문자열 기반 피처를 클러스터링에 활용하려면, 이를 수치화하거나 텍스트 데이터를 분석에 적합한 형태로 변환하는 것이 필요합니다. 몇 가지 접근법을 제안합니다:

1. **카테고리형 문자열 인코딩**:
   * 문자열 값이 카테고리(예: "http\_method", "http\_host" 등)라면 **One-Hot Encoding**이나 **Label Encoding**을 적용할 수 있습니다.
   * 문자열 값이 비교적 적고 명확한 경우 이 방법이 적합합니다.
2. **임베딩 (Embedding)**:
   * 문자열 데이터가 복잡하거나 의미를 포함하는 경우, **Word2Vec**, **GloVe**, 또는 **Sentence Transformers**를 사용해 벡터화할 수 있습니다.
   * 예: http\_uri나 dns\_query\_name처럼 URL이 포함된 필드에 유용.
3. **문자열 길이 및 패턴 기반 특성화**:
   * 문자열의 길이, 특정 문자(예: "/")의 빈도, 도메인 여부 등을 수치적 피처로 변환.
   * dns\_query\_name의 경우, 도메인 수나 하위 도메인 구조를 분석하여 추가 정보를 생성할 수 있습니다.
4. **TF-IDF 벡터화**:
   * 문자열이 텍스트 데이터에 가까운 경우(예: 긴 http\_uri), **TF-IDF**를 사용해 텍스트의 중요도를 계산한 후 벡터화.
5. **해시 기반 인코딩**:
   * 고차원 데이터를 효율적으로 처리하려면 **Feature Hashing**을 사용할 수 있습니다. 이 방식은 공간 효율적이며 대규모 문자열 데이터를 처리하기에 적합합니다.
6. **도메인 지식 활용**:
   * 특정 문자열의 도메인 특성을 기반으로 의미 있는 수치를 파생합니다.
   * 예: http\_code는 상태 코드로, 정상/공격 판단에 직접적으로 기여할 가능성이 큽니다.

**1. Categorical Data Embedding**

* **적용 대상**: http\_method, dns\_query\_type, dns\_query\_class, 등과 같이 유한한 범주를 가진 문자열.
* **방법**:
  + **Learned Embeddings**: 각 카테고리를 고유 임베딩 벡터로 학습. (예: 임베딩 레이어 사용)
  + **One-Hot Encoding + Dense Layer**: One-hot 벡터를 신경망으로 압축하여 학습된 벡터 생성.

**2. Sequence-Based Embedding (Natural Language)**

* **적용 대상**: http\_uri, dns\_query\_name, 등 긴 문자열이나 텍스트 데이터.
* **방법**:
  + **Pre-trained Models**:
    - Word2Vec, GloVe, FastText: 단어 단위 임베딩.
    - Sentence Transformers: 문장 또는 긴 텍스트의 컨텍스트 임베딩.
  + **TF-IDF + PCA**: 텍스트를 TF-IDF로 변환한 후 차원을 축소해 클러스터링에 적합한 형태로 변환.
  + **Custom Tokenization**: 도메인 특화된 토큰화 방식을 설계. 예를 들어, http\_uri의 경로/쿼리/도메인 분리.

**3. Graph-Based Embedding**

* **적용 대상**: src\_port, dst\_port, sa, da 같은 네트워크 연결 정보를 포함한 데이터.
* **방법**:
  + **Node2Vec**: 네트워크 내 노드의 관계를 기반으로 벡터화.
  + **DeepWalk**: 그래프 상에서 임의의 워크를 사용해 노드 임베딩 학습.
  + **Graph Neural Networks (GNN)**: 연결 정보를 포함한 그래프를 학습하여 노드 간의 구조적 특성 추출.

**4. Domain-Specific Embedding**

* **적용 대상**: tls\_cs, dns\_query\_name, tls\_ext\_types, 등 네트워크 프로토콜 관련 문자열.
* **방법**:
  + **Protocol-Specific Parsing**:
    - dns\_query\_name → 도메인 분리 (레벨 수, TLD, 길이 등).
    - tls\_ext\_types → 확장 타입 개수와 분포를 임베딩.
  + **Manual Feature Engineering**: 문자열을 파싱해 네트워크 트래픽 특성을 도출하고 임베딩.

**5. Autoencoder-Based Embedding**

* **적용 대상**: 고차원 문자열 또는 복합적인 피처.
* **방법**:
  + **Variational Autoencoders (VAEs)**: 문자열을 벡터로 변환한 후 잠재 공간에서 임베딩.
  + **Sequence Autoencoders**: LSTM/GRU를 사용해 문자열 시퀀스를 벡터로 학습.

**6. Hashing Vectorizer**

* **적용 대상**: dns\_query\_name, http\_uri, 등 무작위 문자열.
* **방법**:
  + **Feature Hashing**: 문자열을 고정 크기의 해시 벡터로 변환. 메모리 효율적이며 대규모 데이터에 적합.

**7. End-to-End Deep Learning Embedding**

* **적용 대상**: 전체 데이터.
* **방법**:
  + 모델 학습 과정에서 문자열 피처를 임베딩 레이어에 연결하여 학습. 예를 들어:
    - 입력: 문자열 데이터 (임베딩 레이어)
    - 출력: 공격/정상 예측 또는 클러스터링 피처.

**임베딩 결과 활용**

* 임베딩한 벡터는 다른 수치 데이터와 결합해 PCA 또는 t-SNE 같은 차원 축소 기법을 적용.
* 클러스터링 알고리즘으로 K-Means, DBSCAN, Gaussian Mixture Model 등을 활용.