
스마트팩토리 지능화 빅데이터 교육

[교육 일자]

- 5월 27일(월) ~ 5월 30일(목) 18:00~22:00 총 4일/16시간

[교육 강사]

- 김 의 철 박사
- 서울벤처대학원대학교 산업협력단 부설연구소 WISE SMART 센터장
- 영남이공대 사이버보안과 인공지능 빅데이터 강사

[교육 목표]

(1) SMART 산업 IT 기술에 사용되는 Python 학습

- python 을 활용할 수 있는 사용 환경을 구성 할 수 있다.
- python 코드를 이해하고 사용할 수 있다.

(2) Python 을 이용한 데이터를 다루는 기본 기능 실습

- python을 사용하여 데이터 조작 기본 기능 습득
- python을 사용하여 탐색적 데이터 분석 기본 기능 습득
- 데이터 탐색과 분석 결과에 대한 시각화 기본 기능

(3) Python 을 이용한 데이터 분석 예제를 통한 분석 모델 이해

- 머신러닝 분석 모델 예제를 통해서 분석 과정을 이해한다.
- 딥러닝 분석 모델 예제를 통해서 인공지능 학습 과정을 이해한다.

[1 일차]

CH 0. 오리엔테이션 - 교육 내용 소개

CH 1. 스마트 산업과 빅데이터 인공지능 배경

- 1-1. 4차 산업혁명과 스마트 산업
- 1-2. SMART 인공지능 산업에서 데이터를 다루는 기술
- 1-3. SMART 인공지능 산업에서 머신러닝과 AI

CH 2. 파이썬 기초

- 2-1 기초 코딩의 이해 및 파이썬 배경
- 2-2 파이썬 설치 및 코딩 환경 설치
 - **Local** 환경을 활용할 경우 (파이썬 유경험자 및 코딩 경험자 경우)
 - **Google Colaboratory** 환경 (파이썬 입문자 경우)
- 2-3 파이썬 코딩의 기초

[2 일차]

CH 3. 데이터의 이해와 파이썬을 이용한 조작

- 3-1 파이썬을 이용한 데이터 다루기 이해
- 3-2 파이썬 - 데이터 다루기 기초 : Numpy, Pandas
- 3-3 데이터 저장 : SQLite

[3 일차]

CH 4. 데이터 분석 기초와 시각화

- 4-1 데이터 분석 기초 : ETL, EDA, 분석 라이브러리 사용 소개
- 4-2 데이터 및 분석결과 시각화

[4 일차]

CH 5. 머신러닝과 딥러닝 분석모델

- 5-1 머신러닝 예제를 통한 학습 과정 이해
- 5-2 딥러닝 예제를 통한 학습 과정 이해

[예비 추가 진행 내용]

예비추가. 1. 모델의 구체화

- fastAPI
- PySide 6

예비추가. 2. LLM 모델을 사용해보기

- Langchain

[1 일차]

CH 1. 스마트 산업과 빅데이터 인공지능 배경

[CH 1 - 학습 목표]

1. 산업 IT 기술이 ICT 기술에서 smart 기술로의 변화되는 4차산업 배경 이해
2. 데이터 수집 관리 기술 및 정보 분석을 위한 머신러닝과 인공지능의 역할을 이해
3. 최근 스마트 기술 산업의 동향

1-1. 4차 산업혁명과 스마트 산업

- 3차 산업혁명은 ICT : 4차 산업혁명 SMART 인공지능으로 변화와 차이

- 3차 산업혁명을 주도한 ICT(Information and Communication Technology)는 정보의 생성, 저장, 처리, 전송하는 컴퓨터, 소프트웨어, 네트워크, 인터넷 등의 기술이 핵심을 이루는 지식정보 혁명.
- 4차 산업혁명은 머신러닝, 사물인터넷(IOT), 빅데이터, 클라우드 등의 기술을 바탕으로 단순한 정보의 연결이 아니라
 1. 초-연결성 (hyperconnectivity) : 디지털 트윈, VR, AR
 2. 초지능화(superintelligence) : 인공지능,
 3. 예측(prediction) : 다차원 빅데이터를 이용한 동적상태예측과 지능적 자율판단
 4. 융합기술(Convergence technologies) : 예술, 금융, 기술 등 전분야 융합 가능성

- 주요 기술

인공 지능(AI), 사물 인터넷(IoT), 클라우드 컴퓨팅(Cloud Computing), 빅 데이터(Big Data), 모바일(Mobile), 무인 운송 수단, 3D 프린팅(3D Printing), 나노 기술(Nano Technology), 로봇 공학(Robot Engineering)

(1) Digital Twin

Nvidia omniverse(<https://www.nvidia.com/ko-kr/omniverse/>)

(2) VR Graphic

메타휴먼(<https://www.unrealengine.com/ko/metahuman>)

메타휴먼 예시(https://www.youtube.com/watch?v=7IAWhk_aVvc)

(3) AI Agent

game AI Agent(https://youtu.be/5R8xZb6J3r0?si=G8lISzmVT_cldCPG)

(4) Robot

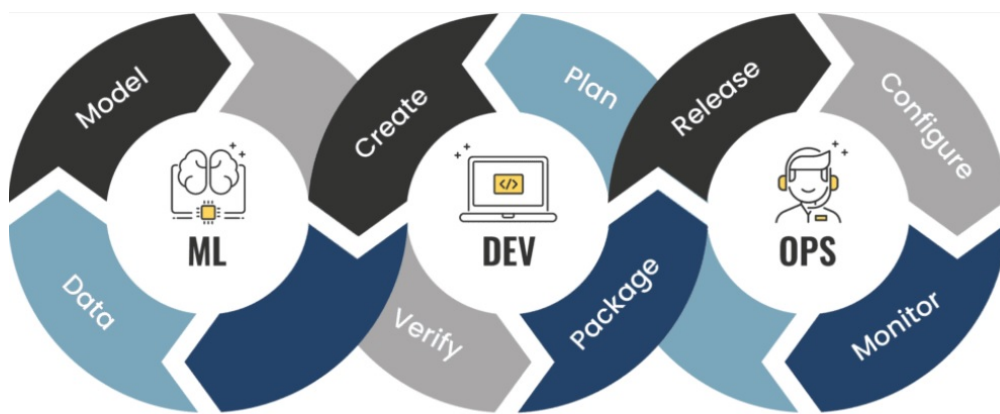
openAI Figure 1(<https://www.youtube.com/watch?v=Sq1QZB5baNw>)

boston dynamics(https://youtu.be/29ECwExc-_M?si=b5t4SDXd9pIO073D)

Tesla optimus(<https://youtu.be/vibNjdbtaEM?si=WF7MnrVdfu8T4GuG>)

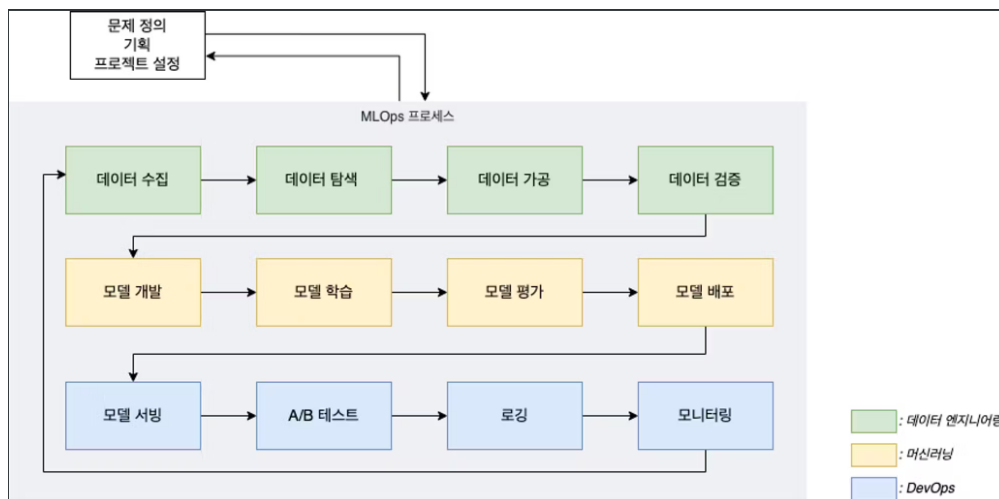
- 산업현장의 현실적인 개념: 현장의 데이터 수집과 관리 자동화에서 분석과 통제 자동화
 - 현장의 상황과 정보를 수집 전달 한다. (센서와 계측) : 현장처럼 자세히 모니터링.
 - 결국 단순한 수집 전달이 아니라 통제 관리자와 현장이 상호작용.
 - 현장 기기와 중앙 시스템에서 인공지능이 관리자의 논리적 관리 기능을 수행.

1-2. SMART 인공지능 산업에서 데이터를 다루는 기술



MLOps = ML + Dev + Ops 출처: <https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference>

- MLOps의 과정



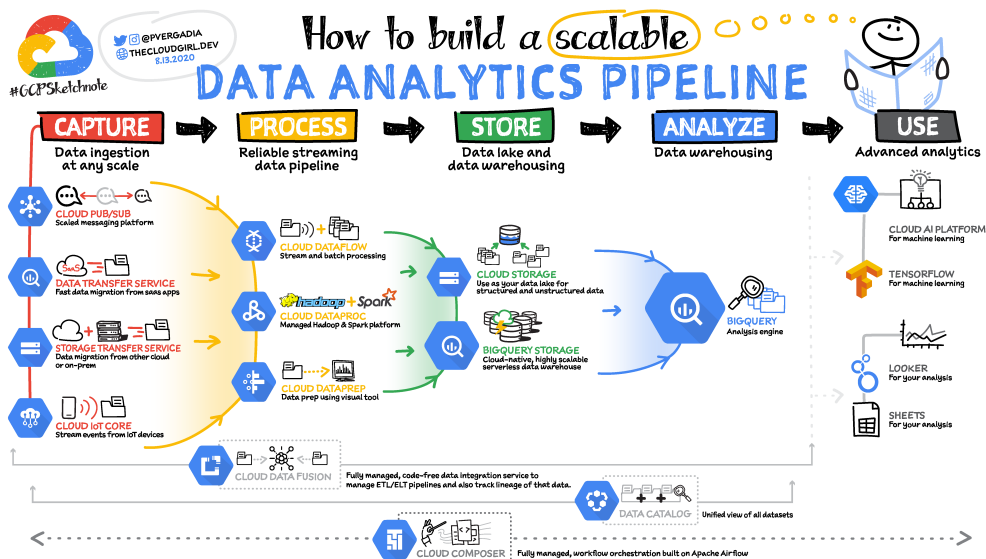
MLOps 과정 출처: <https://blog.taehun.dev/from-zero-to-hero-mlops-tools-1>

- smart system의 Data Warehouse (서비스에 필요한 데이터를 수집, 가공, 저장, 분석 하여 데이터마트를 중심으로 제공하는 시스템)
- Data Pipeline인 구축과 Data Lake (모든 raw한 data의 원천)

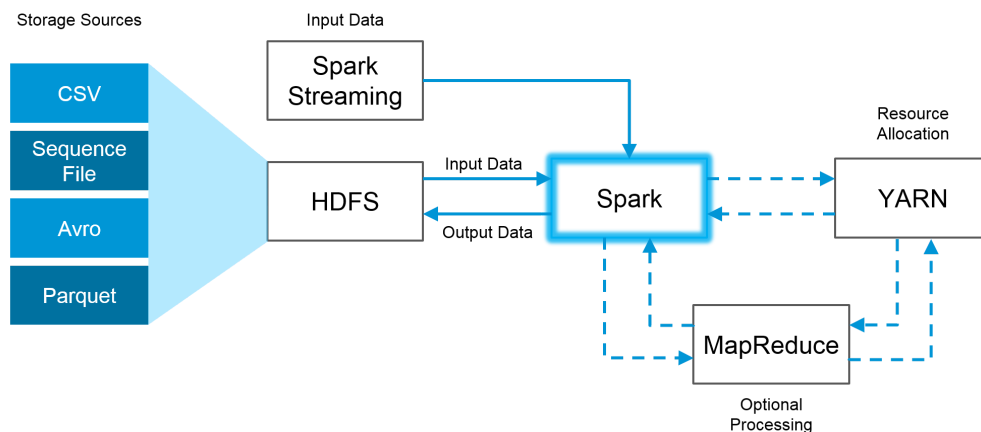


dbt-snowflake-airflow 출처:

https://quickstarts.snowflake.com/guide/data_engineering_with_apache_airflow/index.html#0



출처: <https://www.freecodecamp.org/news/scalable-data-analytics-pipeline/>



출처: Spark is fully compatible with the Hadoop eco-system and works smoothly with HDFS
(<https://towardsdatascience.com>)

1-3. SMART 인공지능 산업에서 머신러닝과 AI

- 기계적 과정의 자동화에서 논리적 과정의 자동화
- 데이터로부터 법칙 발견 - AI 딥러닝
- 피드백 (결과를 전달하는 기술)











CH 2. 파이썬 기초

[CH 2 - 학습 목표]

- 1. SMART 산업의 IT 기술 및 DATA 분석에 주류 프로그램 언어인 Python의 기초 배경 이해
- 2. Python을 사용하기 위한 환경설정을 구성하고 이해
- 3. Python 기초 문법을 학습하고 코드의 기초 내용을 이해

2-1 기초 코딩의 이해 및 파이썬 배경

- 스마트 산업 기술에서의 파이썬 프로그래밍 언어의 위치

Jan 2024	Jan 2023	Change	Programming Language		Ratings	Change
1	1			Python	13.97%	-2.39%
2	2			C	11.44%	-4.81%
3	3			C++	9.96%	-2.95%
4	4			Java	7.87%	-4.34%
5	5			C#	7.16%	+1.43%
6	7	▲		JavaScript	2.77%	-0.11%
7	10	▲		PHP	1.79%	+0.40%
8	6	▼		Visual Basic	1.60%	-3.04%
9	8	▼		SQL	1.46%	-1.04%
10	20	▲		Scratch	1.44%	+0.86%

2024년 1월 TIOBE 프로그래밍 언어 순위 (출처: <https://www.tiobe.com/tiobe-index/>)

여전히 최고로 많은 커뮤니티를 지닌 언어.

접근성과 사용자 집단의 영향력

- AI 시대의 프로그래밍과 Python의 전망

(<https://pyscript.net/>) - Front로 발전가능성?? 의문

- 속도와 성능의 발전 가능성

<https://www.modular.com/max/mojo> - 기존의 인터프리터 언어의 한계를 벗어날 것인가?

- 파이썬 언어의 기초 배경 지식

완전한 독자적인 어플리케이션 개발은 아직까지 어려움.

그러나 머신러닝과 AI 발전에 깊게 뿌리 박고 있어 한동안 대체가 어려움.

자체적으로 발전하며 다른 플랫폼에서 진화하는 중.

2-2 파이썬 설치 및 코딩 환경 설치

<https://www.python.org/>



- anaconda 등의 python 관련 플랫폼 환경도 있도 좋은 방법임.
- IDE 로 pycharm, spider 등도 많이 사용됨.

Local 환경을 활용할 경우 (파이썬 기초 이상 경우)

- 개발 환경 준비
 1. Jupyter Lab (<https://jupyter.org/>)
 2. Visual Studio Code (<https://code.visualstudio.com/>)
- python 버전 관리 및 의존성 관리

(1) pyenv : python 버전 관리

1. 권한조정

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine
```

2. pyenv-win install

```
Invoke-WebRequest -UseBasicParsing -Uri "https://raw.githubusercontent.com/pyenv-win/pyenv-win/master/pyenv-win/install-pyenv-win.ps1" -OutFile ".\install-pyenv-win.ps1"; &"./install-pyenv-win.ps1"
```

3. 환경변수 설정

```
# STEP 1 :: PYENV 설정
[System.Environment]::SetEnvironmentVariable('PYENV',$env:USERPROFILE +
"\.pyenv\pyenv-win\","User")
[System.Environment]::SetEnvironmentVariable('PYENV_ROOT',$env:USERPROFILE +
"\.pyenv\pyenv-win\","User")
[System.Environment]::SetEnvironmentVariable('PYENV_HOME',$env:USERPROFILE +
"\.pyenv\pyenv-win\","User")

# STEP 2 :: PATH에 추가설정 (일반적으로 PYENV 설치 중 자동입력됨)
[System.Environment]::SetEnvironmentVariable('path', $env:USERPROFILE +
"\.pyenv\pyenv-win\bin;" + $env:USERPROFILE + "\.pyenv\pyenv-win\shims;" +
[System.Environment]::GetEnvironmentVariable('path', "User"),"User")

# 경로확인
$env:Path

# pyenv 실행확인
pyenv

# pyenv에서 설치가능한 python 버전리스트 update
pyenv update
pyenv install --list
pyenv versions # 설치된 python 확인
pyenv global [특정 버전]
pyenv local [특정 버전]
```

(2) poetry : python project 라이브러리 의존성 관리

<https://python-poetry.org/>

```
poetry new          # 프로젝트 디렉토리 구조 생성
poetry init         # 기존 경로에 toml 파일 생성

poetry add [라이브러리] # 해당 라이브러리 설치
poetry remove [라이브러리] # 해당 라이브러리 제거
poetry search [라이브러리] # 해당 라이브러리 검색

poetry install # 의존성 설치
poetry update  # 의존성 라이브러리 업데이트
poetry show --tree # 의존성 확인 (트리구조로)

poetry shell # 가상환경 진입 , toml 파일이 있어야함.
poetry env info # 가상환경 정보
```

Google Colaboratory 환경 (파이썬 입문자 경우)

- Google 계정 생성
- Drive 환경 설명
- Colaboratory 사용 환경 설명

2-3 파이썬 코딩의 기초

예제 주소:

```
https://github.com/hugwolf77/SVU_industrial_AI/tree/main
```

- 파이썬 기초 문법
- 예제 코드를 통한 파이썬 기초 문법 실습

< 2 일차 >

CH 3. 데이터의 이해와 파이썬을 이용한 조작

[CH 3 - 학습 목표]

1. Python 생태계에서 데이터를 다루는 환경 이해
2. python 의 데이터 조작 도구 라이브러리 이해
3. python 과 SQL 의 연동 이해

3-1 파이썬을 이용한 데이터 다루기 이해

- python 생태계에서 데이터 다루기
 - (1) 저장장소
 - local 디스크 : file, DB, 메모리
 - Cloud 저장소 : cloud DB, aws-S3, azure-storage, GCP-Drive, hadoop, NetDB
 - (2) 저장형태
 - 스프레드시트 형태
 - DB형태
 - NoSQL(MongoDB, Cassandra), SQL(PstgrasQL, Orcle, SQL-server, MariaDB, MySQL)
 - VectorDB (Pinecone, Chroma, Weaviate, Faiss, Qdrant), Time-seriesDB(fluxDB, OpenTSDB), GraphDB(Neo4j, Graphite), RDBMS
 - 특정 데이터 포맷 형태
 - serialized, Blob
 - CSV, Json, XML, YAML
 - Doc, Image, Audio, Video
- python 소스코드에서 데이터를 다루는 라이브러리
 - serialized : pickle, shelve, dill
 - json, xml, PyYAML
 - array : numpy => Numba (JIT-complier), Cupy
 - DataFrame : pandas => GeoPandas, Dask, Ray, Vaex, Polars
- python 을 이용한 DB 엔진 활용과 SQL 기초의 이해

3-2 파이썬 - 데이터 다루기 기초

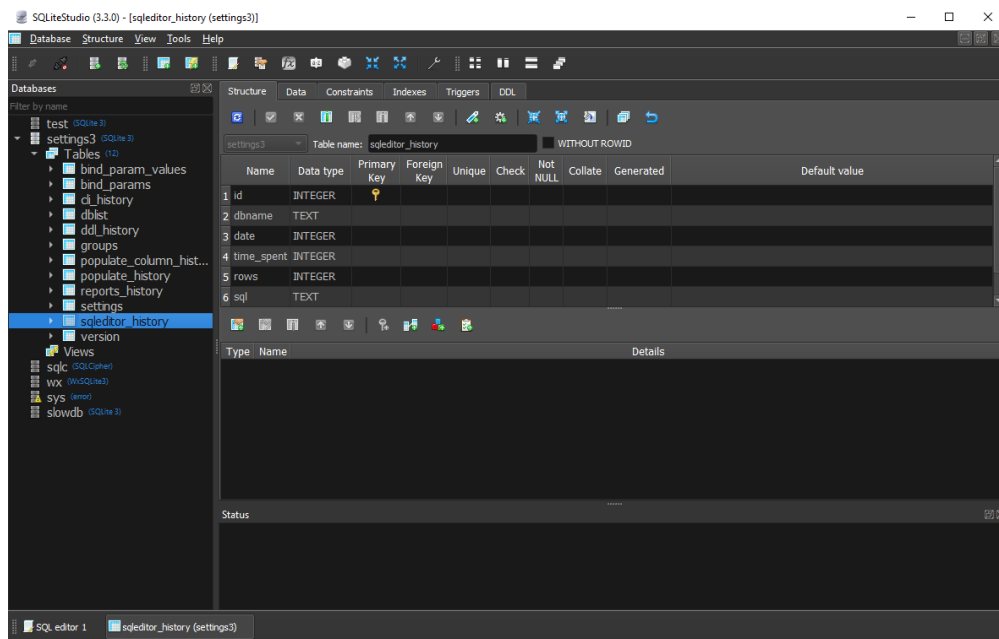
- Numpy 예제 실습 : 수치분석을 위한 라이브러리
- Pandas 예제 실습 :

3-3 데이터 저장

- 데이터베이스와 SQL 소개
 - 특수한 타입과 목적의 시스템이 아닌 이상 여전히 메인 DB 형태는 RDBMS와 NoSQL 형태의 DB 엔진들이다, 그중에서도 SQL를 사용하여 데이터를 관리하는 것이 기본이 되고 있다.
 - *SQL:Structured Query Language*
 - a. DB에서 데이터를 추출하고 조작하는 데에 사용하는 데이터 처리 언어
 - b. DB에 저장된 데이터를 효율적으로 추출하고 분석할 수 있기 때문에 SQL은 빅-데이터를 다루기 위한 필수적인 언어로 자리 잡음. (기초적인 쿼리 언어 간단한 절차나 통계는 가능)
 - c. 하지만 복잡한 모델이나 분석기법은 사용할 수 없음
 - *SQLite3*
 - a. *SQLite*는 별도의 서버 프로세스가 필요 없고 SQL 언어의 비표준 변형을 사용하여 데이터베이스에 액세스할 수 있는 경량 디스크 기반 데이터베이스를 제공하는 C 라이브러리.
 - c. *SQLite*를 사용하여 응용 프로그램을 프로토타입 한 다음 PostgreSQL 이나 Oracle과 같은 더 큰 데이터베이스로 코드를 이식할 수 있음.
 - c. *SQLite3* 는 *SQLite* 데이터베이스용 python 기본 인터페이스 라이브러리 모듈
 - *Sqlalchemy*
 - ORM:Object Relational Mapping : 객체지향 언어를 이용하여 호환되지 않는 type system 간의 데이터 전환을 의미. 관계형 데이터베이스에 객체지향 언어로 접근할 수 있게끔 매핑을 해주는 다리 역할.
 - 1. 장점
 - SQL을 별도로 익히지 않아도 DB를 활용.
 - DB를 변경할 때 쿼리를 하나하나 수정하지 않아도 됨.
 - SQL injection 를 방지할 수 있음.
 - 2. 단점
 - SQL을 아는 사람이라면 ORM을 또 별도로 배워야 함..
 - 복잡한 쿼리가 필요한 경우 성능 저하를 일으키거나 ORM으로 치환이 어려움. => 최근 업데이트로 foreign key, constraints 등과 형상(schema)를 지원하는 기능이 강화되면서 다시 고려해 보아야 할 관점이 됨.
- python 과 *SQLite* 를 이용한 RDBMS 기초 이해 및 실습

https://github.com/hugwolf77/SVU_industrial_AI/tree/main

* SQLite 를 위한 DB-client
[SQLite-STUDIO](<https://sqlitestudio.pl/>)



< 3 일차 >

CH 4. 데이터 분석 기초와 시각화

[CH 4 - 학습 목표]

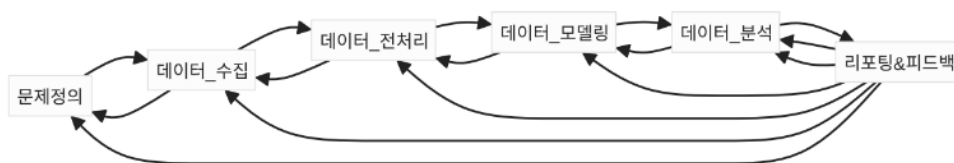
1. 데이터 분석의 기초 과정의 이해
2. 데이터 ETL 과 EDA 의 이해
3. 데이터 분석과 시각화 기능 이해

4-1 데이터 분석 기초

- ETL과 EDA에 대한 이해
- 데이터를 이용한 기본적인 정보 분석 라이브러리 사용 소개
- scikit-learn, scipy, simpy, statemodel

분석과 모델개발 과정

데이터 분석 절차 (데이터 분석에도 해당)



(1) 문제정의 :

목표와 목적에 의해서(분석을 통해서 목표하는 또는 기대하는 결과) 분석하고자 하는 대상에 대한 구체적 문제정의(대상정의)

(2) 데이터 원천 선정:

문제정의로 부터 분석에 필요한 데이터와 데이터 원천에 대하여 확정한다.

(3) 데이터 수집:

확정된 데이터 원천으로 부터 데이터를 수집한다.

(4) 데이터의 검정 및 평가 :

수집된 데이터의 정확성, 일관성, 완전성 등 수집된 데이터를 검정하고 사용할수 있는 데이터 인지 평가

(5) 수집된 데이터의 선택 :

분석에 사용될 수 있는지 실효성 등을 평가하여 변수 선택.

(6) 데이터 전처리: preprocessing

결측치 처리, 이상치 제거, 변수 변환 등을 통해서 분석 모델에 사용할 수 있도록 정제하는 과정

(7) 데이터 모델링:

문제정의에 맞게 데이터를 분석하기 위한 분석 기법, 분석 모델을 수립하는 과정

(8) 데이터 분석:

실제 분석 기법 및 모델을 적용하여 데이터를 분석

(9) 리포팅&피드백:

최종 분석 결과를 보고서로 작성 및 결과에 대한 피드백 확인

1. ETL (EXTRACT, TRANSFORM, LOAD)

2. EDA (EXPLORATORY DATA ANALYSIS)

4-2 데이터 및 분석결과 시각화

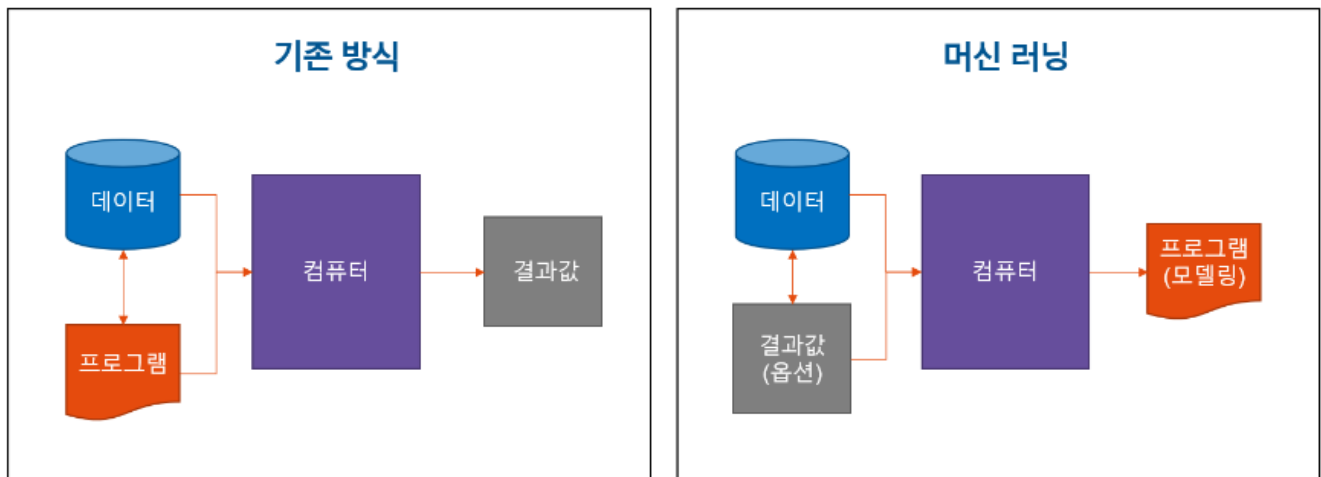
- matplotlib, seaborn 예제
- ployly, Dash,

< 4 일차 >

CH 5. 머신러닝과 딥러닝 분석모델

[CH 5 - 학습 목표]

1. 데이터를 분석 모델의 기초 이해
2. 머신러닝 과정 기초 이해
3. 딥러닝 과정 기초 이해



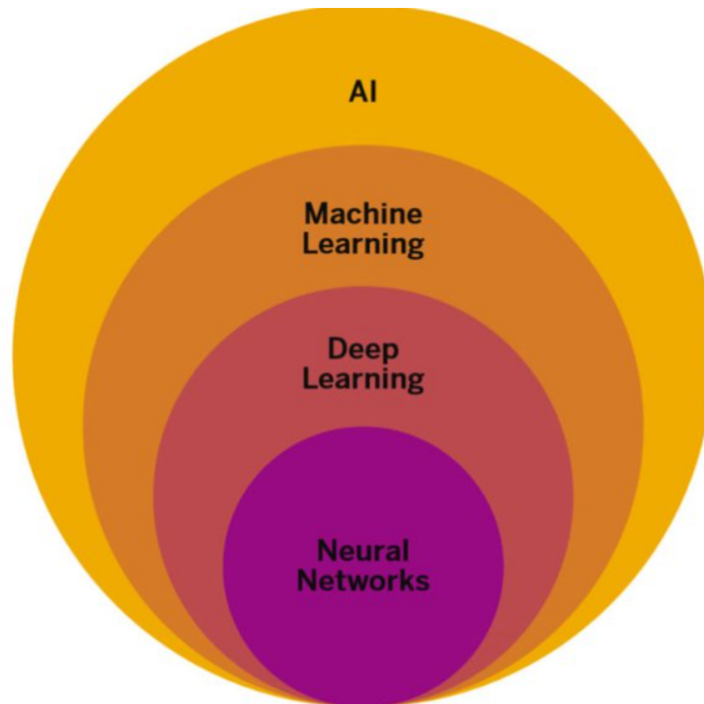
분석모델의 학습 형태에 따른 분류

- 지도 학습(Supervised Learning)
- 비지도 학습(Unsupervised Learning)
- 준지도 학습(Semi-Supervised Learning)
- 강화 학습(Reinforcement Learning)

기존 분석방식과 머신러닝 분석 방식의 차이 : 접근 관점과 방법 차이

- 논리적 절차와 파라미터를 사람이 설정 후 결과값을 통해서 검증하고 보완하는 방식
- 데이터로 부터 결정절차와 파라미터 등의 규칙을 찾아내는 과정

분석방식의 범위



분석모델의 목적에 따른 분류

1. 분류 모델:

- 데이터 포인트가 특정 카테고리에 속하는지 예측하는 데 사용됩니다.
- 예시: 스팸 메일 필터링, 고객 유형 분류, 신용 위험 평가

2. 예측 모델:

- 과거 데이터를 기반으로 미래 값을 예측하는 데 사용됩니다.
- 예시: 주가 예측, 매출 예측, 고객 이탈 예측

3. 군집화 모델:

- 유사한 특성을 가진 데이터 포인트를 그룹으로 묶는 데 사용됩니다.
- 예시: 고객 세분화, 시장 조사, 생물학적 데이터 분석

4. 이상 탐지 모델:

- 데이터에서 정상 범위를 벗어나는 데이터 포인트를 식별하는 데 사용됩니다.
- 예시: 사기 거래 감지, 네트워크 침입 감지, 의료 진단

5. 추천 모델:

- 사용자에게 적합한 제품, 서비스 또는 콘텐츠를 추천하는 데 사용됩니다.
- 예시: 온라인 쇼핑 추천, 영화 추천, 음악 추천

기타 목적 모델

1. 생성형 모델:

- 사용자에게 적합한 제품, 서비스 또는 콘텐츠를 추천하는 데 사용됩니다.
- 예시: 온라인 쇼핑 추천, 영화 추천, 음악 추천

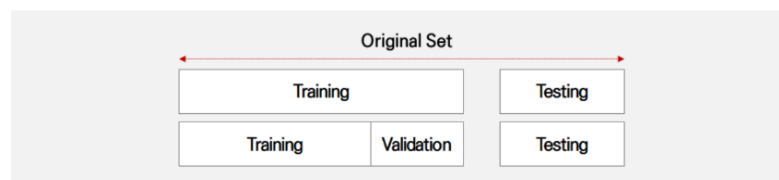
2. AGI:

- 사용자에게 적합한 제품, 서비스 또는 콘텐츠를 추천하는 데 사용됩니다.
- 예시: 온라인 쇼핑 추천, 영화 추천, 음악 추천

5-1 머신러닝 예제를 통한 학습 과정 이해

1. data preprocessing : 모델에 맞게 데이터 형태 조정
2. data split : 학습데이터와 테스트 데이터로 분리

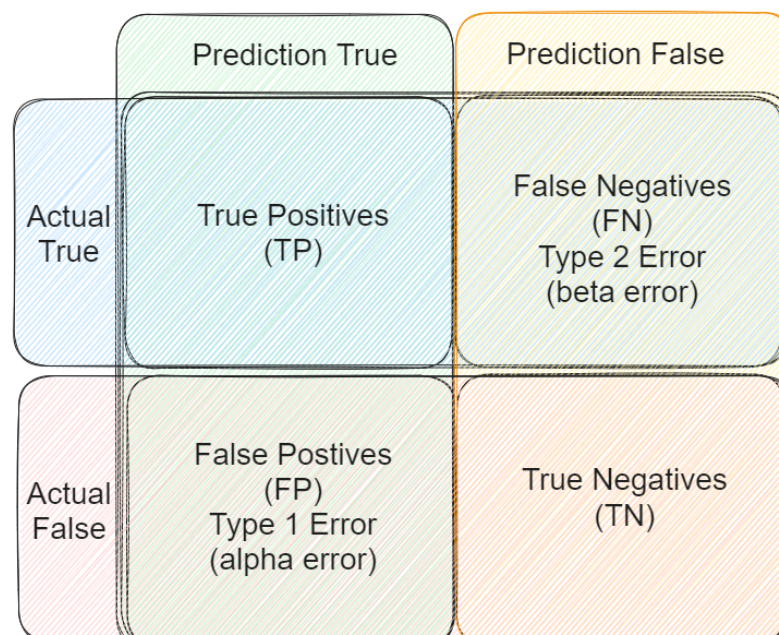
학습 데이터의 분할



3. model training : 모델 학습
4. model test : 모델의 성능 테스트
5. model tuning : 모델 조정

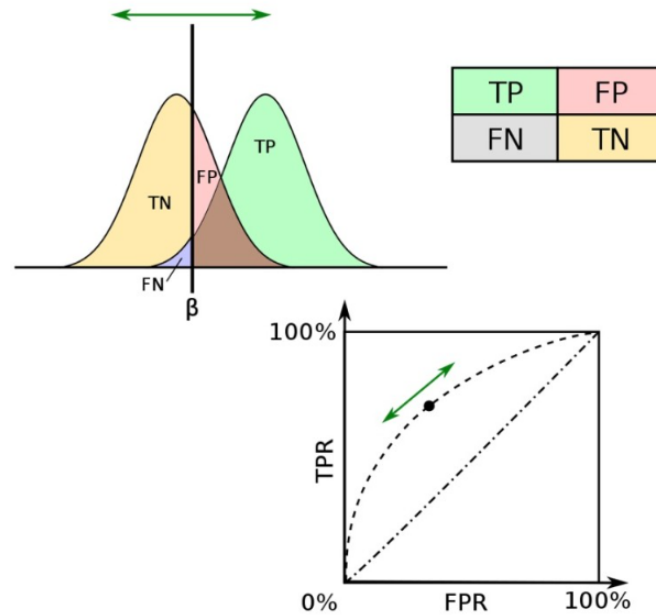
5-2 데이터 및 분석결과 시각화

- Confusion Matrix



- $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FN} + \text{FP})$
- $\text{Recall}(\text{Sensitivity}) = (\text{TP}) / (\text{TP} + \text{FN})$
- $\text{Specificity} = (\text{TN}) / (\text{TN} + \text{FP})$
- $\text{Precision} = (\text{TP}) / (\text{TP} + \text{FP})$
- $\text{F1-score} = (\text{Recall} \times \text{Precision}) / ((\text{Recall} + \text{Precision}))$: 일종의 조화평균

AUC and ROC (Reciever Operating Characteristic curve and Area Under the Curve)



- ROC curve (Reciever Operating Characteristics, 수신자 조작 특성 곡선)
- 기준값(threshold)이 달라짐에 따라 분류모델의 성능이 어떻게 변하는지를 나타내기 위해 그리는 곡선.
- 다양한 분류 임계값에서 TPR과 FPR을 보여줌

AUC (Area Under the Curve)

- ROC 곡선 아래 영역, 가능한 모든 분류 임계값에서 집계된 성능 측정값을 제공

[예비 추가 진행 내용]

예비. 1. 모델의 구체화

- fastAPI

: FastAPI는 Python을 기반으로 개발된 현대적이고 빠른 웹 프레임워크로, 특히 API 개발에 초점을 맞추어 설계되었고 높은 성능과 직관적인 문법을 제공

- python framework

- Django
- Flask
- FastAPI
- 기타 : Pyramid, bottle, Tornado, Falcon, CherryPy, Sanic

- PySide 6 : QT 계열의 윈도우 GUI 프레임워크

- PyQt
- Tkinter (and Bootstrap)
- Kivy
- 기타 : Wx, Pygame, Wx python(or Wax), PySimpleGUI, py GUI
- Libavg
- 간단한 것, turtle
- Py SDL2 (window form)

- 예제 project 주소

```
https://github.com/hugwolf77/YN_SS_AIclass
```

- git clone https://github.com/hugwolf77/YN_SS_AIclass.git

예비. 2. LLM 모델을 사용해보기

- Langchain : ChatGPT 와 같은 LLM을 사용하여 애플리케이션을 개발하기 위한 프레임워크

- LLM (Large Language Model) : Transformer 를 기반으로 하는 생성형 언어모델을 딥러닝 모델들을 뜻한다.

예) ChatGPT, Claude3, Gemini, LLaMA 등

- 핵심구성

(1) LLM : 여러 LLM 모델을 사용할 수 있음. API 를 이용하거나 local 설치

(2) Prompt Templates : LLM에 전송하는 Requests의 구성과 형식 등을 구성한다. 이를 통해서 LLM 이 생성하는 응답에 크게 영향을 주게 된다.

(3) Output Parsers : LLM 으로부터 오는 Responses 결과를 실제 서비스하는 형식으로 parsing 하는 것

- huggingface : 머신러닝 모델과 데이터셋 등의 레퍼런스와 프레임워크를 제공하는 플랫폼 서버
- Meta 의 <https://ollama.com/> 에서 ollama pull llama3