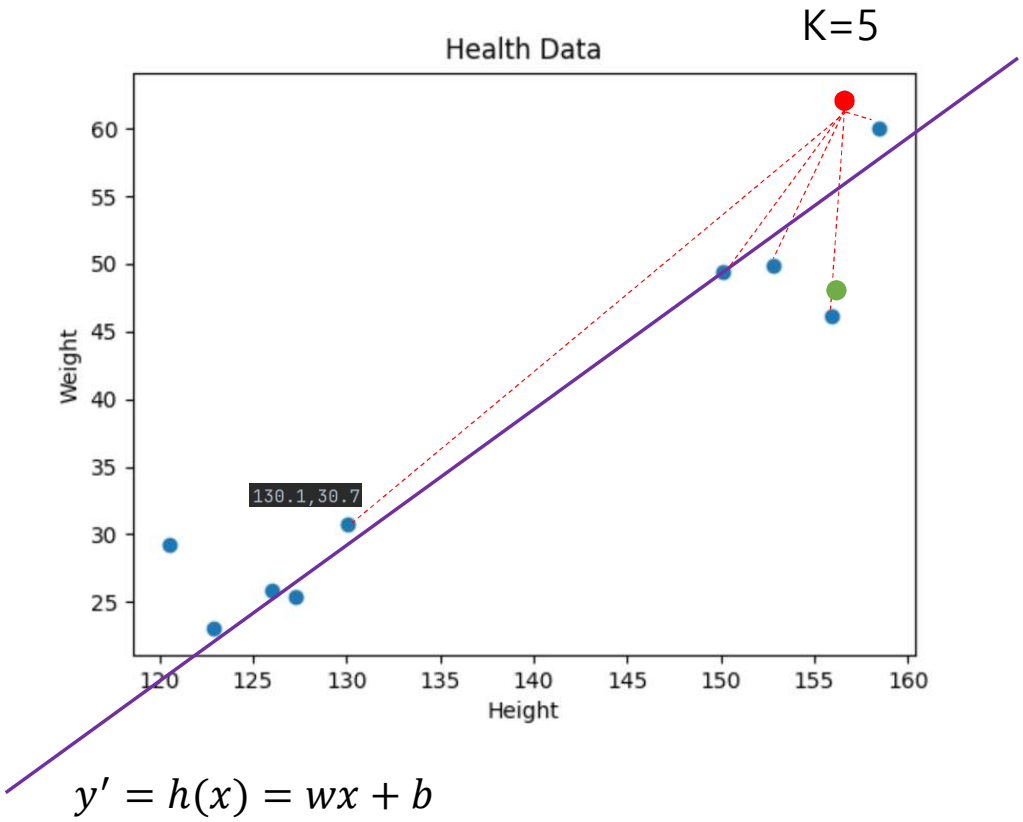
하이퍼파라미터

하이퍼파라미터(Hyperparameter)

- 사용자가 직접 결정할 수 있는 파라미터
- 일반적으로, 많은 실험에 의한 성능 분석을 통해 최적의 하이퍼파라미터 결정 가능
- 모델 학습 시 모델 내부에서 자동으로 결정되는 (모델)파라미터와는 다름
- (모델)파라미터는 학습 데이터에 의해 자동으로, 하이퍼파라미터는 사용자에 의해 수동으로 결정



$$y' = h(x) = wx + b$$

하이퍼파라미터 튜닝

- 모델의 성능을 향상시키기 위해, 하이퍼파라미터 값을 조절하는 것
- 학습/검증 데이터 점수나 교차검증을 통해 하이퍼파라미터 튜닝

d	1	3	5	7	10

의사결정나무 (max_depth)

p \ K	1	3	5	7	10
1	(1, 1)	(1, 3)	(1, 5)	(1, 7)	(1, 10)
2	(2, 1)	(2, 3)	(2, 5)	(2, 7)	(2, 10)

KNN (n_neighbors, p)

의사결정나무 분류

d	1	3	5	7	10

```
import seaborn as sns
titanic = sns.load_dataset("titanic")
data = titanic[["sex", "age", "sibsp", "adult_male", "parch"]].copy()
t = titanic["survived"]
data["age"].fillna(30, inplace=True)
data["sex"].replace("male", 1, inplace=True)
data["sex"].replace("female", 0, inplace=True)
```

```
from sklearn.model_selection import train_test_split
data2, test_data, t2, test_target = train_test_split(
    data, t, test_size=0.3, random_state=42, stratify=t)
train_data, val_data, train_target, val_target = train_test_split(
    data2, t2, test_size=0.2, random_state=42, stratify=t2)

from sklearn.tree import DecisionTreeClassifier
best_score = best_d = 0
score = []
for d in [1, 3, 5, 7, 10]:
```

구현
(random_state=42)

```
    if best_score < val_score:
        best_score = val_score
        best_d = d
```

```
[0.792, 0.824, 0.832, 0.8, 0.792]
```

max_depth

```
print(best_score, best_d)
```

```
0.832 5
```

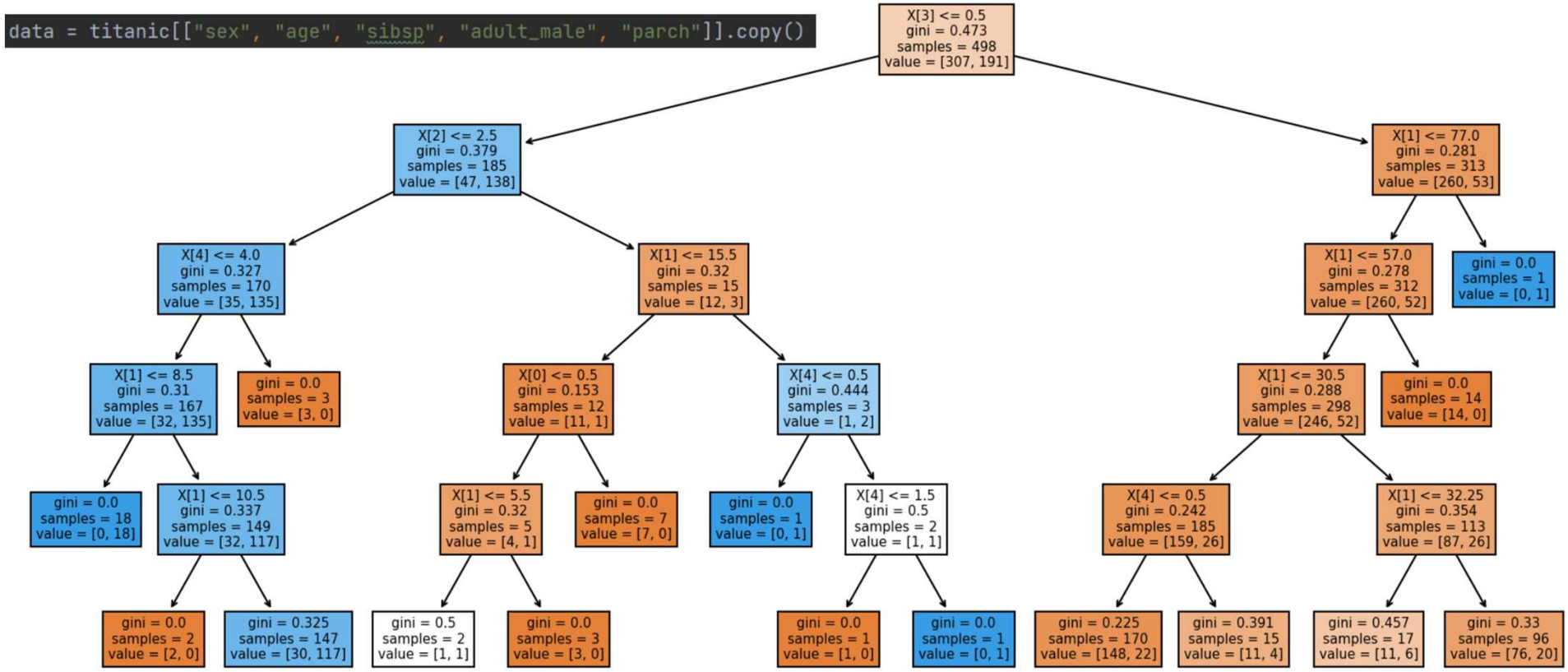
구현(random_state=42)

```
print("Train-Eval:", dt.score(train_data, train_target))
print("Val-Eval  :", dt.score(val_data, val_target))
print("Test-Eval :", dt.score(test_data, test_target))
```

```
Train-Eval: 0.8333333333333334
Val-Eval  : 0.832
Test-Eval : 0.8022388059701493
```

의사결정나무 결과

```
data = titanic[["sex", "age", "sibsp", "adult_male", "parch"]].copy()
```



KNN 분류

p \ K	1	3	5	7	10
1	(1, 1)	(1, 3)	(1, 5)	(1, 7)	(1, 10)
2	(2, 1)	(2, 3)	(2, 5)	(2, 7)	(2, 10)

```
import seaborn as sns
titanic = sns.load_dataset("titanic")
data = titanic[["sex", "age", "sibsp", "adult_male", "parch"]].copy()
t = titanic["survived"]
data["age"].fillna(30, inplace=True)
data["sex"].replace("male", 1, inplace=True)
data["sex"].replace("female", 0, inplace=True)
```

```
from sklearn.model_selection import train_test_split
data2, test_data, t2, test_target = train_test_split(
    data, t, test_size=0.3, random_state=42, stratify=t)
train_data, val_data, train_target, val_target = train_test_split(
    data2, t2, test_size=0.2, random_state=42, stratify=t2)
```

```
from sklearn.neighbors import KNeighborsClassifier
best_score = best_n = best_m = 0
```

구현
(best_n=K, best_m=p)

```
    if best_score < val_score:
        best_score = val_score
        best_n = n
        best_m = m
```

n_neighbors

p

```
[0.784, 0.792, 0.784, 0.8, 0.768]
[0.776, 0.784, 0.76, 0.784, 0.736]
```

검증점수

```
print(best_score, best_n, best_m)
```

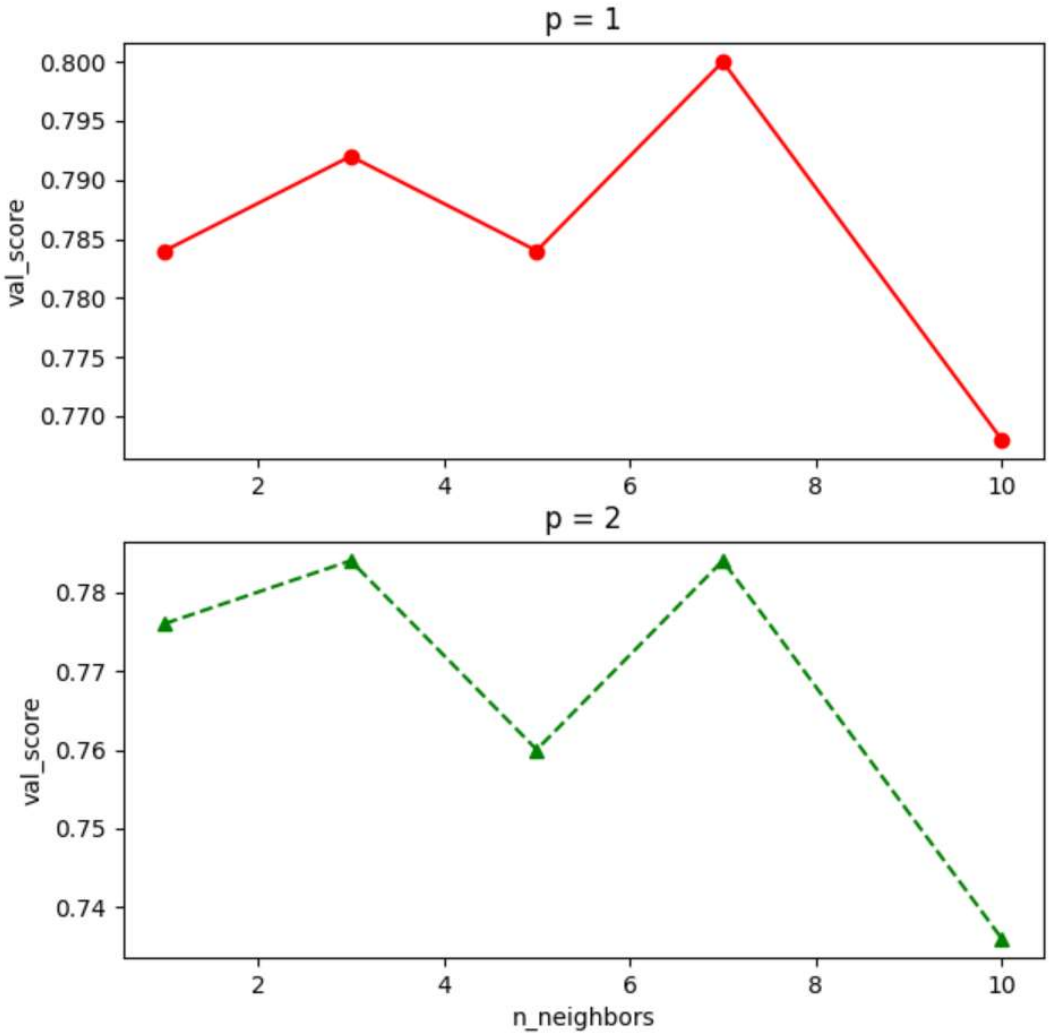
```
0.792 7 1
```

분류 결과

```
import matplotlib.pyplot as plt

구현

plt.show()
```



테스트 결과

n_neighbors	p	train_score	val_score	test_score
1	1	**0.889**	0.784	0.768
3	1	0.847	0.792	0.757
5	1	0.843	0.784	0.764
7	1	0.823	**0.800**	**0.776**
10	1	0.819	0.768	0.742
1	2	**0.889**	0.776	0.761
3	2	0.847	0.784	0.753
5	2	0.831	0.760	0.761
7	2	0.825	0.784	0.764
10	2	0.801	0.736	0.727

K-폴드(K-Fold) 교차검증

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    data, t, test_size = 0.3, random_state = 42, stratify = t)

from sklearn.model_selection import cross_validate
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
best_score = best_n = best_m = 0
```

구현
(cv=10)

10-폴드
교차검증

n_neighbors

p

```
[0.6965949820788532, 0.7480286738351254, 0.7831285202252944, 0.8121607782898106, 0.778469022017409]
[0.6901689708141321, 0.74152585765489, 0.7734254992319509, 0.7975934459805426, 0.7672555043522785]
```

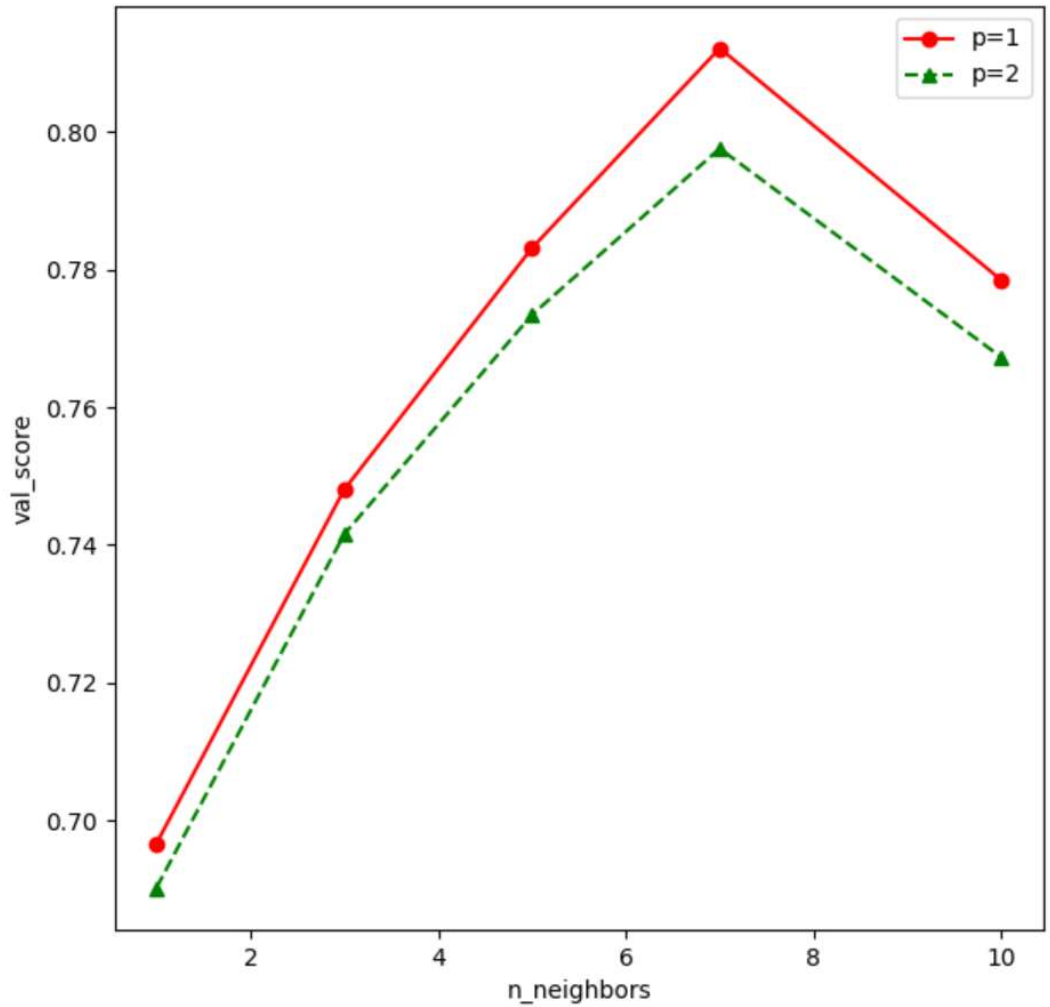
평균검증값

교차검증 결과

```
import matplotlib.pyplot as plt

구현

plt.show()
```



참고자료

- 지능기전공학부 최유경 교수님 자료, https://github.com/sejongresearch/2021.MachineLearning
- 코랩(Colab), https://colab.research.google.com/
- 파이썬(Python), https://www.python.org/doc/
- 사이킷런(sckit-learn), https://scikit-learn.org/stable/index.html
- 판다스(pandas), https://pandas.pydata.org/
- 맷플롯립(matplotlib), https://matplotlib.org/
- 씨본(seaborn), https://seaborn.pydata.org/
- 캐글(Kaggle), https://www.kaggle.com/
- 넘파이(numpy), https://numpy.org/doc/stable/
- 스택오퍼플러우(stackoverflow), https://stackoverflow.com/