- 머신러닝 분석 과정은 데이터 가공 과정을 거친 후 모델이 데이터를 학습한 뒤 알고리즘을 평가하는 프로세스로 구성됨
- 모델 예측성능을 평가하는 것은 결국 학습모델이 실젯값을 얼마나 정확하게 맞추었는지를 나타내는 것
- 머신러닝 모델은 성능평가지표(Evaluation Metric)을 통해서 여러 가지 방법으로 성능을 평가
- 일반적으로 분석 알고리즘이 회귀분석인지 분류분석인지에 따라 여러 종류로 구분

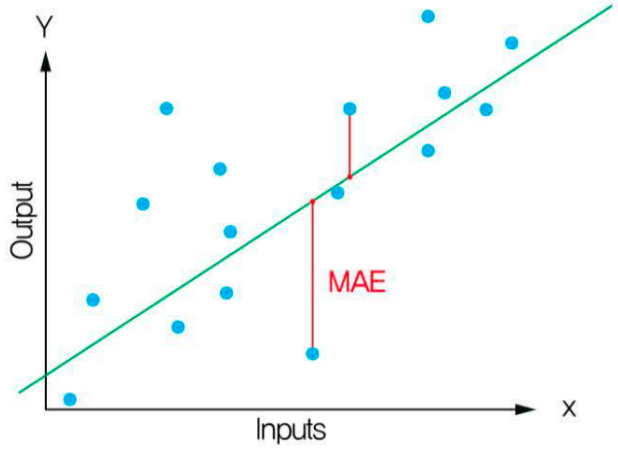
회귀 모델

- 실젯값과 예측값의 차이를 기반으로 한 지표들을 중심으로 성능평가지표가 발전
- 실젯값과 예측값의 편차를 구해서 단순히 합을 하게 되면 0이 되므로 절대값을 구하거나 제곱한 뒤 평균값을 구한다.

MAE

- Mean Absolute Error
- 실젯값과 예측값의 차이를 절댓값으로 변환해 평균한 것

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$



- 특징
  - 에러의 크기가 그대로 반영
  - 이상치에 영향을 받음

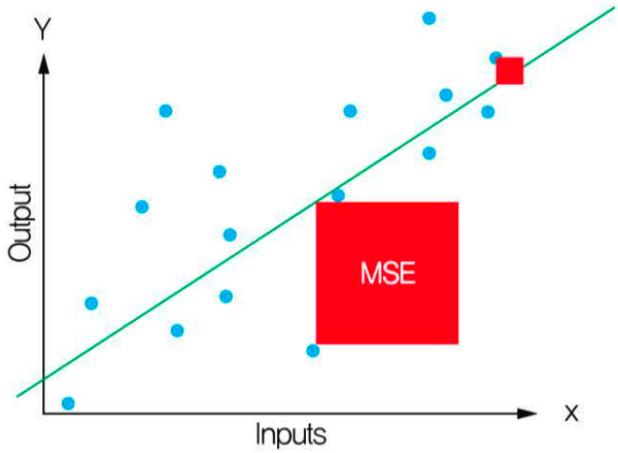
```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)
```

MSE

- Mean Squared Error
- 실젯값과 예측값의 차이를 제곱해 평균한 것

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$



- 특징
  - 실젯값과 예측값 차이의 면적 합을 의미
  - 특이값이 존재하면 수치가 증가

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
```

RMSE

- 평균 제곱근 오차(root mean square error)
- 실젯값과 예측값의 차이를 제곱해 평균한 것에 루트를 씌운 것

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

- 특징
  - 에러에 제곱을 하면 에러가 클수록 그에 따른 가중치가 높이 반영
  - 오차가 커질수록 이 값은 더욱 커지므로 예측에 얼마나 많은 오차가 있는지 가늠하게 함
  - 이때 손실이 기하급수적으로 증가하는 상황에서 실제 오류평균보다 값이 더 커지지 않도록 상쇄하기 위해 사용

```
from sklearn.metrics import mean_squared_error
import numpy as np
```

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

MSLE

- Mean Squared Log Error
- 실젯값과 예측값의 차이를 제곱해 평균한 것에 로그를 적용한 것

$$MSLE = \log\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2\right)$$

- 특징
  - RMSE와 같이 손실이 기하급수적으로 증가하는 상황에서 실제 오류평균보다 값이 더 커지지 않도록 상쇄하기 위해 사용

```
from sklearn.metrics import mean_squared_log_error
msle = mean_squared_log_error(y_test, y_pred)
```

MAPE

- Mean Absolute Percentage Error
- MAE를 퍼센트로 변환한 것

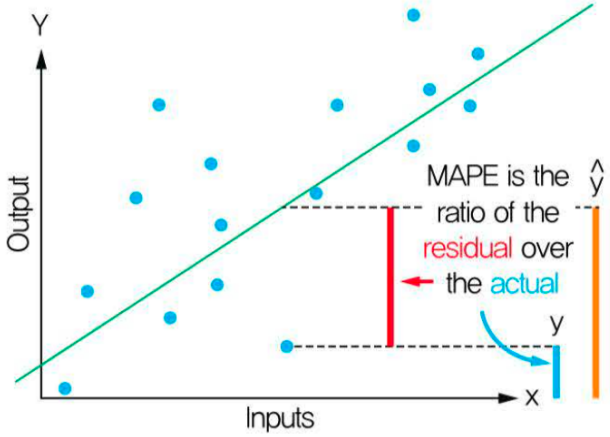
$$MAPE = \frac{n}{100}\sum_{i=1}^{n}\left|\frac{y_i - \hat{f}(x_i)}{y_i}\right|$$

- 특징
  - 오차가 예측값에서 차지하는 정도를 나타냄

```
import numpy as np

def MAPE(y_test, y_pred):
        mape = np.mean(np.abs((y_test - y_pred)/y_test)) * 100
        return mape

mape = MAPE(y_test, y_pred)
```



분류모델

- 분류 모델은 정확도를 기본적으로 선택하여 모델에 대한 평가를 진행하지만 이때 이진분류에서는 단순히 정확도로만 모델을 평가했을 때 잘못된 평가를 내릴 수 있음
- 결정값이 0과 1로 구성된 이진분류에서는 정확도가 아닌 다른 성능평가지표가 더 중요하게 여겨지는 경우도 있음

정확도

- Accuracy
- 실제 데이터에서 예측 데이터가 얼마나 같은지 판단하는 지표

$$\text{Accuracy} = \frac{\text{예측결과가 동일한 데이터 건수}}{\text{전체 예측 데이터 건수}}$$

- 특징
  - 데이터 구성에 따라 머신러닝 모델의 성능을 왜곡할 가능성이 있음

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
```

문제점

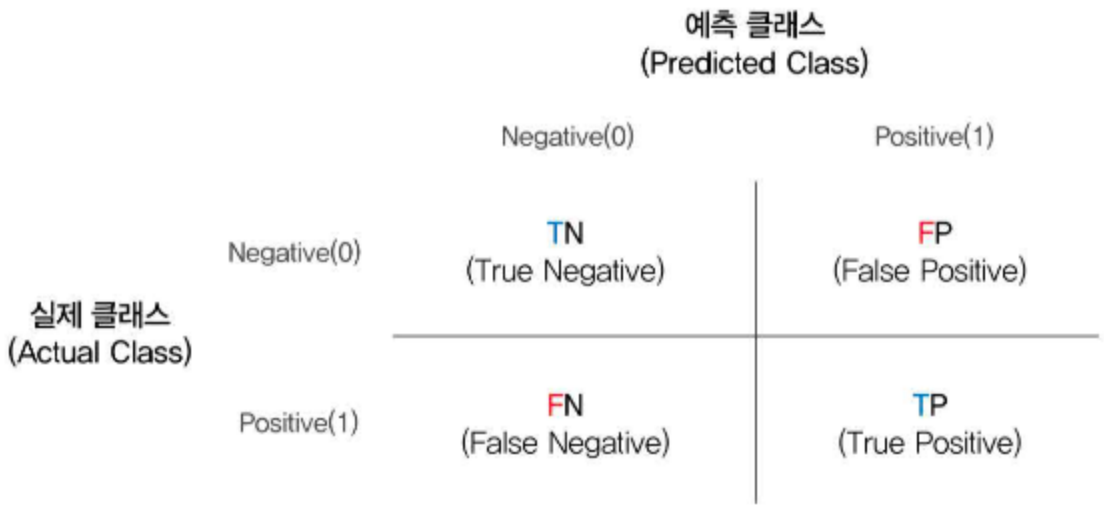
- 예를 들어 3%의 확률로 발생하는 질병을 예측하는 분류모델을 만들어야 한다고 가정하자
- 이 모델이 어떠한 데이터를 입력하든지 상관 없이 모두 질병이 발생하지 않는다는 예측결과를 내놓는다면 전체 예측 데이터 건수에서 3%정도만 질병이기 때문에 나머지 97%는 질병이 아닌 데이터이기 때문에 정확도는 97%가 된다.
- 불균형한 데이터셋에서 정확도로 평가하면 모델을 잘못 평가할수도 있다
- 이를 보완하기 위해 여러 가지 분류지표가 함께 고려되어야 한다.

혼동행렬

- Confusion Matrix
- 정확도의 한계점을 보완하기 위해 혼동행렬을 활용할 수 있음
- 이진 분류의 예측 오류가 얼마이고 어떠한 유형의 오류가 발생하고 있는지 나타내는 지표
- 특징
  - 4분면 행렬에서 실제 레이블 클래스 값과 예측 클래스 값이 어떠한 유형을 가지고 매핑되는지 나타냄

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```



TN

- True Negative
- 예측값을 Negative(0)으로 예측했고, 실제 값도 Negative(0)

FP

- False Positive
- 예측값을 Positive(1)로 예측했지만, 실제 값은 Negative(0)

FN

- False Negative
- 예측값을 Negative(0)으로 예측했지만, 실제 값은 Positive(1)

TP

- True Positive
- 예측값을 Positivie(1)로 예측했고, 실제 값도 Positivie(1)

정확도

- TN + TP / TN + FP + FN + TP

정밀도, 재현율

- 정밀도와 재현율은 Positive 데이터 예측에 집중한 성능평가지표
- 불균형 데이터에서 Positive로 예측한 값이 없다면 (Ture Positive) 정밀도와 재현율은 모두 0이다.

정밀도

- Precision
- Positive로 예측한 것들 중 실제로 Positive인 것들의 비율
- 특징
  - Positive 예측성능을 더욱 정밀하게 측정하기 위한 평가지표
  - 양성 예측도라고도 불림
  - 정밀도가 상대적으로 중요한 경우 :
    - 실제 Negative인 데이터를 Positive로 잘못 예측 했을 때 업무상 큰 영향이 발생할 때
- TP / (FP + TP)

```
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
```

재현율

- Recall
- 실제 Positive인 것들 중 Positive로 예측한 것들의 비율
- 특징
  - 민감도(Sensitivity) 또는 TPR(True Positive Rate)라고 불림
  - 재현율이 상대적으로 중요한 경우 :
    - 실제 Positive인 데이터를 Negative로 잘못 예측 했을 때 업무상 큰 영향이 발생할 때
- TP / (FN + TP)

```
from sklearn.metrics import recall_score

recall = recall_score(y_test, y_pred)
```

F1 스코어

- F1 스코어는 정밀도와 재현율을 결합한 분류 성능지표
- 실제 Positive인 것들 중 Positive로 예측한 것들의 비율

- 정밀도와 재현율이 어느 한 쪽으로 치우치지 않고 적절한 조화를 이룰 때 상대적으로 높은 수치를 나타냄

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

```
from sklearn.metrics import f1_score

f1 = f1_score(y_test, y_pred)
```

ROC 곡선과 AUC 스코어

- ROC 곡선과 이를 기반으로 하는 AUC 스코어는 이진 분류모델의 주요 성능평가지표

ROC 곡선

- FPR(False Positive Rate)이 변할 때 TPR(True Positive Rate)이 변하하는 것을 나타내는 곡선(ROC)
- FPR = FP / (FP + TN) = 1 - TNR = 1 - 특이성
- 특징
  - TPR을 y축으로, FPR을 x축으로 하는 그래프
  - 분류 결정 임계값을 조절하면서 FPR이 0부터 1까지 변할 때 TPR의 변화값을 그래프에 나타냄
  - 우상향 그래프로 그려지는 특징이 있음

$$TNR = \mathrm{TN}/(\mathrm{FP}+\mathrm{TN})$$
$$FPR = \mathrm{FP}/(\mathrm{FP}+\mathrm{TN}) = 1-\mathrm{TNR}$$

```
from sklearn.metrics import roc_curve

fpr, tpr, thres = roc_curve(y_test, y_pred, pos_label = 1)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr)
```

AUC 스코어

- ROC 곡선 자체는 FPR과 TPR의 변화 값을 보는 데 이용
- ROC 곡선 아래의 면적 값을 분류 성능지표로서 사용할 수 있음
- Area Under the ROC Curve
- ROC 곡선 아래의 면적
- 1에 가까울수록 예측성능이 우수하다고 판단
- 특징
  - AUC 값이 커지려면 FPR이 작을 때 TPR 값이 커야 함
  - 우상향 직선에서 멀어지고 왼쪽 상단의 모서리 쪽으로 가파르게 곡선이 이동할수록 AUC가 1에 가까워 짐
  - 랜덤 수준의 AUC값은 0.5

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thres = roc_curve(y_test, y_pred, pos_label = 1)
auc = auc(fpr, tpr)
```

참고

- 평가 지표마다 고유의 의미가 존재함
- 하지만 단일 평가 지표만을 사용하여 모델을 평가할 경우, 단편적인 결론을 내릴 뿐만 아니라 잘못하면 올바르지 못한 결론을 내릴 수도 있음
- 따라서 서로 보완성이 존재하는 지표들을 종합적으로 고려하여 모델을 평가해야 더 쉽고 정확하게 모델에 존재하는 문제들을 찾아낼 수 있음