Logistic regression

Sang Yup Lee

Logistic regression

- Logistic regression
  - 지도학습 알고리즘의 한 종류
    - 분류의 문제에 적용: 즉, 종속 변수가 명목 변수인 경우
  - 선형회귀모형과 전반적인 작동 순서가 유사
    - 정답이 있는 데이터를 학습 데이터와 평가 데이터로 구분
    - 학습 데이터에 수학적 모형 (로지스틱 회귀 모형)을 적용
      - 독립변수들과 종속변수의 관계를 파악하기 위한 목적
      - 학습을 통해 optimal한 파라미터의 값을 도출
        - 비용함수의 값을 최소화하는 파라미터의 값: 경사하강법 사용
    - 평가 데이터를 이용하여 모형의 성능 평가
    - 문제에 대한 데이터에 적용
    - 중요한 것: **수학적 모형 & 비용함수**

Logistic regression

- Binomial logistic regression
  - Values that y can take 0 and 1, 1 when an event has occurred
  - multinomial logistic 도 유사
- 모형의 형태
  - Logistic regression은 y가 1일 확률 (P(y=1|X)과 y가 0일 확률을 이용을 모형의 종속변수로 이용
  - P(y=1|X) = $^{1}/_{(1+e^{-z})}$
    - $z = b_0 + b_1X_1 + b_2X_2 + \ldots + b_kX_k$
    - 비선형모형 (파라미터의 의미 주의)
  - P(y=0|X) = 1 – P(y=1|X)
- 실제 종속변수가 취하는 값은 0 또는 1 이지만, 로지스틱 모형의 종속변수는 y=1 일 확률 즉, P(y=1|X)
  - 보통 P(y=1) > 0.5 인 경우, y를 1로 예측; 그렇지 않은 경우 y=0으로 예측

Logistic regression

- Cost function
  - 교차 엔트로피 (Cross Entropy)
  - $E = -\left[\sum_{i=1}^{N} y_i \ln p(y_i = 1|X) + (1 - y_i) \ln p(y_i = 0|X)\right]$
  - training data를 이용하여 이값을 최소화하는 파라미터의 값을 찾는 것
  - 비용함수의 값은 모형을 통한 예측이 잘못 될 수록 증가
  - i 번째 관측치에 대한 비용
    - $-\{y_i \ln p(y_i = 1|X) + (1 - y_i) \ln p(y_i = 0|X)\}$
    - 실제 종속변수의 값이 1인 경우, 즉, y_i=1, 모형이 종속변수를 잘 예측하는 경우는
      - $p(y_i = 1|X)$의 값이 크게 나와야 함
      - 예측을 잘 못하는 경우는 비용함수의 값이 커지게 됨

비용함수

- 비용함수 관련 중요 내용
  - 특정 수학적 모형에서 사용하는 비용함수는 수학적 모형에 따라서 달라지지 않는다.
  - 적용되는 문제의 종류에 따라서 달라진다.
    - 회귀문제: (일반적으로) MSE
    - 분류문제: (일반적으로) 교차 엔트로피

Python coding

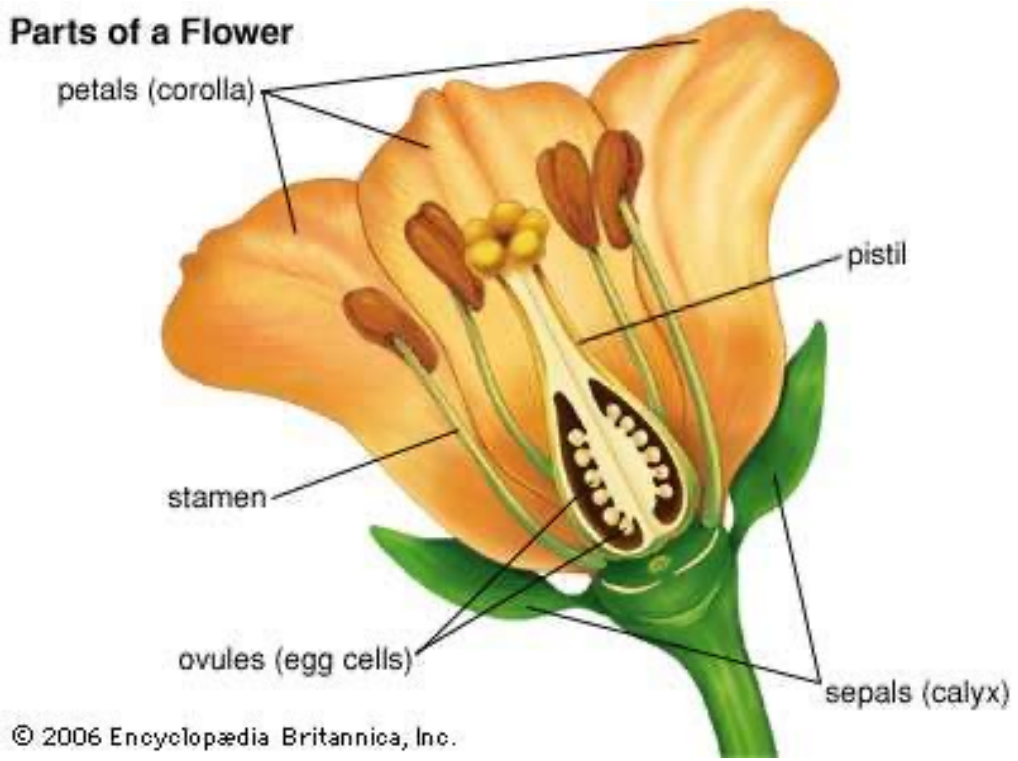
- To predict the species of an Iris flower (붓꽃)
- DV: the species of an Iris flower
  - It takes two different values

versicolor

virginica

Python coding

- Features (IVs)
  - Sepal (꽃받침):
    - length and width
  - Petal (꽃잎)
    - length and width



Python coding

- Python code
  - Refer to "LR_iris_example.ipynb"
  - Logistic regression model
    - When versicolor = 0, virginica = 1
    - P(y=1|X) = $^{1}/_{(1+e^{-z})}$
      - $z = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$
    - P(y=0|X) = 1 – P(y=1|X)

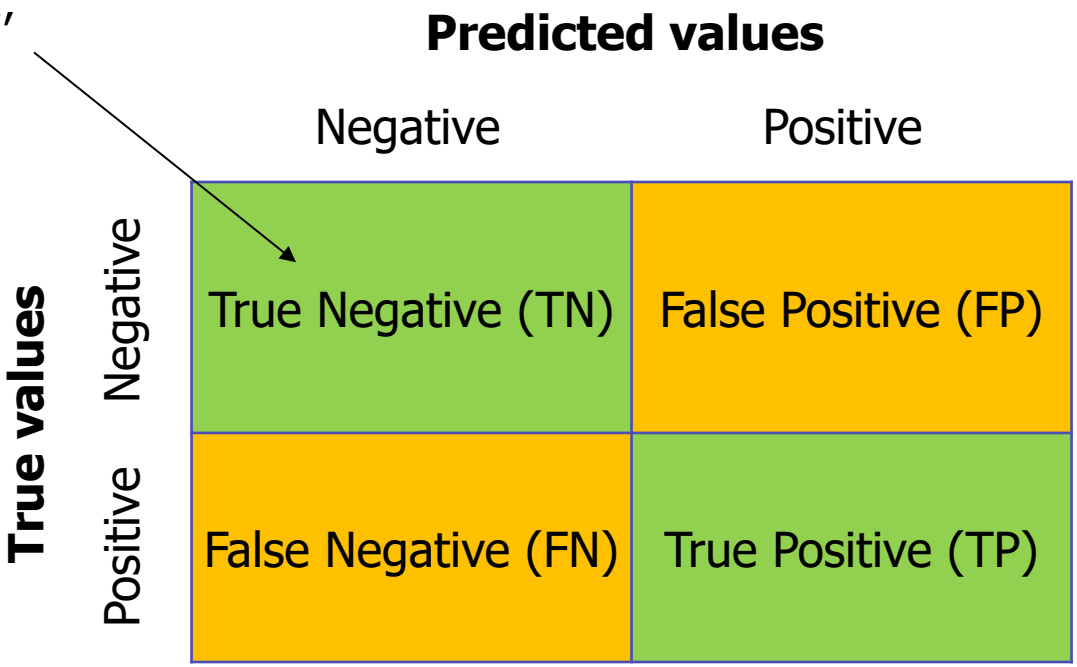
Model evaluation metrics

- 분류모형의 성능 평가 지표
  - Accuracy, recall, precision, F1 등
  - 계산을 위해서 confusion matrix를 사용
- Confusion matrix
  - 종속변수의 실제값과 모형을 통해서 예측이 된 종속변수 값에 따른 관측치들의 분포를 나타내는 matrix

Model evaluation metrics

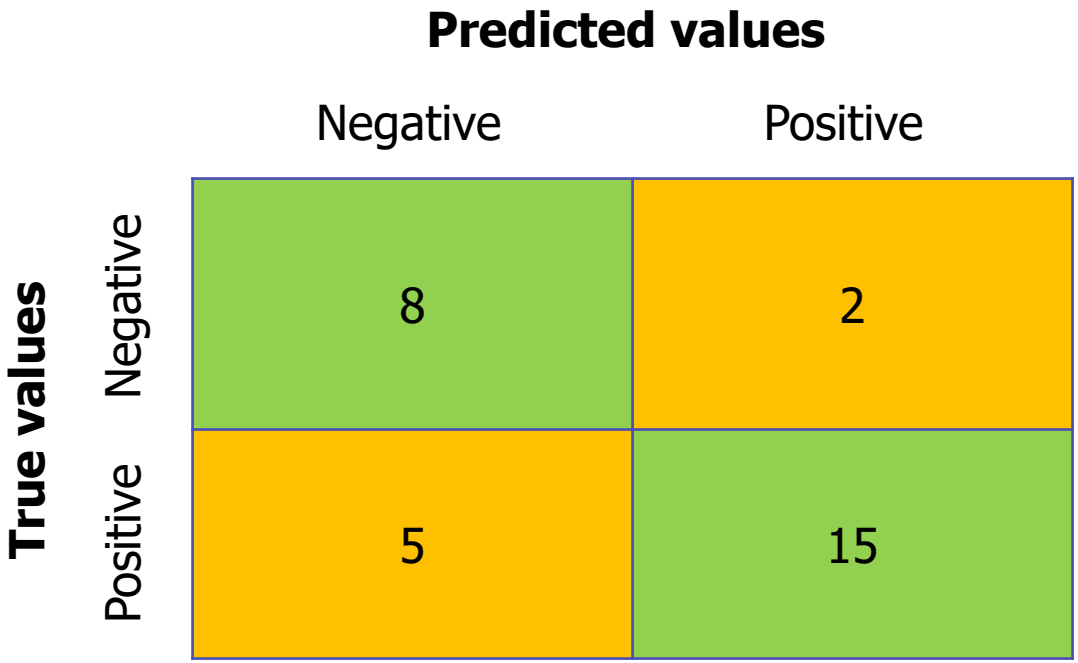
- Confusion matrix
  - y가 취할수 있는값: positive (1), negative (0)

실제 y의 값이 neg이고,
예측된 y값도 neg인
관측치의 수



Model evaluation metrics

- Example



Model evaluation metrics

- Performance measures
  - Accuracy: 전체에서 제대로 예측된 관측치의 비중
    - $\frac{TP+TN}{FP+FN+TP+TN}$
  - Recall for positive class (sensitivity): 실제 positive 관측치 중에서 실제로 positive 로 예측된 비중
    - $\frac{TP}{TP+FN}$
    - 해석: 특정 class가 얼마나 정확하게 예측되는가?
    - Example: 실제로 암인 사람들 중에서 암에 걸렸다고 예측된 사람들?
  - Recall for negative class (specificity): 실제 negative 관측치 중에서 실제로 negative 로 예측된 비중
    - $\frac{TN}{TN+FP}$

Model evaluation metrics

- Performance measures
  - Precision: positive (또는 negative)로 예측된 관측치 중에서 실제로 positive (또는 negative)인 관측치의 비중
    - Precision for positive class: $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$
    - 해석: 도출된 결과가 얼마나 정확한가?
    - Example: 암으로 예측이 된 환자들 중에서 실제로 암인 사람들?
    - Precision for negative class: $\frac{\mathrm{TP}}{\mathrm{TN+FN}}$
  - F1
    - F1 Score considers both precision and recall
    - Precision과 recall의 조화평균
    - $2\frac{PRE*REC}{PRE+REC}$

In Python

- Confusion matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_predictions)
```

```
array([[10,  1],
       [ 2, 17]], dtype=int64)
```

이전 슬라이드의 confusion matrix와 동일한 배치

In Python

- Performance measures

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predictions))
```

	precision	recall	f1-score	support
0	0.83	0.91	0.87	11
1	0.94	0.89	0.92	19
accuracy			0.90	30
macro avg	0.89	0.90	0.89	30
weighted avg	0.90	0.90	0.90	30

=(0.83+0.94)/2

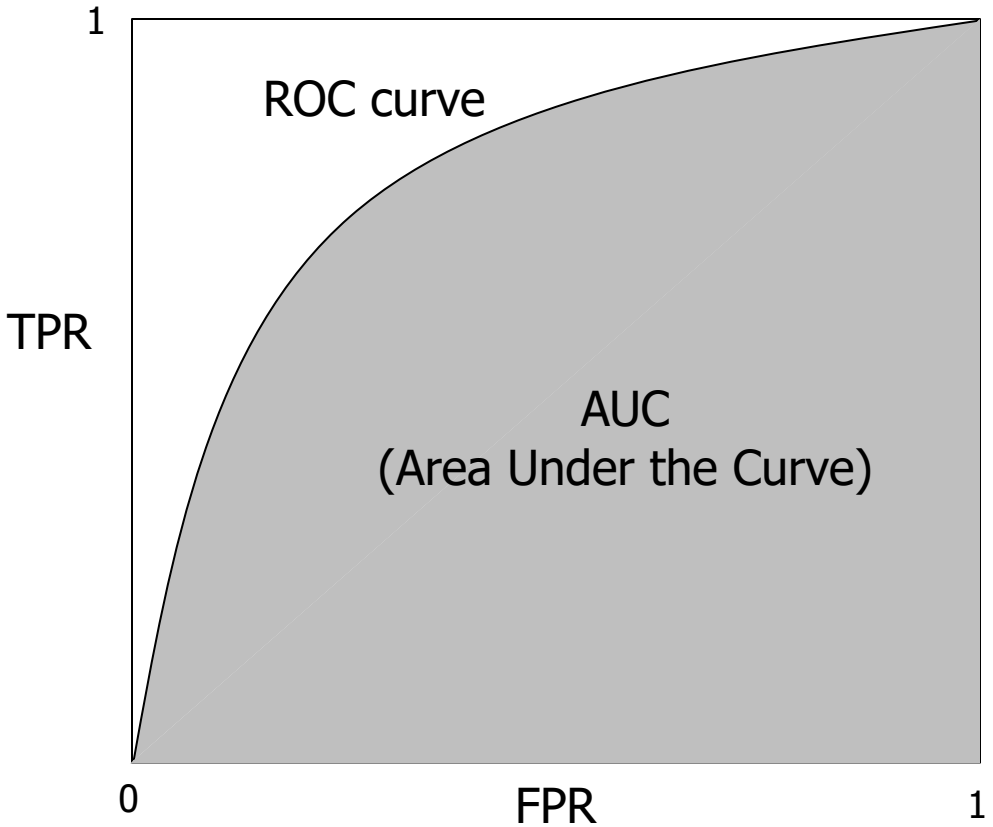
=(0.83*11+0.94*19)/30

Performance measures

- ROC (Receiver Operating Characteristics) curve

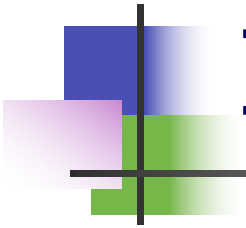
TPR = TP/(TP+FN)
FPR = FP/(FP+TN)

ROC curve:
종속변수값을
무엇으로 예측할
것인지의 기준이
되는 확률
(threshold
probability)값에
따른 TPR과
FPR 값들의 집합



AUC 값이 1에
가까울수록 모형의
성능이 좋은것

See
https://towardsdatascience.com/understanding-the-roc-curve-in-three-visual-steps-795b1399481c



In Python

- from sklearn.metrics import roc_curve
- from sklearn.metrics import roc_auc_score
- See “LR_iris_example.ipynb”

HYPERPARAMETER TUNING

Machine Learning

- 파라미터의 종류
  - 학습을 통해서 그 값이 결정되는 파라미터
  - 사용자가 그 값을 결정하는 파라미터 => 이러한 파라미터를 hyperparameter라고 함
  - 어떠한 hyperparameter를 갖는지는 모형 마다 달라짐
  - Hyperparameter가 취할 수 있는 값은 여러가지
    - 이중 모형의 성능을 좋게하는 값을 선택하는 것이 필요

Logistic Regression

- LogisticRegression class에서의 Hyperparameter 의 예
  - C, penalty, solver 등
  - 그렇다면 어떠한 값으로 설정을 하는게 좋을까요?
- Hyperparameter의 값 결정
  - 사람이 결정하기 때문에 자동적으로 최적의 값을 결정하기 어려워
  - 사전 지식을 가지고 몇가지 값을 시도
  - 모형의 성능이 더 좋은 것을 선택

Logistic Regression

- Model tuning
  - hyperparameter의 값을 변경하는 것 (혹은 그러한 과정을 거쳐서 성능이 더 좋은 모형을 찾는 것)
- Example
  - Penalty 종류에 따른 모형의 성능 비교
- 주의!
  - Hyperparameter tuning의 결과를 파악하기 위해서 평가 데이터를 사용하면 안됨
  - Test dataset은 모형의 최종 평가 목적으로 사용되므로 학습이나 튜닝에 사용되면 안됨 => 즉, 모형의 성능을 개선하는 목적으로 사용하면 안되고, 오직 모형의 성능을 평가하는 목적으로 사용

Logistic Regression

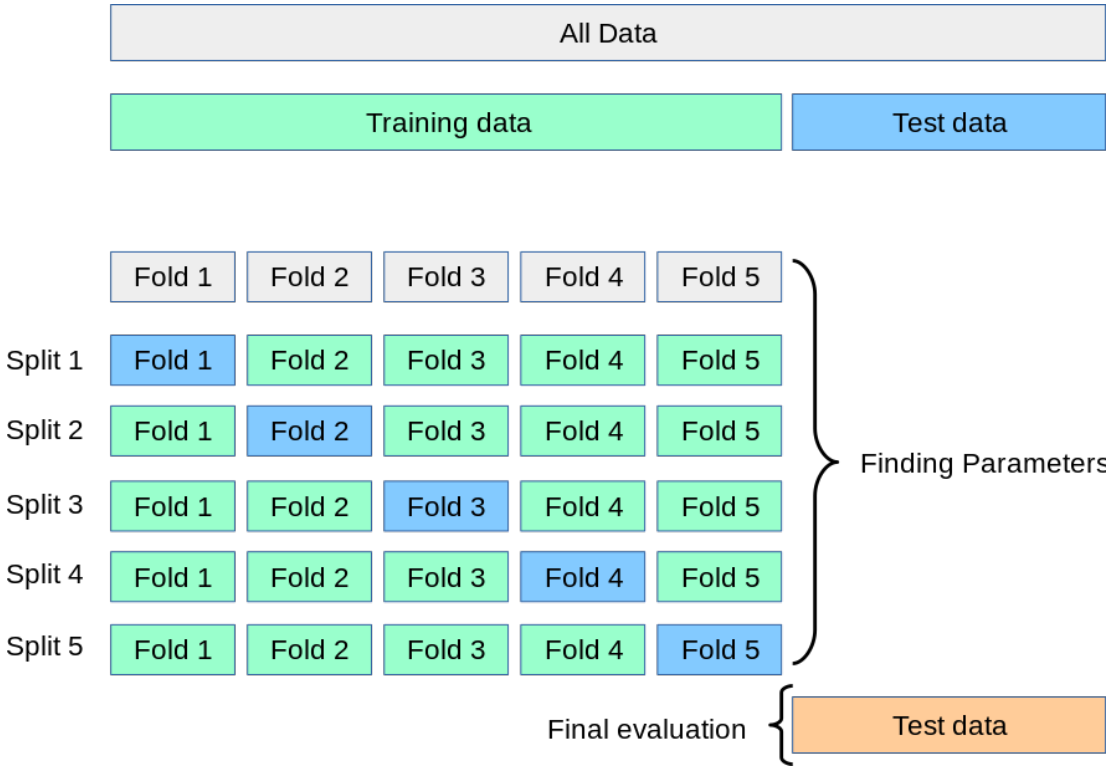
- Validation dataset
  - Model tuning의 목적으로 사용
  - 학습데이터의 일부를 validation dataset으로 사용
- Penalty 유형의 예
  - 두 모형 중에 어떠한 모형의 성능이 좋은가를 평가하기 위해서는 validation dataset을 사용 => 그 후, 더 성능이 좋은 모형을 이용하여 test data를 사용하여 평가

Logistic Regression

- Validation
  - Validation dataset을 이용해서 모형의 성능을 파악해 보는 것 (최종 평가가 아님)
- K-fold cross validation
  - The training set is split into k smaller sets, called “folds”. The following procedure is followed for each of the k “folds”:
    - A model is trained using k-1 of the folds as training data;
    - The resulting model is validated (i.e., tested) on the remaining part of the data
  - This process is repeated k times
  - Validation을 여러번 수행하는 이유 => 모형의 일반화 정도를 높이기 위해서

Logistic Regression

- Example: 5-fold cross validation



Logistic Regression

- K-fold cross validation
  - from sklearn.model_selection import cross_val_score
  - scores = cross_val_score(model, X_train, y_train, cv=5)
- Refer to "LR_iris_example.ipynb"

Grid search

- How to find the optimal values of hyperparameters?
- What is it?
  - Grid-search is used to find the optimal *hyperparameters* of a model which results in the most 'accurate' predictions.
- When to use?
  - When a hyperparameter can take numerous values
  - For a set of values that the user sets, the grid search method automatically finds the best hyperparameter values that leads to the best model (i.e, the model with best performance)

Q & A