Machine Learning with Python

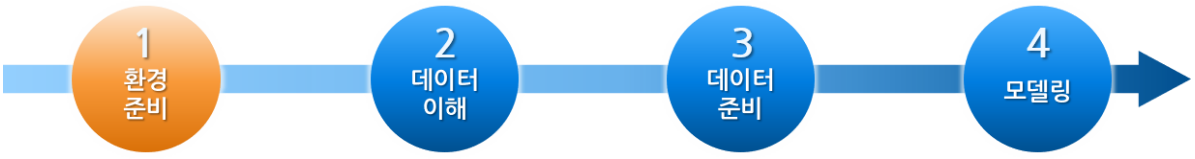
Life is too short, You need Python

실습 내용

- 머신러닝 모델링을 위한 코딩은 무조건 할 수 있어야 합니다.
- 코딩 내용을 자세히 알지 못해도 무작정 코딩을 진행해봅니다.
- Titanic 데이터를 대상으로 모델링을 진행합니다.
- kNN 알고리즘을 사용합니다.
- 다양한 방법으로 모델 성능을 평가합니다.

1.환경 준비

- 기본 **라이브러리**와 대상 **데이터**를 가져와 이후 과정을 준비합니다.



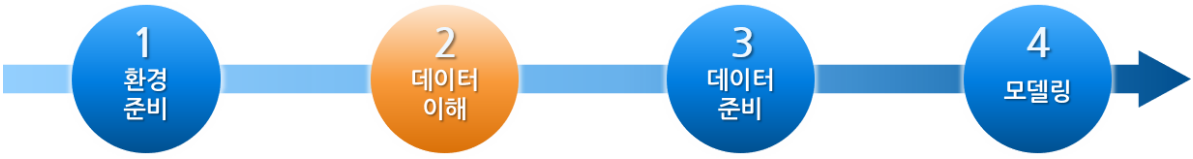
In [89]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

```
warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format = 'retina'
```

In [90]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/Jangrae/csv/master/titanic.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 **충분히 이해**할 수 있도록 다양한 **탐색** 과정을 수행합니다.



In [91]:
```
# 상/하위 몇 개 행 확인
data.head()
```

Out[91]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Er
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [92]:
```
# 하위 몇 개 행 확인
data.tail()
```

Out[92]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	

In [93]:
```
# 변수 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [94]:
```
# 기술통계 확인
data.describe()
```

Out[94]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [95]:
```
# 상관관계 확인
data.corr(numeric_only=True)
```

Out[95]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

3.데이터 준비

- **전처리** 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.

1) 변수 제거

- 분석에 의미가 없다고 판단되는 변수는 제거합니다.

In [96]:
```
# 제거 대상: PassengerId, Name, Ticket, Cabin
drop_cols = ['PassengerId', 'Name', 'Ticket', 'Cabin']
data = data.drop(drop_cols, axis=1)

# 확인
data.head()
```

Out[96]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

2) 결측치 처리

- 결측치가 있으면 제거하거나 적절한 값으로 채웁니다.

In [97]:
```
data.isna().sum()
```

Out[97]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [98]:
```
# Pclass별 Age의 중앙값
P_age = data.groupby(by='Pclass', as_index=False)['Age'].transform('median')
```

In [99]:
```
P_age
```

Out[99]:
```
0      24.0
1      37.0
2      24.0
3      37.0
4      24.0
       ...
886    29.0
887    37.0
888    24.0
889    37.0
890    24.0
Name: Age, Length: 891, dtype: float64
```

In [100…
```
# Age 결측치를 중앙값으로 채우기
data['Age'].fillna(P_age, inplace=True)
```

In [101…
```
# 최빈값
data['Embarked'].mode()[0]
```

Out[101]:
```
'S'
```

In [102…
```
# Embarked 최빈값 'S'로 채우기
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
```

In [103…
```
data.isna().sum()
```

Out[103]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

3) x, y 분리

- 우선 target 변수를 명확히 지정합니다.
- target을 제외한 나머지 변수들 데이터는 x로 선언합니다.
- target 변수 데이터는 y로 선언합니다.
- 이 결과로 만들어진 x는 데이터프레임, y는 시리즈가 됩니다.
- 이후 모든 작업은 x, y를 대상으로 진행합니다.

In [79]:
```
# target 확인
target = 'Survived'

# 데이터 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

4) 가변수화

- 범주형 변수를 가변수화 합니다.

In [104…
```
# 가변수화 대상: Pclass, Sex, Embarked
dumm_cols = ['Pclass', 'Sex', 'Embarked']


# 가변수화
x = pd.get_dummies(data, columns=dumm_cols, drop_first=True, dtype=int)

#  확인
x.head()
```

Out[104]:

	Survived	Age	SibSp	Parch	Fare	Pclass_2	Pclass_3	Sex_male	Embarked_Q	Embarked_S
0	0	22.0	1	0	7.2500	0	1	1	0	1
1	1	38.0	1	0	71.2833	0	0	0	0	0
2	1	26.0	0	0	7.9250	0	1	0	0	1
3	1	35.0	1	0	53.1000	0	0	0	0	1
4	0	35.0	0	0	8.0500	0	1	1	0	1

In [105…
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    891 non-null    int64
 1   Age         891 non-null    float64
 2   SibSp       891 non-null    int64
 3   Parch       891 non-null    int64
 4   Fare        891 non-null    float64
 5   Pclass_2    891 non-null    int32
 6   Pclass_3    891 non-null    int32
 7   Sex_male    891 non-null    int32
 8   Embarked_Q  891 non-null    int32
 9   Embarked_S  891 non-null    int32
dtypes: float64(2), int32(5), int64(3)
memory usage: 52.3 KB
```

5) 학습용, 평가용 데이터 분리

- 학습용, 평가용 데이터를 적절한 비율로 분리합니다.
- 반복 실행 시 동일한 결과를 얻기 위해 random_state 옵션을 지정합니다.

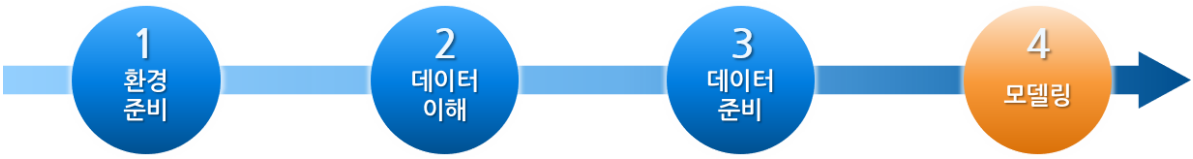
In [106…

```
# 모듈 불러오기
from sklearn.model_selection import train_test_split

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

4.모델링

- 본격적으로 모델을 **선언**하고 **학습**하고 **평가**하는 과정을 진행합니다.
- 우선 **회귀** 문제인지 **분류** 문제인지 명확히 구분합니다.



- 회귀 문제 인가요? 분류 문제인가요?
- 회귀인지 분류인지에 따라 사용할 알고리즘과 평가 방법이 달라집니다.
- 우선 다음 알고리즘을 사용합니다.
  - 알고리즘: KNeighborsClassifier

In [107…

```
# 1단계: 불러오기
from sklearn.neighbors import KNeighborsClassifier
```

In [108…

```
# 2단계: 선언하기
model = KNeighborsClassifier()
```

In [109…
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[109]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [110…
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

5.분류 성능 평가

- 다양한 성능 지표로 분류 모델 성능을 평가합니다.

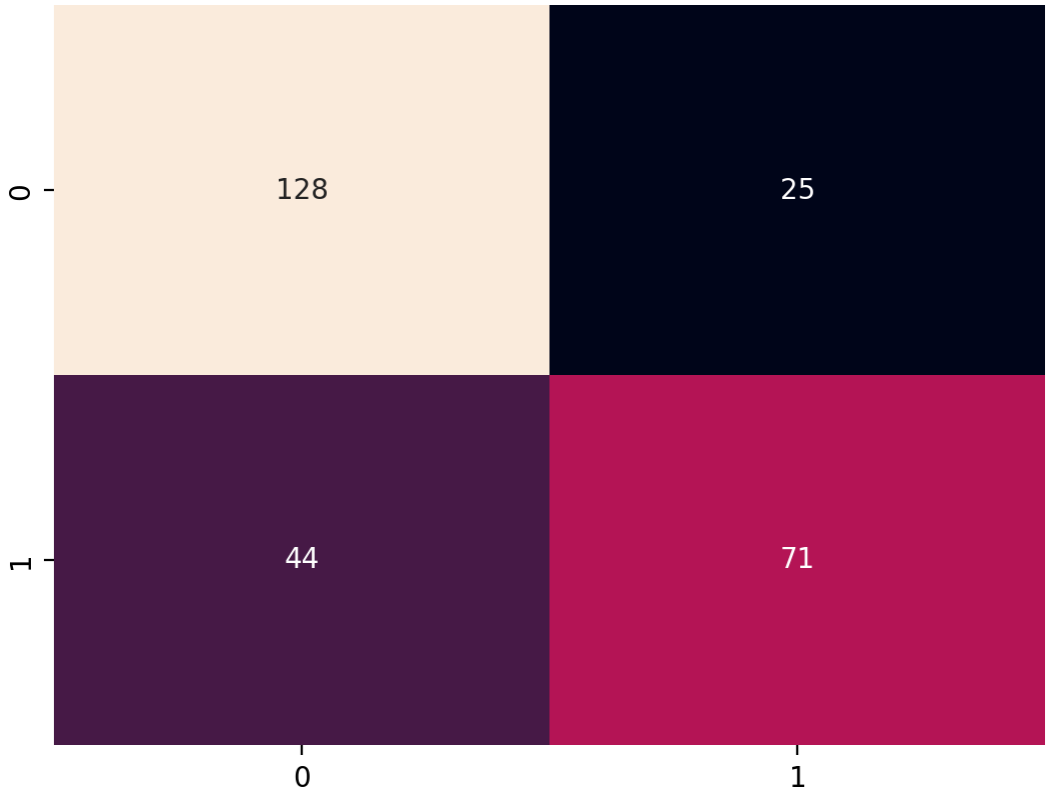
1) Confusion Matrix

In [111…
```
# 모듈 불러오기
from sklearn.metrics import confusion_matrix

# 성능 평가
print('Confusion Matrix:', confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix: [[128  25]
 [ 44  71]]
```

In [129…
```
# 혼동행렬 시각화
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cbar=False, fmt='d')
plt.show()
```



2) Accuracy

In [117…
```
# 모듈 불러오기
from sklearn.metrics import accuracy_score

# 성능 평가
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7425373134328358
```

3) Precision

In [118…
```
# 모듈 불러오기
from sklearn.metrics import precision_score

# 성능 평가
print('Precision:', precision_score(y_test, y_pred, average=None))
```

```
Precision: [0.74418605 0.73958333]
```

4) Recall

In [119…
```
# 모듈 불러오기
from sklearn.metrics import recall_score

# 성능 평가
print('Recall:', recall_score(y_test, y_pred, average=None))
```

```
Recall: [0.83660131 0.6173913 ]
```

5) F1-Score

In [121…
```
# 모듈 불러오기
from sklearn.metrics import f1_score

# 성능 평가
print('F1:', f1_score(y_test, y_pred, average=None))
```

```
F1: [0.78769231 0.67298578]
```

6) Classification Report

In [126…
```
# 모듈 불러오기
from sklearn.metrics import classification_report

# 성능 평가
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.84      0.79       153
           1       0.74      0.62      0.67       115

    accuracy                           0.74       268
   macro avg       0.74      0.73      0.73       268
weighted avg       0.74      0.74      0.74       268
```

In [125…
```
# 참조
print('학습성능', model.score(x_train, y_train))
print('평가성능', model.score(x_test, y_test))
```

```
학습성능 0.8507223113964687
평가성능 0.7425373134328358
```

In [ ]: