변수 선택법: 후진 제거법, 전진 선택법

- 머신러닝 모델의 성능을 향상시키고, 과적합을 방지하며, 모델의 해석력을 높이기 위해 데이터셋에서 가장 관련성이 높거나 유용한 피처(변수)의 부분집합을 선택하는 과정
- 불필요하거나 중복되는 피처를 제거함으로써 모델의 복잡도를 줄이고, 학습 시간을 단축하며, 일반화 성능을 개선하는 것이 목표

종류

- 변수 선택법은 크게 **필터(Filter) 방식, 래퍼(Wrapper) 방식, 임베디드(Embedded) 방식**으로 나눌 수 있습니다.
- **후진 제거법과 전진 선택법**은 래퍼 방식에 해당합니다.

적용 가능한 상황

- **고차원 데이터**: 피처의 수가 샘플 수보다 많거나 매우 많은 경우, 차원의 저주(Curse of Dimensionality) 문제를 해결하기 위해 사용됩니다.
- **노이즈가 많은 데이터**: 모델 성능에 부정적인 영향을 미치는 불필요한 피처를 제거하여 모델의 강건성(robustness)을 높일 때.
- **모델 해석력 향상**: 모델의 예측에 기여하는 핵심 피처만을 남겨 모델의 의사결정 과정을 더 쉽게 이해하고 설명할 수 있도록 할 때.
- **과적합 방지**: 불필요한 피처가 모델에 과도하게 학습되는 것을 방지하여 새로운 데이터에 대한 예측 성능을 높일 때.
- **학습 시간 단축 및 계산 비용 절감**: 피처의 수를 줄여 모델 학습 및 예측에 필요한 시간과 자원을 절약할 때.

1. 후진 제거법 (Backward Elimination)

- 모든 피처를 포함한 모델에서 시작하여, 통계적으로 가장 유의미하지 않은 피처를 하나씩 제거해나가면서 모델의 성능이 가장 좋아지는 피처 조합을 찾는 방법
    - 유의미하지 않은 피처 예시: p-value가 가장 높은
- 각 단계에서 피처를 제거할 때마다 모델을 재학습하고 성능을 평가합니다.

주의사항

- **계산 비용**: 피처의 수가 많을수록 각 단계에서 모델을 재학습해야 하므로 계산 비용이 많이 들 수 있습니다.
- **지역 최적해**: 한 번 제거된 피처는 다시 추가되지 않으므로, 전역 최적해(Global Optimum)가 아닌 지역 최적해(Local Optimum)에 빠질 위험이 있습니다.
- **상호작용 효과 무시**: 제거된 피처가 다른 피처와 중요한 상호작용 효과를 가질 수 있음에도 불구하고 이를 고려하지 못할 수 있습니다.

`statsmodels`를 이용한 OLS 모델

- `significance_level`: 피처를 제거할지 결정하는 유의수준
    - 일반적으로 `0.05`를 사용하며, 이 값보다 p-value가 높으면 해당 피처는 통계적으로 유의미하지 않다고 판단하여 제거합니다.

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# 1. 데이터 준비
X, y = make_regression(n_samples=100, n_features=10, n_informative=5,
random_state=42)
feature_names = [f'X{i}' for i in range(X.shape[1])]
X_df = pd.DataFrame(X, columns=feature_names)
X_df = sm.add_constant(X_df) # 상수항 추가

X_train, X_test, y_train, y_test = train_test_split(X_df, y, test_size=0.2,
random_state=42)

def backward_elimination(X_data, y_data, significance_level=0.05):
    features = list(X_data.columns)
    while len(features) > 1: # 상수항(const)은 남겨둠
        model = sm.OLS(y_data, X_data[features]).fit()
        p_values = model.pvalues
        max_p_value = p_values.drop('const', errors='ignore').max() # 상수항 제외

        if max_p_value > significance_level:
            redundant_feature = p_values.drop('const', errors='ignore').idxmax()
            features.remove(redundant_feature)
            print(f"제거된 피처: {redundant_feature}, p-value: {max_p_value:.4f}")
        else:
            break
    return features

# 2. 후진 제거법 적용
print("--- 후진 제거법 시작 ---")
selected_features_be = backward_elimination(X_train, y_train)
print(f"최종 선택된 피처: {selected_features_be}")
'''
제거된 피처: X5, p-value: 0.1832
제거된 피처: X6, p-value: 0.7410
제거된 피처: X7, p-value: 0.2208
최종 선택된 피처: ['const', 'X0', 'X1', 'X2', 'X3', 'X4', 'X8', 'X9']
'''

# 3. 최종 모델 학습 및 평가
final_model_be = sm.OLS(y_train, X_train[selected_features_be]).fit()
print("\n--- 후진 제거법 최종 모델 요약 ---")
print(final_model_be.summary())
'''
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
```

```
Method:                 Least Squares   F-statistic:                1.011e+31
Date:                Mon, 13 Oct 2025   Prob (F-statistic):              0.00
Time:                        23:07:59   Log-Likelihood:                 2283.6
No. Observations:                  80   AIC:                            -4551.
Df Residuals:                      72   BIC:                            -4532.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -8.882e-15   1.16e-14     -0.767      0.446    -3.2e-14    1.42e-14
X0          7.194e-14   1.14e-14      6.317      0.000    4.92e-14    9.46e-14
X1            63.6430   1.23e-14   5.18e+15      0.000      63.643      63.643
X2            70.6476   1.21e-14   5.83e+15      0.000      70.648      70.648
X3            10.4568   1.12e-14   9.36e+14      0.000      10.457      10.457
X4            16.7483   1.25e-14   1.34e+15      0.000      16.748      16.748
X8             3.1586   1.16e-14   2.71e+14      0.000       3.159       3.159
X9          3.997e-14   1.29e-14      3.106      0.003    1.43e-14    6.56e-14
==============================================================================
Omnibus:                        1.007   Durbin-Watson:                   1.736
Prob(Omnibus):                  0.604   Jarque-Bera (JB):                1.051
Skew:                          -0.254   Prob(JB):                        0.591
Kurtosis:                       2.759   Cond. No.                         1.65
==============================================================================
'''

y_pred_be = final_model_be.predict(X_test[selected_features_be])
mse_be = mean_squared_error(y_test, y_pred_be)
print(f"테스트 세트 MSE (후진 제거법): {mse_be:.4f}") # 0.0000
```

결과 해석 방법

- `final_model_be.summary()`: 최종 선택된 피처들로 학습된 모델의 통계적 요약을 제공합니다. 각 피처의 계수, p-value, R-squared 값 등을 확인할 수 있습니다.
- `mean_squared_error`: 최종 모델의 예측 성능을 평가합니다. 후진 제거법을 통해 선택된 피처들이 모델의 예측 오차를 얼마나 줄였는지 확인할 수 있습니다.

---

2. 전진 선택법 (Forward Selection)

- 아무 피처도 없는 모델에서 시작하여, 통계적으로 가장 유의미한 피처를 하나씩 추가해나가면서 모델의 성능이 가장 좋아지는 피처 조합을 찾는 방법
  - 유의미한 피처 예시: p-value가 가장 낮은
- 각 단계에서 피처를 추가할 때마다 모델을 재학습하고 성능을 평가합니다.

주의사항

- **계산 비용**: 후진 제거법과 마찬가지로 피처의 수가 많을수록 계산 비용이 많이 들 수 있습니다.
- **지역 최적해**: 한 번 추가된 피처는 다시 제거되지 않으므로, 전역 최적해(Global Optimum)가 아닌 지역 최적해에 빠질 위험이 있습니다.

- **중복 피처**: 이미 추가된 피처와 강한 상관관계를 가지는 피처가 추가될 경우, 다중공선성 문제를 야기할 수 있습니다.

`statsmodels`를 이용한 OLS 모델

- `significance_level`: 피처를 추가할지 결정하는 유의수준
  - 일반적으로 `0.05`를 사용하며, 이 값보다 p-value가 낮으면 해당 피처는 통계적으로 유의미하다고 판단하여 추가합니다.

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# 1. 데이터 준비 (후진 제거법 예시와 동일)
X, y = make_regression(n_samples=100, n_features=10, n_informative=5,
random_state=42)
feature_names = [f'X{i}' for i in range(X.shape[1])]
X_df = pd.DataFrame(X, columns=feature_names)
X_df = sm.add_constant(X_df) # 상수항 추가

X_train, X_test, y_train, y_test = train_test_split(X_df, y, test_size=0.2,
random_state=42)

def forward_selection(X_data, y_data, significance_level=0.05):
    initial_features = ['const'] # 상수항부터 시작
    remaining_features = list(set(X_data.columns) - set(initial_features))
    selected_features = list(initial_features)

    while remaining_features:
        best_p_value = 1.0
        best_feature = None

        for feature in remaining_features:
            model = sm.OLS(y_data, X_data[selected_features + [feature]]).fit()
            p_value = model.pvalues[feature]

            if p_value < best_p_value:
                best_p_value = p_value
                best_feature = feature

        if best_feature and best_p_value < significance_level:
            selected_features.append(best_feature)
            remaining_features.remove(best_feature)
            print(f"추가된 피처: {best_feature}, p-value: {best_p_value:.4f}")
        else:
            break
    return selected_features

# 2. 전진 선택법 적용
```

```
print("--- 전진 선택법 시작 ---")
selected_features_fs = forward_selection(X_train, y_train)
print(f"최종 선택된 피처: {selected_features_fs}")
'''
추가된 피처: X2, p-value: 0.0000
추가된 피처: X1, p-value: 0.0000
추가된 피처: X4, p-value: 0.0000
추가된 피처: X3, p-value: 0.0000
추가된 피처: X8, p-value: 0.0000
추가된 피처: X0, p-value: 0.0000
추가된 피처: X9, p-value: 0.0000
추가된 피처: X6, p-value: 0.0001
추가된 피처: X5, p-value: 0.0001
추가된 피처: X7, p-value: 0.0083
최종 선택된 피처: ['const', 'X2', 'X1', 'X4', 'X3', 'X8', 'X0', 'X9', 'X6', 'X5',
'X7']
'''

# 3. 최종 모델 학습 및 평가
final_model_fs = sm.OLS(y_train, X_train[selected_features_fs]).fit()
print("\n--- 전진 선택법 최종 모델 요약 ---")
print(final_model_fs.summary())
'''
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.014e+31
Date:                Mon, 13 Oct 2025   Prob (F-statistic):               0.00
Time:                        23:10:48   Log-Likelihood:                 2299.7
No. Observations:                  80   AIC:                            -4577.
Df Residuals:                      69   BIC:                            -4551.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.043e-14   9.69e-15      2.107      0.039    1.09e-15    3.98e-14
X2            70.6476   1.04e-14   6.82e+15      0.000      70.648      70.648
X1            63.6430   1.04e-14   6.09e+15      0.000      63.643      63.643
X4            16.7483   1.06e-14   1.58e+15      0.000      16.748      16.748
X3            10.4568   9.61e-15   1.09e+15      0.000      10.457      10.457
X8             3.1586   9.86e-15    3.2e+14      0.000       3.159       3.159
X0         -5.151e-14   9.85e-15     -5.230      0.000   -7.12e-14   -3.19e-14
X9         -1.776e-14   1.08e-14     -1.641      0.105   -3.94e-14    3.83e-15
X6         -2.665e-14   9.93e-15     -2.682      0.009   -4.65e-14   -6.83e-15
X5         -6.217e-15   1.13e-14     -0.549      0.585   -2.88e-14    1.64e-14
X7          -3.02e-14   1.11e-14     -2.717      0.008   -5.24e-14   -8.03e-15
==============================================================================
Omnibus:                        2.323   Durbin-Watson:                   1.794
Prob(Omnibus):                  0.313   Jarque-Bera (JB):                1.706
Skew:                           0.178   Prob(JB):                        0.426
Kurtosis:                       3.620   Cond. No.                         1.87
==============================================================================
```

```
'''

y_pred_fs = final_model_fs.predict(X_test[selected_features_fs])
mse_fs = mean_squared_error(y_test, y_pred_fs)
print(f"테스트 세트 MSE (전진 선택법): {mse_fs:.4f}") # 0.0000
```

결과 해석 방법

- `final_model_fs.summary()`: 최종 선택된 피처들로 학습된 모델의 통계적 요약을 제공합니다. 각 피처의 계수, p-value, R-squared 값 등을 확인할 수 있습니다.
- `mean_squared_error`: 최종 모델의 예측 성능을 평가합니다. 전진 선택법을 통해 선택된 피처들이 모델의 예측 오차를 얼마나 줄였는지 확인할 수 있습니다.

장단점 및 대안

방법	장점	단점
후진 제거법	- 모든 피처를 고려하여 시작하므로, 중요한 피처가 누락될 가능성이 적음	- 계산 비용이 높음 - 지역 최적해에 빠질 수 있음 - 피처 간 상호작용을 고려하기 어려움
전진 선택법	- 계산 비용이 상대적으로 낮음 - 구현이 직관적	- 지역 최적해에 빠질 수 있음 - 한 번 추가된 피처는 제거되지 않음 - 중요한 피처가 초기에 선택되지 않으면 누락될 수 있음

대안

- **단계적 선택법 (Stepwise Selection)**: 전진 선택법과 후진 제거법을 결합한 방식으로, 각 단계에서 피처를 추가하거나 제거하면서 최적의 피처 조합을 찾습니다. 가장 널리 사용되는 방법 중 하나입니다.
- **재귀적 피처 제거 (Recursive Feature Elimination, RFE)**: 모델을 반복적으로 학습시키면서 중요도가 가장 낮은 피처를 하나씩 제거하여 최적의 피처 조합을 찾는 방법입니다. `sklearn.feature_selection.RFE`를 사용합니다.
- **L1 정규화 (Lasso Regression)**: 모델 학습 과정에서 중요도가 낮은 피처의 계수를 0으로 만들어 자동으로 피처 선택을 수행합니다. `sklearn.linear_model.Lasso`를 사용합니다.
- **트리 기반 피처 선택**: 트리 기반 모델의 `feature_importances_`를 활용하여 중요도가 높은 피처를 선택합니다. `sklearn.feature_selection.SelectFromModel`을 사용할 수 있습니다.
- **필터 방식 (Filter Methods)**: 피처와 타겟 변수 간의 통계적 관계(예: 상관계수, 카이제곱 통계량)를 기반으로 피처를 선택합니다. 모델 학습 없이 독립적으로 피처를 평가하므로 계산이 빠릅니다. `sklearn.feature_selection.SelectKBest`, `SelectPercentile` 등을 사용합니다.

Select KBest

`sklearn.feature_selection.SelectKBest(score_func=<function f_classif>, *, k=10)`
가장 중요한 K개의 특징(feature) 을 자동으로 선택하는 기능입니다. 즉, 입력 데이터(X)의 여러 특성 중에서 통계적으로 유의미한 상위 K개만 남기는 역할을 합니다.

- 하이퍼 파라미터

    - `f_classif`: ANOVA F-value between label/feature for classification tasks.
    - `mutual_info_classif`: Mutual information for a discrete target.
    - `chi2`: Chi-squared stats of non-negative features for classification tasks.
    - `f_regression`:F-value between label/feature for regression tasks.
    - `mutual_info_regression`: Mutual information for a continuous target.
    - `SelectPercentile`: Select features based on percentile of the highest scores.
    - `SelectFpr`: Select features based on a false positive rate test.
    - `SelectFdr`: Select features based on an estimated false discovery rate.
    - `SelectFwe`: Select features based on family-wise error rate.
    - `GenericUnivariateSelect`: Univariate feature selector with configurable mode.
- 지원 메소드
    - `get_support`: Get a mask, or integer index, of the features selected
    - `fit(X, y)`: Run score function on (X, y) and get the appropriate features.

```
# 파이프라인 예시
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.feature_selection import SelectKBest, f_classif

pipe = Pipeline([
    ('select', SelectKBest(score_func=f_classif, k=3)),
    ('model', SVC())
])

pipe.fit(X, y)
```