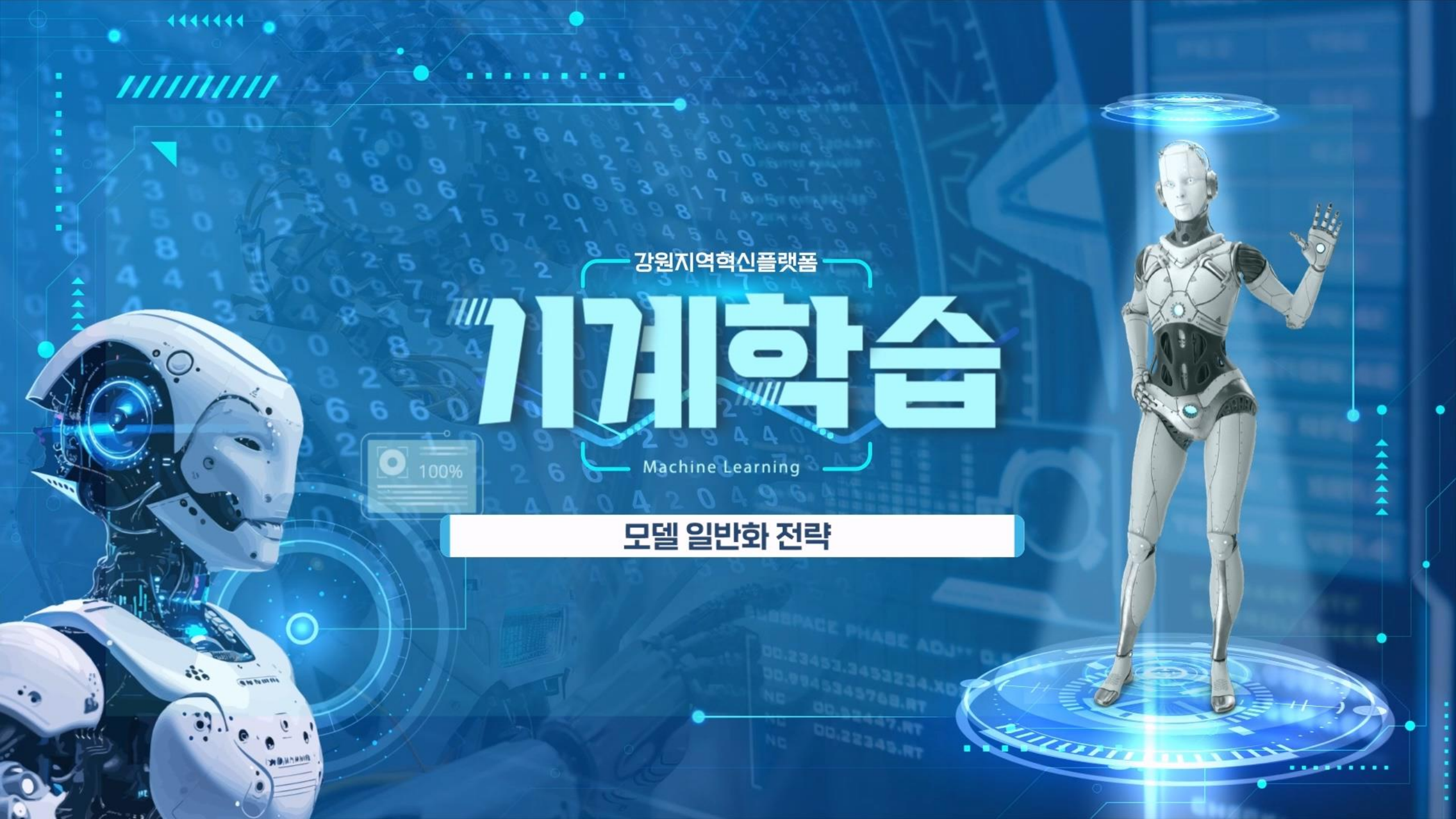
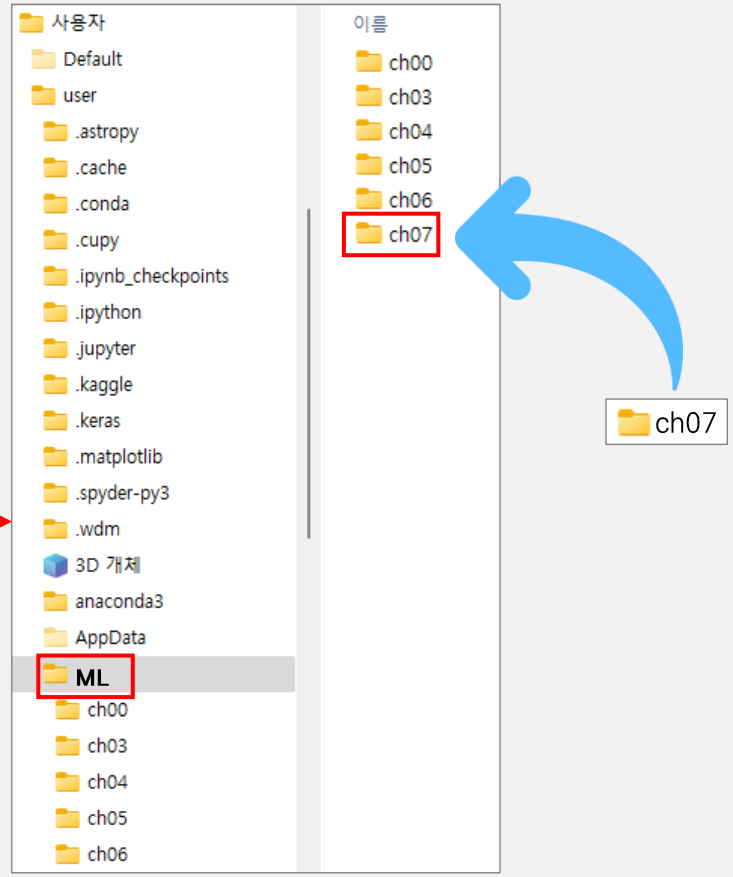

강원지역혁신플랫폼
기계학습
Machine Learning
100%
모델 일반화 전략

학습목표

- 모델 일반화 전략의 개념을 설명할 수 있습니다.

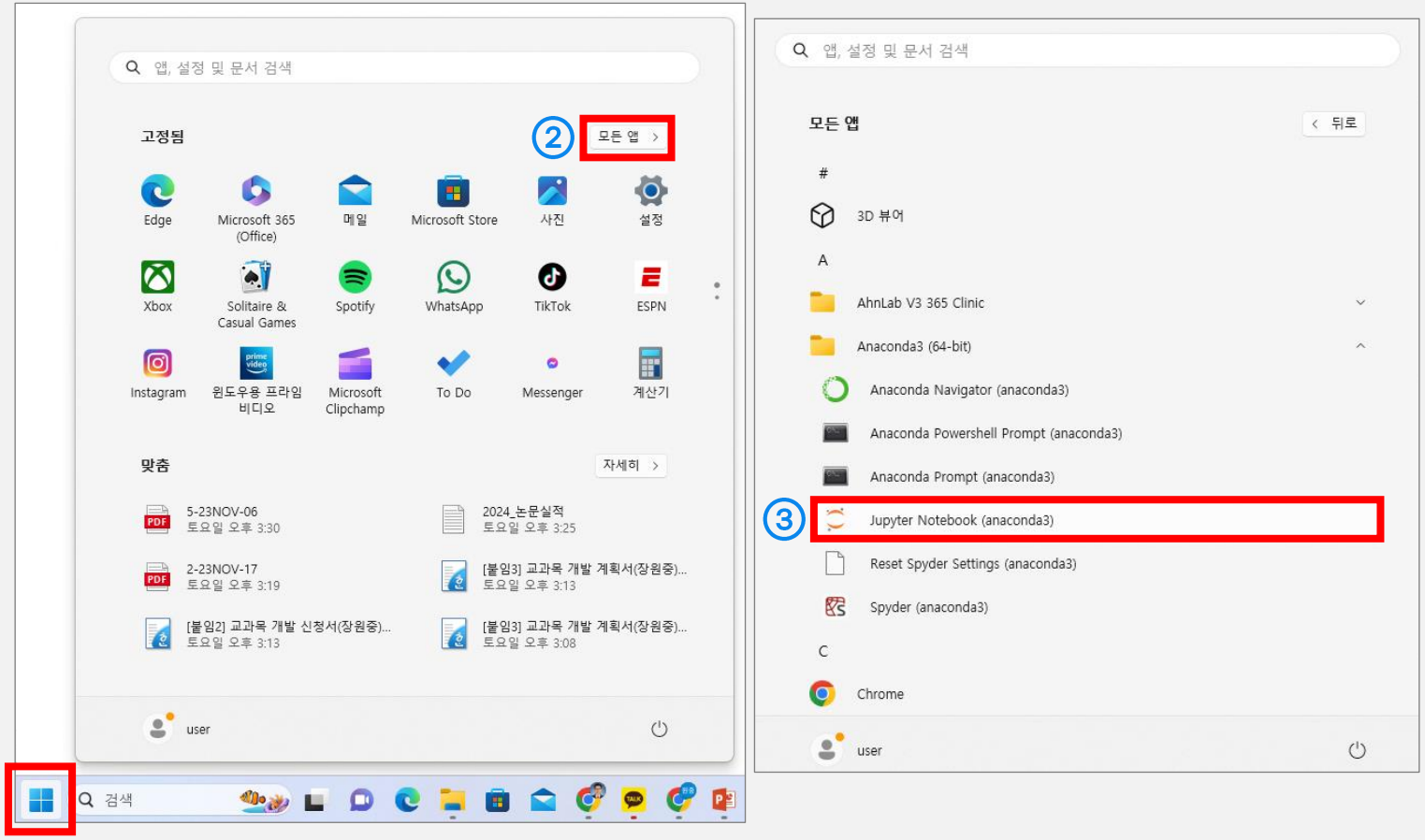
01 | 7주차 실습코드 복사하기

- (권장) 아래와 같은 경로에 실행 소스가 존재하면 환경 구축 완료
  - 7주차 실습코드 다운로드 → 압축해제 → ch07 폴더를 ML 하위 폴더로 복사
  - c:₩Users₩user>ML> 컴퓨터이름 또는 사용자계정



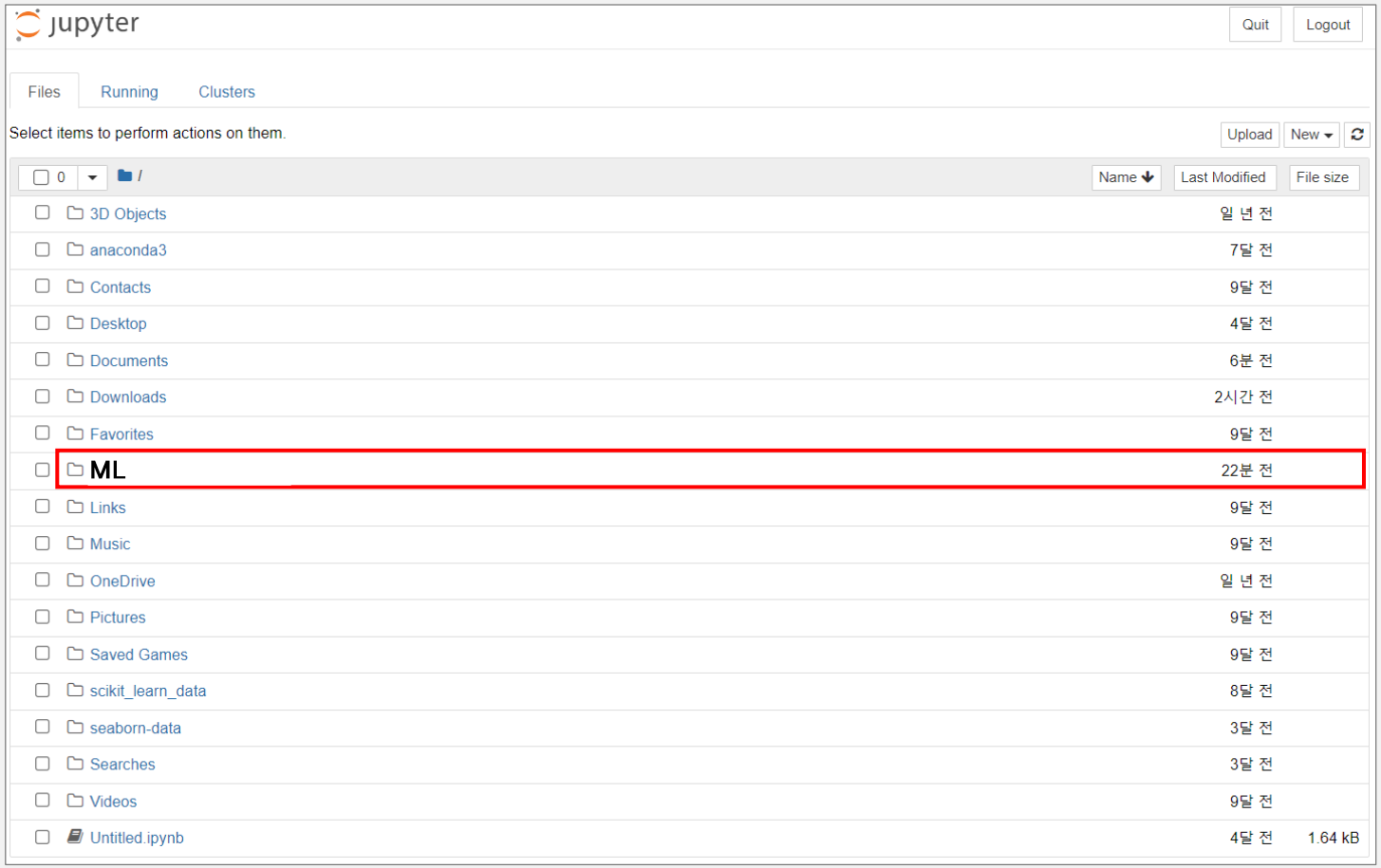
02 | Jupyter Notebook 실행하기

✦ ①시작 메뉴 클릭 > ②모든 앱 버튼 클릭 > ③Anaconda3(64-bit)
> “Jupyter Notebook (anaconda)” 메뉴 클릭하기



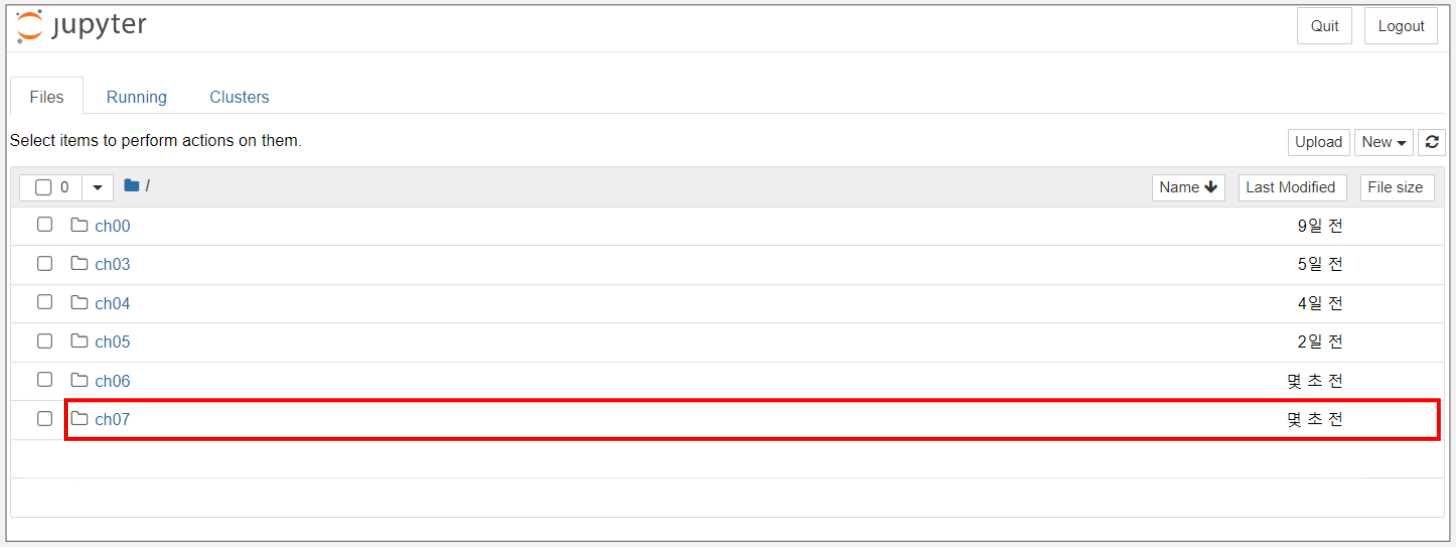
03 | ML 폴더

ML 폴더를 클릭하기



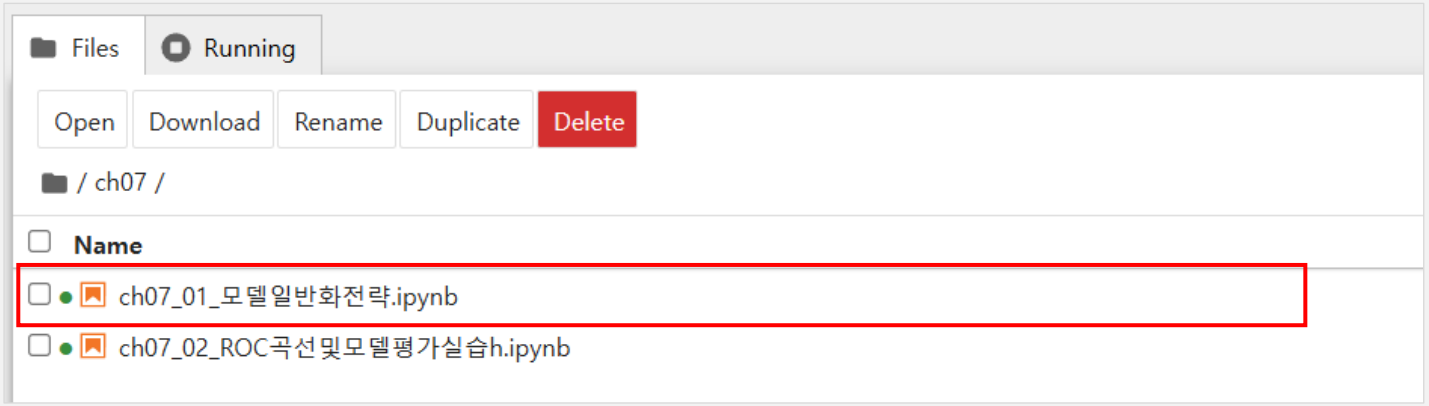
04 | ch07 폴더

ch07 폴더 클릭하기



05 | ch07_01_모델일반화전략.ipynb

✦ ch07_01_모델일반화전략.ipynb 파일 클릭하기



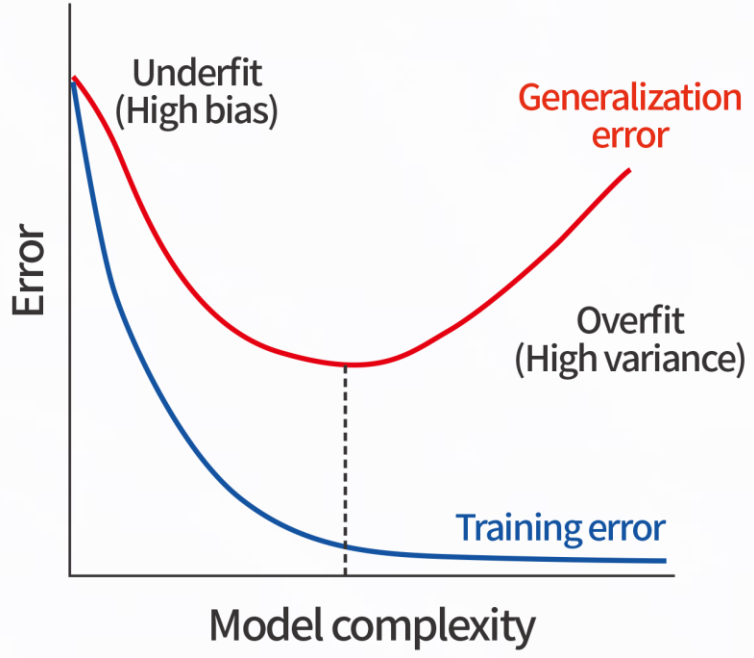
06 | 모델 일반화 전략

모델 일반화 전략

- 기계학습은 데이터로부터 모델의 성능이 결정되는 특징(data-driven)이 있으므로, 데이터가 충분히 많아야 성능을 발휘할 수 있음
  - 데이터가 부족한 경우에는 과적합(overfitting)문제가 발생할 수 있음
    - 모델이 학습 데이터의 특징에만 적합하여 새로운 데이터(unseen data)에 대한 예측 능력이 떨어지게 되는 것임

06 | 모델 일반화 전략

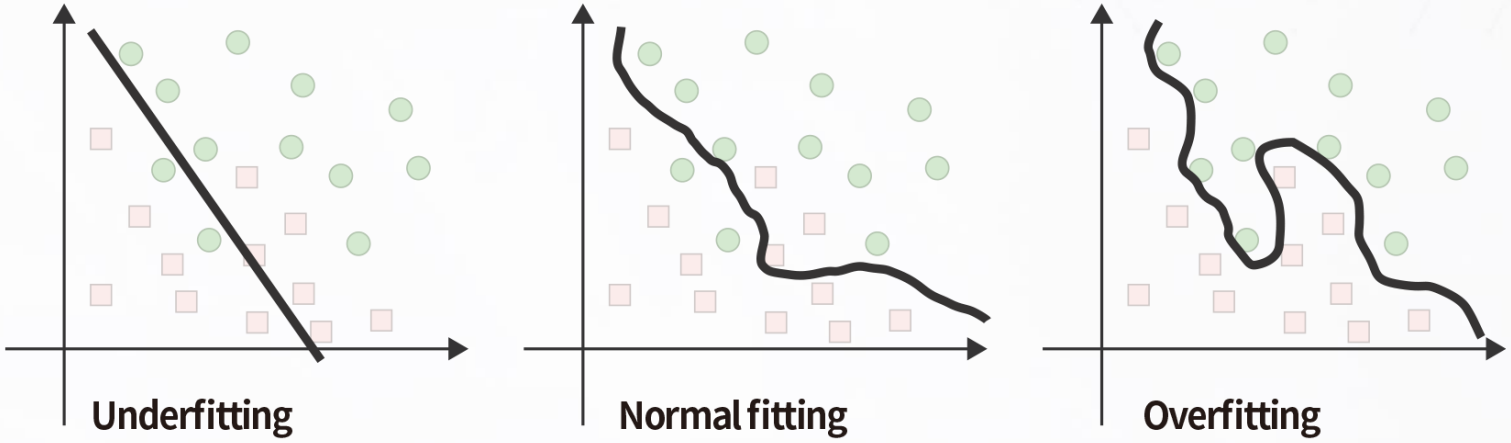
- 아래 그래프와 같이 학습 데이터에 대해서는 오차가 감소하지만, 실제 데이터에 대해서는 오차가 증가하는 지점이 존재함
  - 아래 그림에서는 테스트 에러가 감소하다 갑자기 치솟는 부분에서 과대적합이 발생했다고 볼 수 있음



06 | 모델 일반화 전략

훈련 데이터를 올바르게 학습시키기 위해서는 과대적합과 과소적합의 중간점을 찾는 것이 바람직함

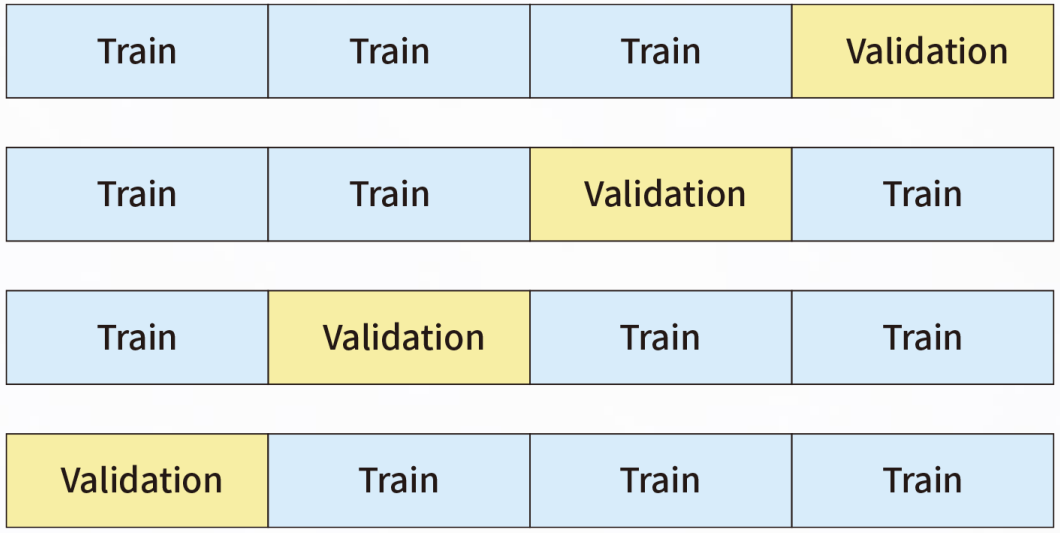
- 아래 그림에서 보이는 바와 같이 너무 잘 분류해도, 분류하지 못해도 올바른 모델이라고 할 수 없음



출처:https://tkleen.cafe24.com/1-머신러닝-학습-알고리즘의-구현/

06 | 모델 일반화 전략

- 테스트 데이터는 학습에서 모델에 간접적으로라도 영향을 미치면 안 되기 때문에 테스트 데이터로 검증을 해서는 안됨
  - 그래서, 검증(validation)데이터셋을 따로 두어 매 훈련마다 검증 데이터셋에 대해 평가하여 모델을 튜닝해야 함
    - 아래 그림과 같이 K-겹 교차 검증(K-fold cross-validation)을 사용하여 검증함



K-Fold

06 | 모델 일반화 전략

다음은 train_test_split()함수로 아이리스 데이트 셋을 훈련 데이터와 테스트 데이터를 7:3 비율로 분리하는 코드이다.

- 아래와 같이 데이터를 섞어서 종속 변수의 레이블을 기준으로 7:3 비율로 훈련 데이터와 테스트 데이터가 분리된 것을 볼 수 있음

```
X_train, X_test, y_train, y_test = train_test_split(iris.iloc[:, :-1],
                                                    iris.iloc[:, -1], test_size=0.3,
                                                    shuffle=True, stratify=iris['species'],
                                                    random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(y_train.value_counts())
print(y_test.value_counts())
```

```
(105, 4) (45, 4) (105,) (45,)
setosa        35
versicolor    35
virginica     35
Name: species, dtype: int64
setosa        15
versicolor    15
virginica     15
Name: species, dtype: int64
```

06 | 모델 일반화 전략

다음은 의사결정나무 모델을 생성하고 훈련 데이터로 학습을 수행하는 코드이다.

- 아래와 같이 모델이 훈련 데이터와 훈련 데이터 정답으로 학습된 것을 알 수 있음

```
model = DecisionTreeClassifier(random_state=1234) # 모델 생성
model.fit(X_train, y_train)                       # 학습
```

06 | 모델 일반화 전략

다음은 학습된 모델에 테스트 데이터로 모델 성능을 평가하는 코드이다.

- 모델 성능 평가결과 정확도가 약 97.78%인 것을 알 수 있음

```
pred = model.predict(X_test)
accuracy = np.round(accuracy_score(y_test, pred), 4)
print('\n## 검증 정확도:', accuracy)    # 검증 정확도: 0.9778
```

06 | 모델 일반화 전략

다음은 StratifiedShuffleSplit()함수로 아이리스 데이트 셋을 훈련 데이터와 테스트 데이터를 7:3 비율로 분리하고, 의사결정나무 모델로 10겹 교차검증을 수행하는 코드이다.

- 아래와 같이 데이터를 섞어서 종속 변수의 레이블을 기준으로 7:3 비율로 훈련 데이터와 테스트 데이터가 분리된 것을 볼 수 있음
  - 10겹 교차 검증의 평균 정확도는 약 95.33%인 것을 볼 수 있음

```
cv = StratifiedShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
results = cross_val_score(model, df_iris.iloc[:, :-1], df_iris.iloc[:, -1], cv=cv)
fin_result = np.round(np.mean(results), 4)

for i, _ in enumerate(results):
    print("{}번째 교차검증 정확도: {}".format(i, _))

print("교차검증 최종 정확도: {}".format(fin_result))
```

```
0번째 교차검증 정확도: 0.9555555555555556
1번째 교차검증 정확도: 0.9333333333333333
2번째 교차검증 정확도: 0.9555555555555556
3번째 교차검증 정확도: 0.9555555555555556
4번째 교차검증 정확도: 0.9555555555555556
5번째 교차검증 정확도: 0.9555555555555556
6번째 교차검증 정확도: 0.9333333333333333
7번째 교차검증 정확도: 0.9777777777777777
8번째 교차검증 정확도: 0.9777777777777777
9번째 교차검증 정확도: 0.9333333333333333
교차검증 최종 정확도: 0.9533
```

06 | 모델 일반화 전략

- 모델 최적화 전략을 위해 GridSearchCV()함수를 이용한 하이퍼파라미터 탐색을 수행할 수 있음
  - 하이퍼파라미터는 기계가 스스로 찾을 수 없어서 사람이 직접 지정해야 하는 파라미터를 의미함
    - 하이퍼파라미터를 찾는 방법도 분석가가 경험적으로 얻은 노하우로 찾는 것이 일반적임

06 | 모델 일반화 전략

- scikit-learn은 하이퍼파라미터를 찾는 GridSearchCV 기능을 제공함
  - 하이퍼파라미터 조합에 대한 경우의 수를 모두 격자(grid)에 나열함
    - 모든 조합을 일일이 학습 및 성능 측정하는 기능임
    - 시간이 오래 걸린다는 단점이 있음

06 | 모델 일반화 전략

다음은 의사결정나무 모델로 아이리스 데이터 셋에 GridSearch()함수를 이용해 모델의 최적의 하이퍼파라미터를 찾는 코드이다.

- 아래와 같이 StratifiedShuffleSplit()함수로 데이터를 섞어서 종속 변수의 레이블을 기준으로 7:3 비율로 훈련 데이터와 테스트 데이터가 분리된 것을 볼 수 있음

```
cv = StratifiedShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
estimator = DecisionTreeClassifier()
parameters = {'max_depth' : [4,5,6,10], 'criterion' : ['gini', 'entropy'], 'splitter' : ['best', 'random'],
              'min_weight_fraction_leaf' : [0.0, 0.1, 0.2, 0.3], 'random_state' : [7, 23, 42, 78],
              'min_impurity_decrease': [0.0, 0.05, 0.1, 0.2]}

model = GridSearchCV(estimator=estimator, param_grid=parameters, cv=cv,
                     verbose=-1, n_jobs=-1, refit=True)
model.fit(df_iris.iloc[:, :-1], df_iris.iloc[:, -1])
```

06 | 모델 일반화 전략

다음은 학습한 모델의 최적의 하이퍼파라미터 값을 확인하는 코드이다.

아래와 같이 최적의 하이퍼파라미터 값을 모델에 적용한 경우 약 95.8%의 정확도인 것을 알 수 있음

```
print("Best Estimator:\n", model.best_estimator_);print()
print("Best Params:\n", model.best_params_);print()
print("Best Score:\n", np.round(model.best_score_,4));print()
```

```
Best Estimator:
 DecisionTreeClassifier(max_depth=4, min_weight_fraction_leaf=0.1,
                        random_state=78, splitter='random')

Best Params:
 {'criterion': 'gini', 'max_depth': 4, 'min_impurity_decrease': 0.0, 'min_weight_fraction_leaf': 0.1, 'random_state': 78, 'splitter': 'random'}

Best Score:
 0.9578
```

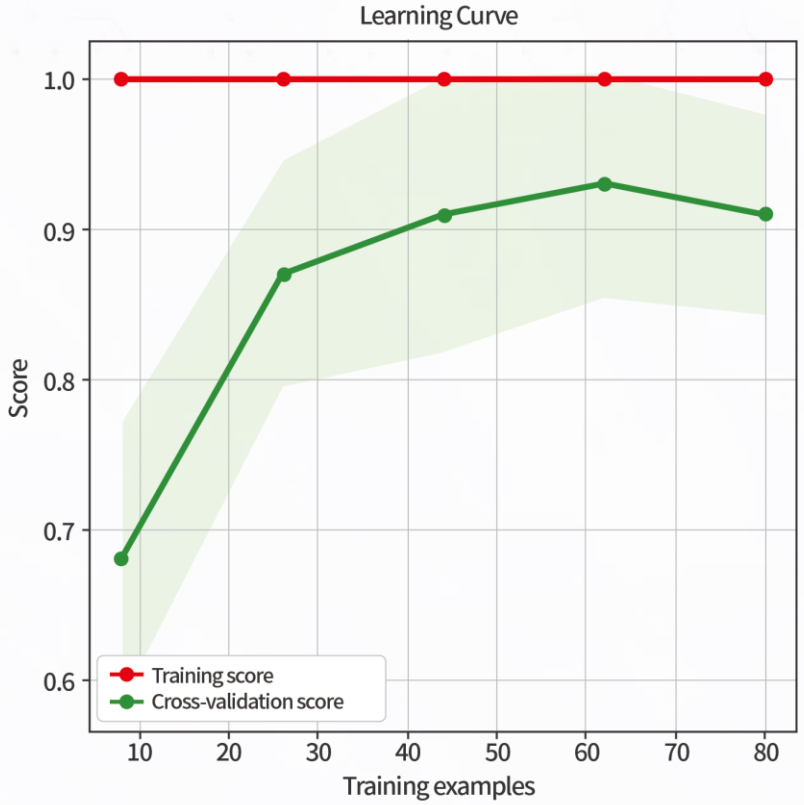
07 | 학습 곡선

학습 곡선(Learning Curve)

- 데이터의 양이 충분히 많을 경우는 훈련 데이터와 검증 데이터를 무작위로 나누더라도 동일한 데이터 분포를 유지함
  - 교차검증 기법은 데이터가 충분하지 않을 때 필요한 기법임
    - 데이터의 양이 충분한지 판단하기 위해서 학습 곡선(Learning Curve)을 그려봄
      - 학습 곡선은 x축을 학습 데이터의 개수
      - y축을 성능 점수
      - 학습 데이터의 양을 조금씩 늘릴 때마다 성능이 어떻게 변화하는지를 보여주는 곡선임
      - 성능 점수는 내부적으로 교차 검증하여 산출함

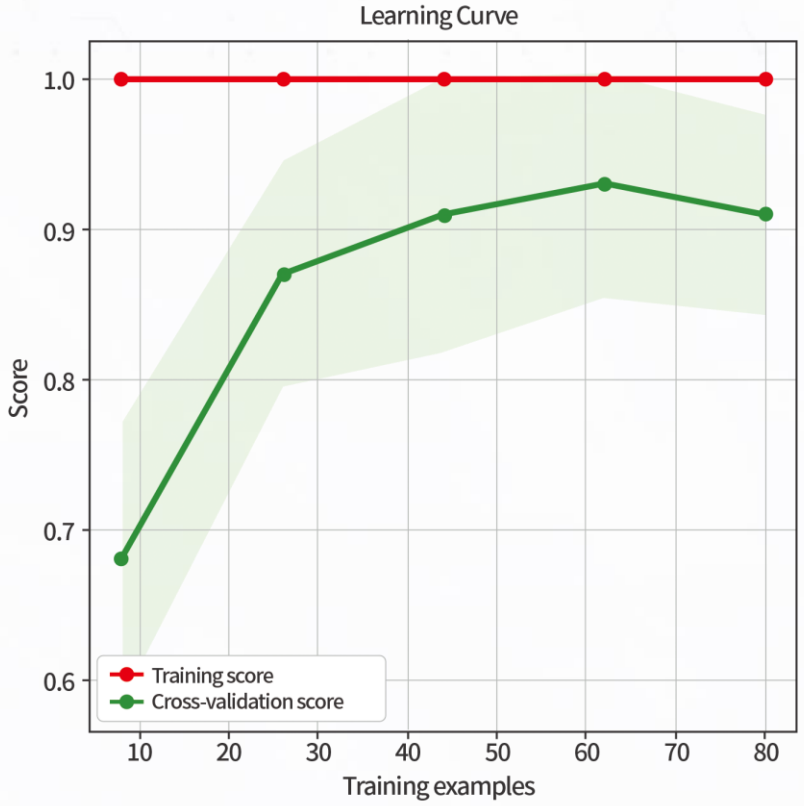
07 | 학습 곡선

- 아래 그림에서 빨간선은 학습에 사용했던 데이터로 검증한 결과이고, 초록선은 교차 검증 결과임
  - 아래 그림에서 초록선이 우상향 하다가 훈련 데이터 샘플 60개를 지나면서 어느 순간 떨어지기 시작하는 것을 볼 수 있는데, 이 지점이 과적합되는 시점임



07 | 학습 곡선

- 아래 그림에서 빨간선은 데이터가 많아지면 일시적으로 떨어질 수도 있음
  - 이는 일시적으로 발생할 수 있는 현상이며 장기적으로 그래프는 수렴함

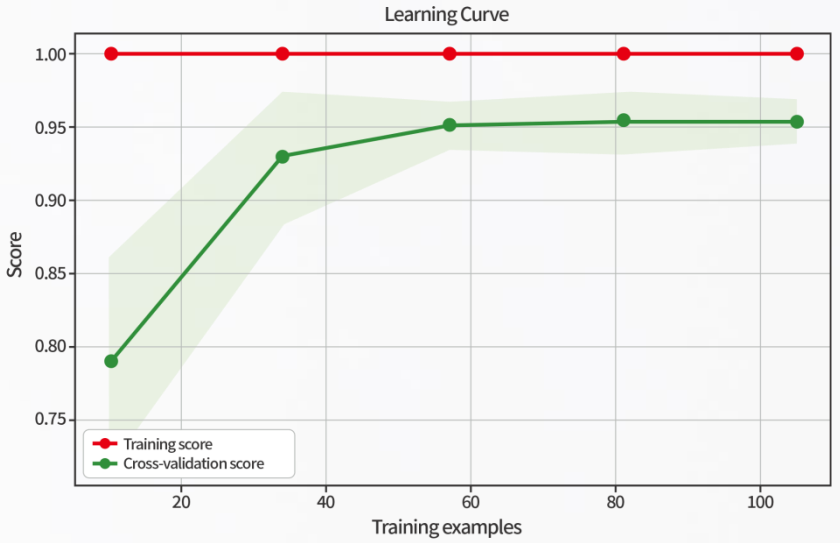


07 | 학습 곡선

다음은 의사결정나무 모델에 아이리스 데이터셋에서 훈련 데이터와 테스트 데이터를 7:3 비율로 분리하여 10겹 교차 검증으로 학습곡선 그래프를 그리는 코드이다.

- 데이터 분리는 StratifiedShuffleSplit()함수로 종속 변수의 각 레이블의 비율을 7:3으로 유지하면서 훈련 데이터와 테스트 데이터를 나눔
  - 학습 곡선을 보면 초록선이 올라가는 모양이므로, 데이터가 더 많이 있을 때 어떻게 될지 알 수 없음

```
cv = StratifiedShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
model = DecisionTreeClassifier(random_state=1234)
plot_learning_curve(model, df_iris.iloc[:, :-1], df_iris.iloc[:, -1], figsize=(10,6), cv=cv)
plt.show()
```



07 | 학습 곡선

- 아래 그림의 학습곡선으로 알 수 있는 것은 현재 의사결정나무 모델은 데이터가 더 있다면 성능이 더 좋아질 가능성이 있다는 것뿐임
  - 아래의 학습곡선은 cv 옵션에서 10겹 교차 검증을 지정하였음
    - 이 옵션을 지정하지 않으면 기본값이 3 Fold가 적용됨
    - 아이리스 데이터셋은 총 150개의 데이터가 들어 있고, 교차 검증으로 30%를 사용함으로 x축의 최대값은 105개 데이터가 있음