<처음보는 용어집>

1. 데이터 사이언스 : 다양한 데이터로부터 지식과 인사이트를 추출하기 위해 과학적 방법론, 프로세스, 알고리즘, 시스템을 사용하는 융합분야

2. 데이터 마이닝(Data Mining) : 대규모 데이터에서 유용한 패턴, 규칙, 관계를 발견하고 추출하는 것

3. 독립변수 : 종속변수에 영향을 미치는, 조작 변경 가능한 변수로 머신러닝의 y=ax+b 에서 x에 해당됨
   x는 feature 혹은 input 변수라고도 불림

4. interpreter 언어 : 컴파일 단계 없이 코드를 한 줄씩 읽고 바로 실행하는 언어로 메모리 호출이 빠름 (출처 : 위키백과)

5. 동적 타이핑(Dynamic Typing) : 프로그래밍 실행 시점에 데이터 타입을 결정하는 방식 (출처 : 위키백과, developer.mozilla)

6. 데이터 리터러시(Data Literacy): 데이터 리터러시는 데이터를 읽고 이해하여 분석 결과를 전달할 수 있는 능력을 의미함. 데이터 리터러시 역량에는 문제 정의, 데이터 수집/관리/분석/시각화, 해석 및 기획 등이 포함됨.(출처: 코드스테이츠)

7. 인공지능(Artificial Intelligence): 인공지능은 학습, 창조, 이미지 인식 등과 같이 주로 인간 지능과 연결된 인지 문제를 해결하는 컴퓨터 공학 분야. 현재 스마트 센서, 데이터 분석, 자동화 등 다양한 인공지능 기술을 활용.(출처: 아마존 웹)

8. 머신러닝(Machine Learning): 머신러닝은 인공지능의 하위 집합으로, 컴퓨터가 데이터를 통해 스스로 학습하고 새로운 규칙을 생성할 수 있도록 하는 기술. 지도 학습, 비지도 학습, 강화 학습 등 다양한 방법으로 구현되며, 다양한 분야에 활용.(출처: 모두의 연구소)

9. 딥러닝(Deep Learning): 딥러닝은 신경망의 하위 분야이며, 머신러닝보다 더 확장 가능한 방식으로 학습함.대량의 데이터를 자동으로 학습할 수 있음.(출처: IBM)

10. 종속변수(Dependent Variable): 종속변수는 다른 변수의 영향을 받는 변수를 의미. 독립변수에 의해 변화하는 변수로, 예측 모델 개발 시 주요 타깃이 됨.

11. 지도 학습(Supervised Learning): 정답이 있는 데이터를 활용해 데이터를 학습시키는 것. (ex. 분류, 회귀)

12. 비지도 학습(Unsupervised Learning): 정답 라벨이 없는 데이터를 비슷한 특징끼리 군집화하여 새로운 데이터에 대한 결과를 예측하는 방법. (ex. 클러스터링, K-means)

13. 강화 학습(Reinforcement Learning): 기계 학습의 한 영역으로, 어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 또는 행동 순서를 선택하는 방법. (출처 : 위키백과)

14. 하이퍼 파라미터(Hyperparameter): 기계 학습에서 학습 속도나 옵티마이저 선택과 같은 매개변수로, 학습 프로세스의 세부사항을 지정하므로 '하이퍼파라미터'라는 이름이 붙음. (출처 : 위키백과)

15. 레이턴시(Latency): 자극과 반응 사이의 시간으로, 컴퓨터 쪽 의미로는 쉽게 말해 '전자 시스템 간 요청 간 시간 지연', 즉 '접근 시간'을 의미함 (출처 : 위키백과)

16. 피쳐(Feature) : 데이터의 특징을 나타내는 변수를 의미하며 독립변수와 동일한 의미로 사용한다. 데이터 Table에서 Column을 의미한다. (출처 : 코칭스터디 강좌)

17. 데이터 핸들링 : 정보 통신 파일의 기본 구성 요소인 레코드를 기반으로 하여 이들의 필드를 조작하는 것을 의미한다. (출처 : 네이버 사전)

18. 차원의 저주 : 데이터의 차원이 높아질 수록 알고리즘의 실행이 아주 까다로워지는 일을 의미한다. (출처 : 위키백과)

19. slicing : 데이터를 가지고 올 때 원하는 데이터 섹션만 가지고 오는 데이터 저치 기법이다.(출처 : 코칭스터디 강좌)

20. ndarray : 배열 객체는 고정 크기 항목의 다차원 동질 배열을 나타낸다. 연관된 데이터 유형 객체는 배열의 각 요소의 형식(바이트 순서, 메모리에서 차지하는 바이트 수, 정수인지 부동 소수점 숫자인지 등)을 설명한다. (numpy 공식 웹사이트)

21. LLM(거대언어모델) : 대용량 인간 언어를 이해하고 생성할 수 있도록 훈련된 인공지능(AI)모델이다. 딥러닝 알고리즘과 통계 모델링을 바탕으로 자연어 처리 작업에 활용된다. 주어진 언어 범위 내에서 정해진 패턴이나 구조, 관계를 학습하는 기존 언어 모델과 달리 대규모 언어 데이터를 학습해 문장 구조 문법, 의미 등을 파악하고 자연스러운 대화 형태로 상호작용이 가능하다. (출처 : 지식백과)

<팀 미션 - 2문제>

제출은 [자료조사] [토론내용] [결론] 형태를 띄어 제출한다.

[작성시 유의사항]
차원의 저주 설명
- 학습데이터 수와 차원의 수의 관계에 대한 개념
- 차원의 저주 현상의 원인

차원의 저주 해결방법 설명
- 차원의 저주 현상의 원인에 따른 근본적인 해결방법에 대해 토의
- 차원의 저주 해결방법 종류들의 특징들에 대해 설명

PCA vs t-SNE 비교분석
- 선형 차원 축소방법인 'PCA'와 비선현 차원 축소방법인 't-SNE'에 대해 깊게 알아보고 두 방법의 차이점 및 장단점 토의

문제 1. 데이터의 차원이 증가함에 따라 나타나는 '차원의 저주 (The curse of dimensionality)' 현상에 대해 조사해 보고 해당 현상의 발생원인과 또 그에 따른 해결방법은 무엇이 있는지 토의해보자. [주제 : 데이터와 차원]

차원의 저주 (The curse of dimensionality) 설명

"차원의 저주(Curse of dimensionality)"란?
: 데이터의 차원이 높아질수록 알고리즘의 실행이 아주 까다로워지는 일이다. 차원의 저주는 저차원 환경에서 발생하지 않는, 고차원 공간에서 데이터를 분석 · 정리할 때 발생하는 다양한 현상을 말한다.

차원이 증가하면 공간의 부피가 너무 빨리 증가하여, 사용 가능한 데이터가 희소해진다. 신뢰할 수 있는 결과를 얻기 위해 필요한 데이터의 양은, "차원"에 따라 기하급수적으로 증가하는 경우가 많다.

같은 데이터지만 1차원에서는 데이터 밀도가 높았던 것이 2차원, 3차원으로 차원이 커질수록 데이터 간 거리가 멀어지고, 밀도가 떨어지게 된다. 이렇게 차원이 증가하면 "빈 공간"이 생기고, 이 공간은 컴퓨터에서 0으로 채워진 공간이다. 즉, 정보가 없는 공간이기 때문에 빈 공간이 많을수록 학습을 시켰을 때 성능이 저하될 수밖에 없다.

따라서 차원의 저주란, 모델 학습을 위해 필요한 학습데이터 수가 차원의 수보다 적어 데이터 분석, 혹은 모델 성능에 부정적인 영향을 주는 현상을 의미한다.

차원 축소의 필요성
: 데이터 차원이 많아질 수록 발생하는 차원의 저주 문제를 해결하고 과적합을 방지하고, 데이터의 이해도와 계산의 효율성을 향상시킬 수 있다. 또한 데이터 시각화를 개선할 수 있다.

차원의 저주 해결방법

: 이론적인 해결 방법은 더 많은 데이터를 확보해 훈련 샘플의 밀도가 충분히 높아질 때까지 데이터를 수집하여 훈련 세트의 절대적인 크기 자체를 키우는 것이다. 하지만 필요한 밀도에 도달하기 위해 요구되는 훈련 샘플의 수는, 차원의 수가 커짐에 따라 기하급수적으로 늘어나기 때문에 이러한 방법은 현실적으로 불가능하다.

차원의 저주를 해결하는 현실적인 방법은 가장 관련된 특성이 있는 차원을 분류하여 그 차원에 대해서만 기술하는 다양한 **"차원 축소 기법"**을 활용하는 것이다.

대표적인 차원 축소 기법에는 크게
1. Feature Selection 과 2. Feature Extraction 이 있다.
그리고 가장 많이 쓰는 차원축소기법은 Feature Selection 인데 그 중 PCA와 LCA를 가장 많이 쓴다.

1) Feature Selection

: 데이터의 특징을 잘 나타내는 중요한 특성만 선택하여 데이터의 차원을 줄인다.
학습 시간을 줄이고, 모델이 단순화 된다는 장점이 있다.

- 방법
  Filter Method : Feature 간 관련성 측정
  Wrapper Method : Feature Subset의 유용성 측정
  Embedded Method : Feature Subset의 유용성을 측정하지만, 내장 metric을 사용

2) Feature Extraction

: 기존 특성을 저차원의 중요 특성으로 압축하여 추출한다.
특성 간의 상관관계를 고려하기에 용이하고, 특성 개수를 많이 줄일 수 있다는 장점이 있다.

- 방법
  주성분 분석(Principal Component Analysis, PCA) : 고차원의 원본 데이터를 저차원의 부분 공간으로 투영하여 데이터를 축소시키는 기법
  선형 판별 분석(Linear Discriminant Analysis, LDA) : 클래스 내부 분산이 최소가 되는, 클래스 중심 간 거리가 최대가 되는 벡터를 찾아 데이터를 투영하여 차원을 축소하는 기법

3) PCA (Principal Component Analysis, PCA)

: PCA는 데이터의 분산을 최대화하는 주성분을 찾아 차원을 축소하는 기법으로 데이터의 주요 특성을 보존하면서 차원을 낮출 수 있어 과적합 문제를 해결할 수 있다.
직교하는 축을 순차적으로 찾아 데이터를 투영하는 방식으로 차원을 축소할 수 있다.

4) LDA (Linear Discriminant Analysis, LDA)

: LDA는 클래스 간 분산은 최대화하고 클래스 내 분산은 최소화하는 방향으로 차원을 축소하는 기법으로 분류 문제에서 데이터의 판별력을 높이는 데 효과적이다.

5) Isomap

: Isomap은 데이터의 고유 구조를 보존하면서 차원을 축소하는 비선형 차원 축소 기법이다.
데이터의 국소적 구조와 전역적 구조를 모두 고려하여 차원을 축소할 수 있다.

- **차원 축소를 통한 데이터 분석 과정**
  1. 데이터의 특성을 고려하여 적절한 차원 축소 기법 선택
  2. 차원 축소를 통해 데이터의 핵심 정보 추출
  3. 축소된 데이터를 활용하여 다양한 분석 수행 (분류, 군집화, 시각화 등)
  4. 분석 결과를 바탕으로 의사결정 및 인사이트 도출

PCA vs t-SNE 비교분석

PCA (Principal Component Analysis)

: 고차원 데이터를 저차원으로 축소하는 선형 차원 축소 기법.
가장 대표적인 차원 축소 알고리즘으로, 먼저 데이터에 가장 가까운 초평면(hyperplane)을 구한 다음, 데이터를 이 초평면에 투영(projection)시키는 방법이다.

데이터의 분산을 최대한 보존하면서 차원을 축소할 수 있다. 주성분 (Principal Component)이라는 새로운 축을 찾아 데이터를 투영하고 주성분은 원래 특성들의 선형 조합으로 구성된다. 주성분 중 분산이 큰 순서대로 선택하여 차원을 축소할 수 있으며, 선형 구조의 데이터에 적합하며, 계산 복잡도가 낮다.

PCA는 다음과 같은 단계로 진행된다.

1. 학습 데이터셋에서 '분산'이 최대인 축(axis)를 찾는다.
2. 여기서 찾은 첫 번째 축과 직교하면서 분산이 최대인 두 번째 축을 찾는다.
3. 첫 번째 축과 두 번째 축에 직교하면서 분산을 최대한 보존하는 세 번째 축을 찾는다.
4. 1~3과 같은 방법으로 데이터셋의 차원(특성 수)만큼의 축을 찾는다.

t-SNE (t-Distributed Stochastic Neighbor Embedding)

: SNE를 개선한 방법으로 고차원의 복잡한 데이터를 저차원에 표현하기 위한 비선형 차원 축소 기법이다. 고차원 데이터 포인트들 간의 유클리드 거리를 이용하여 유사도를 조건부 확률로 바꿔 나타낸다. 유사한 데이터 포인트는 가까이, 다른 데이터 포인트는 멀리 배치한다.

PCA vs t-SNE 비교

: PCA와 t-SNE는 차원을 축소한다는 공통점이 있지만, 여러 차이점도 존재하므로 혼용해서 보완 사용하기도 한다.

	PCA	t-SNE
목적	데이터 구조와 패턴 파악	시각화에 더 초점을 맞춤
방법	데이터 분산을 최대한 '보존'하는 축을 찾아서 차원 축소	고차원 데이터와 저차원 데이터 간의 '유사도'를 계산하고 최적화
데이터 유형	선형적 데이터	선형적 & 비선형적 데이터
주의할 점	- 비선형적 데이터에서는 성능이 떨어짐 - 1차원으로 투영시켰을 때 거리가 가깝거나 멀거나 변별력이 없음	- 저차원에서 유사도 최적화 과정이 포함되기 때문에, PCA에 비해 Computation cost가 높음 - 초기 기준점에 따라 값이 다르기 때문에 돌릴 때마다 시각화 결과가 다르게 나옴

+) UMAP(Uniform Manifold Approximation and Projection)

: 고차원 공간에서의 데이터를 그래프로 만들고, 저차원으로 그래프를 투영(projection)하는 차원 축소 기법으로, 가장 좋은 성능을 내는 알고리즘으로 알려져 있다.

: t-SNE보다 속도가 현저히 빠르고, 데이터의 global 구조를 더 잘 보존한다는 장점이 있다. 또한 리만 기하학과 위상수학에 기반한 탄탄한 이론적 배경을 지니고 있다고 한다. 하지만 Hyperparameter의 영향을 크게 받고, 저차원으로 임베딩 시 데이터 왜곡이 일어날 수 있다는 단점도 존재한다.

<u>실제 산업군에서 각 데이터 분석 기법에 적합한 데이터는?</u>

PCA의 선형 구조 데이터 시각화에 적합한 산업군 예시

제조업: 제품 품질 관리, 공정 최적화 등에 PCA를 활용할 수 있다. 제품의 다양한 특성(예: 무게, 크기, 강도 등)을 고차원 데이터로 표현할 수 있으며, PCA를 통해 주요 특성을 찾아 시각화할 수 있다.

금융업: 주식 포트폴리오 분석, 신용 평가 등에 PCA를 활용할 수 있다. 주식 수익률, 위험 지표 등 고차원 데이터를 PCA로 차원을 축소하여 시각화할 수 있다.

의료 분야: 질병 진단, 유전체 분석 등에 PCA를 활용할 수 있다. 환자의 다양한 생체 지표, 유전자 발현 데이터 등을 PCA로 차원을 축소하여 시각화할 수 있다.

PCA는 선형 구조의 데이터에 적합하므로, 제조업, 금융업, 의료 분야와 같이 다양한 특성을 가진 고차원 데이터를 효과적으로 시각화할 수 있다.

a- SNE의 비선형 구조 데이터 시각화에 적합한 산업군 예시

이미지 처리: 이미지 데이터의 특징을 추출하고 시각화하는 데 t-SNE가 유용하다. 이미지 데이터는 고차원 특성을 가지므로 t-SNE를 통해 비선형 구조를 잘 보존할 수 있다.

자연어 처리: 단어 임베딩, 문서 클러스터링 등에 t-SNE를 활용할 수 있다. 단어나 문서는 고차원 벡터로 표현되므로 t-SNE를 통해 비선형 구조를 잘 보여줄 수 있다.

생물정보학: 유전체 데이터, 단백질 구조 데이터 등의 시각화에 t-SNE가 유용하다. 이러한 데이터는 고차원 특성을 가지며 비선형 구조를 가지므로 t-SNE가 적합하다.

문제 2. 파이썬 라이브러리 중 하나인 'Numpy'의 특징에 대해 서술하고, Numpy의 등장으로 더욱 편리해진 작업들은 무엇이 있는지 조사해보자. [주제 : 파이썬]

[작성시 유의사항]
Numpy 설명
- Numpy 라이브러리의 주 기능과 특징
- Numpy를 사용하는 작업

Numpy 라이브러리의 주 기능과 특징

파이썬의 과학 계산용 패키지인 Numpy(Numerical Python)는 대규모 데이터 연산을 빠르게 처리하기 위해 선형대수나 다양한 수학 함수 기능을 포함하고 있어 데이터 분석, 기계 학습, 통계 분석같은 분야에 널리 쓰인다. 넘파이의 다양한 특징을 나열하면

1. **일반 list에 비해 메모리 효율이 좋고 속도가 빠르다.**
   : 서로 다른 데이터 타입을 가질 수 있는 일반 파이썬 list와 다르게, 넘파이 배열은 모든 요소가 동일한 데이터 타입만 가진다. 넘파이는 배열의 모든 요소가 연속된 메모리 블록에 저장되는데, 이는 데이터 접근과 연산을 빠르게 만들 수 있다.

2. **for문 같은 반복문 없이 배열 처리가 가능하다.**
   : 넘파이는 벡터화 연산을 지원하여 배열 단위로 연산을 수행할 수 있다. 이는 간결한 코드와 실행 속도 향상을 안겨주는 기능으로 모델의 성능을 높여줄 수 있다.

3. **뛰어난 호환성과 통합을 가능한다.**
   : 넘파이는 Pandas나 Matplotlib같은 라이브러리와의 호환성이 좋고 C, C++, 포트란같은 레거시 언어와 통합하여 개발할 수 있다.

4. **강력한 n차원 배열 객체인 ndarray를 지원한다.**
   : 넘파이의 가장 기본 단위인 ndarray(numpy dimensional array)는 빠르고 유연한 자료형으로 다차원 배열을 효율적으로 처리하고 조작하기 위한 넘파이의 핵심 클래스이다.

Numpy를 사용하는 작업

1. 계산 및 데이터 분석

: numpy 라이브러리는 행렬 계산을 포함한 모든 계산의 편의성을 제공하고 표준편차와 상관관계 분석과 같은 통계적 분석과 통계적 계산에도 편의성을 제공한다.

a. 사용예시

```
import numpy as np

# 실험 데이터
data = np.array([1.2, 2.3, 3.1, 4.5, 5.0])

# 평균과 표준 편차 계산
mean = np.mean(data)
std_deviation = np.std(data)

print("Mean : ", mean)
print("Std_deviation : ", std_deviation)
```

```
Mean :  3.22
Std_deviation :  1.396280774056565
```

2. 이미지 처리

: 영상 의료 분석할 때 CT 스캔이나 MRI 이미지 분석이 필요할 때와 같은 이미지 처리 분야에서도 numpy를 사용한다. 이미지 처리 라이브러리를 통해 이미지를 로드하고 필터링하여 특정 패턴이나 이상을 감지할 수 있다.

a. 사용 예시

```
import numpy as np
from skimage import io, color

# 이미지 로드 및 그레이스케일 변환
image = io.imread('/content/cat img.jpg')
gray_image = color.rgb2gray(image)

# 이미지 데이터의 일부 특징 분석
mean_intensity = np.mean(gray_image)
print("Mean intensity: ", mean_intensity)
```

```
Mean intensity:  0.4661178431403892
```

3. 머신러닝

: 다차원, 많은 데이터의 행렬을 계산해야하는 머신러닝에서 numpy를 이용하여 계산을 편리하고 효과적으로 할 수 있어 모델학습과 모델예측을 수월하게 할 수 있다.

a. 사용예시

```
import numpy as np

# 사용자-제품 행렬
ratings = np.array([[5, 3, 0, 1],
                    [4, 0, 0, 1],
                    [1, 1, 0, 5],
                    [1, 0, 0, 4],
                    [0, 1, 5, 4]])

# 행렬 인수 분해를 통한 추천
U, sigma, Vt = np.linalg.svd(ratings, full_matrices=False)
print("User feature matrix U:₩n", U)
print("Diagonal matrix sigma:₩n", sigma)
print("Product feature matrix Vt:₩n", Vt)
```

```
User feature matrix U:
 [[-0.43689593 -0.66924125 -0.29627751  0.48637475]
 [-0.29717498 -0.44308727 -0.05015708 -0.79591123]
 [-0.51589728  0.13631518  0.54893193  0.28612203]
 [-0.39999635  0.11077382  0.48349385 -0.20569271]
 [-0.54282768  0.5700326  -0.61205501 -0.0760895 ]]
Diagonal matrix sigma:
 [9.03171974 6.22925557 3.77397038 1.83890217]
Product feature matrix Vt:
 [[-0.47488998 -0.26234348 -0.3005118  -0.78444124]
 [-0.78203025 -0.20891356  0.45754472  0.36801718]
 [-0.17212379 -0.25224247 -0.81089006  0.49920382]
 [-0.36507752  0.907692   -0.20688838 -0.00329281]]
```

>> 특이값 분해(SVD, 차원축소, 데이터 노이즈 제거, 데이터 압축 등 에서 사용)

>> 각각 왼쪽 특이 벡터, 특이값, 오른쪽 특이 벡터의 전치 를 출력한다.

4. 시뮬레이션 및 모델링

: 날씨 예측과 같은 다양한 학문 분야에서 복잡한 시뮬레이션을 수행할 때 Numpy를 사용하면 수치 모델링 및 시뮬레이션을 간단히 구현할 수 있다.

a. 사용예시

```
import numpy as np

# 기초적인 대기 모델
temperature = np.array([[30, 32, 34],
                        [28, 29, 31],
                        [26, 27, 28]])

# 온도 변화 시뮬레이션
temperature_change = np.random.normal(0, 1, temperature.shape)
new_temperature = temperature + temperature_change
print("변화된 온도₩n", new_temperature)
```

```
변화된 온도
 [[29.63230735 30.37651857 33.49386333]
 [29.17408191 30.6371185  31.89038953]
 [25.50552222 26.51878539 26.91733694]]
```

5. 금융계산

: 시계열 분석을 통한 주식 가격 예측, 리스크 관리, 포트폴리오 최적화 등에서도 사용할 수 있다.

a. 사용예시

```
import numpy as np

# 자산의 수익률 데이터
returns = np.array([[0.05, 0.10, 0.15],
                    [0.07, 0.12, 0.18],
                    [0.04, 0.11, 0.14]])

# 자산의 평균 수익률
mean_returns = np.mean(returns, axis=0)

# 포트폴리오의 최적화
weights = np.random.random(returns.shape[1])
weights /= np.sum(weights)

portfolio_return = np.dot(weights, mean_returns)
print("포트폴리오", {portfolio_return})
```

```
포트폴리오 {0.08711618250570391}
```