분류 모델 평가

Kyungsik Han

본 영상에서 다룰 내용

CODING ✖

- 지도학습 모델 평가
  - 데이터셋
  - 평가지표 및 측정(이진분류)
  - 과적합(overfitting)
  - 오차행렬
  - 분류 평가지표
    - Accuracy, Precision, Recall, F1-score, ROC, AUC
  - 회귀 평가지표
    - R^2

모델 평가

CODING

- 비지도 학습 모델 평가 → 정성적인 작업

- 지도 학습인 회귀와 분류에 집중
  - 데이터 셋 분류: train_test_split()
  - 모델 실행: fit()
  - 모델 평가: score()

모델 평가 예제

CODING

기본 모델 구축 및 성능 평가

In [1]:
```
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [3]:
```
# 인위적인 데이터 셋 생성
X, y = make_blobs(random_state=0)
```

In [4]:
```
# 데이터와 타겟 레이블을 훈련 세트와 테스트 세트로 나눔
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [5]:
```
# Logistic Regression 객체 생성
logreg = LogisticRegression().fit(X_train, y_train)
```

In [6]:
```
# 모델을 테스트 세트로 평가
print("테스트 세트 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

```
테스트 세트 점수: 0.88
```

데이터셋

CODING

- 데이터를 훈련 셋과 테스트 셋으로 나누는 이유
  - 지금까지 본적이 없는 새로운 데이터에 모델이 얼마나 잘 일반화 되는지 측정하기 위함
  - 모델은 학습 과정에 없던 데이터 (테스트 셋)에 대해 예측을 얼마나 잘 하느냐가 중요함

평가 지표와 측정 (이진 분류)

이진 분류의 평가 지표

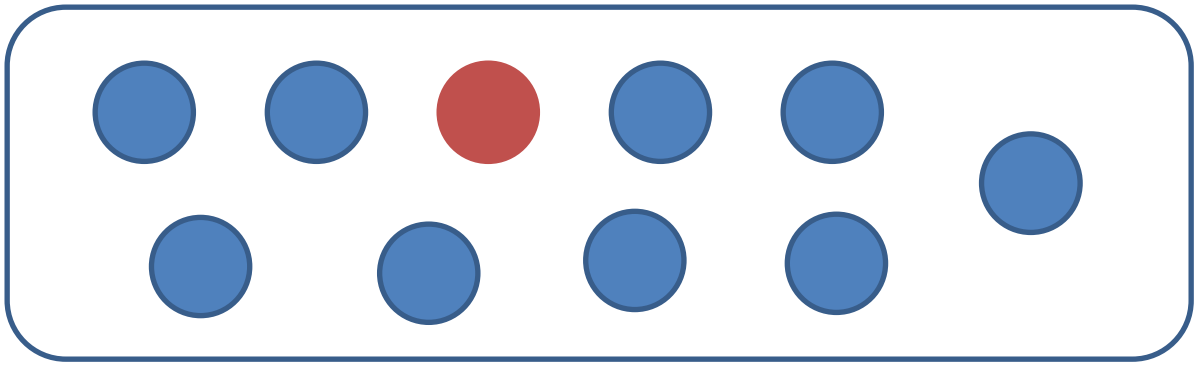
CODING

- 이진 분류 (Binary)
  - 양성 or 음성 클래스

- 암 조기 발견 애플리케이션
  - 양성: 추가 검사가 필요 → 양성 클래스
    - 잘못된 양성 클래스
      - 건강한 사람을 양성으로 분류
      - 거짓 양성 (False Positive) → Type 1 Error
  - 음성: 건강함 → 음성 클래스
    - 잘못된 음성 클래스
      - 암에 걸린 사람을 음성으로 분류
      - 거짓 음성 (False Negative) → Type 2 Error

불균형 데이터셋

- Binary 분류에서 두 class의 데이터셋 크기는 다른 경우가 흔함
  - 불균형 데이터 셋 (imbalanced datasets)



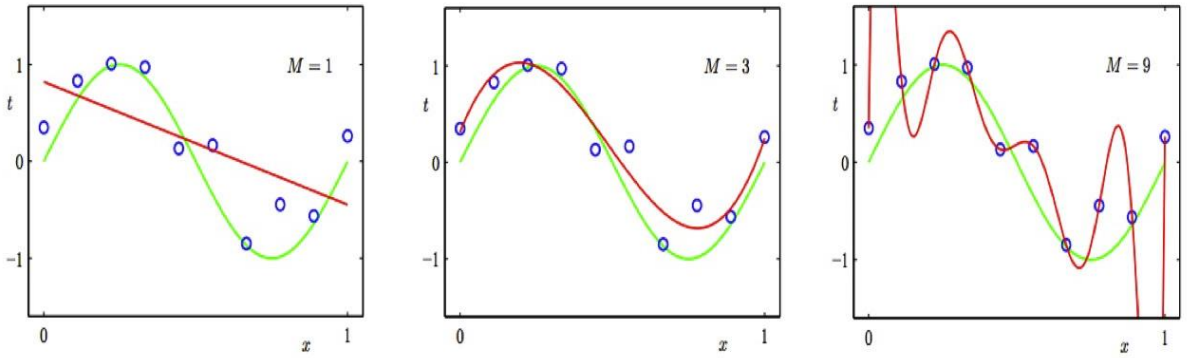
- Cross Validation을 통해서 정확한 결과를 가져오는 것이 매우 중요

Overfitting (과적합)

CODING

과적합을 피해서 모델을 구축하는 것이 매우 중요하다

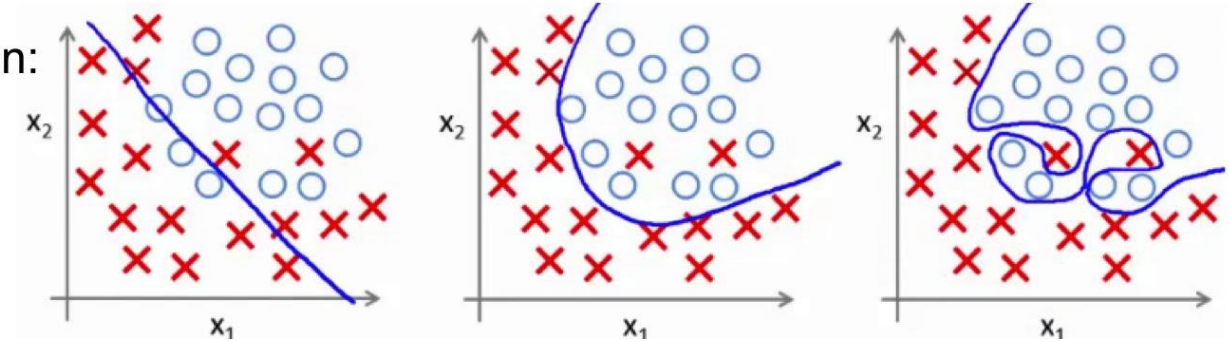
Regression:



predictor too inflexible:
cannot capture pattern

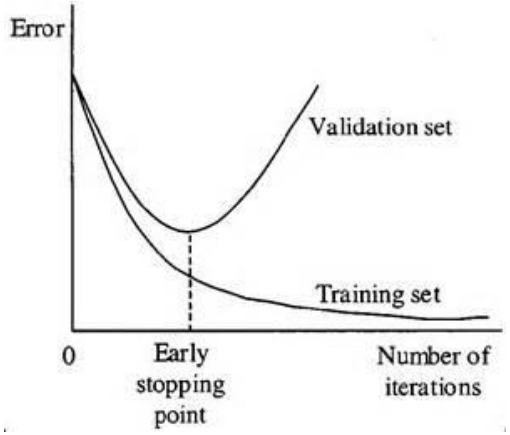
predictor too flexible:
fits noise in the data

Classification:

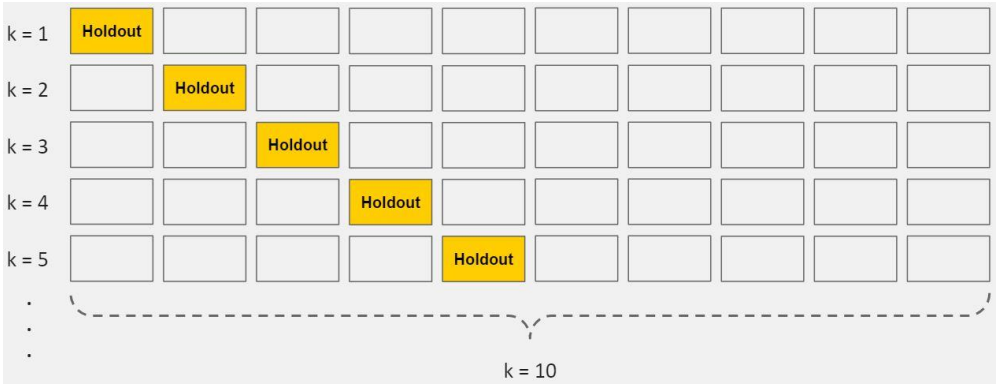


과적합 피하기

- Cross-validation
- Train with more data
- Remove features (차원의 저주)
- Early stopping
- Regularization
- Ensembling (앙상블)



Early stopping



Cross-validation

불균형 데이터셋

CODING

불균형 데이터셋, 오차 행렬

In [28]:
```
from sklearn.datasets import load_digits
import numpy as np

digits = load_digits()
y = digits.target == 9

X_train, X_test, y_train, y_test = train_test_split(
    digits.data, y, random_state=0)
```

DummyClassifier

In [31]:
```
from sklearn.dummy import DummyClassifier
dummy_majority = DummyClassifier().fit(X_train, y_train)
pred_most_frequent = dummy_majority.predict(X_test)
print("예측된 레이블의 고유값: {}".format(np.unique(pred_most_frequent)))
print("테스트 점수: {:.2f}".format(dummy_majority.score(X_test, y_test)))
```

```
예측된 레이블의 고유값: [False  True]
테스트 점수: 0.80
```

DecisionTree Classifier

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=2).fit(X_train, y_train)
pred_tree = tree.predict(X_test)
print("테스트 점수: {:.2f}".format(tree.score(X_test, y_test)))
```

```
테스트 점수: 0.92
```

불균형 데이터셋

CODING

DummyClassifier and LogisticRegression

In [33]:
```
from sklearn.linear_model import LogisticRegression

dummy = DummyClassifier().fit(X_train, y_train)
pred_dummy = dummy.predict(X_test)
print("dummy 점수: {:.2f}".format(dummy.score(X_test, y_test)))

logreg = LogisticRegression(C=0.1).fit(X_train, y_train)
pred_logreg = logreg.predict(X_test)
print("logreg 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

```
dummy 점수: 0.80
logreg 점수: 0.98
```

불균형 데이터셋에서는 정확도를 통해서
모델의 성능을 평가하는 것은 옳지않다

오차 행렬

CODING ✖

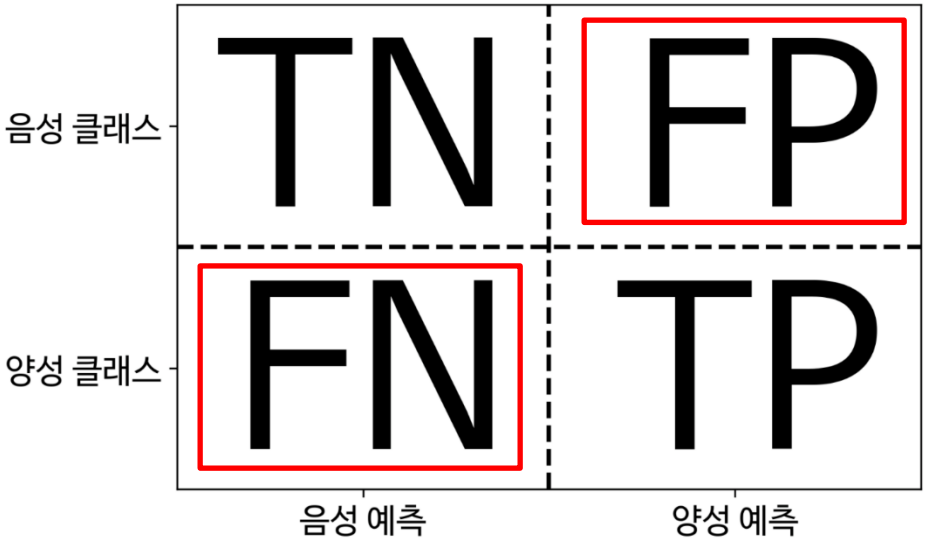
- Confusion Matrix
- 이진 분류 평가 결과를 나타낼 때 가장 널리 사용하는 방법 중 하나

오차행렬

In [34]:
```
from sklearn.metrics import confusion_matrix

confusion = confusion_matrix(y_test, pred_logreg)
print("오차 행렬:\n{}".format(confusion))
```

```
오차 행렬:
[[401   2]
 [  8  39]]
```

오차 행렬



Confusion Matrix (오차 행렬)

CODING ✖

In [35]:
```
print("빈도 기반 더미 모델:")
print(confusion_matrix(y_test, pred_most_frequent))
print("\n무작위 더미 모델:")
print(confusion_matrix(y_test, pred_dummy))
print("\n결정 트리:")
print(confusion_matrix(y_test, pred_tree))
print("\n로지스틱 회귀")
print(confusion_matrix(y_test, pred_logreg))
```

```
빈도 기반 더미 모델:
[[357  46]
 [ 41   6]]

무작위 더미 모델:
[[364  39]
 [ 43   4]]

결정 트리:
[[390  13]
 [ 24  23]]

로지스틱 회귀
[[401   2]
 [  8  39]]
```

평가 지표

CODING ✖

- 정확도 (accuracy)
  - 정확히 예측한 수를 전체 샘플 수로 나눈 값

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- 정밀도 (precision)
  - 양성 예측 중 (TP+FP), 진짜 양성(TP) 의 갯수

$$\text{정밀도} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

평가 지표

CODING ✖

- 재현율 (recall)
  - 전체 양성 샘플 (TP + FN) 중, 진짜 양성(TP)의 갯수
  - 민감도 (sensitivity), 적중률 (hit rate)

$$\text{재현율} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F-점수 (F-score, F1-score)
  - 정밀도와 재연율의 조화 평균

$$\text{F} = 2 \cdot \frac{\text{정밀도} \cdot \text{재현율}}{\text{정밀도} + \text{재현율}}$$

F-score

CODING

정확도, 정밀도, 재현율, F-점수

In [40]:
```
from sklearn.metrics import f1_score

print("무작위 더미 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_dummy)))
print("트리 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_tree)))
print("로지스틱 회귀 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_logreg)))
```

```
무작위 더미 모델의 f1 score: 0.09
트리 모델의 f1 score: 0.55
로지스틱 회귀 모델의 f1 score: 0.89
```

In [41]:
```
print(classification_report(y_test, pred_dummy, target_names=["9 아님", "9"]))
```

```
             precision    recall  f1-score   support

      9 아님       0.89      0.90      0.90       403
          9       0.09      0.09      0.09        47

avg / total       0.81      0.82      0.81       450
```

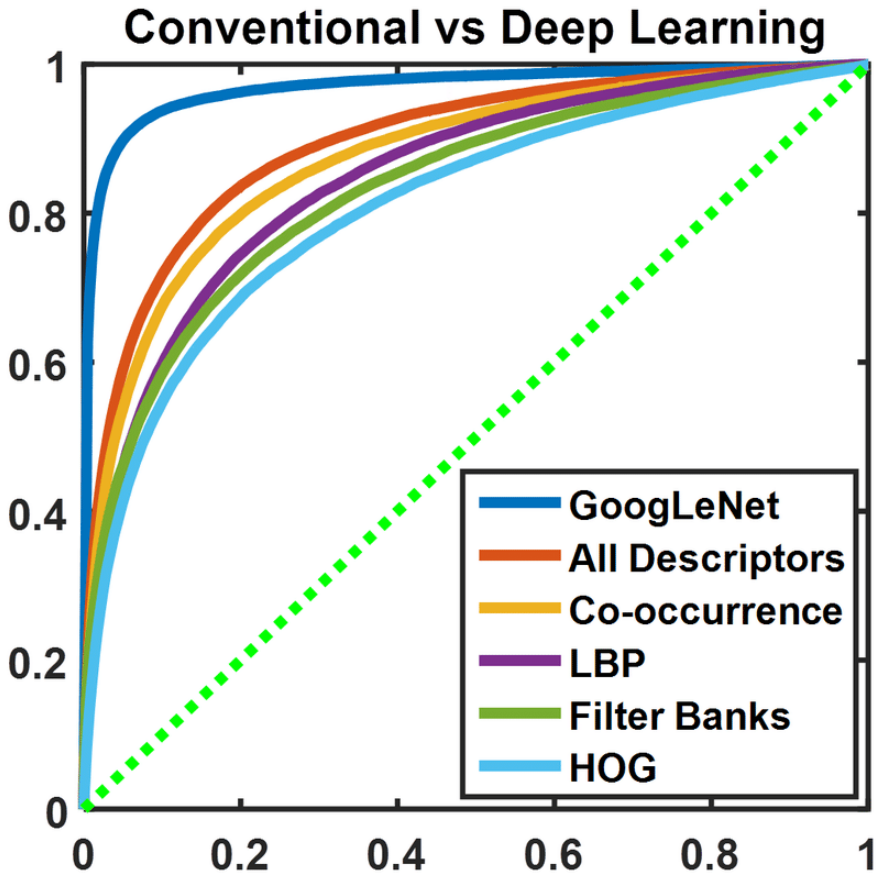
In [42]:
```
print(classification_report(y_test, pred_logreg, target_names=["9 아님", "9"]))
```

```
             precision    recall  f1-score   support

      9 아님       0.98      1.00      0.99       403
          9       0.95      0.83      0.89        47

avg / total       0.98      0.98      0.98       450
```

ROC and AUC

CODING

- ROC (Receiver Operating Characteristics)
  - TPR(True Positive Rate)-FPR(False Positive Rate) 곡선



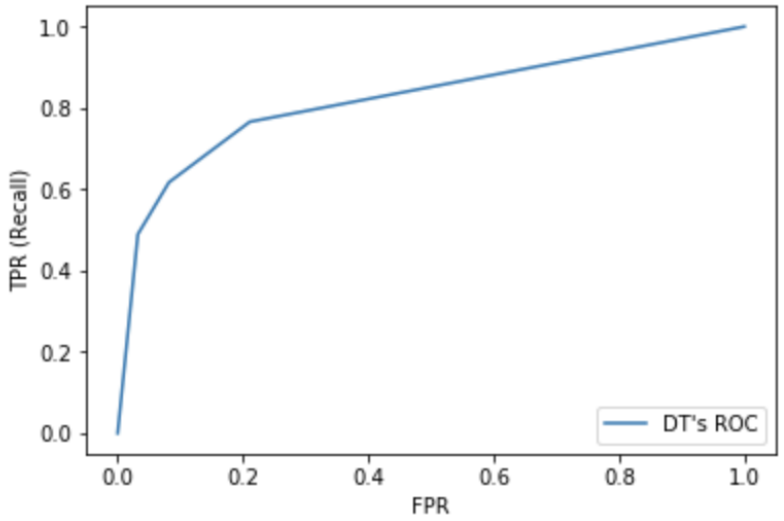
ROC and AUC

CODING

ROC

In [64]:
```
from sklearn.metrics import roc_curve
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, tree.predict_proba(X_test)[:, 1])

plt.plot(fpr_rf, tpr_rf, label="DT's ROC")

plt.xlabel("FPR")
plt.ylabel("TPR (Recall)")

plt.legend(loc=4)
```

Out[64]:
```
<matplotlib.legend.Legend at 0x1a1736f780>
```



ROC and AUC

CODING

- AUC (Area Under the Curve)
  - Curve: ROC 곡선
  - 즉, ROC 곡선 아래의 면적

AUC

In [65]:
```
from sklearn.metrics import roc_auc_score
rf_auc = roc_auc_score(y_test, tree.predict_proba(X_test)[:, 1])
print("DT's AUC: {:.3f}".format(rf_auc))
```

```
DT's AUC: 0.821
```

평가 지표와 측정 (다중 분류)

다중 분류 평가 지표

CODING

- 다중 분류 평가에서는 정확도 이외에 오차 행렬과 분류 리포트(F1 Score)를 주로 사용

다중 분류: 오차 행렬

CODING

Confusion Matrix

다중 분류 평가 지표

In [67]:
```
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, random_state=0)
lr = LogisticRegression().fit(X_train, y_train)
pred = lr.predict(X_test)
print("정확도: {:.3f}".format(accuracy_score(y_test, pred)))
print("오차 행렬:\n{}".format(confusion_matrix(y_test, pred)))
```

```
정확도: 0.953
오차 행렬:
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 39  0  0  0  0  2  0  2  0]
 [ 0  0 41  3  0  0  0  0  0  0]
 [ 0  0  1 43  0  0  0  0  0  1]
 [ 0  0  0  0 38  0  0  0  0  0]
 [ 0  1  0  0  0 47  0  0  0  0]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  1  0  1  1  0  0 45  0  0]
 [ 0  3  1  0  0  0  0  0 43  1]
 [ 0  0  0  1  0  1  0  0  1 44]]
```

다중 분류

CODING ✖

Classification Report

```
In [68]: print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.89	0.91	0.90	43
2	0.95	0.93	0.94	44
3	0.90	0.96	0.92	45
4	0.97	1.00	0.99	38
5	0.98	0.98	0.98	48
6	0.96	1.00	0.98	52
7	1.00	0.94	0.97	48
8	0.93	0.90	0.91	48
9	0.96	0.94	0.95	47
avg / total	0.95	0.95	0.95	450

다중 분류: F1-score

CODING

- sklearn에서 지원
  - average 파라미터로 접근
  - average="micro"
    - 모든 클래스의 FP, FN, TP의 수를 고려해서 계산
  - average="macro"
    - 클래스 크기에 상관없이 모든 클래스 같은 비중으로 다룸
  - average="weighted"
    - 클래스별 샘플 수로 가중치 두어 계산

```
In [70]: print("micro 평균 f1 점수: {:.3f}".format(f1_score(y_test, pred, average="micro")))
         print("macro 평균 f1 점수: {:.3f}".format(f1_score(y_test, pred, average="macro")))
         print("weighted 평균 f1 점수: {:.3f}".format(f1_score(y_test, pred, average="weighted")))

         micro 평균 f1 점수: 0.953
         macro 평균 f1 점수: 0.954
         weighted 평균 f1 점수: 0.953
```

평가 지표와 측정 (회귀)

회귀 평가: R^2

CODING

회귀 분석

In [74]:
```
from sklearn import linear_model
from sklearn import datasets ## imports datasets from scikit-learn
```

In [85]:
```
# loads Boston dataset from datasets library
data = datasets.load_boston()

# define the data/predictors as the pre-set feature names
df = pd.DataFrame(data.data, columns=data.feature_names)

# Put the target (housing value -- MEDV) in another DataFrame
target = pd.DataFrame(data.target, columns=["MEDV"])

# Define X and y
X = df
y = target["MEDV"]
```

In [88]:
```
lm = linear_model.LinearRegression()
model = lm.fit(X,y)

predictions = lm.predict(X)
# prediction results from 0 to 4
print(predictions[0:5])
print('')

# performance: score function
print(lm.score(X,y))
print('')

# print coefficient
print(lm.coef_)
```

Summary

CODING ✖

- 분류: 적절한 평가 지표 선택
  - Imbalanced classes 의 경우는 accuracy는 정확한 측정법이 아님
  - F1 score가 주로 사용
- 회귀: R^2 score 를 사용

다음 영상에서 배울 내용

CODING

- 분류 실습

수고하셨습니다