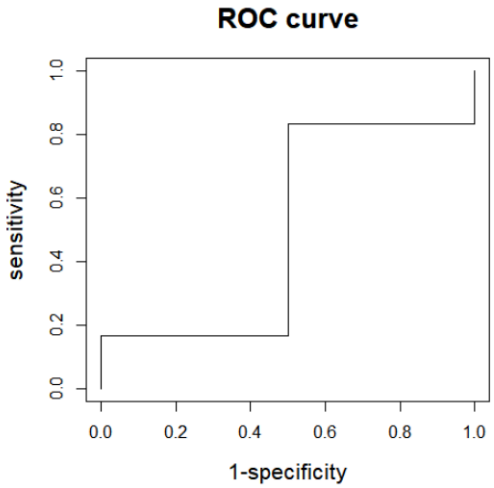
- FPR(False Positive Rate)이 변할 때 TPR(True Positive Rate)가 어떻게 변하는지를 나타내는 곡선
    - FPR이 X축, TPR이 Y축
    - TPR : 재현율과 같으며, 민감도(Sensitivity)라 부름
        - 실제값 Positive(양성)가 정확히 예측되어야 하는 수준
        - 예. 질병이 있는 사람이 질병이 있는 것(양성)으로 판정
    - FPR : 1-특이성(Specificity)
        - 예. 질병이 없는 건강한 사람이 질병이 있는 것으로 예측되는 수준
        - 특이성 : 실제값 Negative(음성)가 정확히 예측되어야 하는 수준
            - 예. 질병이 없는 건강한 사람은 질병이 없는 것(음성)으로 판정

$$\text{True Positive Rate} = \frac{T_p}{T_p + F_N}$$ **민감도(Sensitivity)**

$$\text{True Negative Rate} = \frac{T_n}{T_n + F_p}$$ **특이성(Specificity)**

$$\text{False Positive Rate} = 1 - \frac{T_n}{T_n + F_p} = \frac{F_p}{T_n + F_p}$$

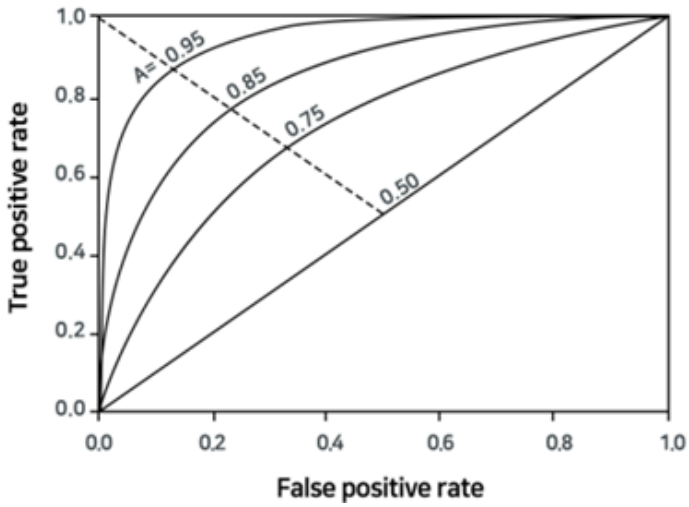
| | | 예측(Predicted) 0 (Negative) | 예측(Predicted) 1 (Positive) | |
|---|---|---|---|---|
| 정답(Actual) | 0 (Negative) | $T_n$ | $F_p$ | 특이성 |
| | 1 (Positive) | $F_n$ | $T_p$ | 민감도 |




- FPR은 0부터 1까지 변경하면서 TPR의 변화 값을 구함
  - 분류 결정 임계값(Positive 예측값을 결정하는 기준)을 변경하면서 결정
- FPR을 0으로 만들려면 분류 결정 임계값을 1로 지정
  - Positive 예측 기준이 높아 데이터를 Positive로 예측할 수 없음
  - FPR이 0인 경우 Positive를 예측할 수 없어 FPR이 0이 됨
- FPR을 1로 만들려면 분류 결정 임계값을 0으로 지정하여 TN을 0으로 만들면 됨
  - 분류기의 Positive 확률 기준이 너무 낮아 다 Positive로 예측
  - Negative를 예측할 수 없으므로 TN이 0이 되고 FPR은 1이 됨

## 7. AUC(Area Under the Curve)

- ROC 곡선 아래 면적
- 대각선의 직선에 대응되면 AUC는 0.5
- 1에 가까울수록 좋은 수치
- FPR이 작을 때 얼마나 큰 TPR을 얻는지에 따라 결정




## 문제

다음 자료에 대한 정확도, 재현율, 정밀도, 특이값, 민감도를 계산하고 어떤 의사가 진단을 잘하는 의사라고 말할 수 있을까?

| 의사A | | 실제 | |
|---|---|---|---|
| | | 췌장암 | 정상 |
| 진단 (예측) | 췌장암 | 10 | 10 |
| | 정상 | 0 | 980 |

| 의사B | | 실제 | |
|---|---|---|---|
| | | 췌장암 | 정상 |
| 진단 (예측) | 췌장암 | 8 | 2 |
| | 정상 | 2 | 980 |

In [ ]:

In [ ]:

---

# 사이킷런의 성능지표 모듈과 관련 함수

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

## Classification metrics

See the Classification metrics section of the user guide for further details.

| | |
|---|---|
| metrics.accuracy_score(y_true, y_pred, *[, ...]) | Accuracy classification score. |
| metrics.auc(x, y) | Compute Area Under the Curve (AUC) using the trapezoidal rule. |
| metrics.average_precision_score(y_true, ...) | Compute average precision (AP) from prediction scores. |
| metrics.balanced_accuracy_score(y_true, ...) | Compute the balanced accuracy. |
| metrics.brier_score_loss(y_true, y_prob, *) | Compute the Brier score loss. |
| metrics.classification_report(y_true, y_pred, *) | Build a text report showing the main classification metrics. |
| metrics.cohen_kappa_score(y1, y2, *[, ...]) | Compute Cohen's kappa: a statistic that measures inter-annotator agreement. |
| metrics.confusion_matrix(y_true, y_pred, *) | Compute confusion matrix to evaluate the accuracy of a classification. |
| metrics.dcg_score(y_true, y_score, *[, k, ...]) | Compute Discounted Cumulative Gain. |
| metrics.det_curve(y_true, y_score[, ...]) | Compute error rates for different probability thresholds. |
| metrics.f1_score(y_true, y_pred, *[, ...]) | Compute the F1 score, also known as balanced F-score or F-measure. |
| metrics.fbeta_score(y_true, y_pred, *, beta) | Compute the F-beta score. |
| metrics.hamming_loss(y_true, y_pred, *[, ...]) | Compute the average Hamming loss. |
| metrics.hinge_loss(y_true, pred_decision, *) | Average hinge loss (non-regularized). |
| metrics.jaccard_score(y_true, y_pred, *[, ...]) | Jaccard similarity coefficient score. |
| metrics.log_loss(y_true, y_pred, *[, eps, ...]) | Log loss, aka logistic loss or cross-entropy loss. |
| metrics.matthews_corrcoef(y_true, y_pred, *) | Compute the Matthews correlation coefficient (MCC). |
| metrics.multilabel_confusion_matrix(y_true, ...) | Compute a confusion matrix for each class or sample. |
| metrics.ndcg_score(y_true, y_score, *[, k, ...]) | Compute Normalized Discounted Cumulative Gain. |
| metrics.precision_recall_curve(y_true, ...) | Compute precision-recall pairs for different probability thresholds. |
| metrics.precision_recall_fscore_support(...) | Compute precision, recall, F-measure and support for each class. |
| metrics.precision_score(y_true, y_pred, *[, ...]) | Compute the precision. |
| metrics.recall_score(y_true, y_pred, *[, ...]) | Compute the recall. |
| metrics.roc_auc_score(y_true, y_score, *[, ...]) | Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. |
| metrics.roc_curve(y_true, y_score, *[, ...]) | Compute Receiver operating characteristic (ROC). |
| metrics.top_k_accuracy_score(y_true, y_score, *) | Top-k Accuracy classification score. |
| metrics.zero_one_loss(y_true, y_pred, *[, ...]) | Zero-one classification loss. |

### Regression metrics

See the Regression metrics section of the user guide for further details.

| | |
|---|---|
| metrics.explained_variance_score(y_true, ...) | Explained variance regression score function. |
| metrics.max_error(y_true, y_pred) | The max_error metric calculates the maximum residual error. |
| metrics.mean_absolute_error(y_true, y_pred, *) | Mean absolute error regression loss. |
| metrics.mean_squared_error(y_true, y_pred, *) | Mean squared error regression loss. |
| metrics.mean_squared_log_error(y_true, y_pred, *) | Mean squared logarithmic error regression loss. |
| metrics.median_absolute_error(y_true, y_pred, *) | Median absolute error regression loss. |
| metrics.mean_absolute_percentage_error(...) | Mean absolute percentage error (MAPE) regression loss. |
| metrics.r2_score(y_true, y_pred, *[, ...]) | $R^2$ (coefficient of determination) regression score function. |
| metrics.mean_poisson_deviance(y_true, y_pred, *) | Mean Poisson deviance regression loss. |
| metrics.mean_gamma_deviance(y_true, y_pred, *) | Mean Gamma deviance regression loss. |
| metrics.mean_tweedie_deviance(y_true, y_pred, *) | Mean Tweedie deviance regression loss. |
| metrics.d2_tweedie_score(y_true, y_pred, *) | D^2 regression score function, fraction of Tweedie deviance explained. |
| metrics.mean_pinball_loss(y_true, y_pred, *) | Pinball loss for quantile regression. |
| metrics.d2_pinball_score(y_true, y_pred, *) | $D^2$ regression score function, fraction of pinball loss explained. |
| metrics.d2_absolute_error_score(y_true, ...) | $D^2$ regression score function, fraction of absolute error explained. |

---

## 평가지표 모듈 임포트

In [ ]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, \
                            precision_score, recall_score, \
                            f1_score, roc_auc_score, \
                            precision_recall_curve, roc_curve
```

# 분류모델 성능평가지표 출력 함수 작성

In [ ]:
```
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)
    g = np.sqrt(precision*recall)
    roc_auc = roc_auc_score(y_test, pred_proba)

    print(f'오차행렬 :\n{confusion}')
    print(f'정확도: {accuracy:.4f}, 정밀도: {precision:.4f}, 재현율: {recall:.4f}')
    print(f'F1: {f1:.5f}, G-measure:{g:.5f}, AUC:{roc_auc:.4f}')
```

In [ ]:
```
def roc_curve_plot(y_test, pred_proba_c1):
    # 임곗값에 따른 FPR, TPR 값을 반환 받음
    fprs, tprs, thresholds = roc_curve(y_test ,pred_proba_c1)

    # ROC Curve를 plot 곡선으로 그림.
    plt.plot(fprs, tprs, label='ROC')

    # 가운데 대각선 직선을 그림.
    plt.plot([0, 1], [0, 1], 'k--', label='Random')

    # FPR X 축의 Scale을 0.1 단위로 변경, X,Y 축명 설정등
    start, end = plt.xlim()
    plt.xticks(np.round(np.arange(start, end, 0.1),2))
    plt.xlim(0,1); plt.ylim(0,1)
    plt.xlabel('FPR( 1 - Specificity )'); plt.ylabel('TPR( Recall )')
    plt.legend()
    plt.show()
```

---