Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Jan. 6 – 17, 2020

Python for Data Analytics

Sklearn

Decision Tree

Lab 1. Decision Tree – step 1

- Titanic dataset을 사용해서 survived or unsurvived를 학습시켜본다.

- data load 는 아래와 같이 seaborn의 데이터를 가져온다.

```
import seaborn as sns
titanic = sns.load_dataset('titanic')
```

Lab 1. Decision Tree – step2

- 데이터 전처리 과정
- 아래와 같이 dataframe에 null값들을 column별로 볼 수 있다.
- 학습을 위한 데이터는 [‘pclass’, ‘sex’, ‘age’, ‘sibsp’, ‘parch’, ‘fare’]이다.
- titanic[’sex’] 는 ‘male’, ‘female’의 string으로 이루어져있다.
- 이를 ‘male’ : 0, ‘female’ : 1 로 바꾸어준다.

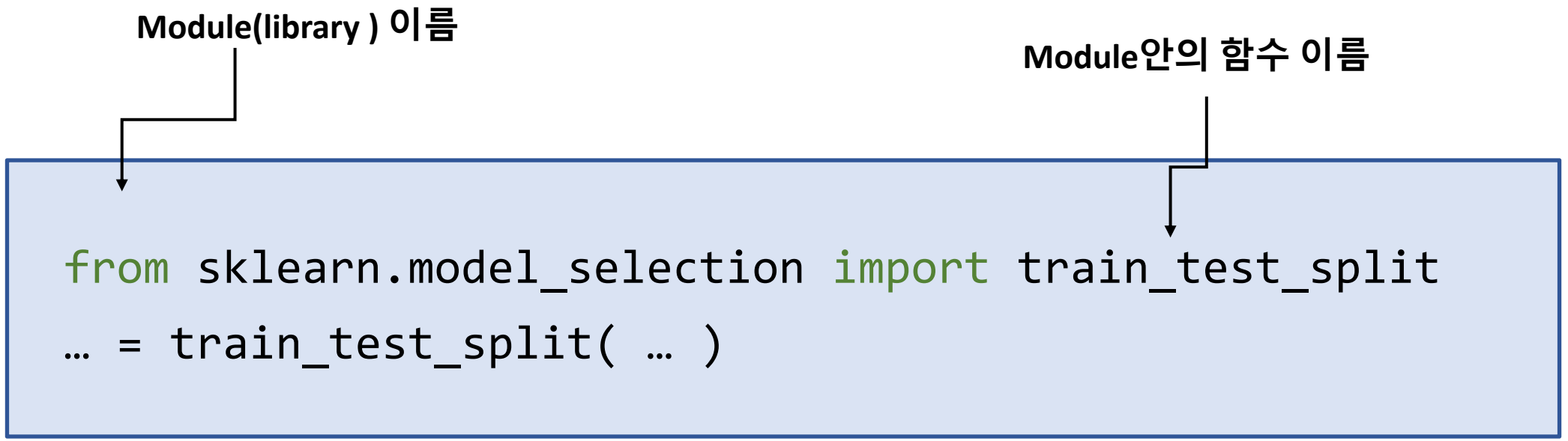
```
print(titanic.isnull().sum())
```

Lab 1. Decision Tree – step3

- Dataframe에서 사용할 데이터는 [‘pclass’, ‘sex’, ‘age’, ‘sibsp’, ‘parch’, ‘fare’]와 같다.
- 이에 사용할 data를 만들어준다.

Lab 1. Decision Tree – step4

- 아래 함수를 사용해서, dataset에서 train dataset과 test dataset을 분리할 수 있다.
  - Import numpy as np와는 다르게 모듈명을 쓰지 않고 함수 이름을 그대로 사용할 수 있다.



Lab 1. Decision Tree – step5

- Decision Tree를 구성해서, 학습시킨다.

Lab 1. Decision Tree – step6

- 학습결과의 성능을 확인해 본다.
- 아래의 함수를 사용해서 각각 성능지표를 확인해본다.

```
from sklearn.metrics import accuracy_score,
precision_score ,recall_score, f1_score
```

Lab 1. Decision Tree – step7

- Step6의 성능지표를 함수가 아닌 numpy만을 이용해서 구현해본다.
- 뒤 페이지의 hint를 참고해서 구현해본다.

Lab 1. Decision Tree – Accuracy

- Accuracy = $\frac{True\ Nagative + True\ Positive}{Total}$
  - Accuracy(정확도)는 전체에서 실제 positive를 positive라 예측한 것과 실제 negative를 negative라 예측한 것의 비율
  - Hint : np.mean(), np.equal()

Predicted(pred)

Actual (label)	**Negative(0)**	**Positive(1)**
Negative(0)	True Negative	False Positive
Positive(1)	False Negative	True Positive

Lab 1. Decision Tree - Precision

- Precision = $\frac{True\ Positive}{True\ Positive + False\ Positive}$

  $= \frac{True\ Positive}{Total\ Predicted\ Positive}$

  - Precision은 예측이 positive라고 했을 때, 실제 positive의 비율
  - Hint : np.sum() && label과 pred중 하나라도 0이면 곱하면 0이 나온다.

Predicted(pred)

Actual (label)	**Negative(0)**	**Positive(1)**
Negative(0)	True Negative	False Positive
Positive(1)	False Negative	True Positive

Lab 1. Decision Tree - Recall

- Recall $= \frac{True\ Positive}{True\ Positive + False\ Negative}$

$= \frac{True\ Positive}{Total\ Actual\ Positive}$

  - Recall은 실제 positive 중에 모델이 positive라 예측한 것의 비율

Predicted(pred)

Actual (label)	**Negative(0)**	**Positive(1)**
Negative(0)	True Negative	False Positive
Positive(1)	False Negative	True Positive

Lab 1. Decision Tree – F_measure

- F_measure = $\frac{2 * precision * recall}{precision + recall}$ (= *f1_score*)
  - 앞서 구현한 Precision, Recall의 조화평균
  - 앞서 구현한 precision과 recall을 이용해서 구현하면 편리하게 구현할 수 있음

Predicted(pred)

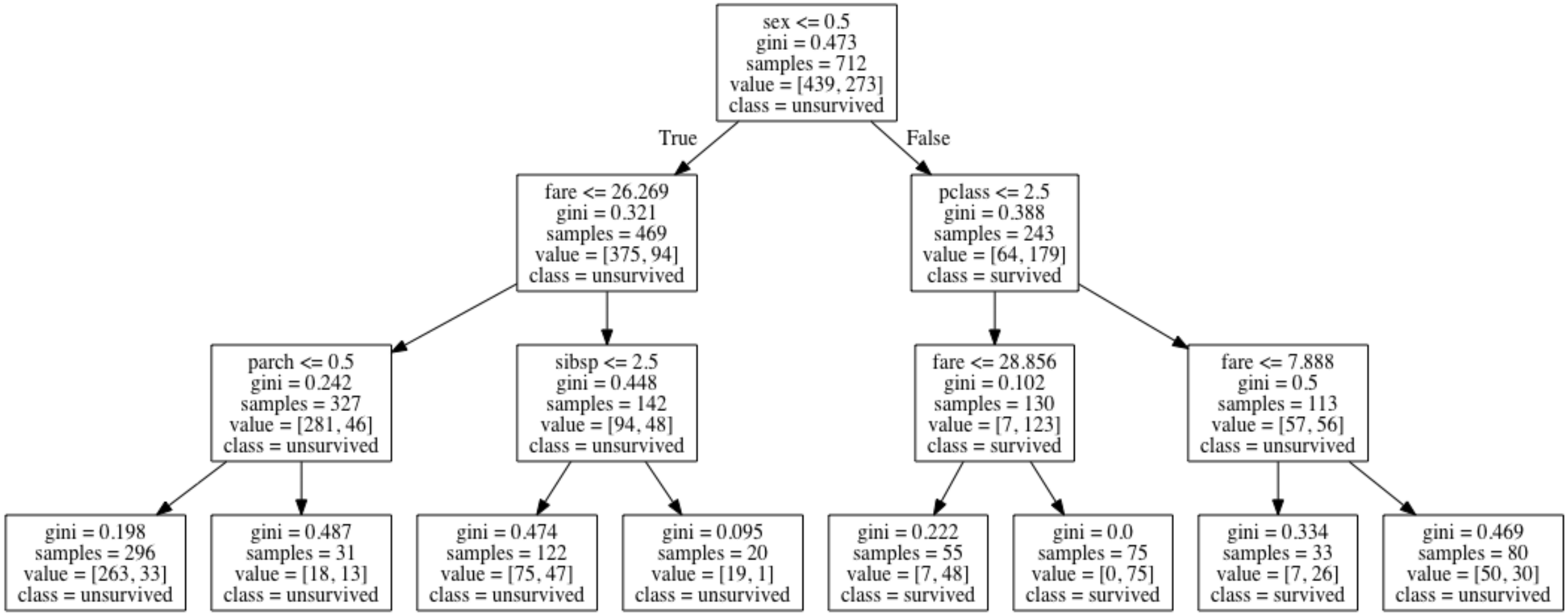
Actual (label)	**Negative(0)**	**Positive(1)**
Negative(0)	True Negative	False Positive
Positive(1)	False Negative	True Positive

Lab 1. Decision Tree – step8

- [‘pclass’, ‘sex’, ‘age’, ‘sibsp’, ‘parch’, ‘fare’] 의 정보로 survived, unsurvived 를 구해보고, 그 확률을 구해본다.
  - Dt.predict_proba() 를 이용해서 확률을 구할 수 있다.

```
jane = [3,1,30,0,0,5]
liam = [2,2,40,1,4,1]
```

Lab 1. Decision Tree – 확인



K-Nearest Neighbor (KNN) Classifier

Lab 2. KNN – step 1

- Dataset을 불려온다.

```
from sklearn import datasets
mnist = datasets.load_digits()
```

- Decision Tree에서 했던 것과 마찬가지로 train, test data를 나눈다.

Lab 2. KNN – step2

- KNN모델을 학습시킨다.
- Decision Tree에서 했던 것과 마찬가지로 함수를 사용해서 accuracy, prediction, recall, f1_measure을 확인해본다.
  - 아래와 같은 함수를 사용하면 쉽게 accuracy 등을 확인할 수 있다.
  - 앞에서 Numpy로 구현한 함수는 이곳에서 측정하면 값이 이상하다. why?

```
from sklearn.metrics import classification_report
Classification_report()
```

Lab 2. KNN – step3

- 램덤한 수(test set에서)를 모델에 넣고, 결과를 확인해본다.
- 그리고 이를 plot해 본다.
- 잘 맞추고 있음을 확인한다.