머신러닝 개요

- 데이터를 기반으로 학습하고 예측하는 알고리즘의 한 분야
- 머신러닝은 컴퓨터가 명시적으로 프로그래밍되지 않고도 데이터에서 학습하고 예측 가능
- 머신러닝의 핵심 사항
    - 데이터: 머신러닝은 데이터를 기반으로 학습. 즉 좋은 데이터가 좋은 모델의 핵심
    - 알고리즘: 머신러닝에는 다양한 알고리즘이 있습니다. 각 알고리즘은 특정 유형의 문제에 더 적합
    - 하이퍼파라미터: 알고리즘의 성능에 영향을 미치는 매개변수를 하이퍼파라미터라고 하는데 하이퍼파라미터는 실험을 통해 최적화
    - 평가 지표: 모델의 성능을 측정하는 방법으로 분류모델은 정확도, 정밀도, 재현율, F1 점수, 회귀모델은 평균 절대 오차, 회귀 작업의 평균 제곱근 오차와 같은 메트릭을 포함
- 머신러닝 유형
    - 지도 학습: 선형 회귀, 로지스틱 회귀, 결정 트리, 랜덤 포레스트, 그래디언트 부스팅, SVM(Support Vector Machines) 및 k-NN(k-nearest neighbours)과 같은 알고리즘은 레이블이 지정된 학습 데이터를 사용하여 입력 기능을 기반으로 대상 변수를 예측하는 데 사용됩니다.
    - 비지도 학습: k-평균, 계층적 클러스터링 및 DBSCAN과 같은 클러스터링 기법 및 PCA(Principal Component Analysis), LDA, SVD 등의 차원 축소 기법
    - 강화 학습: 에이전트가 조치를 취하고 그 대가로 보상이나 처벌을 받음으로써 환경에서 행동하는 방법을 배우는 기계 학습으로 목표는 시간이 지남에 따라 누적 보상을 최대화하는 것이다.
    - 신경망 및 딥러닝: 신경망, 역전파 및 CNN(Convolutional Neural Networks), RNN(Recurrent Neural Networks), Long Short-Term 메모리(LSTM) 네트워크
    - 자연어 : 트랜스포머 기반의 BERT, GPT Series의 발전으로 자연어 처리의 급속한 발전 진행
    - LLM : 거대 생성 언어모델 기반의 ChatGPT 시리즈의 발전으로 인간과 상호작용할 수 있는 수준으로 발전
    - 비전 : Yolo를 이용한 다양한 객체탐지솔루션이 폭넓게 활용되고 Stable Diffusion이 이미지 생성을 위한 주요 솔루션으로 확대 예정

지도 학습

- 학습의 원리
    - 지도 학습에서 알고리즘은 레이블이 지정된 학습 데이터에서 학습하고 해당 데이터를 기반으로 예측을 수행.

    - 지도 학습 알고리즘은 알려진 입력 데이터 세트(특징)와 데이터에 대한 알려진 응답(대상 또는 레이블)을 입력받아 모델을 훈련하여 새 데이터에 대한 응답에 대한 합리적인 예측을 생성.
- 회귀 및 분류라는 두 가지 유형의 문제에 사용.
    - 회귀: 회귀는 대상 또는 종속 변수가 연속적이거나 정렬된 전체 값일 때 사용. 회귀에 사용되는 알고리즘에는 선형 회귀, 결정 트리, 랜덤 포레스트 및 GBM 계열의 회귀트리 모델 포함.
    - 분류: 분류는 대상 변수가 범주형일 때, 간단히 말해서 입력 데이터를 범주로 분류할 때 사용. 분류에 사용되는 알고리즘에는 Logistic Regression, Naive Bayes, Decision Trees, Random Forests, Support Vector Machines 및 Neural Networks 및 GBM 계열의 Boosting 모델 포함.
- 알고리즘이 작동하는 방식
    - 데이터 수집: 데이터를 수집하고 전처리. 데이터는 feature와 label로 구분. 이를 위해 사용되는 데이터는 출력 데이터 세트도 제공되기 때문에 레이블이 지정된 데이터라고 한다.
    - 모델 훈련: 알고리즘은 훈련 데이터를 통해 학습. feature와 label 간의 패턴 또는 관계를 발견하려고 시도.
    - 모델 예측: 모델이 훈련되면 본 적이 없는 새로운 데이터의 결과를 예측하는 데 사용. 이 새 데이터에 대한 입력을 테스트 데이터라고 한다.
    - 평가: 모델의 예측을 실제 값과 비교하여 모델의 정확도를 평가. 정확도, 정밀도, 재현율, F1 점수(분류용), 평균 절대 오차(MAE), 평균 제곱 오차(MSE), 평균 제곱근 오차(RMSE)(회귀용)와 같은 다양한 메트릭이 모델을 평가하는 데 사용.
    - 조정: 모델의 성능이 만족스럽지 않으면 모델로 돌아가 하이퍼파라미터를 조정하거나 다른 모델을 모두 선택 가능
    - 예측: 만족스러운 성능이 달성되면 이제 모델을 사용하여 보이지 않는 새로운 데이터를 예측할 수 있다.

지도학습 - 분류

사이킷런을 사용하여 붓꽃 데이터셋을 로드하고, 이를 사용해 의사결정나무 알고리즘을 이용하여 붓꽃의 종류를 분류합니다.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# 데이터 로드
iris = load_iris()
X = iris.data
y = iris.target

# 학습 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# 모델 초기화 및 학습
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

# 예측
y_pred = dt.predict(X_test)

# 정확도 평가
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9777777777777777
```

지도학습 - 회귀분석

- 캘리포니아 주택 가격 데이터셋을 로드합니다.
- 데이터를 학습용 데이터와 테스트용 데이터로 나눕니다.
- 선형 회귀 모델을 초기화하고 학습시킵니다.
- 학습된 모델을 사용하여 테스트 데이터의 주택 가격을 예측합니다.
- 예측값과 실제값을 비교하여 모델의 성능을 평가합니다.

결과 해석

- Mean Squared Error는 예측값과 실제값의 차이의 제곱의 평균으로, 값이 작을수록 모델의 예측이 정확함을 의미합니다.
- 결과 데이터프레임은 실제 주택 가격과 예측된 주택 가격을 나란히 보여줍니다.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.datasets import fetch_california_housing

# 데이터 로드
califonia = fetch_california_housing()
X, y = califonia.data, califonia.target

# 학습 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 모델 초기화 및 학습
model = LinearRegression()
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)

# 모델 평가
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse: .2f}')

# 결과 일부 출력
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results.head()
```

```
Mean Squared Error:  0.53
```

	Actual	Predicted
0	0.47700	0.726049
1	0.45800	1.767434
2	5.00001	2.710922
3	2.18600	2.835147
4	2.78000	2.606958

비지도학습

- 비지도 학습(Unsupervised Learning)은 레이블이 지정되지 않은 데이터를 사용하여 모델을 학습하는 방법
- 지도 학습과 달리 비지도 학습은 모델의 성능을 평가하기 위한 레이블이 불필요.
- 비지도 학습 알고리즘
  - 클러스터링: 클러스터링 알고리즘에는 K-평균, 평균-점 클러스터링, 밀도기반 군집(DBSCAN) 등

- 차원 축소: 차원 축소 알고리즘에는 주성분 분석(PCA), 선형판별분석(LDA), 특이값 분해(SVD) 등

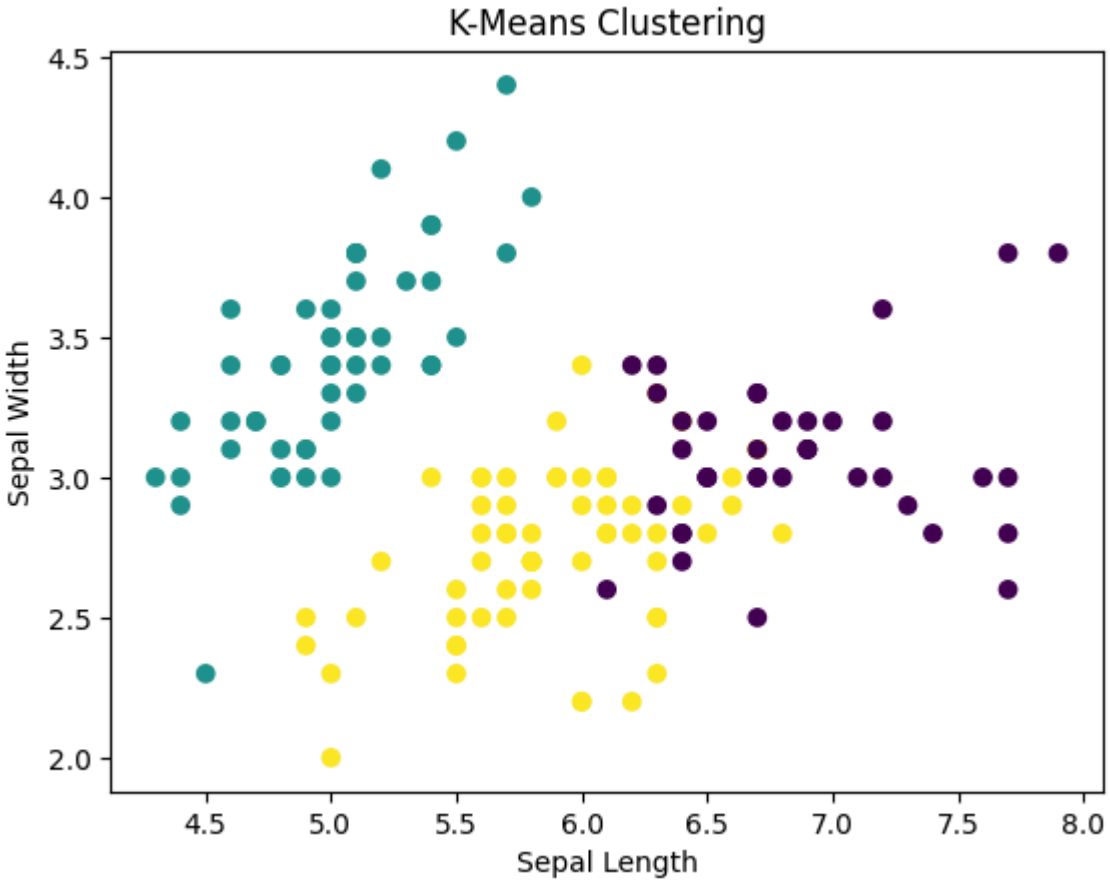
- 비지도 학습은 머신 러닝에서 강력한 도구입니다. 비지도 학습은 데이터를 이해하고 데이터에서 패턴을 찾는 데 사용.

비지도학습 - 군집화(Clustering)

사이킷런을 이용하여 붓꽃 데이터셋을 로드하고, 이를 사용해 K-평균(K-Means) 알고리즘을 이용하여 데이터를 군집화합니다.

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create a K-Means clustering model
kmeans = KMeans(n_clusters=3, n_init='auto', random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
print(pd.Series(labels).value_counts(), '₩n')

# Fit the model to the data
plt.scatter(X[:,0], X[:,1], c=labels, cmap='viridis')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('K-Means Clustering')
plt.show()
print()

# Get the cluster labels for each data point
df = pd.DataFrame(X, columns=iris.feature_names)
df['cluster'] = labels
df.head()
```

```
2    61
1    50
0    39
Name: count, dtype: int64
```



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	cluster
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1

머신러닝 Framework - 사이킷런

- 사이킷런(scikit-learn)은 파이썬 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리.
- 사이킷런은 파이썬 기반의 머신러닝을 위한 가장 쉽고 효율적인 개발 라이브러리를 제공.
- 사이킷런의 특징
  - 파이썬 기반의 다른 머신러닝 패키지도 사이킷런 스타일의 API를 지향할 정도로 쉽고 가장 파이썬스러운 API를 제공.

- 머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 API를 제공.
- 많은 머신러닝 알고리즘이 효율적으로 구현되어 있기에 머신러닝을 처음 배울 때 사용하기 유용.
- 사이킷런은 오픈 소스 라이브러리이기 때문에 누구나 자유롭게 사용 가능
- 사이킷런은 다음과 같은 다양한 머신러닝 알고리즘을 제공.
    - 분류
    - 회귀
    - 클러스터링
    - 차원 축소
    - 특징 추출
    - 모델 선택 및 평가
- 사이킷런은 또한 다음과 같은 다양한 프레임워크와 API를 제공합니다.
    - 데이터 전처리
    - 모델 학습 및 예측
    - 모델 저장 및 복원
    - 모델 모니터링
    - 모델 배포

Python의 기계 학습 라이브러리인 Scikit-Learn을 효과적으로 이해하기 위해 배워야 할 주요 사항

- Python 프로그래밍: Scikit-Learn은 Python 라이브러리이므로 Python 숙련도가 중요. 데이터 유형, 제어 흐름, 기능 이해 및 패키지 작업이 포함.
- 데이터 구조: 목록, 사전 및 집합과 같은 Python의 기본 데이터 구조를 이해하는 것이 중요. 데이터 분석 라이브러리인 Pandas는 특히 DataFrames와 함께 Scikit-Learn에서 사용할 데이터 조작에 필수적.
- NumPy 및 Matplotlib: Scikit-Learn은 숫자 데이터를 효율적으로 처리하는 데 사용되는 NumPy 배열과 잘 작동. 플로팅 라이브러리인 Matplotlib는 데이터를 시각화하고 결과를 모델링하는 데 사용.
- 기본 통계 및 확률: 평균, 중앙값, 모드, 표준 편차, 확률 분포, 가설 테스트 및 상관 계수가 포함.
- 머신 러닝 개념: 지도 학습, 비지도 학습, 강화 학습 등 머신 러닝의 기본 사항을 숙지하고 회귀, 분류, 클러스터링, 차원 감소 및 앙상블 방법과 같은 일반적인 알고리즘 및 개념을 이해.
- Scikit-Learn API: Scikit-Learn API와 일관된 인터페이스를 이해하는 것이 중요. 여기에는 라이브러리를 가져오고, 모델을 만들고, 모델을 데이터에 맞추고, 새 데이터 포인트를 예측하고, 모델의 성능을 평가하는 방법을 아는 것이 포함.
- 데이터 전처리: 크기 조정, 정규화, 범주형 변수 인코딩 및 누락된 값 처리를 위한 Scikit-Learn의 전처리 도구를 사용하는 방법.

- 모델 평가 및 조정: 교차 검증, 혼동 행렬, ROC 곡선, 정밀도, 재현율, F1 점수 등과 같은 모델 평가를 위한 Scikit-Learn 도구를 사용하는 방법과 GridSearch 및 RandomizedSearch와 같은 기술을 사용하여 모델을 조정하는 방법 이해.
- 파이프라인 생성: Scikit-Learn은 전처리 및 모델 교육 프로세스를 간소화하는 파이프라인을 생성하는 유틸리티를 제공. 이것은 효율적이고 재현 가능한 코드를 만들기 위해 배워야 할 중요한 측면이다.
- 딥 러닝: Scikit-Learn은 주로 딥 러닝에 사용되지 않지만 알아두면 유용할 수 있는 신경망에 대한 일부 지원을 제공.

사이킷런의 주요 모듈

- model_selection: 학습 데이터와 테스트 데이터를 분리하거나 교차 검증, 그리고 Estimator의 하이퍼 파라미터를 튜닝하기 위해서 다양한 함수와 클래스를 제공.
- preprocessing: 데이터를 정규화, 스케일링, 인코딩하는 등 다양한 기능을 제공.
- datasets: 다양한 데이터 세트를 제공.
- linear_model: 선형 회귀, 로지스틱 회귀, 릿지 회귀, 라쏘 회귀 등 다양한 선형 모델을 제공.
- tree: 의사 결정 트리, 랜덤 포레스트, 그래디언트 부스팅 트리 등 다양한 트리 기반 모델을 제공.
- svm: 서포트 벡터 머신, 커널 서포트 벡터 머신 등 다양한 서포트 벡터 머신을 제공.
- neighbors: K-최근접 이웃 알고리즘을 제공.
- ensemble: 앙상블 방법을 제공.
- cluster: 클러스터링 알고리즘을 제공.

데이터 전처리: 스케일링과 정규화

```
# 사이킷런의 StandardScaler를 사용하여 데이터의 평균과 표준 편차를 계산하고 이를 사용하여 데이터를

from sklearn.preprocessing import StandardScaler
import numpy as np

# 예제 데이터
data = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]])

# 표준화
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# 평균과 표준 편차 출력
print('Mean:', scaler.mean_)
print('Standard Deviation:', scaler.scale_)
print('Scaled Data:')
```

```
Mean: [5. 6.]
Standard Deviation: [2.82842712 2.82842712]
Scaled Data:
```

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np

# 예제 데이터
data = np.array([[1, 2, 3,], [4, 5, 6], [7, 8, 9]])

# 표준화
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
print("Standard Scaled Data:₩n", scaled_data)

# 정규화
minmax_scaler = MinMaxScaler()
normalized_data = minmax_scaler.fit_transform(data)
print("Min-Max Normalized Data:₩n", normalized_data)
```

```
Standard Scaled Data:
 [[-1.22474487 -1.22474487 -1.22474487]
 [ 0.          0.          0.        ]
 [ 1.22474487  1.22474487  1.22474487]]
Min-Max Normalized Data:
 [[0.  0.  0. ]
 [0.5 0.5 0.5]
 [1.  1.  1. ]]
```

Q. 와인 데이터셋을 이용한 와인의 종류를 예측하는 모델을 학습한 후 정확도로 평가를 수행하세요.

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# 1. 데이터 로드
wine = load_wine()
X = wine.data
y = wine.target

# 2. 학습용과 테스트용 데이터셋으로 나누기
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# 3. 로지스틱 회귀 모델 생성 및 학습
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

# 4. 예측 및 정확도 계산
y_pred = dt.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.97
```

Q. 당뇨병 데이터셋을 이용하여 당뇨병 진행도를 예측하는 회귀모델을 학습시킨 후 MSE로 평가를 수행하세요. (단, 독립변수들에 대하여 표준화를 적용)

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# 데이터 로드
diabetes = load_diabetes()

# Create a pandas DataFrame from the dataset
df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
df['target'] = diabetes.target

# 학습용과 테스트용 데이터셋으로 나누기
X_train, X_test, y_train, y_test = train_test_split(df.drop('target', axis=1), df['target'], test_

# 표준화
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 선형 회귀 모델 생성 및 학습
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 예측
y_pred = model.predict(X_test_scaled)

# MSE 계산
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 2900.1936284934823
```

데이터셋 로드 및 분할

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# 데이터 로드
iris = load_iris()
X, y = iris.data, iris.target
```

로지스틱 회귀(Logistic Regression)

- 분류 문제를 해결하기 위한 알고리즘으로, 선형 회귀와 마찬가지로 입력 특성의 가중치 합을 계산하는데, 선형 회귀와 달리 결과를 이진 분류(0 또는 1, 참 또는 거짓 등)에 사용할 수 있는 확률로 변환.
- 로지스틱 회귀는 선형 회귀와 비슷하게 모델의 예측과 실제 값 사이의 차이를 최소화하도록 가중치를 학습. 하지만 로지스틱 회귀는 선형 회귀와는 달리 결과를 0과 1 사이의 값으로 제한하는 로지스틱 함수(또는 시그모이드 함수)를 사용.
- 로지스틱 회귀는 각 클래스에 속할 확률을 제공하며, 특정 임계값(일반적으로 0.5)을 초과하는 경우 데이터 포인트를 해당 클래스에 할당. 이는 이진 분류뿐만 아니라 다중 클래스 분류에도 적용될 수 있다(이 경우에는 일대다(OvR) 또는 다항 로지스틱 회귀를 사용할 수 있다).
- 로지스틱 회귀는 출력이 확률이기 때문에, 결과의 해석이 직관적이며 모델의 예측이 불확실한 경우에도 그 정도를 측정할 수 있다. 또한 로지스틱 회귀는 선형 회귀보다 이상치에 덜 민감하며, 모델이 과적합되는 것을 방지하기 위해 규제를 적용할 수 있다는 장점도 있다.
- 로지스틱 함수, 또는 시그모이드 함수는 S-자 형태를 띠는 함수로, 실수 입력값을 0과 1 사이의 출력값으로 변환하는 데 사용. 이 함수는 머신러닝, 특히 이진 분류 문제에서 중요한 역할을 한다.
- 로지스틱 함수의 정의
  - f(x) = 1 / (1 + e^-x)
  - e는 자연 상수(약 2.71828). x는 어떤 실수 값도 가능하며, -무한대에서 무한대까지의 범위를 가지며 이 함수는 모든 실수 입력에 대해 0과 1 사이의 값을 반환.
  - 함수가 결과를 0과 1 사이로 제한하기 때문에, 이는 확률에 대해 논의할 때 특히 유용. 로지스틱 회귀 분석에서 이 함수는 선형 함수의 결과를 확률로 변환하는데 사용.
  - 입력값 x가 커질수록 로지스틱 함수의 출력은 1에 가까워지고, x가 작아질수록 출력은 0에 가까워진다. x가 0일 때 로지스틱 함수의 값은 0.5입니다. 이러한 특성 때문에 로지스틱 함수는 이진 분류 문제에 널리 사용.

데이터셋을 로드하고, 전처리한 후, 로지스틱 회귀 모델을 학습시키고, 모델의 성능을 평가하는 과정

- 타이타닉 데이터셋을 로드하고 필요한 컬럼만 선택합니다.
- 결측치를 처리하고 범주형 변수를 인코딩합니다.
- 데이터를 학습용과 테스트용으로 분리합니다.
- 로지스틱 회귀 모델을 학습시키고 예측을 수행합니다.
- 모델의 정확도, 혼동 행렬, 분류 보고서를 출력합니다.
- ROC AUC 성능을 출력합니다.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

data = data_org.copy()

# 필요한 컬럼 선택 및 결측치 처리
data = data[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
data = data.dropna()

# 범주형 변수 인코딩
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# 특성과 레이블 분리
X = data.drop('Survived', axis=1)
y = data['Survived']

# 학습 데이터와 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 로지스틱 회귀 모델 학습
model = LogisticRegression()
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)

# 모델 평가
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {round(accuracy,4)}')
print('Confusion Matrix:')
print(conf_matrix,'₩n')
print('Classification Report:')
print(class_report,'₩n')

# 확률값 확인
y_pred_prob = model.predict_proba(X_test)[:, 1]

# ROC AUC 평가
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = roc_auc_score(y_test, y_pred_prob)
print(f'ROC_AUC : {round(roc_auc,4)}')
```

```
Accuracy: 0.7483
Confusion Matrix:
[[71 16]
 [20 36]]

Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.78      0.82      0.80        87
              1       0.69      0.64      0.67        56

       accuracy                           0.75       143
      macro avg       0.74      0.73      0.73       143
   weighted avg       0.75      0.75      0.75       143


   ROC_AUC : 0.8164
```

교차 검증

모델의 성능을 평가하기 위한 기법으로, 데이터를 여러 번 나누어 여러 모델을 학습시키고 검증하는 방법입니다. 교차 검증을 통해 모델의 일반화 성능을 더 잘 추정할 수 있습니다.

교차 검증의 과정

- 데이터 분할:
  - 전체 데이터를 여러 개의 폴드(fold)로 나눕니다. 예를 들어, 5-폴드 교차 검증에서는 데이터를 5개의 폴드로 나눕니다.
- 모델 학습 및 평가:
  - 각 폴드에 대해 한 번씩 테스트 데이터로 사용하고 나머지 폴드는 학습 데이터로 사용합니다.
  - 즉, 5-폴드 교차 검증에서는 5번의 학습 및 평가 과정을 거칩니다.
- 성능 측정:
  - 각 폴드에서 얻은 평가 점수를 기록하고, 이를 평균하여 최종 성능을 객관적으로 측정합니다.
- 장점
  - 일반화 성능 추정: 모델이 새로운 데이터에 대해 얼마나 잘 작동할지를 더 객관적으로 추정할 수 있습니다.
  - 데이터 효율성: 데이터를 반복해서 사용하므로 데이터 낭비가 적습니다.
  - 모델 튜닝: 모델의 하이퍼파라미터를 튜닝할 때 유용합니다.

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

# 모델 초기화
model = LogisticRegression(max_iter=200)

# 교차 검증
scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-Validation Scores: {scores}")
print(f"Average Score: {np.mean(scores):.2f}")
```

```
Cross-Validation Scores: [0.76223776 0.81818182 0.78321678 0.76923077 0.81690141]
Average Score: 0.79
```

서포트 벡터 머신(Support Vector Machines, SVM)

- 분류나 회귀, 이상치 탐지 등에 사용되는 강력한 머신러닝 알고리즘 중 하나입니다. SVM은 주로 분류 문제에 사용되며, 이 알고리즘의 핵심 아이디어는 데이터를 고차원 공간으로 변환하여 서로 다른 클래스 간의 최대 마진을 찾는 것입니다.
- SVM은 데이터를 두 개의 클래스로 나누는 결정 경계(결정 초평면이라고도 함)를 찾습니다. 이 결정 경계는 각 클래스의 가장 가까운 훈련 샘플(서포트 벡터라고 함)까지의 거리가 최대가 되는 선을 찾는 것을 목표로 합니다. 이를 '마진 최대화'라고 하며, 이 마진 최대화는 오류를 최소화하고 모델의 일반화 성능을 향상시키는 데 중요한 역할을 합니다.
- SVM은 선형 뿐만 아니라 비선형 분류 문제에도 사용할 수 있습니다. 비선형 문제를 해결하기 위해, SVM은 커널 트릭이라는 기법을 사용하여 데이터를 고차원 공간으로 변환하고, 그 고차원에서 선형 결정 경계를 찾습니다. 이 커널 트릭 덕분에 SVM은 복잡한 분류 문제를 처리할 수 있습니다.
- SVM은 작은 데이터셋에서도 잘 작동하며, 높은 차원의 데이터에 대해 강력한 성능을 발휘합니다. 그러나 데이터셋이 크거나 노이즈가 많은 경우, 그리고 데이터가 선형적으로 구분되지 않는 경우에는 다른 알고리즘(예: 랜덤 포레스트나 신경망)에 비해 성능이 떨어질 수 있습니다.

그리드 서치를 통한 하이퍼파라미터 튜닝

GridSearchCV는 사이킷런에서 제공하는 하이퍼파라미터 튜닝 기법으로, 모델의 성능을 최적화하기 위해 다양한 하이퍼파라미터 조합을 시도하는 방법입니다. GridSearchCV는 사용자가 제공한 하이퍼파라미터 값들의 조합을 모두 시도하여, 가장 좋은 성능을 내는 조합을 찾습니다

GridSearchCV의 과정

- 하이퍼파라미터 그리드 설정: 튜닝할 하이퍼파라미터와 그 후보 값들의 집합을 정의합니다.
- 모델 초기화: 튜닝할 머신러닝 모델을 초기화합니다.
- 교차 검증을 통한 모델 학습 및 평가:
  - 각 하이퍼파라미터 조합에 대해 교차 검증을 수행합니다.
  - 교차 검증을 통해 얻은 평균 성능 점수를 기록합니다.
- 최적의 하이퍼파라미터 선택: 교차 검증 결과를 바탕으로 가장 높은 성능을 보인 하이퍼파라미터 조합을 선택합니다.
- 최적 모델로 학습: 최적의 하이퍼파라미터를 사용하여 모델을 학습시킵니다.

GridSearchCV의 장점

- 자동화된 하이퍼파라미터 튜닝: 다양한 하이퍼파라미터 조합을 자동으로 시도하여 최적의 조합을 찾을 수 있습니다.

- 교차 검증 사용: 교차 검증을 통해 모델의 일반화 성능을 평가하므로, 데이터셋에 대한 과적합을 방지합니다.
- 다양한 모델 지원: 사이킷런의 다양한 모델과 함께 사용할 수 있습니다

아래 코드에서는 다음과 같은 작업을 수행합니다:

- 파라미터 그리드 설정: param_grid 변수에 SVM(Support Vector Machine) 모델의 하이퍼파라미터 C와 kernel의 후보 값들을 정의합니다.
    - C: 규제 파라미터로, 값이 작을수록 규제가 강해집니다.
    - kernel: SVM에서 사용할 커널 함수로, 여기서는 'linear'와 'rbf'를 시도합니다.
- 모델 초기화: SVC 모델을 초기화합니다.
- 그리드 서치 초기화 및 학습:
    - GridSearchCV 객체를 초기화하고, svc 모델과 param_grid를 전달합니다.
    - cv=5는 5-폴드 교차 검증을 의미합니다.
    - fit 메서드를 호출하여 그리드 서치를 수행하고 모델을 학습시킵니다.
- 최적 파라미터와 점수 출력:
    - grid_search.best_params_를 사용하여 최적의 하이퍼파라미터 조합을 출력합니다.
    - grid_search.best_score_를 사용하여 최적 조합에서의 교차 검증 평균 점수를 출력합니다.

Task1_0722. Adult Income 데이터셋을 이용한 전처리 및 분류 모델(소득이 50K 이상인지 예측)을 아래 설명을 참조하여 수행하세요.

[문제 설명]

- Adult Income 데이터셋을 로드합니다.
- 결측치를 처리합니다.
- 이상치를 제외합니다.
- 파생 변수를 작성합니다.
- 범주형 변수를 인코딩합니다.
- 변수 선택 및 독립변수 종속변수를 분리합니다.
- 데이터를 표준화합니다.
- 데이터셋을 학습용과 테스트용으로 나눕니다.
- Logistic Regression 모델 생성 및 학습합니다.
- 예측 및 평가합니다.

```
# 1. 데이터 로드
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
           'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
           'hours-per-week', 'native-country', 'income']
data = pd.read_csv(url, header=None, names=columns, na_values='?', skipinitialspace=True)
```

QR 방법을 사용한 이상치 제거 가이드

IQR 계산

- IQR(Interquartile Range)은 데이터의 중앙 50%를 나타내며, 데이터의 변동성을 측정하는 데 사용됩니다. IQR은 3사분위수(Q3)와 1사분위수(Q1)의 차이로 계산됩니다.
    - Q1: 데이터의 25번째 백분위수(1사분위수)
    - Q3: 데이터의 75번째 백분위수(3사분위수)
    - IQR: Q3 - Q1

이상치 경계 계산

- 일반적으로 IQR의 1.5배를 사용하여 이상치 경계를 설정합니다.
    - Lower Bound: Q1 - 1.5 * IQR
    - Upper Bound: Q3 + 1.5 * IQR
- 이 경계를 벗어나는 데이터 포인트는 이상치로 간주됩니다.

이상치를 제거는 데이터의 분포를 왜곡하는 극단적인 값을 제거하여 데이터 분석 및 모델 성능을 향상시키는 데 도움이 됩니다. 도메인 지식과 데이터의 특성을 고려해서 이상치 제거를 진행합니다.

Adult Income 데이터셋(또는 "Census Income" 데이터셋)은 미국 인구 조사 데이터를 바탕으로 각 개인의 특성에 따른 소득 수준을 예측하는 데 사용됩니다. 이 데이터셋의 컬럼들은 다음과 같습니다:

- age: 나이 (숫자) 개인의 나이를 나타냅니다.
- workclass: 직업 유형 (범주형) 개인의 직업 유형을 나타냅니다. 예를 들어, 'Private', 'Self-emp-not-inc', 'Self-emp-inc', 'Federal-gov', 'Local-gov', 'State-gov', 'Without-pay', 'Never-worked' 등이 있습니다.
- fnlwgt: 최종 가중치 (숫자) 인구 조사에서 각 행이 전체 인구를 대표하는 비율을 나타내는 가중치입니다. 더 큰 값은 더 큰 대표성을 의미합니다.
- education: 교육 수준 (범주형) 개인의 교육 수준을 나타냅니다. 예를 들어, 'Bachelors', 'Some-college', '11th', 'HS-grad', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', '9th', '7th-8th', '12th', 'Masters', '1st-4th', '10th', 'Doctorate', '5th-6th', 'Preschool' 등이 있습니다.
- education-num: 교육 수준(숫자) (숫자) 교육 수준을 숫자로 나타낸 것입니다. 예를 들어, 'Bachelors'는 13, 'HS-grad'는 9 등으로 교육의 연수를 나타냅니다.
- marital-status: 결혼 상태 (범주형) 개인의 결혼 상태를 나타냅니다. 예를 들어, 'Married-civ-spouse', 'Divorced', 'Never-married', 'Separated', 'Widowed', 'Married-spouse-absent',

'Married-AF-spouse' 등이 있습니다.

- occupation: 직업 (범주형) 개인의 직업을 나타냅니다. 예를 들어, 'Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial', 'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving', 'Priv-house-serv', 'Protective-serv', 'Armed-Forces' 등이 있습니다.
- relationship: 가족 관계 (범주형) 개인의 가족 관계를 나타냅니다. 예를 들어, 'Wife', 'Own-child', 'Husband', 'Not-in-family', 'Other-relative', 'Unmarried' 등이 있습니다.
- race: 인종 (범주형) 개인의 인종을 나타냅니다. 예를 들어, 'White', 'Asian-Pac-Islander', 'Amer-Indian-Eskimo', 'Other', 'Black' 등이 있습니다.
- sex: 성별 (범주형) 개인의 성별을 나타냅니다. 'Male' 또는 'Female'입니다.
- capital-gain: 자본 이득 (숫자) 개인의 자본 이득을 나타냅니다. 자본 자산의 매매에서 발생하는 이익입니다.
- capital-loss: 자본 손실 (숫자) 개인의 자본 손실을 나타냅니다. 자본 자산의 매매에서 발생하는 손실입니다.
- hours-per-week: 주당 근무 시간 (숫자) 개인이 주당 일하는 시간을 나타냅니다.
- native-country: 출생 국가 (범주형) 개인의 출생 국가를 나타냅니다. 예를 들어, 'United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada', 'Germany', 'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras', 'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France', 'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua', 'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands' 등이 있습니다.
- income: 소득 수준 (범주형) 개인의 소득 수준을 나타냅니다. ' <=50K' 또는 ' >50K'로, 연 소득이 50,000달러 이하인지 초과인지를 나타냅니다.

QR 방법을 사용한 이상치 제거 가이드

IQR 계산

- IQR(Interquartile Range)은 데이터의 중앙 50%를 나타내며, 데이터의 변동성을 측정하는 데 사용됩니다. IQR은 3사분위수(Q3)와 1사분위수(Q1)의 차이로 계산됩니다.
    - Q1: 데이터의 25번째 백분위수(1사분위수)
    - Q3: 데이터의 75번째 백분위수(3사분위수)
    - IQR: Q3 - Q1

이상치 경계 계산

- 일반적으로 IQR의 1.5배를 사용하여 이상치 경계를 설정합니다.
    - Lower Bound: Q1 - 1.5 * IQR
    - Upper Bound: Q3 + 1.5 * IQR
- 이 경계를 벗어나는 데이터 포인트는 이상치로 간주됩니다.

이상치를 제거는 데이터의 분포를 왜곡하는 극단적인 값을 제거하여 데이터 분석 및 모델 성능을 향상시키는 데 도움이 됩니다. 도메인 지식과 데이터의 특성을 고려해서 이상치 제거를 진행합니다.

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}

svc = SVC()

grid_search = GridSearchCV(svc, param_grid, cv=5)
grid_search.fit(X, y)

print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Score: {grid_search.best_score_:.2f}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-9a5c6d3e31a8> in <cell line: 9>()
      7
      8 grid_search = GridSearchCV(svc, param_grid, cv=5)
----> 9 grid_search.fit(X, y)
     10
     11 print(f"Best Parameters: {grid_search.best_params_}")

NameError: name 'X' is not defined
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.datasets import load_iris

# Load example data
data = load_iris()
X = data.data
y = data.target

# Define the parameter grid
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}

# Initialize the SVC
svc = SVC()

# Set up the grid search
grid_search = GridSearchCV(svc, param_grid, cv=5)

# Fit the grid search
grid_search.fit(X, y)

# Print the best parameters and best score
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Score: {grid_search.best_score_:.2f}")
```

```
Best Parameters: {'C': 1, 'kernel': 'linear'}
Best Score: 0.98
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# 데이터셋 로드
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupati
data = pd.read_csv(url, header=None, names=columns, na_values='?', skipinitialspace=True)

# 결측치가 있는 행 삭제
data.dropna(inplace=True)

# 새로운 파생 변수 생성: capital-diff (capital-gain - capital-loss)
data['capital-diff'] = data['capital-gain'] - data['capital-loss']

# 타겟 변수 인코딩: income (소득이 50K 이상인지 여부)
data['income'] = data['income'].apply(lambda x: 1 if x == '>50K' else 0)

# 피처(특징) 열과 타겟(목표) 정의
X = data.drop('income', axis=1)
y = data['income']

# 숫자형 및 범주형 피처 선택
numeric_features = ['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-w
categorical_features = ['workclass', 'education', 'marital-status', 'occupation', 'relationship',

# 전처리 파이프라인 설정
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

# 로지스틱 회귀 모델 파이프라인 생성
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])

# 데이터를 학습용과 테스트용으로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 모델 학습
clf.fit(X_train, y_train)

# 예측 및 평가
y_pred = clf.predict(X_test)
report = classification_report(y_test, y_pred)

# 평가 결과 출력
```

```
print(report)
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4503
           1       0.76      0.62      0.68      1530

    accuracy                           0.85      6033
   macro avg       0.82      0.78      0.79      6033
weighted avg       0.85      0.85      0.85      6033
```

```
import pandas as pd

# 데이터 로드
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
           'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
           'hours-per-week', 'native-country', 'income']
data = pd.read_csv(url, header=None, names=columns, na_values='?', skipinitialspace=True)

# 결측치 처리
data.dropna(inplace=True)

# 이상치 제거
Q1 = data['capital-gain'].quantile(0.25)
Q3 = data['capital-gain'].quantile(0.75)
IQR = Q3 - Q1
capital_gain_outliers = data[(data['capital-gain'] < (Q1 - 1.5 * IQR)) | (data['capital-gain'] > (

Q1 = data['capital-loss'].quantile(0.25)
Q3 = data['capital-loss'].quantile(0.75)
IQR = Q3 - Q1
capital_loss_outliers = data[(data['capital-loss'] < (Q1 - 1.5 * IQR)) | (data['capital-loss'] > (

data = data.drop(capital_gain_outliers.index)
data = data.drop(capital_loss_outliers.index)

# 데이터프레임의 일부를 출력하여 확인
print(data.head())
```

```
   age         workclass  fnlwgt  education  education-num  ₩
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13
5   37           Private  284582    Masters             14

       marital-status         occupation   relationship   race     sex  ₩
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2           Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female
```

```
5  Married-civ-spouse    Exec-managerial           Wife  White  Female

   capital-gain  capital-loss  hours-per-week native-country income
1             0             0              13  United-States  <=50K
2             0             0              40  United-States  <=50K
3             0             0              40  United-States  <=50K
4             0             0              40           Cuba  <=50K
5             0             0              40  United-States  <=50K
```

```
data['capital-diff'] = data['capital-gain'] - data['capital-loss']
```

One-Hot Encoding

- 비순서형 범주형 변수: 대부분의 범주형 변수는 순서가 없습니다. 예를 들어, workclass, occupation, race, sex 등의 변수는 순서가 없습니다. 이 경우, 각 범주를 고유한 이진 벡터로 변환하는 것이 적합합니다.
- 모델의 가정: Logistic Regression과 같은 선형 모델은 입력 변수들이 서로 독립적이라고 가정합니다. One-hot encoding은 이러한 가정을 유지하는 데 도움이 됩니다.
- 다중공선성 방지: One-hot encoding은 다중공선성 문제를 피하기 위해 첫 번째 범주를 제거할 수 있습니다. 이 방법은 drop_first=True 옵션을 사용하여 구현됩니다.

Label Encoding

- 순서형 범주형 변수: Label encoding은 범주에 순서가 있는 경우에 적합합니다. 예를 들어, 교육 수준(예: 초등학교 < 중학교 < 고등학교 < 대학교)은 순서가 있는 범주형 변수입니다.
- 트리 기반 모델: 결정 트리, 랜덤 포레스트, XGBoost와 같은 트리 기반 모델은 label encoding된 변수를 더 잘 처리할 수 있습니다. 이는 트리 기반 모델이 변수 간의 순서를 처리할 수 있기 때문입니다.

Adult Income 데이터셋의 경우, 대부분의 범주형 변수는 순서가 없으므로 one-hot encoding을 사용하는 것이 적절

```
# 범주형 범수 인코딩
category_features = ['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'ra
data = pd.get_dummies(data, columns=category_features, drop_first=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 26197 entries, 1 to 32559
Data columns (total 97 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   age                                     26197 non-null  int64
 1   fnlwgt                                  26197 non-null  int64
 2   education-num                           26197 non-null  int64
 3   capital-gain                            26197 non-null  int64
 4   capital-loss                            26197 non-null  int64
 5   hours-per-week                          26197 non-null  int64
 6   capital-diff                            26197 non-null  int64
 7   workclass_Local-gov                     26197 non-null  bool
```

```
 8   workclass_Private                26197 non-null  bool
 9   workclass_Self-emp-inc           26197 non-null  bool
 10  workclass_Self-emp-not-inc       26197 non-null  bool
 11  workclass_State-gov              26197 non-null  bool
 12  workclass_Without-pay            26197 non-null  bool
 13  education_11th                   26197 non-null  bool
 14  education_12th                   26197 non-null  bool
 15  education_1st-4th                26197 non-null  bool
 16  education_5th-6th                26197 non-null  bool
 17  education_7th-8th                26197 non-null  bool
 18  education_9th                    26197 non-null  bool
```