

13

## 배깅(회귀)

### [배깅 회귀분석 실습]

# 의사결정나무에서 다루었던 분석 결과와 어떻게 다른지 살펴보겠습니다.

00:30

The screenshot shows a Jupyter Notebook titled "13-3\_배깅(회귀)" with the following code:

```
[예제] kc_house_data.csv를 train과 test 데이터로 분할하고 train 데이터를 활용하여 BaggingRegressor 모델을 만들어라
```

```
In [1]: import pandas as pd
df = pd.read_csv('../data/kc_house_data.csv')
df = df.drop(['id', 'date'], axis=1)

X = df.drop('price', axis=1)
y = df['price']

X = pd.get_dummies(data = X, columns=['waterfront'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2021)
```

```
In [ ]: from sklearn.ensemble import BaggingRegressor
model_bag_reg = BaggingRegressor()
model_bag_reg.fit(X_train, y_train)
print(model_bag_reg.score(X_train, y_train))
print(model_bag_reg.score(X_test, y_test))
```

Out of Bag 샘플을 활용한 성능 측정

```
In [3]: model_bag_reg_oob = BaggingRegressor(n_estimators=100,
oob_score=True)
model_bag_reg_oob.fit(X, y)
print(f'oob score : {model_bag_reg_oob.oob_score_}')

oob score : 0.7519571663576616
oob score : 0.7519571663576616
```

### [배깅 회귀분석 실습 - oob사용]

# 학습 과정에서 oob를 사용할 때에는 분류된 데이터가 아닌, X,y값을 학습시켜주어야 합니다.

01:34

The screenshot shows a Jupyter Notebook titled "13-3\_배깅(회귀)" with the following code:

```
X = pd.get_dummies(data = X, columns=['waterfront'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2021)
```

```
In [2]: from sklearn.ensemble import BaggingRegressor
model_bag_reg = BaggingRegressor()
model_bag_reg.fit(X_train, y_train)
print(model_bag_reg.score(X_train, y_train))
print(model_bag_reg.score(X_test, y_test))

0.9482591476153137
0.7534236745465055
```

Out of Bag 샘플을 활용한 성능 측정

```
In [3]: model_bag_reg_oob = BaggingRegressor(n_estimators=100,
oob_score=True)
model_bag_reg_oob.fit(X, y)
print(f'oob score : {model_bag_reg_oob.oob_score_}')

oob score : 0.7519571663576616
oob score : 0.7519571663576616
```

```
In [4]: model_bag_reg = BaggingRegressor(n_estimators=100)
model_bag_reg.fit(X_train, y_train)
print(model_bag_reg.score(X_train, y_train))
print(model_bag_reg.score(X_test, y_test))

0.9642299466313344
0.9642299466313344
```

## [회귀분석 결과 해석]

# 트리모델을 사용한 회귀분석에서는 회귀 계수 값 자체를 구할 수는 없지만, 어떠한 변수가 중요한 변수인지는 찾을 수 있습니다.

03:02

The image shows a Jupyter Notebook interface with the title '13-3\_배깅(회귀)' and '(unsaved changes)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running code, and viewing output. The notebook contains two code cells. The first cell (In [7]) imports sklearn and retrieves feature importances from a bagged regression model. The output (Out [7]) is a 1D array of 10 values. The second cell (In [6]) imports numpy and pandas, concatenates the feature names with their importances into a DataFrame, and sorts it by importance. The output (Out [6]) is a DataFrame with two columns: 'col\_name' and 'feature\_importance', showing the top 6 features by importance.

```

0.36529476422922
0.758110700689407

In [7]: from sklearn import tree
        model_baq_reg.estimators_[0].feature_importances_

Out[7]: array([0.00951675, 0.02457485, 0.28208744, 0.02851607, 0.00894546,
        0.04282814, 0.00671157, 0.35581146, 0.02454131, 0.01256344,
        0.10275019, 0.0042037 , 0.05311825, 0.03579746, 0.00739053,
        0.00063927])

In [6]: ## 변수의 중요도 확인

import numpy as np
importances = pd.DataFrame(np.mean([tree.feature_importances_ for tree in model_baq_reg.estimators_], axis=0))
feature_importances = pd.concat([pd.DataFrame(X.columns), importances], axis=1)
feature_importances.columns=["col_name", "feature_importance"]
feature_importances

# 변수의 상대적 중요도를 보았을 때, Credit Amount, Account Balance, Duration of Credit [month] 순으로
# 변수 중요도가 큰 것을 파악할 수 있음

Out[6]:
   col_name  feature_importance
0  bedrooms      0.008436
1  bathrooms      0.021851
2   sqft_living  0.288876
3   sqft_lot      0.034866
4    floors      0.005513
5     view      0.022751
  
```