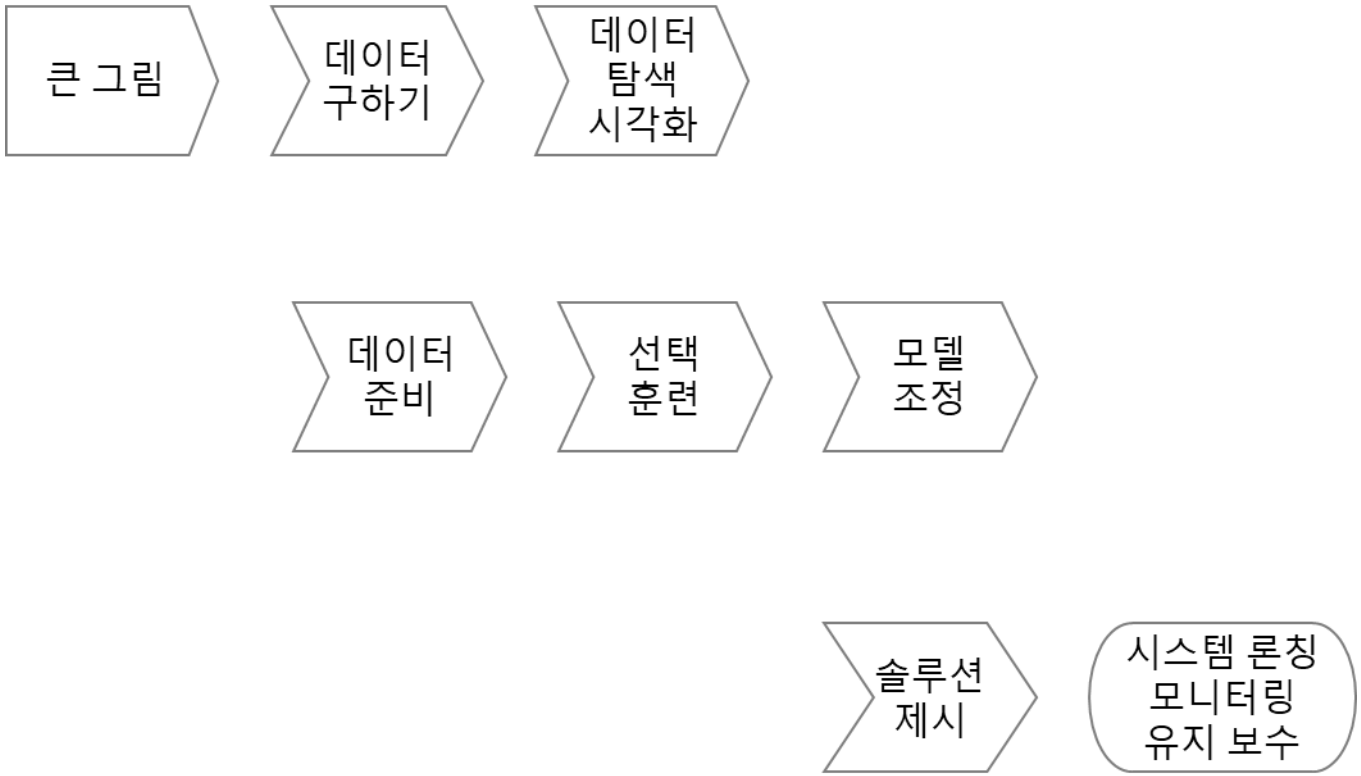
사이킷런

- 권수태 교수

JEONJU UNIVERSITY
The Place for Superstars

1. 프로젝트개발

❖ 프로젝트 개발 전단계



2. 라이브러리

❑ 라이브러리

➢ 현대 프로그래밍 언어는 오픈소스로 공개

- 파이썬, R, Ruby, Perl, Julia, Swift 등
- 제3자 라이브러리 풍부한 장점
- 파이썬은 하루에도 수십 개의 새로운 라이브러리가 공개됨

구분		공식 사이트	튜토리얼 문서
언어	파이썬 (Python)	https://www.python.org	• The Python Tutorial: https://docs.python.org/3/tutorial/
라이브러리 관리	파이파이 (Pypi)	https://pypi.org	• Installing packages: https://packaging.python.org/tutorials/installing-packages • Packaging Python projects: https://packaging.python.org/tutorials/packaging-projects

2. 라이브러리

❑ 라이브러리

라이브러리	넘파이 (Numpy)	https://numpy.org	• Numpy Tutorial: https://numpy.org/devdocs/user/quickstart.html
	맷플롯립 (Matplotlib)	https://matplotlib.org	• User's guide: https://matplotlib.org/users/index.html
	사이킷 런 (Scikit-learn)	https://scikit-learn.org	• User guide: https://scikit-learn.org/stable/user_guide.html
	텐서플로 (TensorFlow)	https://www.tensorflow.org	• Tensorflow Tutorials: https://www.tensorflow.org/tutorials
	케라스 (Keras)	https://keras.io	• Keras documentation: https://keras.io
	파이토치 (PyTorch)	https://pytorch.org	• Welcome to Pytorch tutorials: https://pytorch.org/tutorials

2. 라이브러리

❑ 라이브러리

➢ 기초 라이브러리

- 넘파이(Numpy): 다차원 배열 지원
- 맷플롯립(Matplotlib): 데이터 시각화

➢ 인공지능 라이브러리

- 사이킷런(Scikit-learn): 고전적인 기계 학습 지원
- 텐서플로(TensorFlow): 딥러닝 지원
- 케라스(Keras): 텐서플로를 한 단계 추상화한 라이브러리
- 파이토치(PyTorch): 딥러닝 라이브러리

3. 사이킷런

❖ 사이킷런

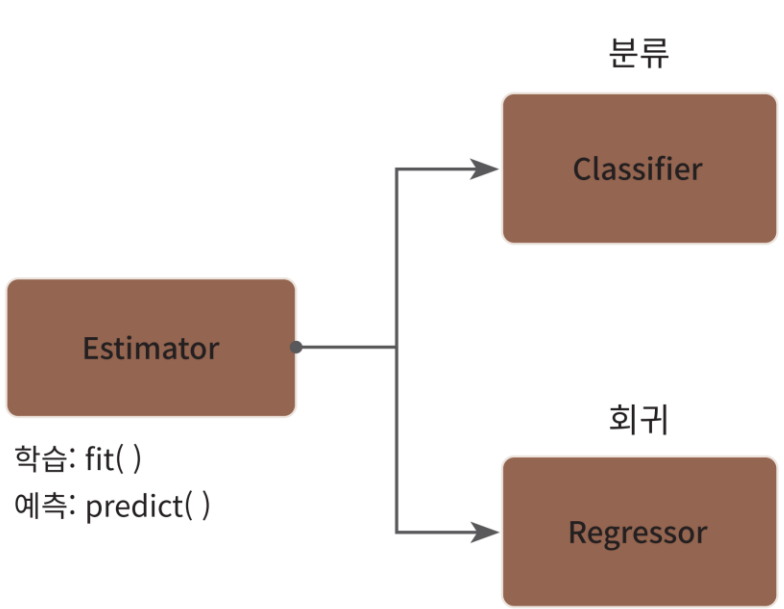
- ➢ 파이썬 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리
- ➢ 파이썬 기반의 머신러닝을 위한 가장 쉽고 효율적인 개발 라이브러리
- ➢ 특징
  - ✓ 머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 API를 제공
  - ✓ 오랜 기간 실전 환경에서 검증됐으며, 매우 많은 환경에서 사용되는 성숙한 라이브러리

3. 사이킷런

❖ 사이킷런

➢ Estimator

✓ 지도학습의 모든 알고리즘을 구현한 클래스



분류 구현 클래스

DecisionTreeClassifier
RandomForestClassifier
GradientBoostingClassifer
GaussianNB
SVC

회귀 구현 클래스

LinearRegression
Ridge
Lasso
RandomForestRegressor
GradientBoostingRegressor

3. 사이킷런

➢ 사이킷런의 주요모듈

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
피처 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 가공 기능 제공(문자열을 숫자형 코드 값으로 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 피처를 우선순위대로 셀렉션 작업을 수행하는 다양한 기능 제공
피처 처리	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 피처를 추출하는 데 사용됨. 예를 들어 텍스트 데이터에서 Count Vectorizer나 Tf-Idf Vectorizer 등을 생성하는 기능 제공. 텍스트 데이터의 피처 추출은 sklearn.feature_extraction.text 모듈에, 이미지 데이터의 피처 추출은 sklearn.feature_extraction.image 모듈에 지원 API가 있음.
피처 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련한 알고리즘을 지원하는 모듈임. PCA, NMF, Truncated SVD 등을 통해 차원 축소 기능을 수행할 수 있음
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드 서치(Grid Search)로 최적 파라미터 추출 등의 API 제공

3. 사이킷런

평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공
	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등을 제공
ML 알고리즘	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘을 지원. 또한 SGD(Stochastic Gradient Descent) 관련 알고리즘도 제공
	sklearn.naive_bayes	나이브 베이즈 알고리즘 제공. 가우시안 NB, 다항 분포 NB 등.
	sklearn.neighbors	최근접 이웃 알고리즘 제공. K-NN 등
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
	sklearn.cluster	비지도 클러스터링 알고리즘 제공 (K-평균, 계층형, DBSCAN 등)
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

3. 사이킷런

➢내장된 예제 데이터세트

API 명	설명
datasets.load_boston()	회귀 용도이며, 미국 보스턴의 집 피처들과 가격에 대한 데이터 세트
datasets.load_breast_cancer()	분류 용도이며, 위스콘신 유방암 피처들과 악성/음성 레이블 데이터 세트
datasets.load_diabetes()	회귀 용도이며, 당뇨 데이터 세트
datasets.load_digits()	분류 용도이며, 0에서 9까지 숫자의 이미지 픽셀 데이터 세트
datasets.load_iris()	분류 용도이며, 붓꽃에 대한 피처를 가진 데이터 세트

분류와 클러스터링을 위한 표본 데이터 생성기

API 명	설명
datasets.make_classifications( )	분류를 위한 데이터 세트를 만듭니다. 특히 높은 상관도, 불필요한 속성 등의 노이즈 효과를 위한 데이터를 무작위로 생성해 줍니다.
datasets.make_blobs( )	클러스터링을 위한 데이터 세트를 무작위로 생성해 줍니다. 군집 지정 개수에 따라 여러 가지 클러스터링을 위한 데이터 세트를 쉽게 만들어 줍니다.

3. 사이킷런

➢ 내장된 예제 데이터세트

✓ fetch 계열의 명령

- 인터넷에서 내려받아 홈 디렉토리 아래의 scikit_learn_data 라는 서브 디렉토리에 저장한 후 추후 불러들이는 데이터
- fetch_covtype() : 회귀분석용 토지조사 자료
- fetch_20newsgroups() : 뉴스그룹 텍스트 자료
- fetch_olivetti_faces(), fetch_lfw_people(), fetch_lfw_pairs() : 얼굴 이미지 자료
- fetch_rcv1() : 로이터 뉴스 말뭉치
- fetch_mldata() : ML 웹사이트에서 다운로드

3. 사이킷런

➢ 내장된 예제 데이터세트

- ✓ 내장된 데이터세트는 일반적으로 딕셔너리 형태로 되어 있음
- ✓ 키는 data, target, target_name, feature_names, DESCR로 구성되어 있음
  - ✓ data : 피처의 데이터 세트를 가리킴
  - ✓ target : 분류시 레이블값, 회귀일때는 숫자 결과값 데이터세트
  - ✓ target_names : 개별 레이블의 이름을 나타냄
  - ✓ feature_names : 피처의 이름을 나타냄
  - ✓ DESCR : 데이터세트에 대한 설명과 각 피처의 설명을 나타냄
  - ✓ data, target 은 넘파이 배열 타입
  - ✓ target_names, feature_names 넘파이 배열 또는 파이썬 리스트 타입
  - ✓ DESCR 스트링 타입
- ✓ 피처의 데이터 값을 반환하기 위해서는 내장 데이터 세트 API를 호출한 뒤에 그 Key 값을 지정하면 됨

3. 사이킷런

❖ Model Selection 모듈

- ➢ 모델, 파라미터, 그리고 학습
  - ✓ 머신러닝은 문제를 해결하는 모델model
  - ✓ 모델의 동작을 결정하는 파라미터parameter
  - ✓ 파라미터를 더 좋은 상태로 변경하는 학습learning 동작으로 이루어짐
- ➢ 모델은 현재의 파라미터를 바탕으로 어떤 행위를 할 것
  - ✓ 그 결과는 보통은 차이가 남. 이것이 모델의 오차error
  - ✓ 오차가 없다 = 학습이 완벽하게 잘 되었다 = 모델이 데이터를 잘 설명한다
- ➢ 학습이란 이 오차가 줄어드는 방향(모델이 데이터를 잘 설명하는 방향)으로 파라미터를 변경하는 일

3. 사이킷런

❖ Model Selection 모듈

- 학습데이터와 테스트 데이터 세트를 분리하거나, 교차 검증 분할 및 평가, Estimator의 하이퍼 파라미터를 튜닝하기 위한 다양한 함수와 클래스 제공
- 학습/테스트 데이터 셋 분리 – train_test_split()
  - ✓ 학습과 예측을 동일한 데이터 세트로 수행: 예측정확도 100%
  - ✓ test_size : 전체데이터에서 테스트 데이터 세트 크기를 얼마로 샘플링할 것인가를 결정. default=25%
  - ✓ train_size : 전체데이터에서 학습용 데이터 세트 크기를 얼마로 샘플링할 것인가를 결정
  - ✓ shuffle : 데이터를 분리하기 전에 데이터를 미리 섞을지를 결정함
  - ✓ random_state : 데이터 세트를 생성하기 위해 주어지는 난수 값(동일한 데이터 세트로 분리하기 위해 사용)

3. 사이킷런

❖ Model Selection 모듈

➢ 교차 검증

✓ 과적합 문제 해결

✓ 데이터 편중을 막기 위해서 별도의 여러 세트로 구성된 학습 데이터세트와 검증 데이터세트에서 학습과 평가를 수행하는 것

3. 사이킷런

❖ Model Selection 모듈

➢ 교차 검증

✓ K 폴드 교차검증

- K 개의 데이터 폴드 세트를 만들어서 K 번 만큼 각 폴트 세트에 학습과 검증 평가를 반복적으로 수행하는 방법

JEONJU UNIVERSITY

3. 사이킷런

❖ Model Selection 모듈

➢ 교차 검증

✓ Stratified K 폴드

- 불균형한 분포도를 가진 레이블(결정 클래스) 데이터 집합을 위한 K 폴드 방식
- K 폴드가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 경우의 문제를 해결
- 원본 데이터의 레이블 분포를 먼저 고려한 뒤 이 분포와 동일하게 학습과 검증 데이터 세트를 분배
- 분류에서의 교차검증으로 사용

3. 사이킷런

❖ Model Selection 모듈

➢ 교차 검증

✓ cross_val_score( )

- 교차검증을 좀 더 편리하게 수행할 수 있게 해주는 API
- K 폴드로 데이터를 학습하고 예측하는 코드의 과정을 한꺼번에 수행해주는 API (내부적으로 stratifiedKFold 이용)
- Cross_val_score(estimator, X, y=none, scoring=None, cv=None, n_jobs=1, verbose=0, fit_params=None, pre_dispatch= ' 2 * n_jobs ')
- X : 피처 데이터세트, y : 레이블 데이터세트, scoring : 예측 성능평가지표, cv : 교차 검증 폴드수
- 반환값은 scoring 파라미터로 지정된 성능 지표 측정값을 배열형태로 반환

3. 사이킷런

❖ Model Selection 모듈

➢ GridSearchCV

- ✓ Classifier 나 Regressor 와 같은 알고리즘에 사용되는 하이퍼 파라미터를 순차적으로 입력하면서 편리하게 최적의 파라미터를 도출할 수 있는 방안을 제공
- ✓ 교차검증을 기반으로 하이퍼 파라미터의 최적값을 찾게 해 줌
- ✓ 즉, 데이터 세트를 cross-validation 을 위한 학습/테스트 세트로 자동으로 분할한 뒤에 하이퍼 파라미터 그리드에 기술된 모든 파라미터를 순차적으로 적용해 최적의 파라미터를 찾을 수 있게 해 줌
- ✓ 학습데이터를 GridSearchCV 를 이용해 최적 하이퍼 파라미터 튜닝을 수행한 뒤에 별도의 테스트 세트에서 이를 평가하는 것이 일반적인 머신러닝 모델 적용방법

3. 사이킷런

❖ Model Selection 모듈

➢ GridSearchCV

✓ GridSearchCV 클래스의 생성자로 들어가는 주요 파라미터

- estimator: classifier, regressor, pipeline
- param_grid: key+리스트 값을 가지는 딕셔너리가 주어짐
- scoring: 예측성능을 측정할 평가방법을 지정
- cv: 교차 검증을 위해 분할되는 학습/테스트 세트의 개수
- refit: 가장 최적의 하이퍼 파라미터를 찾은 뒤 입력된 estimator 객체를 해당 하이퍼파라미터로 재학습시킴
- 학습을 수행하면 최고 성능의 하이퍼파라미터와 평가결과 값을 best_params_, best_score_ 속성에 기록

3. 사이킷런

❖ 데이터 전처리

➢ 머신러닝 알고리즘이 허용하지 않는 것: 결손값 (NaN, Null), 문자열 값

➢ 데이터 인코딩

✓ 레이블 인코딩(Label encoding)

- 카테고리 피처를 코드형 숫자 값으로 변환하는 것
- LabelEncoder 를 객체로 생성한 후 fit()과 transform()을 호출해 레이블 인코딩을 수행

✓ 원-핫 인코딩(One-Hot encoding)

- 피처값의 유형에 따라 새로운 피처를 추가해 고유값에 해당하는 칼럼에만 1을 표시하고 나머지 칼럼에는 0을 표시하는 방식
- 즉, 행 형태로 되어있는 피처의 고유값을 열 형태로 차원을 변환한 뒤, 고유값에 해당하는 칼럼에만 1 표시
- 변환전 모든 문자열 값이 숫자형으로 변환되어야 하며, 입력값으로 2차원 데이터 필요

3. 사이킷런

❖ 데이터 전처리

원본 데이터

상품 분류	가격
TV	1,000,000
냉장고	1,500,000
전자렌지	200,000
컴퓨터	800,000
선풍기	100,000
선풍기	100,000
믹서	50,000
믹서	50,000

숫자로 인코딩

상품 분류	가격
0	1,000,000
1	1,500,000
4	200,000
5	800,000
3	100,000
3	100,000
2	50,000
2	50,000

원-핫 인코딩

TV	냉장고	믹서	선풍기	전자렌지	컴퓨터	가격
1	0	0	0	0	0	1,000,000
0	1	0	0	0	0	1,500,000
0	0	0	0	1	0	200,000
0	0	0	0	0	1	800,000
0	0	0	1	0	0	100,000
0	0	0	1	0	0	100,000
0	0	1	0	0	0	50,000
0	0	1	0	0	0	50,000

3. 사이킷런

❖ 데이터 전처리

➢ 피처 스케일링과 정규화

✓ 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업

✓ StandardScaler

- 개별 피처를 평균 0, 분산 1인 값으로 변환

✓ MinMaxScaler

- 서로 다른 피처의 크기를 통일(0-1)
- 데이터의 분포가 가우시안 분포가 아닐 경우

$$x_i\_new = \frac{x_i - mean(x)}{stdev(x)}$$

$$x_i\_new = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

3. 사이킷런

❖ 데이터 전처리

➢ 학습 데이터와 테스트 데이터의 스케일링 변환시 유의점

- ✓ Scaler 객체를 이용해 데이터의 스케일링 변환시 fit(), transform(), fit_transform() 메소드를 이용
  - fit() : 데이터 변환을 위한 기준 정보(최대/최소값) 설정을 적용
  - Transform() : 설정된 정보를 이용해 데이터를 변환
- ✓ Scaler 객체를 이용해 학습데이터 세트로 fit(), transform() 을 적용하면 테스트 데이터 세트로는 다시 fit()을 수행하지 않고 학습데이터 세트로 fit() 을 수행한 결과를 이용해 transform() 변환을 적용해야 함
- ✓ 즉, 학습데이터로 fit()이 적용된 스케일링 기준 정보를 그대로 테스트 데이터에 적용해야 함
- ✓ 가능하다면 전체 데이터의 스케일링 변환을 적용한 뒤, 학습과 테스트 데이터로 분리
- ✓ 불가능시 테스트 데이터 변환시에는 fit()이나 fit_transform() 을 적용하지 않고, 학습데이터로 이미 fit()된 Scaler 객체를 이용해 transform() 으로 변환

4. 사이킷런 프로그램

❖ 타이타닉 데이터

survival	Survival (0 = No; 1 = Yes)
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name	Name
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
ticket	Ticket Number
fare	Passenger Fare
cabin	Cabin
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

titanic_df = pd.read_csv('./titanic_train.csv')
titanic_df.head(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
print('\n ### train 데이터 정보 ### \n')
print(titanic_df.info())
```

```
 ### train 데이터 정보 ###

<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
None
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
titanic_df['Age'].fillna(titanic_df['Age'].mean(),inplace=True)
titanic_df['Cabin'].fillna('N',inplace=True)
titanic_df['Embarked'].fillna('N',inplace=True)
print('데이터 세트 Null 값 갯수 ',titanic_df.isnull().sum().sum())
```

데이터 세트 Null 값 갯수 0

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
print(' Sex 값 분포 :\n',titanic_df['Sex'].value_counts())
print('\n Cabin 값 분포 :\n',titanic_df['Cabin'].value_counts())
print('\n Embarked 값
분포 :\n',titanic_df['Embarked'].value_counts())
```

```
Sex 값 분포 :
 male      577
female    314
Name: Sex, dtype: int64

 Cabin 값 분포 :
 N            687
G6             4
C23 C25 C27    4
```

```
 Embarked 값 분포 :
 S    644
C    168
Q     77
N      2
Name: Embarked, dtype: int64
```

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
titanic_df['Cabin'] = titanic_df['Cabin'].str[:1]
print(titanic_df['Cabin'])

# 선실등급을 나타내는 첫번째 알파벳 추출
```

```
titanic_df.groupby(['Sex','Survived'])['Survived'].count()
```

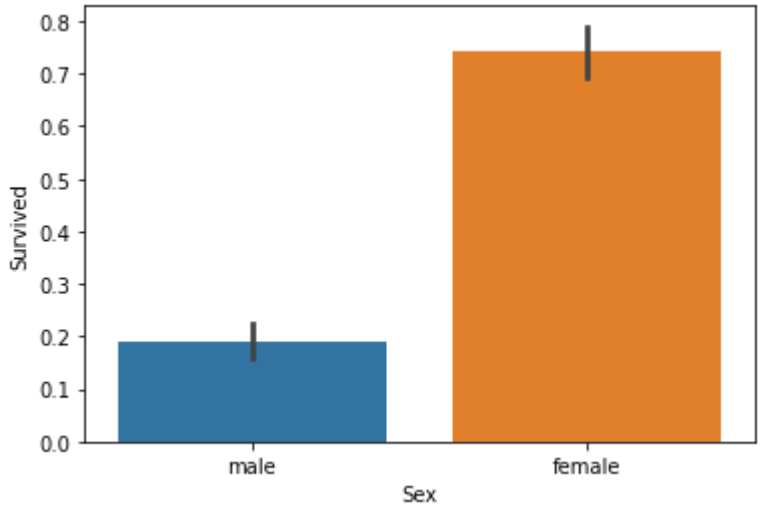
```
Sex     Survived
female  0          81
        1         233
male    0         468
        1         109
Name: Survived, dtype: int64
```

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
sns.barplot(x='Sex', y = 'Survived', data=titanic_df)
```

<AxesSubplot:xlabel='Sex', ylabel='Survived'>

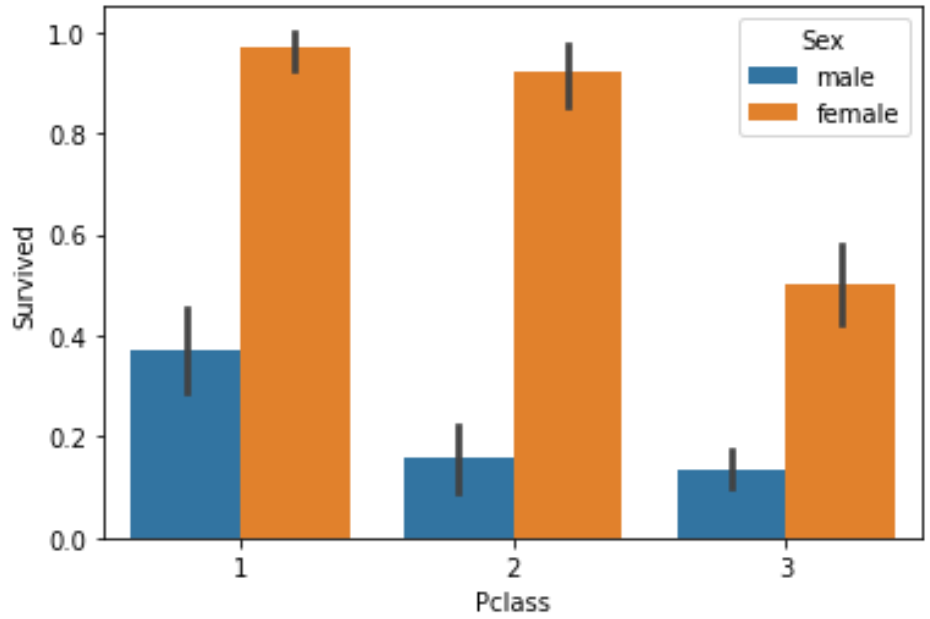


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
sns.barplot(x='Pclass', y='Survived', hue='Sex', data=titanic_df)
```

<AxesSubplot:xlabel='Pclass', ylabel='Survived'>



4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# 입력 age에 따라 구분값을 반환하는 함수 설정. DataFrame의 apply lambda식에 사용.
def get_category(age):
    cat = ''
    if age <= -1: cat = 'Unknown'
    elif age <= 5: cat = 'Baby'
    elif age <= 12: cat = 'Child'
    elif age <= 18: cat = 'Teenager'
    elif age <= 25: cat = 'Student'
    elif age <= 35: cat = 'Young Adult'
    elif age <= 60: cat = 'Adult'
    else : cat = 'Elderly'
    return cat
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

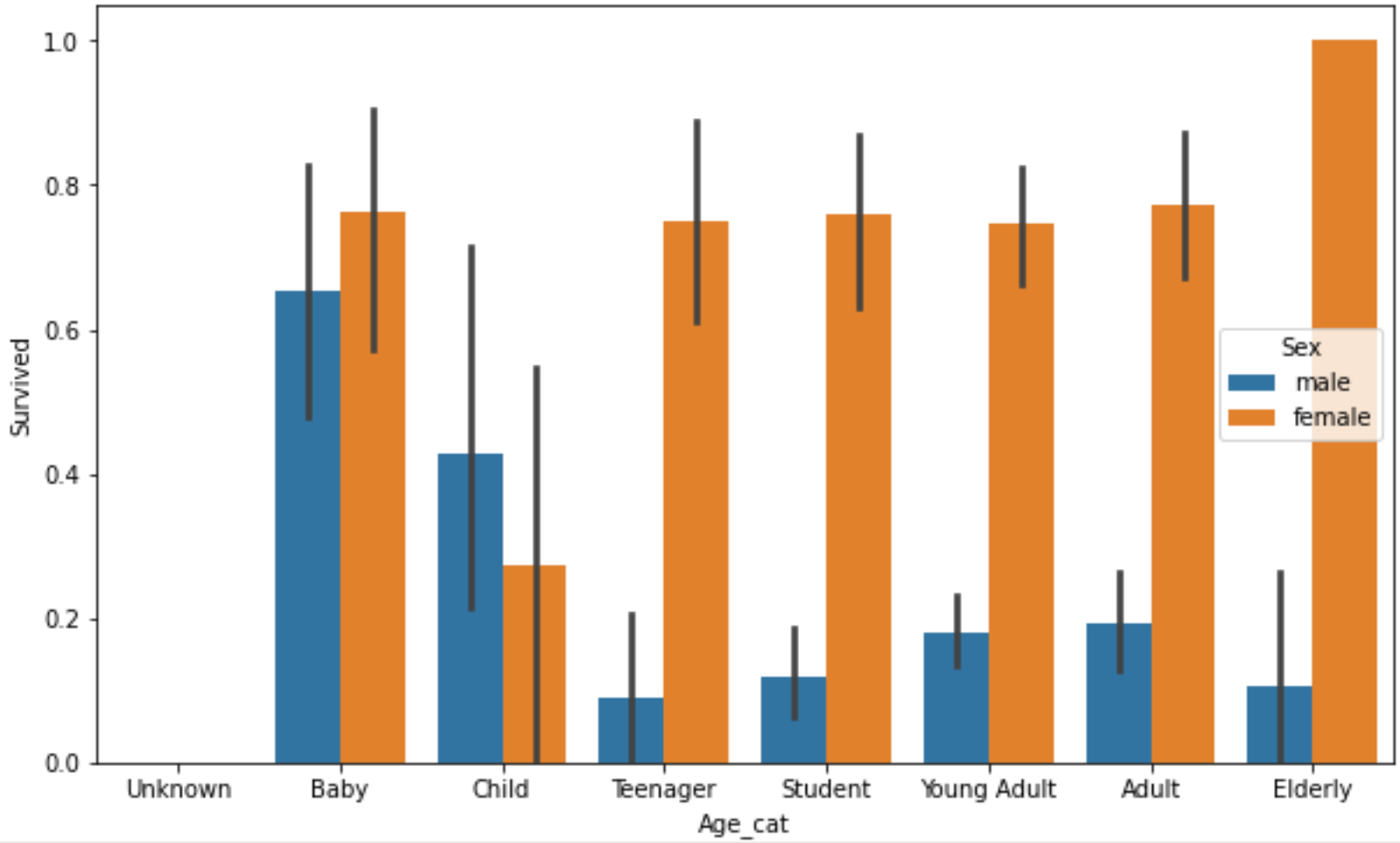
```
plt.figure(figsize=(10,6))

#X축의 값을 순차적으로 표시하기 위한 설정
group_names = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student',
'Young Adult', 'Adult', 'Elderly']

# lambda 식에 위에서 생성한 get_category( ) 함수를 반환값으로 지정.
# get_category(X)는 입력값으로 'Age' 컬럼값을 받아서 해당하는 cat 반환
titanic_df['Age_cat'] = titanic_df['Age'].apply(lambda x :
get_category(x))
sns.barplot(x='Age_cat', y = 'Survived', hue='Sex', data=titanic_df,
order=group_names)
titanic_df.drop('Age_cat', axis=1, inplace=True)
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터



JEONJU UNIVERSITY

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn import preprocessing

def encode_features(dataDF):
    features = ['Cabin', 'Sex', 'Embarked']
    for feature in features:
        le = preprocessing.LabelEncoder()
        le = le.fit(dataDF[feature])
        dataDF[feature] = le.transform(dataDF[feature])

    return dataDF

titanic_df = encode_features(titanic_df)
titanic_df.head()
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	7	3
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	2	0
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	7	3
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	2	3
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	7	3

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn.preprocessing import LabelEncoder

# Null 처리 함수
def fillna(df):
    df['Age'].fillna(df['Age'].mean(),inplace=True)
    df['Cabin'].fillna('N',inplace=True)
    df['Embarked'].fillna('N',inplace=True)
    df['Fare'].fillna(0,inplace=True)
    return df
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# 머신러닝 알고리즘에 불필요한 속성 제거
def drop_features(df):
    df.drop(['PassengerId','Name','Ticket'],axis=1,inplace=True)
    return df

# 레이블 인코딩 수행.
def format_features(df):
    df['Cabin'] = df['Cabin'].str[:1]
    features = ['Cabin','Sex','Embarked']
    for feature in features:
        le = LabelEncoder()
        le = le.fit(df[feature])
        df[feature] = le.transform(df[feature])
    return df
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# 앞에서 설정한 Data Preprocessing 함수 호출
def transform_features(df):
    df = fillna(df)
    df = drop_features(df)
    df = format_features(df)
    return df
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# 원본 데이터를 재로딩 하고, feature데이터 셋과 Label 데이터 셋 추출.

titanic_df = pd.read_csv('./titanic_train.csv')
y_titanic_df = titanic_df['Survived']
X_titanic_df= titanic_df.drop('Survived',axis=1)

X_titanic_df = transform_features(X_titanic_df)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test
      =train_test_split(X_titanic_df, y_titanic_df,
                                        test_size=0.2, random_state=11)
```

4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# 결정트리, Random Forest, 로지스틱 회귀를 위한 사이킷런 Classifier 클래스 생성

dt_clf = DecisionTreeClassifier(random_state=11)
rf_clf = RandomForestClassifier(random_state=11)
lr_clf = LogisticRegression()
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# DecisionTreeClassifier 학습/예측/평가
dt_clf.fit(X_train , y_train)
dt_pred = dt_clf.predict(X_test)
print('DecisionTreeClassifier 정확도: {0:.4f}'.format(accuracy_score(y_test, dt_pred)))

# RandomForestClassifier 학습/예측/평가
rf_clf.fit(X_train , y_train)
rf_pred = rf_clf.predict(X_test)
print('RandomForestClassifier 정확도:{0:.4f}'.format(accuracy_score(y_test, rf_pred)))

# LogisticRegression 학습/예측/평가
lr_clf.fit(X_train , y_train)
lr_pred = lr_clf.predict(X_test)
print('LogisticRegression 정확도: {0:.4f}'.format(accuracy_score(y_test, lr_pred)))
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
DecisionTreeClassifier 정확도: 0.7877
RandomForestClassifier 정확도:0.8547
LogisticRegression 정확도: 0.8492
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn.model_selection import KFold

def exec_kfold(clf, folds=5):
    # 폴드 세트를 5개인 KFold객체를 생성, 폴드 수만큼 예측결과 저장을 위한  리스트 객체 생성.
    kfold = KFold(n_splits=folds)
    scores = []

    # KFold 교차 검증 수행.
    for iter_count , (train_index, test_index) in enumerate(kfold.split(X_titanic_df)):
        # X_titanic_df 데이터에서 교차 검증별로 학습과 검증 데이터를 가리키는 index 생성
        X_train, X_test = X_titanic_df.values[train_index], X_titanic_df.values[test_index]
        y_train, y_test = y_titanic_df.values[train_index], y_titanic_df.values[test_index]
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
        # Classifier 학습, 예측, 정확도 계산
        clf.fit(X_train, y_train)
        predictions = clf.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        scores.append(accuracy)
        print("교차 검증 {0} 정확도: {1:.4f}".format(iter_count, accuracy))

    # 5개 fold에서의 평균 정확도 계산.
    mean_score = np.mean(scores)
    print("평균 정확도: {0:.4f}".format(mean_score))

# exec_kfold 호출
exec_kfold(dt_clf , folds=5)
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
교차 검증 0 정확도: 0.7542
교차 검증 1 정확도: 0.7809
교차 검증 2 정확도: 0.7865
교차 검증 3 정확도: 0.7697
교차 검증 4 정확도: 0.8202
평균 정확도: 0.7823
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(dt_clf, X_titanic_df , y_titanic_df , cv=5)

for iter_count,accuracy in enumerate(scores):
    print("교차 검증 {0} 정확도: {1:.4f}".format(iter_count, accuracy))

print("평균 정확도: {0:.4f}".format(np.mean(scores)))
```

```
교차 검증 0 정확도: 0.7430
교차 검증 1 정확도: 0.7753
교차 검증 2 정확도: 0.7921
교차 검증 3 정확도: 0.7865
교차 검증 4 정확도: 0.8427
평균 정확도: 0.7879
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
from sklearn.model_selection import GridSearchCV

parameters = {'max_depth':[2,3,5,10],
              'min_samples_split':[2,3,5], 'min_samples_leaf':[1,5,8]}

grid_dclf = GridSearchCV(dt_clf , param_grid=parameters ,
scoring='accuracy' , cv=5)
grid_dclf.fit(X_train , y_train)

print('GridSearchCV 최적
하이퍼파라미터 :',grid_dclf.best_params_)
print('GridSearchCV 최고 정확도:{0:.4f}'.format(grid_dclf.best_score_))
best_dclf = grid_dclf.best_estimator_
```


4. 사이킷런 프로그램

❖ 타이타닉 데이터

```
# GridSearchCV의 최적 하이퍼 파라미터로 학습된 Estimator로 예측 및 평가 수행.
dpredictions = best_dclf.predict(X_test)
accuracy = accuracy_score(y_test , dpredictions)
print('테스트 세트에서의 DecisionTreeClassifier 정확도 : {0:.4f}'.format(accuracy))
```

```
GridSearchCV 최적 하이퍼 파라미터 : {'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2}
GridSearchCV 최고 정확도: 0.7992
테스트 세트에서의 DecisionTreeClassifier 정확도 : 0.8715
```


Thank You !

JEONJU UNIVERSITY
The Place for Superstars