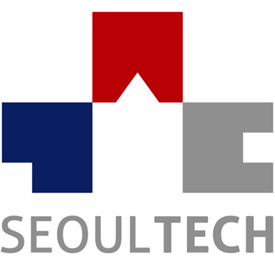
**개발 결과물 사용 설명서**

딥러닝 기반 다변량 스팀 사용 이상 감지 및 영향변수의 원인 분석 기능 제작

2023. 11. 28.



서울과학기술대학교 데이터사이언스학과

연구책임자 심재웅

목차

[1. 컴퓨팅 사양 3](#_Toc152258000)

[2. 파일 구조 3](#_Toc152258001)

[3. 개발 환경 3](#_Toc152258002)

[3.1. Ubuntu (Linux) 3](#_Toc152258003)

[3.2. Window 6](#_Toc152258004)

[4. 주요 라이브러리 9](#_Toc152258005)

[5. 실행 방법 및 예시 10](#_Toc152258006)

# 컴퓨팅 사양

* 운영체제 : Ubuntu 20.04.5 LTS
* CPU : AMD Ryzen 9 7950X 16-Core Processor \* 32
* GPU : NVIDIA GeForce RTX 3090

# 파일 구조

1. Random\_forest.ipynb: Random forest 모델 구축 및 예측에서 주요한 변수 분석
2. 1D-CNN.ipynb: 1D-CNN 모델 구축 및 예측에서 주요한 변수 분석
3. LSTM\_Attention.ipynb: LSTM&Attention 모델 구축 및 예측에서 주요한 변수 분석
4. Dual\_stage\_attention\_RNN.ipynb: DARNN 모델 구축 및 예측에서 주요한 변수 분석
5. requirements\_linux.txt: 프로젝트 공유나 환경 복원 시 필요한 외부 패키지 목록을 담은 파일(Linux)
6. requirements\_window.txt: 프로젝트 공유나 환경 복원 시 필요한 외부 패키지 목록을 담은 파일(Window)
7. df\_ext(2023-04-01~2023-08-31,51250385)\_2023-10-17 10-58-30 -seoultec.xlsx: 학습 및 검증을 위한 데이터셋

# 개발 환경

## Ubuntu (Linux)

* NVIDIA 드라이버 설치
  1. 그래픽 카드에 맞는 NVIDIA 드라이버 버전 확인

|  |
| --- |
| bash : ubuntu-drivers devices |

* 1. 그래픽카드에 호환되는 적절한 nvidia 드라이버 설치

|  |
| --- |
| bash : sudo apt-get install nvidia-driver-xxx |

* 1. nvidia 드라이버를 커널에 로드해주는 nvidia-modprobe도 같이 설치

|  |
| --- |
| bash : sudo apt-get install dkms nvidia-modprobe |

* 1. 설치한 nvidia 드라이버 작동을 위한 업데이트 및 재부팅

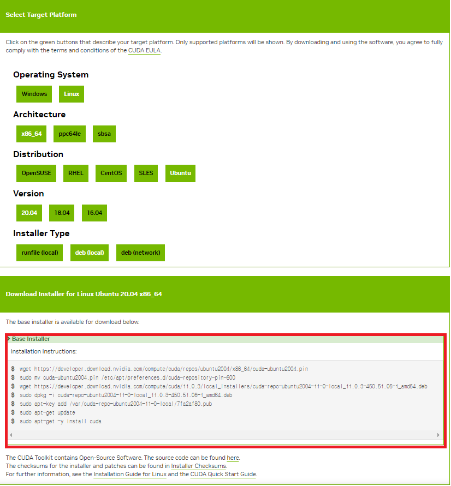
|  |
| --- |
| bash : sudo apt-get update  bash : sudo apt-get upgrade  bash : sudo reboot |

* CUDA 11.2 설치

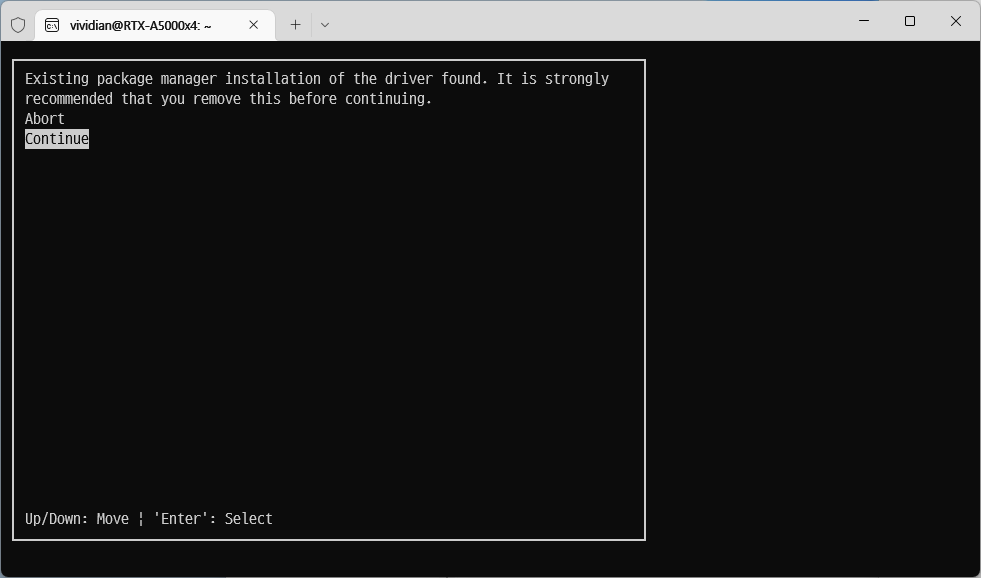
1. CUDA 설치 웹사이트 접속

<https://developer.nvidia.com/cuda-11.2.0-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=2004&target_type=deblocal>

1. NVIDIA에서 OS 조건에 맞는 CUDA11.2 선택



1. 위 이미지의 빨간 상자와 같은 파일 다운로드 명령어를 terminal에 입력
2. 다운로드 후 실행이 완료되면 설치 진행



1. 설치된 Driver 를 제외하고 CUDA Toolkit 11.2, CUDA Demo Suite 11.2, CUDA Documentation 11.2 설치
2. 설치가 완료되면 PATH 설정을 위해 .bashrc를 열여서 가장 마지막 부분에 export 부분을 입력해 설정하도록 하며, source ~/.bashrc를 통해 업데이트

|  |
| --- |
| bash : sudo vi ~/.bashrc # open file  export PATH=/usr/local/cuda-11.2/bin${PATH:+:${PATH}}  export LD\_LIBRARY\_PATH=/usr/local/cuda-11.2/lib64:${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}  bash : source ~/.bashrc # apply after save file |

1. PATH 설정 후 nvcc -V를 통해 설치 버전 확인

* CUDNN 8.1 설치

1. CUDNN 다운로드 웹사이트 접속

<https://developer.nvidia.com/rdp/cudnn-archive>

1. Linux (X85\_64)용 cuDNN Libary를 다운로드
2. 다운로드 받은 cuDNN 압축을 풀고, /usr/local/cuda 디렉토리로 복사

|  |
| --- |
| bash : sudo cp cuda/include/cudnn\*.h /usr/local/cuda-11.2/include  bash : sudo cp cuda/lib64/libcudnn\* /usr/local/cuda-11.2/lib64  bash : sudo chmod a+r /usr/local/cuda-11.2/include/cudnn\*.h /usr/local/cuda-11.2/lib64/libcudnn\* |

1. 설치 후 버전 확인

|  |
| --- |
| bash : cat /usr/local/cuda/include/cudnn\_version.h | grep CUDNN\_MAJOR -A 2 |

* Python 설치
* 터미널 실행 후 아래의 명령어를 통해 Python 3.8 설치

|  |
| --- |
| bash : sudo apt install python3.8 |

* 가상환경 생성
* 실행 시킬 파일이 존재하는 폴더로 진입 후, 아래의 명령어를 통해 가상환경 생성

|  |
| --- |
| bash : python -m venv 가상환경이름 |

* 가상환경 실행
* 아래의 명령어를 통해 가상환경 실행

|  |
| --- |
| bash : source 가상환경이름/bin/activate |

* 활용 라이브러리 설치

1. pytorch 설치: 아래의 명령어를 통해 가상환경 내 pytorch 설치

|  |
| --- |
| bash : pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch\_stable.html |

1. tensorflow 설치: 아래의 명령어를 통해 가상환경 내 tensorflow 설치

|  |
| --- |
| bash : pip install tensorflow-gpu==2.10 |

1. 나머지 package 설치: 아래의 명령어를 통해, 나머지 필요 package들을 설치

|  |
| --- |
| bash : pip install -r requirements\_linux.txt |

## Window

* NVIDIA 드라이버 설치
* Driver 설치 웹사이트 접속 <https://www.nvidia.co.kr/Download/index.aspx?lang=kr>
* 그래픽 카드에 맞는 NVIDIA 드라이버를 다운 및 설치
* CUDA 11.2 설치

다운로드 링크

1. CUDA 설치 웹사이트 접속

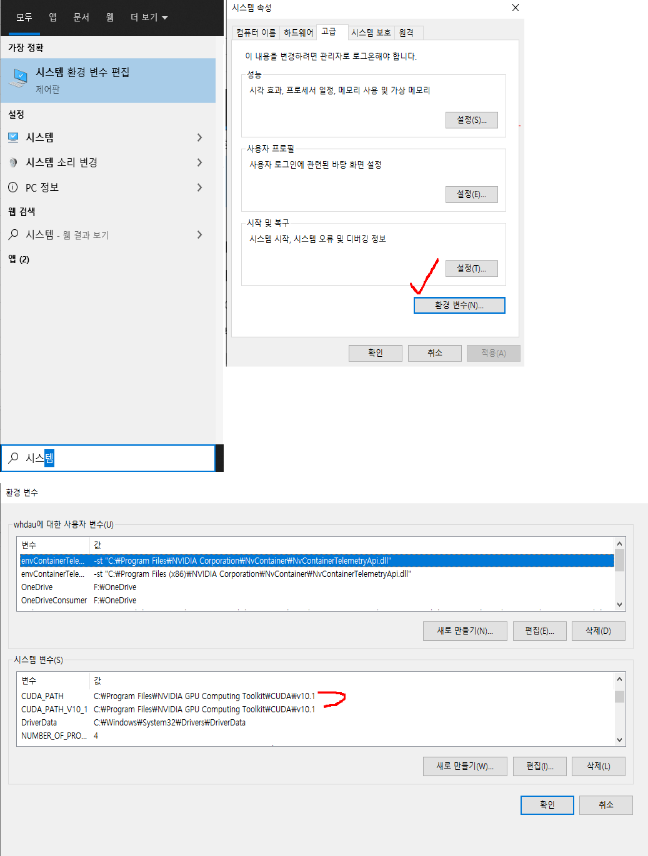
<https://developer.nvidia.com/cuda-11.2.0-download-archive?target_os=Windows&target_arch=x86_64>

1. NVIDIA에서 CUDA11.2 설치 파일 다운로드

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

1. CUDA 11.2 설치
2. 설치확인



텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* CUDNN 8.1 설치

1. CUDNN 다운로드 웹사이트 접속

<https://developer.nvidia.com/rdp/cudnn-archive>

1. NVIDIA에서 CUDNN 8.1 파일 다운로드
2. 다운로드한 압축 파일 해제
3. 압축 해체한 파일을 설치한 CUDA 폴더에 덮어씌우기
4. CUDA 경로 : C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2

* Python 설치
  1. Python 공식 웹사이트에 접속하여 Python 3.8 설치 파일 다운로드

(<https://www.python.org/downloads/>)

* 1. 다운로드한 설치 프로그램을 실행
  2. 설치 과정을 시작하기 전에 "Add Python 3.8 to PATH" 옵션을 선택
  3. "Install Now"을 선택하여 Python 3.8을 설치
* 가상환경 생성
* 실행시킬 파일이 존재하는 폴더로 진입 후, 아래의 명령어를 통해 가상환경 생성

|  |
| --- |
| bash : python -m venv 가상환경이름 |

* 가상환경 실행
* 아래의 명령어를 통해 가상환경 실행

|  |
| --- |
| bash : 가상환경이름\Scripts\activate |

* 활용 라이브러리 설치
  1. pytorch 설치: 아래의 명령어를 통해 가상환경 내 pytorch 설치

|  |
| --- |
| bash : pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch\_stable.html |

* 1. ensorflow 설치: 아래의 명령어를 통해 가상환경 내 tensorflow 설치

|  |
| --- |
| bash : pip install tensorflow-gpu==2.10 |

* 1. 나머지 package 설치: 아래의 명령어를 통해, 나머지 필요 package들을 설치

|  |
| --- |
| bash : pip install -r requirements\_window.txt |

# 주요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
|  | 라이브러리 | 버전 | 용도 |
| Pytorch | torch | 1.7.1+cu110 | 딥러닝 라이브러리 |
| torchaudio | 0.7.2 | PyTorch의 오디오 데이터를 처리하기 위한 확장 |
| torchmetrics | 0.5.0 | PyTorch 모델을 위한 머신러닝 메트릭 모음 |
| torchvision | 0.8.2+cu110 | 이미지 처리를 위한 유용한 데이터셋, 모델, 변환 제공 라이브러리 |
| pytorch-lightning | 1.3.8 | PyTorch의 래퍼로, 더 쉽고 빠른 실험을 위한 라이브러리 |
| Tensorflow | tensorflow-gpu | 2.10.0 | GPU 지원 TensorFlow, 머신러닝 및 신경망 라이브러리 |
| tensorflow-estimator | 2.10.0 | TensorFlow를 사용하여 학습, 평가, 예측을 위한 모델 정의 라이브러리 |
| tensorboard | 2.10.1 | TensorFlow 실행 시각화 도구 |
| tensorboard-data-server | 0.6.1 | TensorBoard에 데이터를 제공하여 모델 학습 진행 상황 및 성능을 시각적으로 모니터링 |
| tensorboard-plugin-wit | 1.8.1 | TensorBoard의 시각화를 확장하여 데이터의 시각적 분석을 개선하는 플러그인 |
| tensorflow-io-gcs-filesystem | 0.34.0 | TensorFlow와 Google Cloud Storage(GCS) 파일 시스템 간 상호 작용을 위한 입출력 모듈 |
|  | numpy | 1.24.4 | 수학 연산을 위한 기본 python패키지 |
|  | pandas | 2.0.3 | 데이터 조작 및 분석을 위한 고수준 데이터 구조 및 조작 도구 |
|  | shap | 0.39.0 | 모델의 예측을 설명하기 위한 라이브러리 |
| Scikit-learn | scikit-learn | 1.3.1 | 머신러닝 알고리즘 및 도구를 제공하는 라이브러리 |
| scikit-image | 0.21.0 | 이미지 분석과 조작을 지원하는 과학적 이미지 처리를 위한 라이브러리 |
| Math | sympy | 1.12 | 수학적 기호 계산을 위한 라이브러리 |
| mpmath | 1.3.0 | 고도의 수학적 기호 및 수치 계산을 위한 라이브러리 |
| matplotlib | matplotlib | 3.7.3 | python에서 2D 도표를 생성을 위한 라이브러리 |
| matplotlib-inline | 0.1.6 | Matplotlib 그래프를 직접 출력하기 위한 도구를 제공하는 라이브러리 |
|  | seaborn | 0.13.0 | matplotlib 기반의 통계적 그래픽 라이브러리 |

# 실행 방법 및 예시

* 주피터 노트북 실행 후 각 파일 실행

|  |
| --- |
| bash : jupyter notebook |

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

* 모델 실행 결과 모델명의 폴더 생성이 된 후, 다음과 같은 파일들 생성

|  |  |  |
| --- | --- | --- |
| 모델명 | 파일명 | 파일 설명 |
| Random Forest | model\_performance.txt | 학습한 모델을 테스트한 결과가 저장된 파일 |
| Time series prediction plot.png | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| Scatter plot.png | 학습된 모델의 예측값 대 실제값 비교 산점도 |
| Random Forest feature plot.png | 모델의 주요 특성 중요도 10개를 추출하고 이를 시각화 |
| 1D-CNN | best\_model.pth | Validation loss가 가장 낮은 epoch 에서의 모델 weight를 저장해둔 파일 |
| last\_model.pth | 가장 마지막에 학습한 모델 weight를 저장해둔 파일 |
| model\_performance.txt | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| loss.png | 모든 epoch에 대해 train loss와 validation loss 그래프 |
| Time Series Prediction.png | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| scatter plot.png | 학습된 모델의 예측값 대 실제값 비교 산점도 |
| shap summary plot.png | 모델의 각 특성이 예측에 미치는 영향력과 방향을 색상과 SHAP 값으로 나타낸 그래프 |
| cumulative shap plot.png | 테스트 데이터셋에 대해 각 특성별로 평균 절대 SHAP 값의 크기를 시각화 한 그래프 |
| LSTM-Attention | best\_model.pth | Validation loss가 가장 낮은 epoch 에서의 모델 weight를 저장해둔 파일 |
| last\_model.pth | 가장 마지막에 학습한 모델 weight를 저장해둔 파일 |
| model\_performance.txt | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| loss.png | 모든 epoch에 대해 train loss와 validation loss 그래프 |
| Time Series Prediction.png | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| scatter plot.png | 학습된 모델의 예측값 대 실제값 비교 산점도 |
| Average Attention Map.png | Time stamp 관련 attention weights 시각화 |
| shap summary plot.png | 모델의 각 특성이 예측에 미치는 영향력과 방향을 색상과 SHAP 값으로 나타낸 그래프 |
| cumulative shap plot.png | 테스트 데이터셋에 대해 각 특성별로 평균 절대 SHAP 값의 크기를 시각화 한 그래프 |
| DARNN | best\_model.ckpt.index | 텐서플로우/케라스 모델의 변수 이름과 그들의 저장 위치를 매핑하는 인덱스 정보 |
| best\_model.ckpt.data-00000-of-00001 | 모델의 실제 가중치와 매개변수를 저장하는 바이너리 데이터 파일 |
| loss.png | 모든 epoch에 대해 train loss와 validation loss 그래프 |
| Time Series Prediction.png | 시간에 따른 학습 모델의 예측값과 실제값 비교 그래프 |
| scatter plot.png | 학습된 모델의 예측값 대 실제값 비교 산점도 |
| Feature Attention Map.png | Feature 관련 attention weights 시각화 |
| Time Stamp Attention Map.png | Time stamp 관련 attention weights 시각화 |