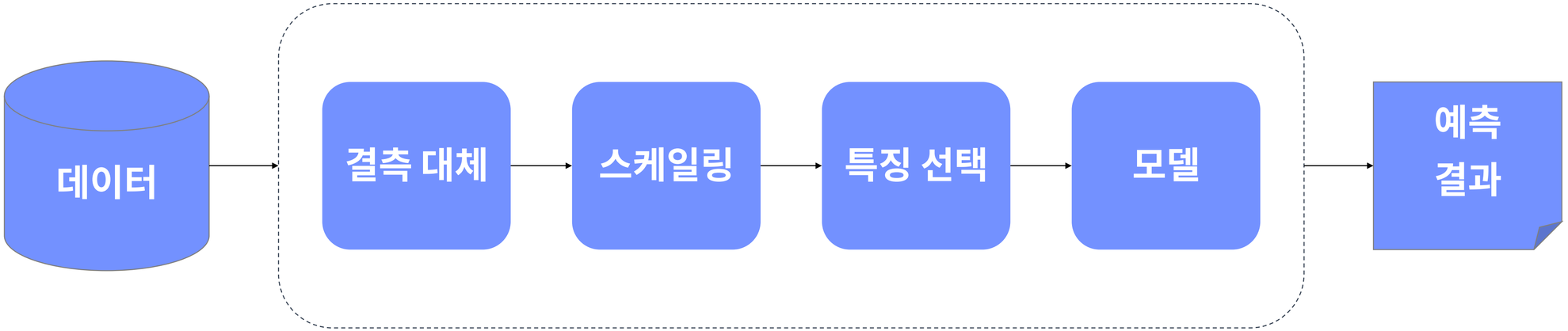
4. 파이프라인과 모델 저장

1 머신러닝 파이프라인

머신러닝 파이프라인이란?

넓은 의미에서 머신러닝 파이프라인은 데이터 수집, 전처리, 모델 학습, 모델 배포, 예측 등을 순차적으로 처리하는 일련의 프로세스입니다.
좁은 의미에서 머신러닝 파이프라인은 새로운 데이터의 라벨을 예측하는 일련의 프로세스입니다.

머신러닝 파이프라인 예시 [1)]



Zöller, M. A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of artificial intelligence research*, 70, 409-472.

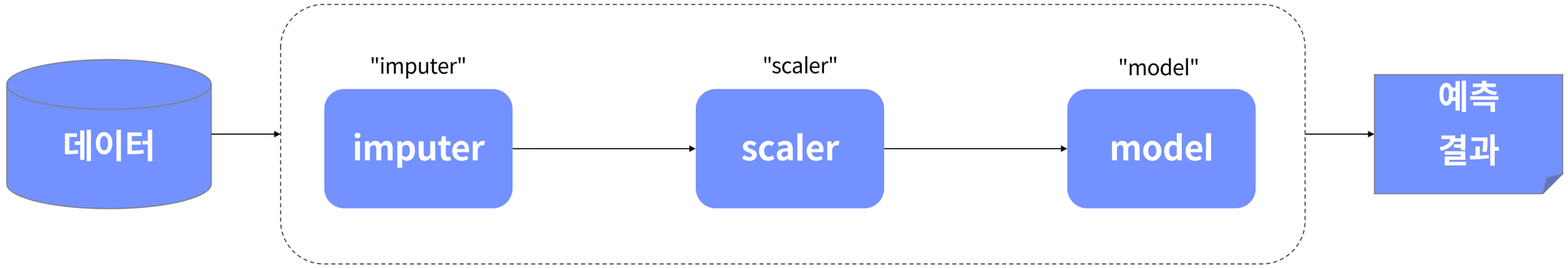
Pipeline 클래스

사이킷런의 Pipeline 클래스를 사용하면 손쉽게 파이프라인을 학습하고 활용할 수 있습니다.

주요 인자

인자	설명
steps	각 요소가 튜플인 리스트로, 이 튜플은 인스턴스의 이름과 인스턴스로 구성됨. steps에 있는 인스턴스가 새로 입력된 샘플을 전처리하거나 예측한 결과를 다음 인스턴스로 전달함

```
steps = [("imputer", imputer), ("scaler", scaler), ("model", model)]
```



예제 데이터 불러오기

파이프라인을 학습하고 활용하는 방법을 자세히 알아보는데 사용할 데이터를 불러옵니다.

데이터 불러오기 예제

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3
4  df = pd.read_csv("../../data/Classification/bands.csv")
5  X = df.drop('y', axis = 1)
6  y = df['y']
7  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2022)
```

파이프라인 구성 요소 정의

파이프라인의 구성 요소를 정의하겠습니다.

파이프라인 구성 요소 정의 예제

```
1  from sklearn.impute import SimpleImputer
2  from sklearn.preprocessing import MinMaxScaler
3  from sklearn.svm import SVC
4
5  imputer = SimpleImputer(strategy = "mean")
6  scaler = MinMaxScaler()
7  model = SVC(kernel = "rbf")
```

- 파이프라인의 구성 요소는 학습도 동시에 하므로 puter, scaler, model 모두 학습하지 않았음

파이프라인 정의 및 학습

파이프라인을 정의하고 학습해보겠습니다.

파이프라인 정의 및 학습 예제

```
1  from sklearn.pipeline import Pipeline
2  P = Pipeline([("imputer", imputer), ("scaler", scaler), ("model", model)])
3  P.fit(X_train, y_train)
4  display(P.predict(X_test))
```

```
array([0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 1], dtype=int64)
```

- 위 결과는 결측이 평균으로 대체되고 스케일링이 된 X_test가 SVC 모델에 입력돼 나온 결과임
- 즉, predict 메서드는 파이프라인의 마지막 요소인 SVC 모델의 메서드를 사용한 것임

Pipeline 클래스의 단점

peline 클래스는 손쉽게 파이프라인을 구축할 수 있다는 장점이 있지만, 몇몇 문제가 있어 실무에서 활용하기 어렵습니다.

사이킷런의 인스턴스가 아닌 다른 클래스의 인스턴스가 파이프라인에 포함되면 정상적으로 작동하지 않을 수 있음

하이퍼 파라미터 튜닝 등을 할 때 전체 파이프라인을 계속해서 수정해야 해서 번거로움

특히, 이미 학습한 인스턴스를 여러 번 재학습해야 할 수 있음

파이프라인 커스터마이징

파이프라인의 각 요소를 독립적으로 학습하고 함수를 사용해 연결하는 방식으로 파이프라인을 구현합니다.

파이프라인 커스터마이징 예제

```
1  imputer.fit(X_train, y_train)
2  scaler.fit(X_train, y_train)
3  model.fit(X_train, y_train)
```

```
1  def my_pipeline(X, imputer, scaler, model):
2      X = imputer.transform(X)
3      X = scaler.transform(X)
4      pred_Y = model.predict(X)
5      return pred_Y
```

```
1  pred_Y = my_pipeline(X_test, imputer, scaler, model)
2  display(pred_Y[:5])
```

```
array([0, 1, 0, 1, 0], dtype=int64)
```

4. 파이프라인과 모델 저장

2 피클을 활용한 모델 저장 및 불러오기

피클 모듈

피클 모듈은 파이썬 객체를 저장하고 불러오는데 사용합니다. 새로운 데이터가 입력될 때마다 모델을 재학습할 수 없으므로, 학습했던 파이프라인 혹은 인스턴스를 피클을 사용해 저장했다가 필요하면 꺼내써야 합니다.

주요 함수

함수	설명	인자
dump	파이썬 객체를 파일에 저장합니다.	• obj: 저장할 인스턴스 • file: 인스턴스를 저장할 파일 (주로 pckl 혹은 pkl 확장자를 가진 바이너리 파일)
load	저장한 인스턴스를 불러옵니다. 저장한 인스턴스가 둘 이상이라면 저장한 순서대로 불러옵니다.	• file: 인스턴스가 저장된 파일

피클 모듈을 활용한 파이프라인 저장 및 불러오기

피클 모듈을 활용해서 파이프라인을 저장하고 불러오겠습니다.

파이프라인 저장하기 예제

```
1  import pickle
2  with open("my_pipeline.pckl", "wb") as f:
3      pickle.dump(imputer, f)
4      pickle.dump(scaler, f)
5      pickle.dump(model, f)
```

- **라인 2**: with 구문을 이용해 my_pipeline.pckl을 바이너리로 쓰기 모드인 "wb"로 설정해서 f로 엽니다.
- **라인 3 – 5**: imputer, scaler, model을 순서대로 f에 저장합니다.

파이프라인 불러오기 예제

```
1  with open("my_pipeline.pckl", "rb") as f:
2      loaded_imputer = pickle.load(f)
3      loaded_scaler = pickle.load(f)
4      loaded_model = pickle.load(f)
```

- **라인 1:** with 구문을 이용해 my_pipeline.pckl을 f로 엽니다. 이번에는 읽는 것이므로 "rb"를 사용했습니다.

사전으로 변환하여 저장하기

파이프라인이 길어지면 인스턴스를 저장한 순서를 기억해야 하는 문제가 있습니다. 그래서 강사는 아래와 같이 주로 사전으로 병합하여 저장합니다.

파이프라인 저장하기 예제

```
1  my_pipeline_dict = {"imputer":imputer, "scaler":scaler, "model":model}
2  with open("my_pipeline_dict.pckl", "wb") as f:
3          pickle.dump(my_pipeline_dict, f)
```