1 <처음보는 용어집>

1) EDA: EDA 는 Exploratory Data Analysis 의 약자로, 탐색적 데이터 분석을 의미합니다. 데이터셋을 탐색하고 시각화하여 데이터의 특성과 패턴을 이해하는 과정을 말합니다. (출처 : ChatGPT)
2) 브로드캐스팅: 브로드캐스팅은 데이터 분석이나 프로그래밍에서 배열 또는 행렬 연산을 수행할 때, 크기가 다른 배열을 자동으로 동일한 크기로 맞추는 기능을 의미합니다. 이를 통해 효율적인 연산을 가능하게 합니다. (출처 : ChatGPT)
3) soft skill: 소프트 스킬은 개인의 인간적인 능력과 특성을 의미합니다. 커뮤니케이션, 리더십, 문제 해결, 협업 등과 같은 역량을 말하며, 데이터 사이언스 분야에서는 분석 결과를 효과적으로 전달하거나 팀원들과 원활한 협업을 할 수 있는 능력을 포함합니다. (출처 : ChatGPT)
4) hard skill: 하드 스킬은 주로 기술적인 역량이나 전문 지식을 의미합니다. 데이터 사이언스 분야에서는 프로그래밍, 통계, 머신러닝, 데이터베이스 관리 등과 같은 기술적인 스킬을 말합니다. 이러한 하드 스킬은 분석과 모델링에 필요한 핵심 역량을 제공합니다. (출처 : ChatGPT)
5) ETL: ETL 은 Extract, Transform, Load 의 약자로, 데이터 웨어하우스나 데이터베이스에 데이터를 추출하고 변환하여 적재하는 과정을 말합니다. 데이터를 추출하고 필요한 형태로 가공한 후 목적에 맞게 데이터베이스에 적재하는 작업을 포함합니다. 이는 데이터 사이언스에서 데이터 전처리 과정의 일부로 중요한 역할을 합니다. (출처 : ChatGPT)
6) 변수 : 수학적 공간 차원
7) 성김 현상 : 고차원에서 데이터가 일반적으로 몰려있는 경향을 나타내는 현상으로 차원의 증가에 따라 데이터 포인트가 특정 영역에 집중되는 현상
8) 차원 축소 : 저차원의 표현이 고차원 원본 데이터의 의미 있는 특성을 원래 차원에가깝게 유지할 수 있고 고차원 공간에서 저차원 공간으로 데이터를 변환하는 것
9) 선형 차원 축소 : 선형 변환을 사용하는 방법으로 주로 주성분 분석(PCA, Principal Component Analysis) 기법에 기반한다. PCA 는 데이터의 분산을 최대로 보존하면서 주요한 정보를 추출하기 위해 공분산 행렬의 고유벡터를 계산하여 데이터를 새로운 저차원 공간으로 투영
10) 비선형 차원 축소 : 비선형 변환을 사용하여 데이터의 차원을 축소하는 기법으로 데이터가 복잡한 비선형 구조를 가지고 있을 때 유용
11) 선형대수: 벡터 공간, 벡터, 선형 변환, 행렬, 연립 선형 방정식 등을 연구하는 대수학의 한 분야(출처: 위키피디아)

12) 벡터: numpy 에서, 벡터는 숫자를 원소로 가지는 리스트나 배열을 뜻한다.
13) 여기서, 세로로 나열된 벡터는 열 벡터라고 하고, 가로로 나열된 벡터는 행 벡터라고 한다. 이 때, 벡터에 있는 숫자들의 개수로 벡터의 차원이 결정된다. 벡터는 공간에서 한 점을 나타내고, 원점으로부터 상대적인 위치를 표현하기도 한다.(출처: https://velog.io/@babydeveloper/BoostCamp-Day-6)
14) weight: 머신러닝에서 weight 라는 값은, 비용함수 (cost function)이 가장 작은값을 가지게 하는 값이다. weight 값이 0 을 가지게 되면 입력 데이터가 무시될 수 있기 때문에, weight 가 0 이 되는 것은 피해야 한다. (출처: https://copycode.tistory.com/173)
15) axis: 파이썬의 배열에서 축을 의미한다. 1 차원 배열에서는 축이 axis0, 2 차원 배열에서는 axis 0, axis 1 로 2 개가 된다 (각각 행, 열). 3 차원에서는 축이 3 개로, 높이, 행, 열이 된다. 4 차원에서는 축이 4 개가 된다. (출처: https://chancoding.tistory.com/11)
16) 크롤링: 웹 페이지를 그대로 가져와서 데이터를 추출해 내는 행위 (출처: 나무위키)
17) 데이터 마이닝(Data Mining) : 대규모로 저장된 데이터안에서 체계적이고 자동적으로 통계적 규칙이나 짜임을 분석하여 가치있는 정보를 배내는 과정(출처: 위키백과)
18) condition rule : 조건을 설정하고 그 조건에 따라 특정한 동작이나 결과를 결정하는 규칙 (출처: ChatGPT)
19) 스프레드시트 : 표 형식으로 데이터의 조직, 분석, 저장을 가능케 하는 상호작용 컴퓨터 애플리케이션. 계산을 위해 사용되는 표 형식의 계산용지를 컴퓨터에서 사용할 수 있게 구현한 표 계산 프로그램(출처: 위키백과)
20) dynamic typing : 프로그래밍 언어에서 변수의 데이터 타입을 미리 선언하지 않고, 실행 시점에서 변수에 할당된 값에 따라 데이터 타입이 동적으로 결정되는 개념(출처: ChatGPT)
21) fancy indexing : 배열의 값에 접근할 때, 인덱스 배열을 통해서 여러 배열 요소에 접근하기 위한 indexing 방법(출처: 파이썬 데이터사이언스 핸드북)

2 <팀 미션 - 2 문제>

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]

1. 차원의 저주 설명

- 자료조사

  - 차원의 저주(The curse of dimensionality)는 데이터의 차원이 증가함에 따라 발생하는 현상으로, 데이터 분석과 머신러닝 작업에 어려움을 초래합니다.

  - 차원의 저주(Curse of Dimensionality)는 고차원 공간에서 데이터 분포의 특성이 변화하는 현상이다.

  - 차원의 저주란, 데이터의 차수가 큰 고차원 데이터를 다룰 때 종종 마주하는 문제이다. 데이터의 차원이 커질수록 데이터의 공간은 매우 빠르게 증가하는데(지수함수로), 이렇게 거대한 공간의 데이터는 현실적으로 처리하기가 어렵다

  - 고차원의 데이터에서 학습 데이터수가 부족하여 특정 학습 데이터에 따라 특정 값의 방향으로 변화하여 데이터 분포나 적절한 모델 추정이 어려워지는 것을 이야기한다.

- 토의

  - 차원의 저주는 차원이 증가할수록 데이터 공간의 크기가 기하급수적으로 증가하기에 발생. 이 경우 데이터가 고차원 공간에서 희소해지는 문제가 생기고 이러한 문제는 데이터 간의 거리 계산이 어렵고, 모델의 학습과 일반화가 어려워지게 하는 문제를 일으킴.

  - 차원의 저주는 데이터 희소성(고차원 공간에서 데이터가 희박해지는 경우), 차원의 폭발적인 증가(가능한 데이터의 조합이 기하급수적으로 증가하는 문제), 모델의 과적합(적은 수의 데이터 포인트에 대해 정확한 예측을 하지만, 새로운 데이터에 대해 일반화하기 어려운 문제), 차원의 무관한 특성(특성들 간의 상관 관계가 줄어들고, 중요한 특성을 식별하기 어려우며, 불필요한 특성들이 노이즈로 작용) 등의 측면으로 발생함.

- 결론

  - 차원의 수: 차원은 변수의 수학적 공간으로 차원의 수는 데이터의 변수(피처)의 수.

  - 학습데이터 수: 모델을 학습시킬 때 필요한 데이터의 수로, 일반화할 수 있는 모델을 만들기 위해서는 특정 개수 이상의 학습데이터가 필요함.

- 학습 데이터 수와 차원의 수의 관계: 차원의 수가 증가하면 학습 데이터의 수도 증가하게 됨. 데이터의 차원이 증가하면 그에 따라 가능한 데이터의 조합이 기하급수적으로 증가함. 예를 들어 1 개의 특성이 10 개의 값을 가질 수 있다면, 특성이 1 개일 때는 가능한 데이터의 조합이 10 개이지만, 10 개의 특성이 있는 데이터의 경우 10^10 개의 데이터 조합이 생김. 동일한 학습데이터 수를 가지고 있을 때 차원이 늘어나면 데이터가 없는 것으로 처리되는 빈 공간의 수도 늘어나기에, 동일한 성능을 위해서는 차원의 수가 늘어나는 만큼 학습데이터의 수도 늘어나게 됨.
- 차원의 저주는 차원의 개수와 밀접한 관련이 있으며, 차원의 크기에 비해 학습데이터가 부족하여 발생함.

2. 차원의 저주 해결방법 설명

- 자료조사

  - 차원축소: 고차원의 데이터를 저차원으로 변환하는 기법. 데이터의 차원을 줄이면서도 가능한 한 많은 정보를 유지할 수 있음.
  - 피처 선택과 피처 추출: 피처 선택은 주어진 피처에서 유용한 피처만을 선택하는 것이고 피처 추출은 기존의 피처를 변환하여 새로운 피처를 생성하는 것. 이를 통해 데이터의 차원을 줄이고 유의미한 정보를 유지할 수 있음
  - 데이터 수집과 데이터 품질 개선: 더 많은 데이터를 수집하거나, 데이터의 노이즈를 제거하고 보정함으로써 차원의 저주로 인한 문제 완화.

- 토의

  - 차원 축소: 특정 피처를 선택할 수도 있지만, 피처와 데이터의 양이 많다면 데이터적으로 접근하여 여러 알고리즘을 통해 차원을 축소하거나 고차원을 저차원으로 낮추는 과정인 피처 추출의 방법을 사용하는 것이 더 효율적이고 효과적일 것임
  - 데이터 수집 단계에서 피처의 수를 과하게 잡지 말고 도메인적으로 중요한 피처들만을 측정하여 데이터를 수집하는 것도 중요한 방법일 것으로 보임.

  차원의 저주 해결방법의 추가적인 방법

  - 군집화 기법 활용: 데이터를 군집화하여 데이터 포인트 간의 관련성을 파악하고, 저차원에서의 군집성을 이용하여 차원 축소를 수행할 수 있다. 이는데이터의 유사성을 보존하면서 차원을 줄이는 데 도움이 된다.
  - 모델 선택과 규제화: 차원의 저주를 고려하여 적절한 모델을 선택하고 규제화

기법을 사용한다. 규제화는 모델의 복잡성을 제한하고, 과적합을 방지하는 데 도움이 된다. 예를 들어, L1 또는 L2 규제를 사용하는 선형 모델, 트리 기반 앙상블 모델의 가지치기 등이 있다.

- 데이터 샘플링 기법: 고차원 데이터에서는 학습 데이터의 밀도를 보존하면서 샘플을 선택하는 적절한 샘플링 기법을 사용할 수 있다. 예를 들어, K-Means 클러스터링을 사용하여 각 클러스터에서 대표적인 샘플을 선택하는 방법이 있다.

- 결론

- 차원의 저주를 해결하기 위해서는 근본적으로 과하게 많은 차원의 수를 줄이는 것이 중요함. 이를 위하여 주로 차원 축소 방법이나 피처 선택 및 피처 추출 방법이 사용됨.

- 차원 축소(Dimensionality Reduction)는 고차원의 데이터를 저차원으로 변환하는 기법으로 데이터의 차원을 줄이면서도 가능한 한 많은 정보를 유지시키는 방법. 이는 데이터의 차원을 줄이고 불필요한 특성을 제거하여 차원의 저주를 완화하는 데 도움이 됨. 주요 알고리즘으로는 PCA, t-SNE, UMAP 등이 있음.

- 피처 선택(Feature Selection)과 피처 추출(Feature extraction)에서 피처 선택은 가장 유용한 일부 피처만 선택하는 것을 의미하며, 피처 추출은 피처를 변환하여 새로운 피처를 생성하는 것을 말함. 두 방법 모두 불필요한 피처를 줄이는 과정. 피처 선택 알고리즘에는 상호정보량(mutual information), 가중치 기반 피처 선택, 상관 관계 분석, L1 규제 등이 있고 피처 추출의 알고리즘에는 자동인코더(autoencoder) 등이 있음.

- 차원의 저주를 해결하기 위해서는 데이터 수집과 가공 단계에서 좋은 데이터를 만들어내고 충분한 양을 학습시킬 수도 있음.

3. PCA vs t-SNE 비교분석

- 자료조사(PCA)

- PCA는 선형 차원 축소 알고리즘으로, 데이터의 분산을 최대한 보존하면서 새로운 축을 찾아 데이터를 변환합니다. 이러한 축은 데이터의 분산이 가장 큰 방향으로 정렬됩니다. PCA는 데이터 의 주성분(principal component)을 추출하여 차원을 줄일 수 있으며, 데이터의 변동성을 잘 보존 하는 특징이 있습니다. 그러나 비선형 구조를 잘 표현하지 못하는 한계가 있습니다.

- 선형 차원 축소는 데이터를 저차원 공간으로 변환하는 과정에서 선형 변환을 사용하는 방법이다. 이는 주로 주성분 분석(PCA, Principal

Component Analysis) 기법에 기반한다. PCA는 데이터의 분산을 최대로 보존하면서 주요한 정보를 추출하기 위해 공분산 행렬의 고유벡터를 계산하여 데이터를 새로운 저차원 공간으로 투영한다. 이러한 선형 변환은 주로 데이터의 선형적 상관 관계를 고려한다.

- 주성분 분석이라고도 하는데 데이터의 분산을 최대한 보존하면서, 서로 직교하는 방향을 찾아, 샘플들을 저차원으로 변형한다. 더 자세히 말하면 PCA 방법은 원본 데이터셋과 투영된 데이터셋 간의 평균 제곱거리를 최소화하는 축을 찾고 (최대한 분산을 보존), 이후 학습 데이터셋에서 분산이 최대인 축을 찾는다 (그래야 올바르게 clustering 할 수 있음). 그러나 문제는, 선형이기 때문에 뭉쳐져 있는 데이터가 뭉게지는 단점이 있다는 것이다.

- 선형 차원 축소방법으로 다른 말로는 주성분 분석이라고 한다. 데이터 행렬의 공분산 행렬을 구하여 데이터의 분산을 최대한 보존할 수 있는 새로운 벡터로 데이터를 변환하는 과정으로 차원을 줄인다

- 자료조사(t-SNE)

  - t-SNE는 비선형 차원 축소 알고리즘으로, 데이터의 군집 구조를 시각화하기 위해 사용됩니다. t-SNE는 고차원 데이터를 저차원(일반적으로 2차원 또는 3차원)으로 매핑하여 데이터 포인트 간 유사성을 보존하면서 시각적으로 구별되도록 합니다. 따라서 t-SNE는 시각화 목적이 강한 경우에 유용합니다. 그러나 데이터의 전역적 구조를 보존하는 것보다 군집 간의 상대적인 구조를 강조하는 경향이 있으며, 계산 비용이 많이 드는 단점이 있습니다.

  - 비선형 차원 축소는 선형 변환이 아닌 비선형 변환을 사용하여 데이터의 차원을 축소하는 기법이다. 비선형 차원 축소는 데이터가 복잡한 비선형 구조를 가지고 있을 때 유용하다. 이러한 방법은 데이터를 고차원 공간으로 매핑한 후, 저차원 공간으로 변환하는 방식으로 작동한다. 대표적인 비선형 차원 축소 알고리즘으로는 t-SNE(t-Distributed Stochastic Neighbor Embedding)이 있으며 t-SNE는 고차원 데이터의 유사성을 보존하면서 데이터를 저차원 공간으로 투영한다. 이를 통해 데이터의 복잡한 구조와 군집성을 시각화하는 데 주로 사용된다.

  - t-SNE 분석으로 복잡한 데이터를 2차원 또는 3차원 데이터로 축소시키는 것이다. 이를 사용하면 비선형적이기 때문에, 고차원의 데이터 셋을 시각화하는 것에 대한 성능이 좋다. 메커니즘은 조건부 확률식으로 각 데이터

포인트의 이웃을 골라 내는 것이다. 이 때 고차원공간과 저차원 공간에서의 확률값 간의 차이를 최소화한다. 그러나 이 과정은 input feature를 확인하기 어렵다는 단점이 있어서 주로 시각화를 할 때 사용된다. 또 데이터의 개수가 많을수록 연산량이 아주 많이 늘어나 연산량 부하가 걸린다는 단점이 있다.
- 비선형 차원 축소방법으로, 주로 데이터 시각화를 위해 사용한다. 고차원의 데이터의 유사성과 해당 차원을 저차원으로 낮췄을 때 저차원 공간에서의 데이터의 유사성이 확률적으로 최소가 되는 저차원으로 변환시키는 과정을 반복한다.

- 토의

1. PCA (Principal Component Analysis):
    ① 선형 차원 축소 알고리즘
    ② 데이터의 분산을 최대한 보존하는 방식으로 주요한 정보를 추출
    ③ 주로 데이터의 선형 구조를 고려
    ④ 데이터의 공분산 행렬의 고유벡터를 계산하여 데이터를 새로운 저차원 공간으로 투영
    ⑤ 주성분(PC)이라는 새로운 축을 생성하고, 이를 기반으로 데이터를 재구성할 수 있음
    ⑥ PCA는 차원 축소와 함께 데이터 압축, 잡음 제거, 시각화 등에도 활용될 수 있음
- 2. t-SNE (t-Distributed Stochastic Neighbor Embedding):
    ① 비선형 차원 축소 알고리즘
    ② 고차원 데이터의 유사성을 보존하면서 데이터를 저차원 공간으로 투영
    ③ 데이터의 비선형 구조와 군집성을 시각화하는 데 주로 사용
    ④ 주로 시각화 목적으로 사용되며, 데이터의 군집을 시각적으로 구분하기 쉽게함
    ⑤ t-SNE는 데이터 쌍 간의 유사도를 측정하여 고차원과 저차원 공간 사이의 유사도를 최대화하는 방식으로 작동
    ⑥ 이 알고리즘은 주로 시각화를 위해 사용되기 때문에, 데이터의 재구성이나 압축에는 적합하지 않을 수 있음
    ⑦ t-SNE는 계산 비용이 높기 때문에 대규모 데이터셋에는 적합하지 않을 수 있음
- 결론

- 결론적으로, PCA는 선형 구조를 고려하여 데이터를 저차원으로 투영하고 재구성하는 반면에, t-SNE는 비선형 구조와 군집성을 시각화하는 데에 주로 사용된다. PCA는 계산 효율성과 데이터의 재구성 가능성 측면에서 우수하지만, t-SNE는 시각화 측면에서 우수하다. 따라서 분석하고자 하는 데이터의 특성과 목적에 따라 적절한 알고리즘을 사용해야 한다.

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

1. Numpy 특징

- 자료조사
  - ✓ 다차원 배열 (Multidimensional arrays): Numpy의 핵심 기능은 다차원 배열인 ndarray(NumPy N-dimensional Array) 객체입니다. 이러한 다차원 배열은 동일한 데이터 타입의 원소들을 저장하며, 빠르고 효율적인 수치 연산을 가능하게 합니다. 다차원 배열을 사용하면 행렬 연산, 벡터화 연산 등 다양한 수학적 연산을 간편하게 수행할 수 있습니다.
  - ✓ 브로드캐스팅 (Broadcasting): Numpy는 다른 크기의 배열 간에도 연산을 수행할 수 있는 브로드캐스팅 기능을 제공합니다. 브로드캐스팅은 배열 간의 차원이 다를 때 자동으로 크기를 맞춰주는 기능으로, 반복문을 사용하지 않고도 배열 간의 연산을 효율적으로 수행할 수 있도록 합니다.
  - ✓ 벡터화 연산 (Vectorized operations): Numpy는 벡터화 연산을 지원하여 반복문을 사용하지 않고도 배열의 모든 요소에 대한 연산을 한 번에 수행할 수 있습니다. 이를 통해 코드의 가독성을 향상시키고 실행 속도를 개선할 수 있습니다.
  - ✓ 선형 대수 연산 (Linear algebra operations): Numpy는 선형 대수 연산을 지원하는 다양한 함수와 메서드를 제공합니다. 이를 활용하여 행렬의 곱셈, 역행렬 계산, 특잇값 분해 등과 같은 선형 대수 연산을 쉽게 수행할 수 있습니다.
  - ✓ 빠른 연산 속도: Numpy는 C로 구현되어 있어 많은 연산을 고도로 최적화되어 처리 속도가 빠릅니다. 따라서 대용량 데이터 처리나 과학적 계산에 적합하며, 파이썬의 기본 리스트보다 더욱 빠른 연산을 제공합니다.
- 결론

  NumPy는 파이썬에서 수치 계산을 위한 핵심 라이브러리다. Numerical Python의 약자로, 다차원 배열 객체와 배열을 다루기 위한 다양한 함수를 제공한다.
  Numpy는 다차원 행렬 구조인 ndarray를 통해 벡터 및 행렬을 사용하는 선형 대수 계산에서 주로 사용된다. 예를 들어 벡터의 산술연산, 다차원 배열, 선형 대수, 난수 생성, 푸리에 변환 등등의 계산을 지원한다.
  NumPy의 가장 중요한 기능은 다차원 배열인 ndarray(N-dimensional array)다. 이 배열은 동일한 타입의 원소를 가지며, 각 차원의 크기를 가지고 있다. NumPy의 배열은 C나 Fortran과 같은 저수준 언어로 구현되어 있어 매우 효율적인 메모리 사용과 빠른 연산이 가능하다. 따라서 NumPy를 사용하면 배열 기반 데이터 생성, 조작, 처리, 저장등의 작업의 효율을 증대 시킬 수 있다. 주요 기능으로는 다차원 배열 생성, 배열 연산, 브로트 캐스팅, 배열 인덱싱 및 슬라이싱, 선형 대수 연산

등이 가능하다. Numpy는 편의성을 갖추고 있고, 속도도 빠르다는 장점이 있다.

2. Numpy 라이브러리
    1) numpy.ndarray: NumPy의 핵심 라이브러리인 다차원 배열 객체로 ndarray는 동일한 데이터 타입을 가지는 다차원 배열을 생성하고 다룰 수 있는 기능을 제공한다.
    2) numpy.random: 난수 생성과 관련된 기능을 제공하는 라이브러리로 NumPy는 다양한 확률 분포를 따르는 난수를 생성하고, 배열을 무작위로 섞거나 샘플링하는 기능을 제공한다. 통계적 시뮬레이션 및 머신러닝에서의 데이터 분할 등 다양한 작업에 유용하게 사용된다.
    3) numpy.linalg: 선형 대수 연산을 수행하는 기능을 제공하는 라이브러리로 행렬의 분해(예: LU, QR, SVD), 선형 방정식의 해 구하기, 고유값 및 고유벡터 계산 등과 같은 선형 대수 연산을 수행할 수 있다. 이러한 기능은 데이터 분석, 머신러닝, 신호처리 등 다양한 영역에서 사용된다.
    4) numpy.fft: 고속 푸리에 변환(Fast Fourier Transform, FFT)과 관련된 기능을 제공하는 라이브러리로 FFT는 시계열 데이터나 신호 처리 등에서 주파수 도메인에서의 분석을 수행하는 데 사용된다. NumPy의 FFT 기능을 활용하면 신호의 주파수 스펙트럼, 필터링, 컨벌루션 등을 처리할 수 있다.
    5) numpy.ma: 결측값(missing value)을 다루기 위한 기능을 제공하는 라이브러리로 ma는 "마스크드 어레이(masked array)"를 의미하며, 이를 통해 배열에서 특정 원소를 결측값으로 처리하고 해당 결측값을 제외한 연산을 수행할 수 있다. 결측값처리는 데이터 분석에서 중요한 부분이며, NumPy의 ma 기능은 이를 효과적으로 처리할 수 있도록 도와준다.

3. Numpy를 사용하여 더욱 편리해진 작업
    1) 배열 기반 데이터 처리: Numpy는 다차원 배열을 사용하여 데이터를 저장하고 처리하는데 효율적입니다. 따라서 데이터 분석, 수치 계산, 신호 처리 등 다양한 작업에서 Numpy를 활용하여 데이터를 효율적으로 처리할 수 있습니다.
    2) 수학 및 통계 연산: Numpy는 다양한 수학 및 통계 연산을 지원합니다. 배열 간의 연산, 수학 함수 (sin, cos, exp 등), 통계 함수 (mean, std, sum 등), 난수 생성 등을 쉽게 수행할 수 있습니다.
    3) 이미지 처리: Numpy는 이미지를 다차원 배열로 표현하고 처리하는 데에 매우 유용합니다. 이미지 필터링, 크기 조정, 회전 등의 작업을 Numpy 배열을 통해 쉽게 수행할 수 있습니다.
    4) 시뮬레이션 및 모델링: Numpy는 과학적 모델링과 시뮬레이션 작업을 지원합니다. 물리학, 생물학, 경제학 등 다양한 분야에서 수치 시뮬레이션을 위해 Numpy를

사용합니다.

5) 기계 학습: Numpy는 기계 학습 알고리즘을 구현하는 데에도 널리 사용됩니다. 다차원 배열을 사용하여 데이터를 표현하고, 벡터화 연산과 브로드캐스팅 기능을 활용하여 효율적인 학습 알고리즘을 개발할 수 있습니다.