Preprocess

✅ **Orange 3에서 Preprocess 위젯을 사용할지 여부는 데이터의 특성과 분석 목적에 따라 달라져!**

**📌 Preprocess 위젯을 사용할 필요가 있는 경우**

**1️⃣ 데이터 정리가 필요한 경우 (Data Cleaning)**

* 결측치(Missing Values) → Impute Missing Values
* 이상치(Outliers) 제거 → Remove Sparse Features
* 중복 데이터 제거

**2️⃣ 정규화(Normalization) 또는 표준화(Standardization)**

* 속성 값의 크기가 다르면 모델 학습이 어려울 수 있음
* 예: Min-Max Scaling, Z-score Normalization
* 특히 거리 기반 알고리즘(KNN, SVM 등)을 사용할 경우 **필수!**

**3️⃣ 특징 선택(Feature Selection) & 차원 축소(Dimensionality Reduction)**

* 정보가 적은 속성 제거 (Select Relevant Features)
* PCA(주성분 분석) 적용하여 차원 축소

**4️⃣ 텍스트 데이터 전처리 (Text Preprocessing)**

* 불용어(Stop Words) 제거, 소문자 변환, 표제어 추출(Lemmatization)
* Preprocess Text + Bag of Words

**📌 Preprocess 위젯 없이 진행할 수 있는 경우**

**1️⃣ 데이터가 이미 정리되어 있는 경우**

* 데이터셋에 결측치나 이상치가 없음
* 속성 값의 범위가 비슷하여 정규화가 필요 없음

**2️⃣ 모델이 자동으로 처리할 수 있는 경우**

* 트리 기반 모델(랜덤 포레스트, 결정 트리 등)은 데이터 정규화가 필요 없음

**3️⃣ 특징 선택이 필요하지 않은 경우**

* 데이터 속성이 적거나 모든 속성이 중요한 경우

**📊 결론: 언제 사용해야 할까?**

| **분석 목적** | **Preprocess 사용 여부** |
| --- | --- |
| KNN, SVM, 로지스틱 회귀 사용 | ✅ **정규화 필수** |
| 트리 기반 모델 (Random Forest, Decision Tree) | ❌ 필요 없음 |
| 결측치가 있는 경우 | ✅ 필요 |
| 텍스트 데이터 분석 | ✅ 필요 |
| 다중 공선성 문제 해결 (Feature Selection) | ✅ 필요 |
| PCA를 통한 차원 축소 | ✅ 필요 |

**👉 따라서 데이터의 특성과 분석 목적에 따라 Preprocess를 사용할지 결정해야 해!**

Preprocess 추가 설명

**📌 Min-Max Scaling vs. Z-score Normalization**

두 방법 모두 **데이터 정규화(Normalization)** 기법이지만, 적용 방식과 적합한 경우가 다름.

**1️⃣ Min-Max Scaling (최소-최대 정규화)**

✅ **설명:**

* 데이터 값을 **0과 1** 또는 특정 범위 [a,b][a, b][a,b]로 변환
* 최소값이 0, 최대값이 1이 되도록 변환
* **데이터 분포가 일정할 때**(예: 온도, 가격) 많이 사용

✅ **특징:**

* **이상치(Outliers)에 민감** → 이상치가 있으면 전체 스케일이 왜곡됨
* **데이터 범위를 유지** → 원래 데이터의 상대적인 크기가 유지됨

✅ **예제:**

| **원래 값** | **Min-Max Scaling 결과 (0~1)** |
| --- | --- |
| 10 | 0.0 |
| 15 | 0.5 |
| 20 | 1.0 |

✅ **적합한 경우:**  
✔ **이미 정해진 범위(0~1) 내에서 값이 필요할 때** (예: 신경망 입력)  
✔ **정확한 범위 유지가 필요한 경우**

**2️⃣ Z-score Normalization (표준화, Standardization)**

✅ **설명:**

* 데이터의 평균(𝜇)을 0, 표준편차(𝜎)를 1로 변환
* 즉, 데이터가 **평균을 중심으로 표준 정규분포**를 따르도록 조정

✅ **특징:**

* **이상치(Outliers)에 덜 민감**
* **데이터의 평균이 0, 표준편차가 1**이 됨 → 정규분포를 따르는 데이터에서 특히 효과적

✅ **예제:**

| **원래 값** | **Z-score 정규화 결과** |
| --- | --- |
| 10 | -1.0 |
| 15 | 0.0 |
| 20 | 1.0 |

✅ **적합한 경우:**  
✔ **데이터가 정규분포를 따를 때**  
✔ **거리 기반 알고리즘 (KNN, SVM, PCA 등) 사용 시**

**📌 차이점 요약**

|  | **Min-Max Scaling** | **Z-score Normalization** |
| --- | --- | --- |
| 변환 후 값 범위 | [0,1] (또는 [a,b][a,b][a,b]) | 평균=0, 표준편차=1 |
| 이상치 영향 | 크다 (영향 받음) | 적다 (영향 덜 받음) |
| 적합한 경우 | 신경망, 정해진 범위 필요할 때 | 거리 기반 알고리즘, 정규분포 데이터 |
| **차원 축소(Dimensionality Reduction)란?** 차원 축소는 **고차원 데이터를 저차원으로 변환하는 과정**으로, 데이터의 주요 정보를 최대한 유지하면서 불필요한 특성을 제거하는 방법이다. **1️⃣ 차원 축소의 필요성** ✅ **고차원 문제 (Curse of Dimensionality) 해결**   * 차원이 증가하면 데이터가 희소(Sparse)해지고, 학습이 어려워짐 * 거리 기반 알고리즘(KNN, SVM 등)에서 성능 저하 발생   ✅ **모델의 과적합(Overfitting) 방지**   * 불필요한 속성이 많으면 노이즈가 포함되어 과적합 가능성 증가 * 차원 축소를 통해 모델이 더 일반화됨   ✅ **계산 속도 및 저장 공간 절약**   * 데이터 차원이 줄어들면 **연산량 감소** * 메모리 사용량 줄이고 빠른 모델 학습 가능   ✅ **데이터 시각화 가능**   * 차원이 너무 높으면 시각화 어려움 * 2D 또는 3D로 변환하여 **산점도(Scatter Plot) 등으로 표현 가능**  **2️⃣ 차원 축소 기법 종류** 차원 축소 방법은 \*\*특징 선택(Feature Selection)\*\*과 **특징 추출(Feature Extraction)** 두 가지로 나뉜다. **(1) 특징 선택 (Feature Selection)** → 중요한 속성만 골라서 사용 ✅ ****1. 필터 방법 (Filter Methods)****  * 데이터의 통계적 속성을 이용하여 중요한 속성 선택 * **방법:**   + **분산 기반 선택** (Low Variance Filter) → 분산이 낮은 특성 제거   + **상관관계 기반 선택** (Correlation Coefficient) → 상관관계가 높은 특성 제거  ✅ ****2. 래퍼 방법 (Wrapper Methods)****  * 모델을 학습시켜 가장 성능이 좋은 특성을 선택 * **방법:**   + **Recursive Feature Elimination (RFE)** → 중요하지 않은 속성을 하나씩 제거   + **Forward/Backward Selection** → 하나씩 추가/제거하며 성능 비교  ✅ ****3. 임베디드 방법 (Embedded Methods)****  * 모델 자체가 중요한 특성을 자동으로 선택 * **예제:**   + **L1 정규화(Lasso Regression)**   + **랜덤 포레스트(Random Forest) 중요도 기반 선택**  **(2) 특징 추출 (Feature Extraction)** → 기존 데이터를 새로운 저차원 데이터로 변환 ✅ ****1. 주성분 분석 (PCA, Principal Component Analysis)**** ✔ **핵심 개념:**   * 데이터의 분산을 가장 잘 설명하는 방향(주성분, Principal Components)으로 변환 * 고차원의 데이터를 저차원 공간으로 축소   ✔ **적용 예시:**   * **100개의 특성을 가진 데이터 → 2~3개 주성분으로 변환 가능** * **이미지 처리, 음성 분석, 금융 데이터에서 많이 사용**   ✔ **Orange 3에서 PCA 실행 방법:**   * Preprocess 위젯에서 Principal Component Analysis (PCA) 선택 * 원하는 차원 수 설정 후 적용  ✅ ****2. 선형판별분석 (LDA, Linear Discriminant Analysis)**** ✔ **핵심 개념:**   * **클래스 간 분리를 극대화하는 방향으로 변환** * **PCA와 다르게 지도학습(Supervised Learning) 기반** * 분류(Classification) 문제에서 효과적   ✔ **예제:**   * **얼굴 인식(Face Recognition)** * **텍스트 분류(Text Classification)**   ✔ **차이점:**   * **PCA**는 분산을 극대화, **LDA**는 클래스 분리를 극대화  ✅ ****3. t-SNE (t-Distributed Stochastic Neighbor Embedding)**** ✔ **핵심 개념:**   * 고차원 데이터를 **2D 또는 3D로 시각화하는 데 특화된 방법** * 데이터 포인트 간의 관계를 유지하며 차원 축소   ✔ **예제:**   * **클러스터링(Clustering)에서 데이터 구조 시각화** * **자연어 처리(NLP)에서 단어 벡터 시각화**   ✔ **Orange 3에서 t-SNE 실행 방법:** 1️⃣ t-SNE 위젯 추가 2️⃣ 데이터 입력 후 Number of components = 2 or 3 설정 3️⃣ **시각화(Scatter Plot) 연결하여 결과 확인** **3️⃣ 차원 축소 기법 비교**  | **기법** | **설명** | **적용 방식** | **지도/비지도** | | --- | --- | --- | --- | | **PCA** | 데이터의 분산을 최대한 유지하며 차원 축소 | 선형 변환 | 비지도 | | **LDA** | 클래스 간 분리를 최대화하는 방향으로 변환 | 선형 변환 | 지도 | | **t-SNE** | 데이터 포인트 간 거리 유지하며 저차원 표현 | 비선형 변환 | 비지도 | | **Feature Selection** | 중요 속성만 선택하여 차원 축소 | 직접 선택 | 지도/비지도 |   **L1, L2 정규화 (Regularization)란?**  정규화(Regularization)는 **모델의 복잡도를 줄이고 과적합(Overfitting)을 방지**하는 기법이다.   * **L1 정규화 (Lasso, Least Absolute Shrinkage and Selection Operator)**   + 불필요한 특성(Feature)을 0으로 만들어 변수 선택(Feature Selection) 역할 수행 * **L2 정규화 (Ridge Regression)**   + 가중치를 0에 가깝게 만들어 모델의 복잡도를 낮춤 |  |  |

### ****Orange3에서 Preprocessing(전처리)이 필요한지 판별하는 방법****

데이터 분석 과정에서 전처리가 필요한지 판단하는 것은 **데이터의 특성, 분석 목적, 모델의 요구사항**에 따라 달라진다. Orange3에서는 아래 과정을 통해 전처리 필요 여부를 판별할 수 있다.

## **✅ 1. Data Table 위젯으로 데이터 확인**

**👉 가장 기본적인 방법으로, 원본 데이터의 상태를 직접 확인하는 과정**  
1️⃣ **Data Table 위젯을 추가**  
2️⃣ **데이터 파일(csv, xlsx 등)을 File 위젯으로 불러온 후 연결**  
3️⃣ **데이터를 열어서 아래 문제점이 있는지 확인**

🚨 **Preprocessing(전처리)이 필요한 경우**

* 결측값(Missing Values)이 있는가?
* 특성(Feature) 간 값의 범위가 다르거나 이상치(Outliers)가 존재하는가?
* 명목형(Nominal) 데이터가 많아서 숫자로 변환이 필요한가?
* 데이터가 너무 많거나 차원이 높은가? (Dimensionality Reduction 필요)
* 특성이 많지만 중요하지 않은 변수가 있는가? (Feature Selection 필요)

**📌 만약 위 문제점이 발견된다면, Preprocess 위젯을 사용해야 한다!**

## **✅ 2. Feature Statistics 위젯으로 데이터 분포 확인**

**👉 속성(Feature)들의 통계값을 확인하고 전처리 필요성을 분석**  
1️⃣ **Feature Statistics 위젯 추가 후 데이터 연결**  
2️⃣ **각 속성의 평균(Mean), 분산(Variance), 최소-최대 값(Min-Max) 등을 확인**

🚨 **Preprocessing이 필요한 경우**

* 특정 속성의 값이 한쪽으로 편향된 경우 (Log Transformation 필요)
* 분산이 낮은 속성이 있는 경우 (Feature Selection 필요)
* 데이터 값의 범위가 큰 차이를 보이는 경우 (Normalization 필요)

✅ **분석 결과 정규화(Normalization) 또는 특성 선택(Feature Selection)이 필요하면 Preprocess 위젯을 추가**

## **✅ 3. Distributions & Box Plot으로 데이터 이상치 확인**

**👉 데이터 이상치를 찾고, 제거해야 하는지 판단**  
1️⃣ **Distributions 또는 Box Plot 위젯 추가**  
2️⃣ **속성을 선택하여 데이터 분포 확인**

🚨 **Preprocessing이 필요한 경우**

* 특정 속성이 극단적인 이상치(Outliers)를 포함하는 경우
* 데이터가 한쪽으로 치우친 경우 (정규화 필요)

✅ **이상치가 많다면 Impute Missing Values 또는 Remove Outliers 기능 사용**

## **✅ 4. Correlations 위젯으로 변수 간 관계 확인**

**👉 속성 간 상관관계를 분석하여 차원 축소(Dimensionality Reduction) 필요성 판단**  
1️⃣ **Correlations 위젯 추가 후 데이터 연결**  
2️⃣ **Pearson 또는 auc 상관계수를 확인**

🚨 **Preprocessing이 필요한 경우**

* 상관계수가 높은 속성이 많으면 다중공선성(Multicollinearity) 문제 발생 → 차원 축소 필요
* 독립 변수가 너무 많고 상관성이 낮으면, PCA(Principal Component Analysis) 활용 가능

✅ **PCA 또는 Feature Selection 적용 필요 여부를 판단할 수 있음**

## **📌 최종 정리: Orange3에서 전처리가 필요한 경우를 판별하는 과정**

| **전처리 필요 여부** | **사용 위젯** | **주요 확인 사항** | **해결 방법** |
| --- | --- | --- | --- |
| **결측값 확인** | Data Table | 값이 비어 있는지 확인 | Impute Missing Values |
| **데이터 범위 차이** | Feature Statistics | 속성 간 값의 범위 비교 | Normalize Features |
| **이상치(Outliers) 존재** | Box Plot / Distributions | 극단적인 값 확인 | Remove Outliers |
| **차원 축소 필요** | Correlations | 높은 상관계수 확인 | PCA 또는 Feature Selection |
| **텍스트 데이터** | Data Table | 문자열(텍스트) 포함 여부 | Preprocess Text |

➡ **위 과정을 거쳐 전처리가 필요하면, Preprocess 위젯을 추가하여 정리하면 된다!** 🚀

### ****🔍 Test and Score vs Prediction 차이점 & 사용법****

Orange3에서 **Test and Score**와 **Prediction** 위젯은 모두 모델의 성능을 평가하는 데 사용되지만, 목적과 적용 방식이 다릅니다.

## **📌 Test and Score**

✅ **주요 역할:**

* **모델의 성능 평가**
* **교차 검증(Cross Validation)**, **훈련-테스트 분할(Train-Test Split)** 등 사용
* **정확도(Accuracy), AUC, F1-score, Precision, Recall** 등 다양한 성능 지표 제공

✅ **적합한 상황:**

* **모델을 선택하고 성능을 비교할 때**
* **모델이 과적합(Overfitting) 여부를 판단할 때**
* **훈련 데이터와 테스트 데이터를 나누어 성능을 평가할 때**
* **교차 검증을 통해 모델의 일반화 성능을 확인할 때**

✅ **사용 예시:**

1. 데이터 샘플링 후, **훈련 데이터(Train)와 테스트 데이터(Test)로 분할**
2. **모델(예: Logistic Regression, SVM, Random Forest 등)을 연결**
3. **Test and Score 위젯을 사용하여 모델의 성능을 비교**

## **📌 Prediction**

✅ **주요 역할:**

* **새로운 데이터에 대해 모델이 예측한 결과를 확인**
* 모델이 학습한 내용을 바탕으로 **새로운 데이터에 대한 출력값(예측값)을 생성**
* **실제 데이터와 비교하여 개별 예측 오류를 분석할 수 있음**

✅ **적합한 상황:**

* **새로운 데이터가 들어왔을 때 모델이 예측한 결과를 확인할 때**
* **회귀(Regression) 또는 분류(Classification) 모델이 실제로 어떻게 예측하는지 볼 때**
* **특정 데이터가 어떤 클래스로 분류되는지 확인할 때**
* **회귀 모델에서 예측값과 실제값의 차이를 분석할 때**

✅ **사용 예시:**

1. **학습된 모델(Logistic Regression, Decision Tree 등)을 Prediction 위젯과 연결**
2. **새로운 데이터셋을 연결하여 예측 수행**
3. **결과를 확인하고 실제값과 비교**

## **📌 Test and Score vs Prediction 정리**

| **구분** | **Test and Score** | **Prediction** |
| --- | --- | --- |
| **목적** | 모델 성능 평가 | 새로운 데이터 예측 |
| **사용 방식** | 교차 검증, Train-Test 평가 | 모델 학습 후 새로운 데이터 적용 |
| **출력** | AUC, Precision, Recall, F1-score 등 성능 지표 | 예측된 클래스(Label) 또는 값(Regression) |
| **적용 상황** | 여러 모델 비교, 과적합 확인 | 실제 데이터에 대한 모델 적용 |
| **예제** | 모델 성능 비교 (예: Logistic Regression vs Random Forest) | 고객의 구매 여부 예측 |

## **📌 어떤 경우에 어떤 위젯을 사용해야 할까?**

* **모델을 검증하고 성능을 평가**하고 싶다면 → ✅ **Test and Score**
* **새로운 데이터에 대해 예측을 수행**하고 싶다면 → ✅ **Prediction**

## **📌 결론**

👉 **Test and Score**는 **모델이 얼마나 좋은지 평가하는 용도**  
👉 **Prediction**은 **새로운 데이터를 모델에 적용해 결과를 얻는 용도**

즉, 모델을 만들고 **Test and Score**에서 성능이 좋으면 **Prediction**을 사용해 실제 데이터에 적용하는 것이 일반적인 과정입니다! 🚀

### ****📌 과적합(Overfitting) 판단을 위한 주요 수치****

### ****과적합 – 모델이 훈련 데이터 만큼 테스트 데이터, 새로운 데이터에도 좋은 성능을 유지하는지.****

과적합을 확인하려면 **훈련 데이터(Training)와 테스트 데이터(Test)에서의 성능 차이**를 비교하는 것이 핵심입니다. 아래 지표들을 활용해 과적합 여부를 판단할 수 있습니다.

## **✅ 1. 훈련 데이터 vs 테스트 데이터 성능 비교**

훈련 데이터에서 **성능이 매우 높지만, 테스트 데이터에서 성능이 낮다면** 과적합이 발생한 것입니다.

### ****🔹 Test and Score 결과에서 확인 가능****

* **정확도(Accuracy)**
* **AUC (Area Under Curve)**
* **F1-score**
* **Precision (정밀도) / Recall (재현율)**
* **MSE / RMSE (회귀 모델의 경우)**

**💡 과적합 가능성이 높은 경우:**  
✅ **훈련 데이터(Train) 성능이 높은데, 테스트 데이터(Test) 성능이 낮다**  
✅ 예)

* **Train 정확도: 95%**
* **Test 정확도: 75%** → 과적합 가능성 높음 🚨

## **✅ 2. Cross Validation 성능 변동 확인**

**교차 검증(K-Fold Cross Validation)을 수행한 후 Fold 간 성능 차이를 확인**

* **Test and Score에서 Cross Validation 설정 후 실행**
* **Fold별 성능 편차가 크다면 모델이 특정 데이터에 과적합되었을 가능성이 있음**

**💡 과적합 가능성이 높은 경우:**  
✅ Fold별 성능 차이가 심하다  
✅ 예)

* **Fold 1 정확도: 95%**
* **Fold 2 정확도: 70%** → 모델이 특정 데이터에 과적합 🚨

## **✅ 3. 학습 곡선(Training Curve) 분석**

훈련 데이터와 테스트 데이터의 **손실값(Loss)** 변화를 확인하는 방법

🔹 **훈련 데이터 손실이 급격히 낮아지는데, 테스트 손실이 감소하지 않으면 과적합 가능성**  
🔹 **Epoch이 증가할수록 성능이 나빠지는 경우 조기 종료(Early Stopping) 필요**

## **✅ 4. Regularization(L1/L2 정규화) 적용 후 성능 비교**

* **L1 (Lasso) 정규화**: 불필요한 변수 제거 → 모델 단순화
* **L2 (Ridge) 정규화**: 가중치(W) 크기 줄이기 → 모델 복잡도 감소
* **정규화 적용 전후 성능 변화를 확인**

**💡 과적합 가능성이 높은 경우:**  
✅ 정규화 적용 전: **Train 정확도 98%, Test 정확도 70%**  
✅ 정규화 적용 후: **Train 정확도 85%, Test 정확도 83%** → **일반화 성능 개선됨** 🚀

## **📌 결론: 과적합 여부 판단 기준 정리**

| **지표** | **과적합 가능성이 높은 경우 🚨** |
| --- | --- |
| **Train-Test 성능 차이** | Train 성능이 높고 Test 성능이 낮음 |
| **Cross Validation 편차** | Fold별 성능 차이가 큼 |
| **학습 곡선(Loss Curve)** | Train Loss는 감소, Test Loss는 정체 or 증가 |
| **정규화 적용 전후 비교** | 정규화 후 Train-Test 성능 차이가 줄어듦 |

✅ **Test and Score에서 Train vs Test 성능을 비교하는 것이 가장 직관적인 방법**  
✅ **Cross Validation을 활용해 일반화 성능 확인**  
✅ **필요하면 L1/L2 정규화 적용하여 모델 단순화**

👉 위 기준을 활용해서 **Orange3에서 모델이 과적합되었는지 확인**하고 **적절한 조치를 취하면 모델의 일반화 성능을 향상시킬 수 있습니다!** 🚀

## **Orange3의 Discretize 위젯 기능 설명**

**Discretize 위젯**은 **연속형(Continuous) 데이터를 범주형(Categorical) 데이터로 변환(Discretization, 구간화)** 하는 기능을 제공합니다.  
즉, **숫자로 표현된 연속적인 값들을 여러 개의 구간(bin)으로 나누어 범주화하는 역할**을 합니다.

**💡 언제 사용해야 할까?**  
✅ 연속형 변수(예: 나이, 소득, 온도)를 범주형으로 변환하여 분류 모델에서 활용하고 싶을 때  
✅ 데이터의 분포를 분석하기 위해 값의 범위를 나누고 싶을 때  
✅ 결합분석(Association Rule Mining) 등에서 연속형 변수를 범주형으로 변환할 필요가 있을 때

## **🔹 Discretize 위젯의 주요 기능**

### ****✅ 1. 자동 구간화(Auto Binning)****

📌 Orange3에서는 자동으로 연속형 변수를 여러 개의 범주로 나눌 수 있습니다.  
**구간을 나누는 기준(방법) 선택 가능**

1️⃣ **Equal-Width (동일 간격 구간화)**

* 데이터의 최소값과 최대값을 기준으로 **동일한 너비(Width)** 로 구간을 나눔
* 예: 020, 20~30
* **데이터가 균등하게 분포할 때 적합**

2️⃣ **Equal-Frequency (동일 빈도 구간화)**

* 각 구간에 동일한 개수의 데이터를 포함하도록 구간을 설정
* **데이터가 비대칭적(편향)일 때 유용**

3️⃣ **Entropy-Based (엔트로피 기반 구간화)**

* 정보 엔트로피를 최소화하는 방식으로 구간을 설정
* **결정 트리(Decision Tree)와 같은 모델에서 유용**

4️⃣ **Manual (사용자 지정 구간화)**

* 사용자가 직접 범위를 지정하여 구간을 나눌 수 있음
* 특정 기준에 맞춰 원하는 대로 범주를 설정 가능

### ****✅ 2. Discretization 적용 과정****

1️⃣ **Discretize 위젯을 추가하고 연속형 변수를 선택**  
2️⃣ **적절한 구간화 방법(Equal-Width, Equal-Frequency 등) 선택**  
3️⃣ **변환된 데이터를 Data Table 또는 시각화 위젯(Scatter Plot, Box Plot)에서 확인**

### ****🔹 Discretization 예제****

**예제: 나이(Age) 데이터를 구간화하기**  
연속형 변수 **나이(Age: 0~100세)를 3개 그룹으로 구간화**

| **Age (연속형)** | **Discretized Age (범주형)** |
| --- | --- |
| 5 | 어린이 (0~18) |
| 22 | 청년 (19~40) |
| 65 | 성인 (41~100) |

➡ **이제 "나이" 변수가 범주형 데이터로 변환됨!** 🎯

## **📌 결론: Discretize 위젯 활용 요약**

✅ 연속형 변수를 범주형으로 변환하여 분류 모델에서 활용 가능  
✅ 다양한 구간화 방법(Equal-Width, Equal-Frequency 등) 제공  
✅ 데이터 분포에 맞는 적절한 구간화를 선택하여 분석 정확도 향상

🔹 **💡 Orange3에서 Discretize 위젯을 사용하면 연속형 데이터를 범주형으로 변환하여 더 효과적인 분석이 가능!** 🚀

## **Orange3의 Edit Domain 위젯 기능 설명**

**Edit Domain 위젯**은 데이터셋의 **변수(특성, 속성) 정보를 수정**하는 기능을 제공합니다.  
즉, **변수의 타입을 변경하거나, 레이블을 수정하거나, 불필요한 특성을 제거할 때 유용**합니다.

## **🔹 Edit Domain 위젯의 주요 기능**

### ****✅ 1. 변수 유형 변경 (Change Variable Type)****

📌 변수의 데이터 타입을 변경할 수 있습니다.

* **연속형(Continuous) ↔ 범주형(Categorical) 변환 가능**
  + 예: 숫자로 표현된 등급(1, 2, 3)을 **범주형 값("Low", "Medium", "High")** 로 변경 가능
  + 예: 범주형 값("Yes", "No")을 **0, 1의 숫자형 값으로 변경 가능**
* **텍스트(Text) 데이터 타입 변환**
  + 예: 원래는 메타 데이터로 분류된 텍스트 데이터를 범주형 또는 연속형으로 변환 가능

### ****✅ 2. 변수 이름 수정 (Rename Attributes)****

📌 변수의 이름을 원하는 대로 변경할 수 있습니다.

* 예: var\_1 → customer\_age
* 예: attr\_5 → income\_category

➡ **데이터의 가독성을 높이고 분석을 용이하게 함**

### ****✅ 3. 변수 역할 변경 (Change Attribute Role)****

📌 Orange3에서는 변수를 **"특성(Feature), 레이블(Target), 메타(Meta)"** 등의 역할로 지정할 수 있습니다.

* **Feature (특성)**: 모델이 학습할 입력 값
* **Target (레이블/목표 변수)**: 예측해야 하는 값 (ex: 가격, 카테고리)
* **Meta (메타 데이터)**: 분석에는 사용되지 않지만 정보로 저장 (ex: 데이터 ID, 설명 텍스트)

➡ **분석 모델을 만들 때 타겟 변수를 지정할 때 유용**

### ****✅ 4. 변수 삭제 및 선택 (Select or Remove Attributes)****

📌 불필요한 변수(특성)를 제거하거나 필요한 변수만 선택할 수 있습니다.

* 예: 데이터셋에서 customer\_id 같은 ID 변수는 분석에 필요 없으므로 제거 가능
* 예: 모델 학습에 불필요한 메타 데이터를 제거 가능

➡ **불필요한 변수를 제거하여 데이터셋을 정리하고 분석 성능을 향상**

### ****✅ 5. 범주형 데이터 수정 (Edit Categorical Values)****

📌 범주형 데이터의 값(label)을 수정할 수 있습니다.

* 예:
  + "male" → "M", "female" → "F" 로 변경 가능
  + "low", "medium", "high"를 "L", "M", "H"로 변경 가능

➡ **데이터를 더 깔끔하게 정리하고 분석을 쉽게 함**

## **🔹 Edit Domain 위젯 사용법**

1️⃣ **Edit Domain 위젯을 추가하고 데이터와 연결**  
2️⃣ **변수 유형(Type), 역할(Role), 이름(Name) 등을 변경**  
3️⃣ **필요 없는 변수를 삭제하고 필요한 변수만 선택**  
4️⃣ **변경된 데이터셋을 다른 위젯(Data Table, Scatter Plot 등)에서 활용**

## **📌 결론: Edit Domain 위젯 활용 요약**

✅ 변수의 데이터 타입(연속형 ↔ 범주형) 변경 가능  
✅ 변수 이름을 수정하여 가독성 향상  
✅ 타겟 변수(Target) 설정 가능  
✅ 불필요한 변수를 제거하여 데이터셋 최적화  
✅ 범주형 데이터(Label) 수정 가능

**💡 Edit Domain 위젯을 활용하면 데이터 전처리를 더욱 쉽게 수행할 수 있음! 🚀**

**📌 Orange 3에서 트리 가지치기(Pruning) 적용 방법**

**트리 모델(Decision Tree)은 가지치기(Pruning)를 통해 불필요한 분기를 제거하여 과적합(Overfitting)을 방지할 수 있습니다.**  
Orange 3에서는 **"Tree" 위젯의 설정 옵션**을 조정하여 가지치기를 적용할 수 있습니다.

**🔹 1️⃣ Tree 위젯에서 가지치기 설정하기**

1️⃣ **Tree 위젯을 추가**

* **"Learner"** 카테고리에서 **"Tree"** 위젯을 캔버스로 끌어옵니다.
* 기존의 **Test & Score** 또는 **Prediction**과 연결합니다.

2️⃣ **Tree 위젯을 더블 클릭하여 설정 열기**

3️⃣ **가지치기(Pruning) 관련 옵션 조정**

* **"Limit depth of tree" (트리 깊이 제한)** → 기본값을 낮게 설정 (예: 5~10)
  + 너무 깊으면 과적합 가능성이 증가
* **"Limit number of leaves" (리프 노드 개수 제한)** → 적절한 값 설정 (예: 10~20)
  + 너무 많은 리프 노드는 데이터에 과적합
* **"Minimum instances in leaves" (최소 샘플 수 제한)** → 값 증가 (예: 5~10)
  + 최소 샘플 수가 작으면 트리가 과적합됨

4️⃣ **설정 저장 후 "Apply Automatically" 체크**

* 자동 적용하여 결과를 바로 확인

**🔹 2️⃣ Test & Score로 성능 확인하기**

트리 가지치기 적용 후 **"Test & Score"** 위젯을 사용하여 성능이 어떻게 변했는지 확인합니다.

* **Accuracy, F1-score, AUC 값**을 비교하여 가지치기가 효과적인지 판단
* 만약 성능이 너무 떨어진다면 가지치기 강도를 낮춤

**📌 결론**

✅ Orange 3에서는 **Tree 위젯에서 가지치기 설정**을 조정하여 과적합 방지 가능  
✅ 가지치기 후 **Test & Score로 성능 평가**하여 최적의 설정을 찾는 것이 중요  
✅ 필요하면 **Random Forest** 같은 앙상블 모델을 활용하여 성능 향상 🚀

**📌 추가로, 트리 모델의 성능이 여전히 좋지 않다면, 다른 모델(예: Random Forest, SVM)도 고려해볼 수 있습니다.** 🎯

양식의 맨 위

양식의 맨 아래

**📌 과적합(Overfitting)의 반대 개념: 과소적합(Underfitting)**

✅ **과소적합(Underfitting)이란?**  
모델이 **데이터의 패턴을 충분히 학습하지 못해서 성능이 낮은 경우**를 의미합니다.  
즉, 모델이 너무 단순해서 **훈련 데이터뿐만 아니라 테스트 데이터에서도 성능이 낮음**.

**🔹 과적합(Overfitting) vs. 과소적합(Underfitting) 비교**

|  | **과소적합(Underfitting)** | **적절한 학습(Balanced Fit)** | **과적합(Overfitting)** |
| --- | --- | --- | --- |
| **설명** | 모델이 너무 단순해서 데이터 패턴을 학습하지 못함 | 일반화 성능이 좋은 최적의 상태 | 모델이 너무 복잡해서 훈련 데이터에 과하게 맞춤 |
| **훈련 데이터 성능** | 낮음 | 적절함 | 매우 높음 |
| **테스트 데이터 성능** | 낮음 | 적절함 | 낮음 (일반화 부족) |
| **해결 방법** | 더 복잡한 모델 사용, 특징 추가 | 현재 상태 유지 | 모델 단순화, 규제(Regularization) 적용 |

**🔹 Orange 3에서 과소적합(Underfitting) 방지 방법**

1️⃣ **더 복잡한 모델 사용**

* 선형 회귀 대신 **다항 회귀(Polynomial Regression)**
* 단순 결정 트리 대신 **랜덤 포레스트(Random Forest) 또는 부스팅(Boosting) 모델**

2️⃣ **더 많은 특징(Feature) 추가**

* 중요한 변수를 추가하여 모델이 더 많은 패턴을 학습하도록 유도

3️⃣ **학습 데이터 증가**

* 데이터 양이 부족하면 학습이 부족할 수 있음
* 데이터 증강(Augmentation) 또는 추가 데이터 수집

4️⃣ **Regularization(정규화) 조절**

* 과소적합이 발생했다면, **L1 또는 L2 정규화의 강도를 낮춰** 모델이 더 많은 패턴을 학습하도록 유도

**📌 결론**

* 과적합: 모델이 너무 복잡해서 훈련 데이터에 과하게 맞춤 → **일반화 성능↓**
* 과소적합: 모델이 너무 단순해서 패턴을 학습하지 못함 → **전체 성능↓**
* Orange 3에서는 **트리 깊이 조절, 정규화, 모델 변경** 등을 통해 해결 가능 🚀

📌 **즉, 목표는 "적절한 학습(Balanced Fit)"을 찾는 것!**  
**과적합과 과소적합 사이에서 최적의 모델을 찾는 것이 중요합니다.** 🎯

## **🔍 특정 클래스만 적중률이 높은 경우, 고려해야 할 점**

### ****1️⃣ 과적합(Overfitting) 가능성****

* 훈련 데이터에서 **특정 클래스의 패턴을 과하게 학습**해서 다른 클래스의 예측 성능이 낮을 수 있음.
* 특히 **트리 모델은 깊이가 너무 깊어지면 과적합되기 쉬움.**  
  ▶ **해결 방법**:
  + **트리 가지치기(Pruning)** 적용
  + **트리의 깊이 제한(Max Depth 설정)**
  + **랜덤 포레스트(Random Forest)나 부스팅(Boosting) 모델 사용**

### ****2️⃣ 데이터 불균형(Imbalanced Data) 문제****

* 특정 클래스의 데이터 수가 다른 클래스보다 **훨씬 많으면** 모델이 **다수 클래스를 더 잘 맞추게 됨.**  
  ▶ **해결 방법**:
  + \*\*SMOTE(Synthetic Minority Over-sampling Technique)\*\*로 데이터 균형 조정
  + **클래스 가중치(Class Weight) 조정**

### ****3️⃣ 모델이 특정 클래스의 특징을 더 잘 학습한 경우****

* 특정 클래스의 데이터가 더 분명한 패턴을 가지면 해당 클래스에서 높은 정확도가 나올 수 있음.
* 예를 들어, 자동차 가격을 예측하는 경우, **고가 차량의 특징이 더 명확하면 "고가 차량" 분류가 더 잘될 수도 있음.**  
  ▶ **해결 방법**:
  + 다른 **특징(Feature) 추가**하여 모델의 일반화 성능을 향상
  + **다른 알고리즘 사용 (예: SVM, 신경망)**

## **🛠 Orange 3에서 문제 해결 방법**

🔹 **트리 모델 과적합 조정** 1️⃣ **Pruning (가지치기) 적용**

* **Tree 위젯 설정**에서 **"Pruning Size", "Max Depth" 조절**
* 가지치기를 통해 불필요한 분기를 제거하여 일반화 성능 향상

2️⃣ **랜덤 포레스트(Random Forest) 적용**

* 단일 트리 대신 여러 개의 트리를 조합하면 과적합 감소

3️⃣ **테스트 데이터 정확도 확인**

* **Test & Score 위젯**에서 **훈련 정확도와 테스트 정확도 비교**
* 테스트 성능이 낮다면 과적합 가능성이 있음

🔹 **데이터 불균형 문제 해결** 1️⃣ **데이터 샘플링 조정 (SMOTE)**

* Orange 3의 **"Impute Missing Values"** 또는 **"Select Rows"** 활용

2️⃣ **클래스 가중치 설정**

* 모델 학습 시, 적은 클래스에 더 많은 가중치 부여

### ****📌 결론****

✔ 특정 클래스의 적중률이 높을 때, **반드시 과적합이라고 단정할 수 없음.**  
✔ **데이터 불균형, 모델 특성 등을 함께 고려하여 판단**해야 함.  
✔ **해결 방법:** 트리 가지치기(Pruning), 랜덤 포레스트 활용, 데이터 균형 조정 등 🚀

💡 **Orange 3에서 먼저 "Confusion Matrix"와 "Test & Score"를 확인한 후, 과적합 여부를 분석하는 것이 중요합니다!**

Classification 분석은 tree or logistic regression.

Continuous or numeric target value의 경우 linear regression and treeㄱ