1. 딥러닝 프레임워크의 필요성 및 기본 구성 요소

- **딥러닝 프레임워크의 역할**: 딥러닝 기술에는 DNN, CNN, RNN 등 검증된 여러 알고리즘이 존재합니다 [1]. 이러한 알고리즘들을 매번 새로 구현하는 것은 비효율적이므로 [2], 프레임워크는 검증된 알고리즘들을 쉽게 구현할 수 있도록 라이브러리를 제공하여 개발을 빠르고 손쉽게 만듭니다 [3].
- **머신러닝/딥러닝 실행의 4 대 구성 요소** [4]:
    1. **데이터**: 예측 또는 판별과 같은 문제 해결의 기반이 되는 필수 요소입니다 [5].
    2. **컴퓨터 (하드웨어)**: 코드를 실행하는 주체로, CPU 또는 고속 처리에 특화된 GPU 를 선택하여 사용합니다 [6].
    3. **플랫폼**: 구현된 알고리즘 코드를 컴퓨터에서 실행하게 도와주는 환경입니다. 대표적으로 Python 과 TensorFlow 가 있습니다 [7].
    4. **프로그램 (코드)**: 특정 문제를 해결하기 위해 작성된 코드 결과물입니다 [8].

---

2. 딥러닝 작업 환경 구축 방법: 클라우드 vs 개인 PC

딥러닝을 만들고 작동시키는 방법은 크게 두 가지로 나뉩니다 [9].

방법 1: 클라우드 기반 환경 (Google Colab)

- **정의**: 구글이 제공하는 클라우드 기반 딥러닝 실행 플랫폼입니다 [10]. 웹 브라우저만 있으면 접속하여 코드를 실행하고 결과를 확인할 수 있습니다 [11].
- **장점**:
    - **환경 설정의 편리함**: 주요 패키지(TensorFlow 등)가 이미 설치되어 있어 환경 설정에 드는 노력이 매우 적습니다 [12121212].
    - **저사양 PC 에서도 사용 가능**: 실제 연산은 구글 클라우드에서 수행되므로 개인 PC 는 고사양이 아니어도 됩니다 [13].
    - **무료 GPU 제공**: 개인 PC 에 GPU 가 없어도 구글의 GPU 를 무료로 사용하여 딥러닝 연산을 가속할 수 있습니다 [14141414]. GPU 는 텐서 연산에 매우 효율적이라 CPU 보다 빠른 속도를 제공합니다 [15].
    - **접근성**: 구글 계정만 있으면 빠르고 쉽게 온라인으로 코드를 테스트할 수 있습니다 [16].
- **단점**:
    - **패키지 비영속성**: 직접 새로 설치한 패키지는 세션이 종료되면 사라져 재설치가 필요합니다 [17].
    - **시간 제한**:
        - 최대 세션 유지 시간은 **12 시간**으로, 이 시간 안에 실행이 끝나야 합니다 [18].
        - **90 분** 동안 마우스 클릭 등 아무런 상호작용(interaction)이 없으면 연결이 자동으로 종료됩니다 [19].
    - **사용 한계**: 12 시간 이상 소요되는 복잡한 작업이나 본격적인 프로젝트에는 부적합하며, 이 경우 개인 PC 에 환경을 구축하는 것이 좋습니다 [20202020].

- **기타**: 기관이나 연구소에서는 외부 클라우드가 아닌, 고성능 중앙 서버를 두고 내부 구성원들이 접속해 사용하는 **내부 클라우드(하이브리드)** 형태도 존재합니다 [21212121].

방법 2: 개인 PC 에 직접 설치

- **정의**: 개인 컴퓨터에 필요한 프로그램을 직접 설치하여 코드를 실행하는 방식입니다 [22].
- **특징**:
  - **컴퓨터 성능에 직접적 영향**: 코드 실행 속도가 전적으로 개인 PC 의 성능에 좌우됩니다 [23]. 코드가 복잡하고 많은 리소스를 요구하면 고사양 PC 가 필요합니다 [24].
  - **설치 순서**:
    1. **아나콘다(Anaconda) 설치**: Python 뿐만 아니라 pandas, numpy 등 딥러닝에 필수적인 주요 패키지들이 포함된 통합 패키지를 설치하는 것이 효율적입니다 [25252525].
    2. **TensorFlow 설치**: 아나콘다 설치 후, 딥러닝 프레임워크인 TensorFlow 를 설치합니다 [26].
    3. **PyCharm 설치**: 코드 작성을 위한 통합 개발 환경(IDE)으로, 파이썬 개발에 널리 쓰이는 PyCharm 을 설치합니다 [27272727].
- **가상환경 (Virtual Environment)의 중요성**:
  - **개념**: 하나의 컴퓨터에서 여러 프로젝트를 독립된 환경으로 관리하기 위한 기능입니다 [28].
  - **필요성**: 프로젝트마다 요구하는 Python 이나 라이브러리 버전이 다를 수 있습니다 (예: 프로젝트 A 는 TensorFlow 1.4, 프로젝트 B 는 TensorFlow 2.7 필요)[29]. 단일 환경에서는 이를 동시에 충족하기 어렵기 때문에 [30303030], 각 프로젝트에 맞는 가상환경을 만들어 종속성 문제를 해결합니다 [31].
  - **PyCharm 에서 생성**: PyCharm 에서 새 프로젝트를 만들 때, Conda 를 기반으로 특정 Python 버전을 지정하여 가상환경을 설정할 수 있습니다 [32323232].

---

3. 실습: 폐암 수술 환자 생존율 예측

3.1. 데이터 준비

- **데이터 설명**: 폴란드 의과대학에서 공개한 실제 폐암 수술 환자의 의료 기록 데이터입니다 [33].
  - 총 470 명의 환자 데이터(470 개 행)로 구성됩니다 [34].
  - 17 개의 수술 전 환자 상태(종양 유형, 흡연 여부 등)를 나타내는 속성(Attribute)이 있습니다 [35353535].
  - 18 번째 열은 수술 후 생존 결과를 나타내며, **생존 시 1, 사망 시 0** 으로 표기됩니다 [36363636].
- **데이터 다운로드**:
  - GitHub 에서 CSV 파일의 내용을 복사하여 메모장에 붙여넣거나 [37373737], '다른 이름으로 저장'을 통해 다운로드할 수 있습니다 [38].

    - **PyCharm** 에서는 프로젝트 폴더 내에 저장하여 바로 사용할 수 있습니다 [39393939].
    - **Google Colab** 에서는 로컬 파일을 클라우드 환경으로 업로드하는 과정이 필요합니다 [40404040].

3.2. 코드 실행 환경별 절차

- **Google Colab 에서 실행**:
    1. 새 노트를 생성하고 예제 코드를 붙여넣습니다 [41414141].
    2. files.upload() 함수를 사용하여 로컬에 저장된 데이터셋(.csv) 파일을 Colab 환경으로 업로드합니다 [42424242].
    3. 코드를 실행하여 결과를 확인합니다. (정확도 약 84.68% [43])
    4. GPU 를 사용하려면 '수정' -> '노트 설정'에서 하드웨어 가속기를 'GPU'로 변경해야 합니다 [44].
- **PyCharm 에서 실행**:
    1. 프로젝트 폴더에 예제 코드 파일(.py)과 데이터 파일(.csv)을 위치시킵니다 [45].
    2. PyCharm 터미널에서 pip install tensorflow 명령어로 TensorFlow 를 설치합니다 [46].
    3. 코드를 실행하면 동일한 결과를 얻을 수 있습니다 [47].
    4. **Python Console** 을 활용하면 코드를 한 줄씩 실행하며 변수의 상태를 중간중간 확인할 수 있어 디버깅에 용이합니다 [48484848].

3.3. 딥러닝 코드 핵심 분석

- **라이브러리 임포트**: numpy, pandas, tensorflow 등 필요한 라이브러리를 불러옵니다 [49].
    - from tensorflow.keras.models import Sequential: TensorFlow 의 Keras API 에서 Sequential 모델을 직접 임포트하여, 이후 코드에서 Sequential()로 간결하게 사용할 수 있습니다 [50505050].
- **데이터 로드 및 분리**:
    - np.loadtxt(): CSV 파일을 불러와 Numpy 배열로 변환합니다 [51].
    - 데이터를 **속성(X)**과 **클래스/정답(Y)**으로 분리합니다 [52].
        - X: 0 번부터 16 번 열까지의 17 개 속성 데이터 [53].
        - Y: 17 번 열의 생존 여부 데이터 (0 또는 1)[54].
- **딥러닝 모델 설계 (구조 결정)**:
    - model = Sequential(): 신경망의 각 층(Layer)을 순차적으로 쌓을 수 있는 모델을 생성합니다 [55].
    - model.add(Dense(...)): 모델에 완전 연결 계층(Dense Layer)을 추가합니다 [56].
        - **입력층**: input_dim=17 은 17 개의 입력 속성을 의미합니다 [57]. 첫 번째 은닉층은 30 개의 노드로 구성됩니다.
        - **은닉층**: 활성화 함수로 relu 를 사용합니다 [58].
        - **출력층**: 노드가 1 개이며, 생존(1) 또는 사망(0)의 이진 분류 문제이므로 활성화 함수로 sigmoid 를 사용합니다 [59595959].
- **모델 컴파일 (학습 방식 설정)**:

    - model.compile(): 생성된 모델이 효과적으로 학습할 수 있도록 학습 과정을 설정합니다 [60].
    - loss='binary_crossentropy': 이진 분류 문제에 사용하는 손실 함수를 지정합니다.
    - optimizer='adam': 가중치(weight)를 효율적으로 업데이트하기 위한 최적화 알고리즘으로 'adam'을 사용합니다 [61].
    - metrics=['accuracy']: 학습 과정을 평가할 지표로 '정확도'를 사용합니다.
- **모델 학습**:
    - model.fit(X, y, ...): 준비된 데이터(X, y)를 모델에 입력하여 학습을 실행합니다 [62].
    - epochs=100: 전체 데이터셋을 100 번 반복해서 학습시킵니다 [63]. 1 에포크(epoch)는 전체 데이터(470 명)를 모두 사용하는 것을 의미합니다 [64].
    - batch_size=10: 전체 데이터를 한 번에 처리하지 않고, 10 개씩 작은 묶음(batch)으로 나누어 학습을 진행합니다 [65].