

인공지능 개론

데이터 분석 및 전처리

호서대학교 조학수

marius1406@gmail.com

목 차

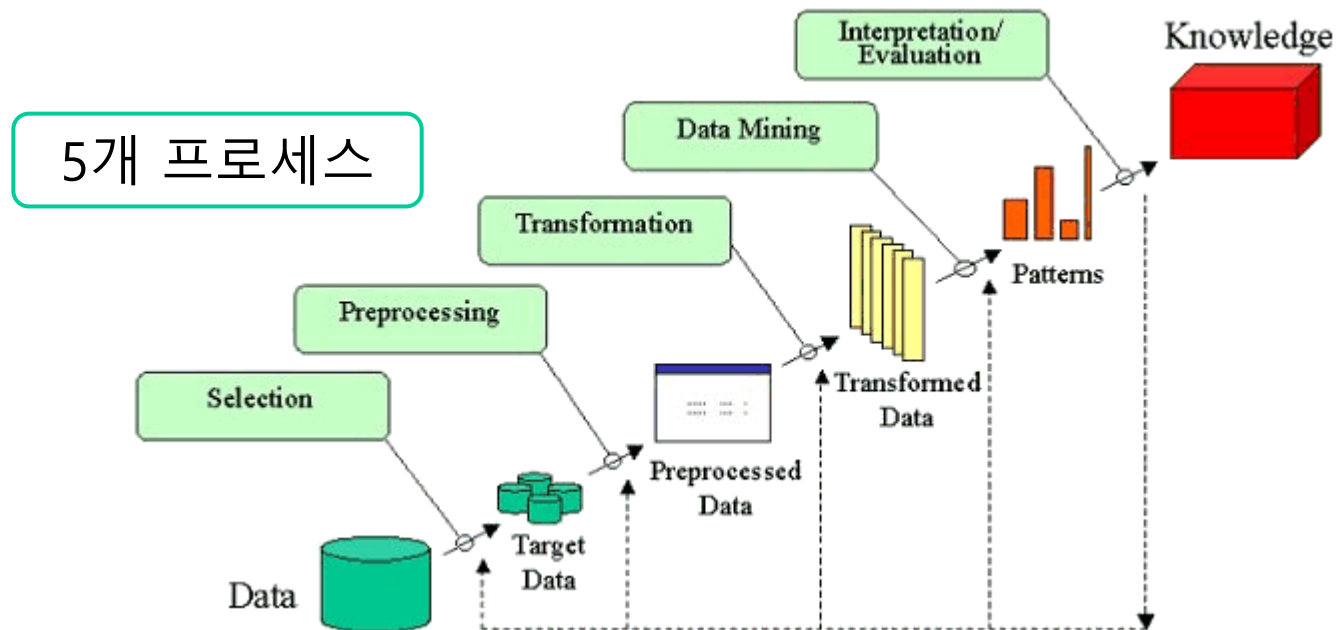
1. 데이터 분석 방법론
2. 실습 환경 구성
3. 주요 라이브러리
4. 데이터 분석
5. 탐색적 분석

1 데이터 분석 방법론

데이터 분석 방법론

1. KDD 분석 방법론 (KDD Model)

KDD(Knowledge Discovery in Database)는 1996년 Fayyad가 체계적으로 정리한 **데이터 마이닝 프로세스**로써 데이터베이스에서 의미 있는 지식을 탐색하는 데이터 마이닝부터, 기계학습, 인공지능, 패턴인식, 데이터 시각화 등에서 응용될 수 있는 구조를 가지고 있다. KDD는 데이터에서 패턴을 찾는 과정을 다음과 같이 5개의 프로세스로 제시하고 있다.



데이터 분석 방법론

1. KDD 분석 방법론 (KDD Model)

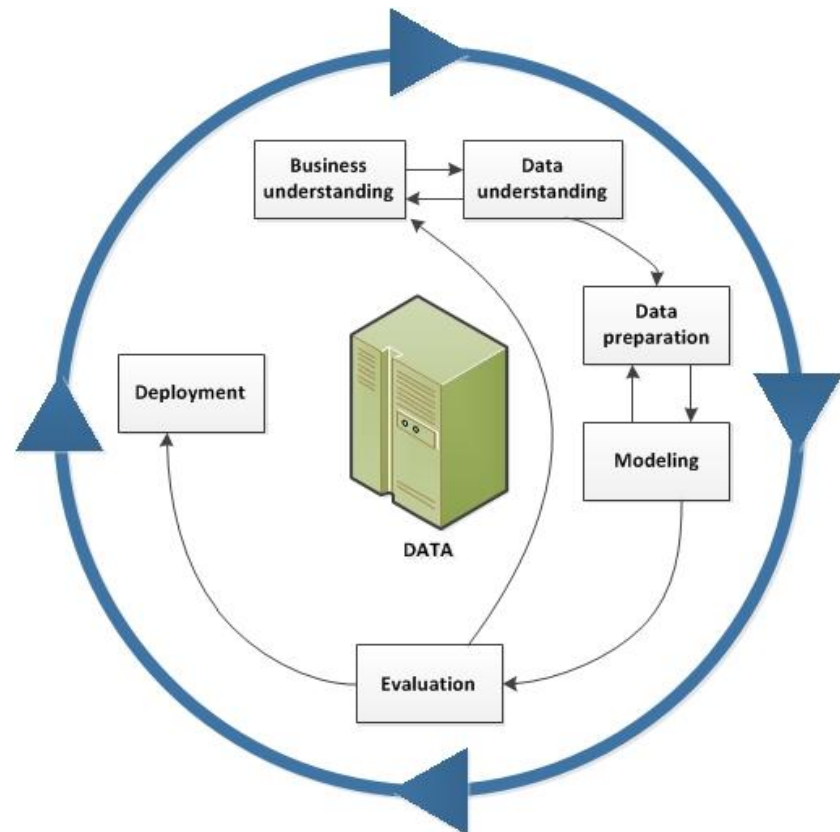
단계	설명
데이터셋 선택 (Selection)	분석대상의 비즈니스 도메인에 대한 이해와 프로젝트의 목표 설정 데이터베이스 또는 원시 데이터에서 분석에 필요한 데이터를 선택 필요 시 추가 데이터셋을 생성
데이터 전처리 (Preprocessing)	분석 대상용 데이터셋에 포함되어 있는 잡음(Noise) 과 이상값(Outlier) , 결측치(Missing Value) 를 식별,제거 또는 의미 있는 데이터로 처리 추가적인 데이터셋이 필요하면 데이터셋 선택 프로세스를 반복
데이터 변환 (Transformation)	분석용 데이터셋에서 분석 목적에 맞는 변수를 선택 데이터의 차원을 축소 하여 데이터 마이닝이 쉽게 데이터셋 변환
데이터 마이닝 (Data Mining)	분석용 데이터셋을 이용 분석 목적에 맞는 데이터 마이닝 기법을 선택 하고 데이터 마이닝 알고리즘을 선택 유의미한 데이터의 패턴을 찾거나 규칙을 추출
데이터 마이닝 결과 평가 (Evaluation)	데이터 마이닝 결과에 대한 해석과 유효성 평가 시각화를 통한 지식(통찰)을 획득 발견된 지식을 업무에 활용

데이터 분석 방법론

2. CRISP-DM 분석 방법론

DaimlerChrysler, SPSS, NCR 등이 참여하여 1996년 유럽연합의 ESPRIT 프로젝트에서 시작되어 1999년 발표된 데이터 마이닝 표준 프로세스 모델

6단계로 구성되어 산업 전반에 활용 가능하고 단계간 반복적 피드백을 통해 프로젝트의 완성도를 높임



데이터 분석 방법론

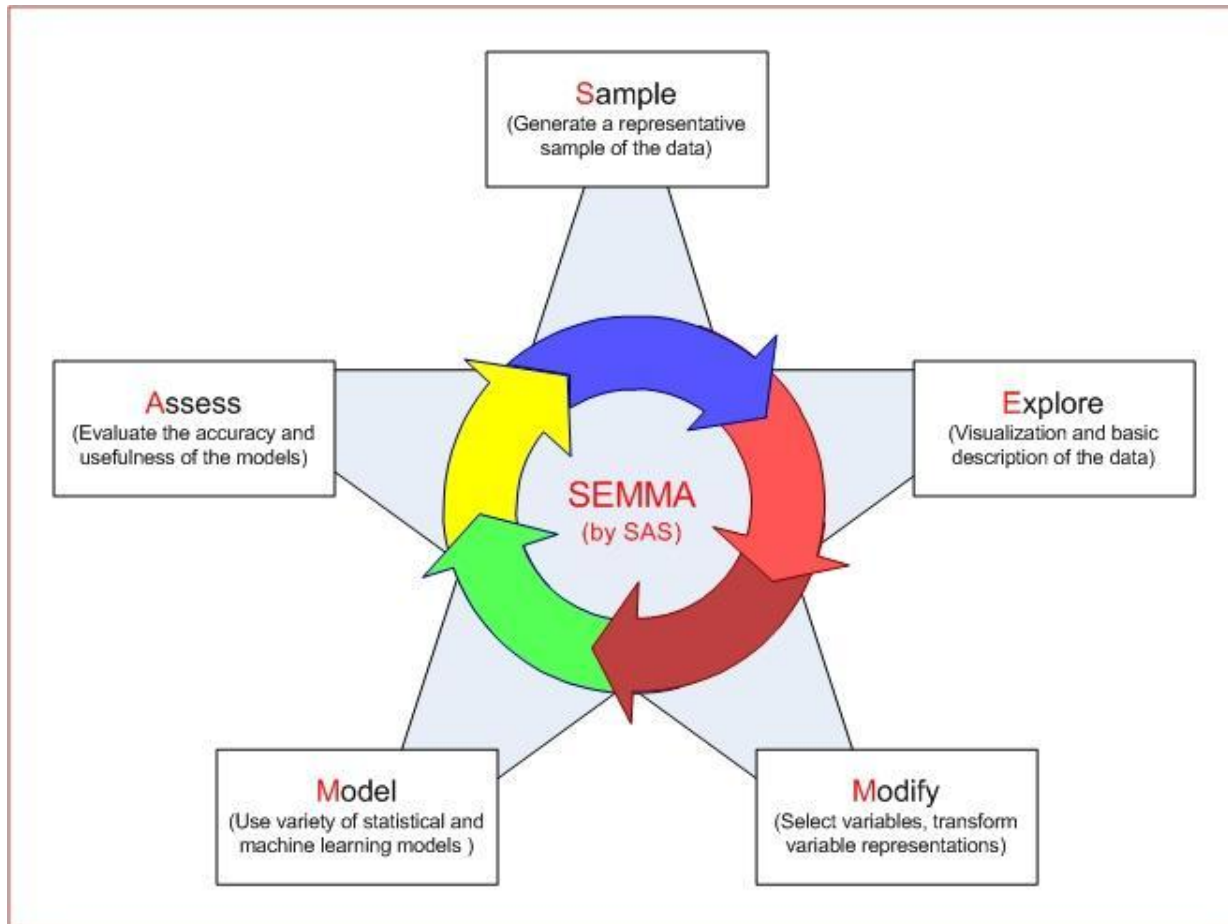
2. CRISP-DM 분석 방법론

단계	설명
업무 이해 (Business Understanding)	비즈니스의 목적과 데이터마이닝의 목표를 수립하고 프로젝트 계획 수립 • 업무 목적 파악 / 상황 파악 / 데이터 마이닝 목표 설정 / 프로젝트 계획 수립
데이터 이해 (Data Understanding)	분석에 필요한 Initial Data를 분석하고 품질을 검토하여 분석용 데이터 확보를 위한 준비단계 • 초기 데이터 수집 / 데이터 기술 분석 / 데이터 탐색 / 데이터 품질 확인
데이터 준비 (Data Preparation)	데이터를 획득하여 선별, 통합, 정제과정을 통해 분석용 데이터셋을 편성 • 분석용 데이터셋 선택 / 데이터 정제 / 분석용 데이터셋 편성 / 데이터 통합 / 데이터 포매팅
모델링 (Modeling)	다양한 분석 기법을 활용하여 모델링을 하고 설계된 테스트 계획에 따라 평가 • 모델링 기법 선택 / 모델 테스트 계획 설계 / 모델 작성 / 모델 평가
평가 (Evaluation)	분석결과를 평가하고 과정을 리뷰 • 분석결과 평가 / 모델링 과정 평가 / 모델 적용성 평가
전개 (Deployment)	전개 및 모니터링 계획을 수립하고 과제 종료 • 전개 계획 수립 / 모니터링과 유지보수 계획 수립 / 프로젝트 종료보고

데이터 분석 방법론

3. SEMMA 분석 방법론

SEMMA(Sample, Explore, Modify, Model and Assess)는 SAS Institute에서 개발한 데이터 마이닝 방법론으로 SAS Miner에서 사용하는 분석 절차



데이터 분석 방법론

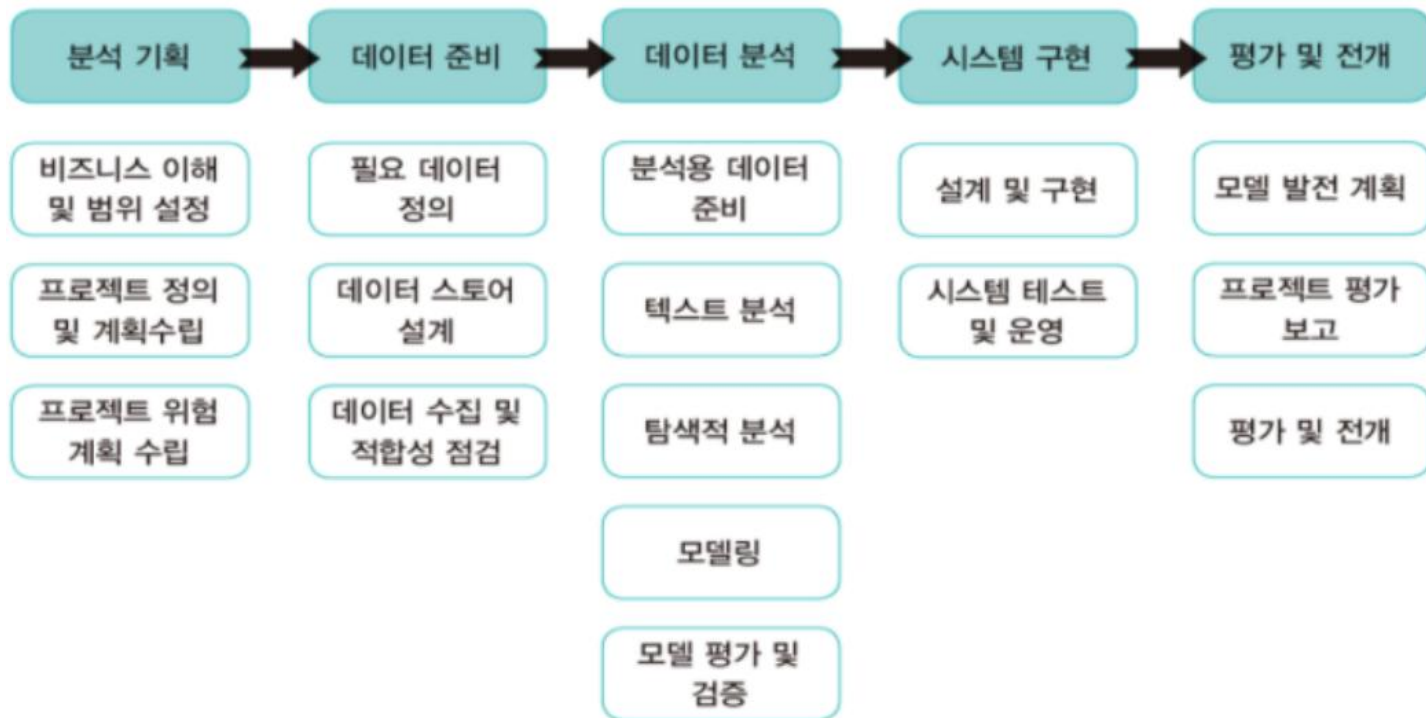
3. SEMMA 분석 방법론

단계	설명
추출 (Sample)	<ul style="list-style-type: none"> • 분석할 데이터 생성 (통계적 추출, 조건 추출) • 모델링 및 모델 평가를 위한 데이터 준비
탐색 (Explore)	<ul style="list-style-type: none"> • 분석 데이터 탐색 • 데이터 조감을 위한 데이터 오류 검색 • 데이터 현황 이해 및 Integrity 확보
수정 (Modify)	<ul style="list-style-type: none"> • 분석 데이터 수정·변환 (수량화, 표준화, 계층화, 그룹화) • 데이터가 지닌 정보의 표현 극대화 • 최적의 모델이 구축되도록 변수를 생성, 선택, 변경
모델링 (Modeling)	<ul style="list-style-type: none"> • 모델 구축 (Neural Network, DT, Logistic Regression 등) • 데이터의 숨겨진 패턴 발견 • 복수의 모델과 알고리즘 적용
평가 (Assess)	<ul style="list-style-type: none"> • 모델 평가 및 검증 • 서로 다른 모델 동시 비교

데이터 분석 방법론

4. 빅데이터 분석 방법론

데이터산업진흥원에서 제시하는 빅데이터 분석 방법론 참조모델로, 데이터 분석 프로젝트를 위해 표준적으로 적용할 수 있는 프로세스를 5단계로 정리



데이터 분석 방법론

4. 빅데이터 분석 방법론

단계	설명
분석 기획 (Planning)	<p>분석하려는 비즈니스를 이해하고 도메인의 문제점을 파악하여 빅데이터 분석 프로젝트의 범위를 확정하는 단계 프로젝트의 정의 및 수행계획 수립</p> <ul style="list-style-type: none"> • 비즈니스 이해 및 범위 설정 / 프로젝트 정의 및 계획 수립
데이터 준비 (Preparing)	<p>비즈니스 요구사항을 데이터 차원에서 파악하고 필요 데이터 정의 전사 차원의 데이터 스토어(Data Store) 준비 데이터의 품질 확보를 위해 품질 통제와 품질보증 프로세스 수행</p> <ul style="list-style-type: none"> • 필요 데이터 정의 / 데이터 스토어 설계 / 데이터 수집 및 정합성 점검
데이터 분석 (Data Analyzing)	<p>정형/비정형 데이터를 분석 진행 데이터 스토어(Data Store)에서 분석에 필요한 데이터셋을 준비 탐색적 분석, 모델링과 모델 평가 태스크 진행</p> <ul style="list-style-type: none"> • 분석용 데이터 준비 / 텍스트 분석 / 탐색적 분석 / 모델링 / 모델 평가 및 검증 / 모델 적용 및 운영방안 수립
시스템 구현 (Developing)	<p>데이터 분석을 진행하여 모델을 도출 이를 운영중인 시스템에 적용하거나 시스템 구현 단계를 진행</p> <ul style="list-style-type: none"> • 설계 및 구현 / 시스템 테스트 및 운영
평가 및 전개 (Deploying)	<p>프로젝트의 목적을 달성 했는지 평가 시스템 구현 단계에서 구축된 모델의 발전계획을 수립 빅데이터 분석 프로젝트의 종료 및 전개 프로세스로 구성 수행된 프로젝트를 객관적이고 정량적으로 평가, 내부 활용 및 자산화</p> <ul style="list-style-type: none"> • 모델발전계획수립/프로젝트평가및보고

데이터 분석 방법론

나만의 분석 프로세스 만들어 보기

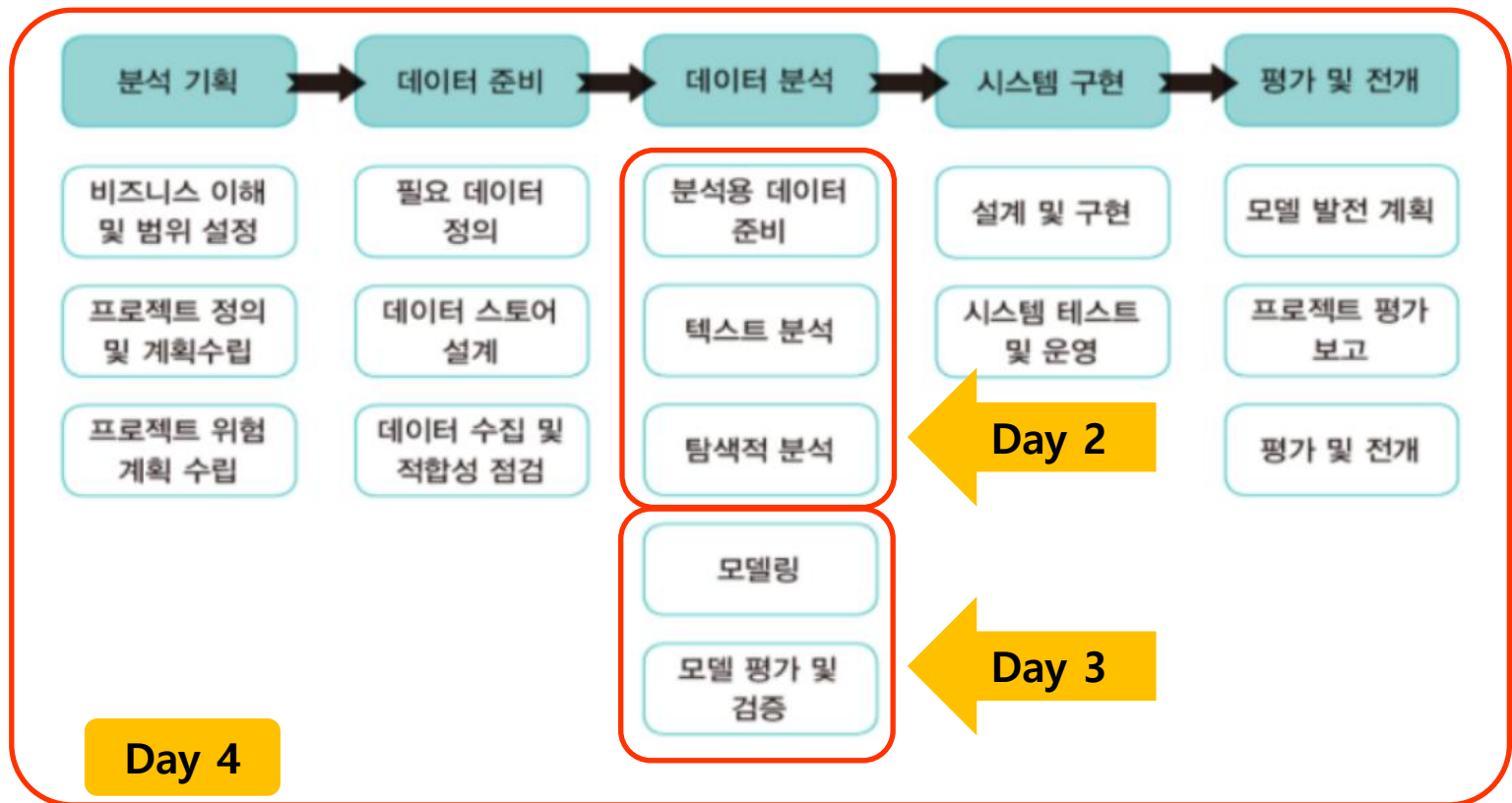
(실습) KDD 분석 방법론, CRISP-DM 분석 방법론,
SEMMA 분석 방법론, 빅데이터 분석 방법론에서
공통적으로 발견되는 프로세스 요소가 있는가?

데이터 분석 방법론

이과정에서는

데이터 분석 방법론 중 '빅데이터 분석 방법론' 을 기준으로,

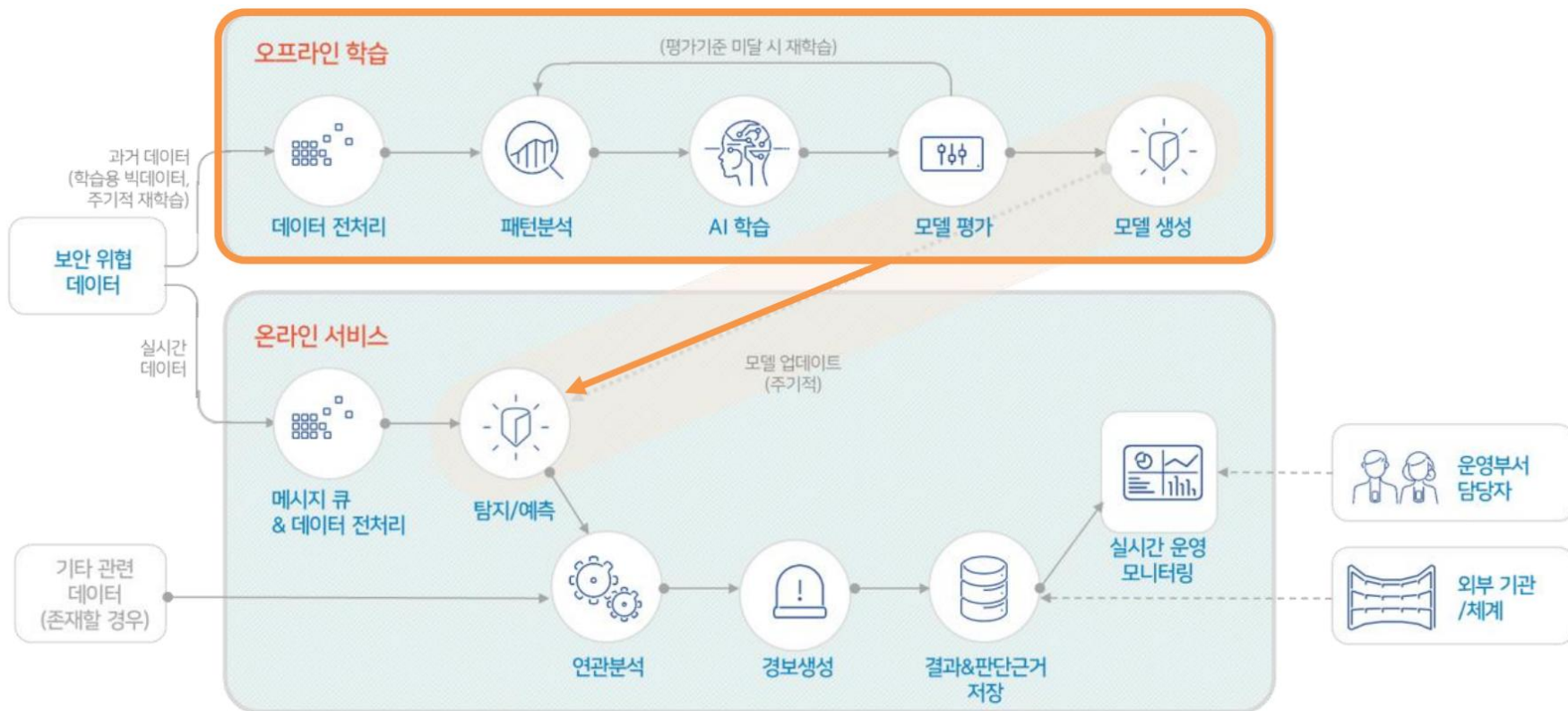
- Day 2의 전체 머신러닝 프로세스 중 데이터 처리 부분
- Day 3의 모델링 부분, Day 4에서 실제 문제에 대해 종합적으로 적용 연습함



데이터 분석 방법론

실제 AI 서비스 구조

실제 AI 서비스 구조도 예시이다. 이러한 분석 방법론을 통해 구현된 AI모델은 실제 AI 서비스 상에서 다른 서비스 컴포넌트들과 함께 유기적으로 동작하게 된다.

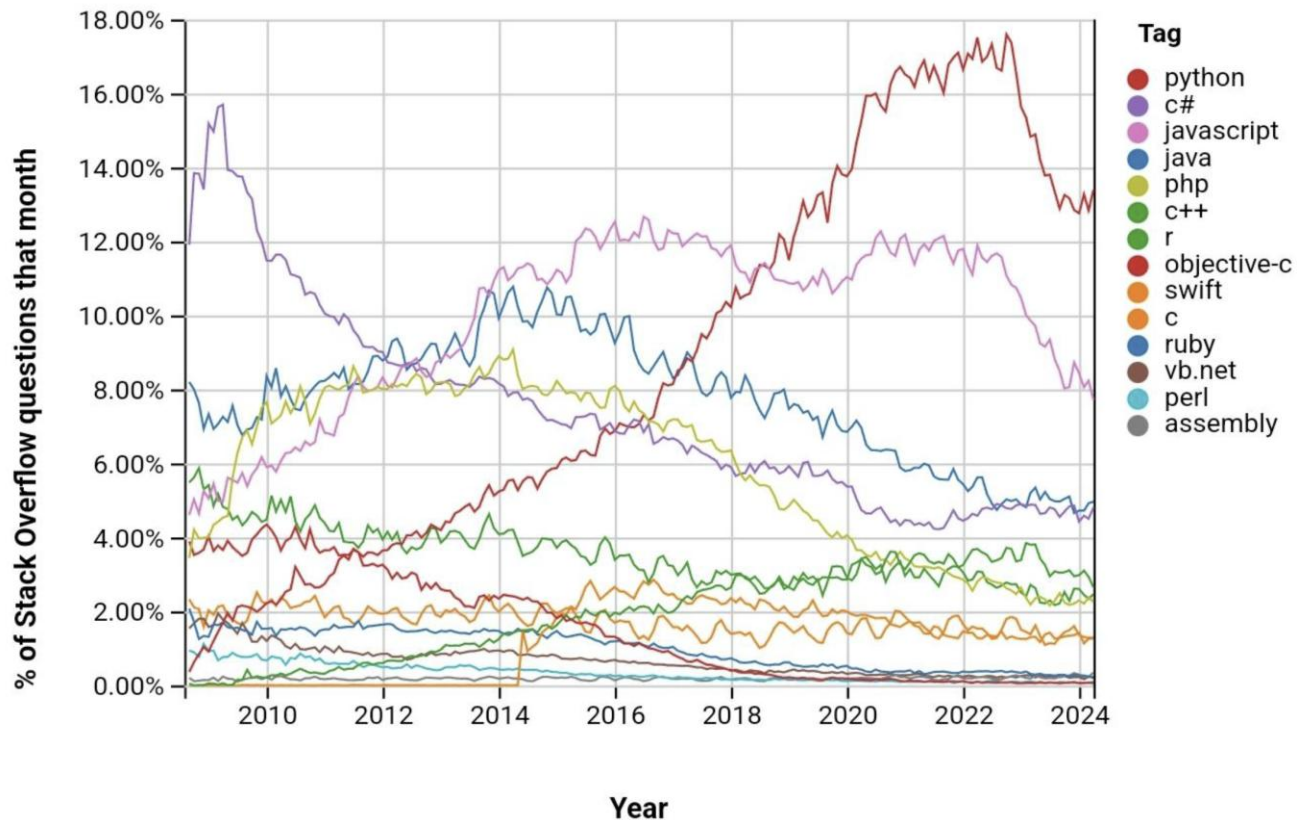


2 실습환경 구성

실습 환경 구성

데이터 분석 언어로서의 Python

본 과정에서는 TensorFlow, Keras, PyTorch 등의 딥러닝 관련 라이브러리 지원이 월등한 Python 주로 사용



실습 환경 구성

Python 장점

- 배우기 쉬운 대화형 인터프리터 언어
- C언어를 포함한 다른 프로그래밍 언어와 연동성 높음
- 데이터 분석과 인공지능을 위한 다양한 라이브러리
- 많은 레퍼런스 자료와 커뮤니티



실습 환경 구성

Jupyter Notebook

- 웹 브라우저에서 파이썬 인터프리터를 실행할 수 있는 개발환경
- 주피터 세션에서 셀 단위로 실행하고 결과가 세션에 유지
- 코드 블록(셀)을 원하는 순서대로 실행 가능하고 몇 번이고 다시 실행 가능

▼ 시작하기

지금 읽고 계신 문서는 정적 웹페이지가 아니라 코드를 작성하고 실행할 수 있는 대화형 환경인 **Colab 메모장**입니다.
예를 들어 다음은 값을 계산하여 변수로 저장하고 결과를 출력하는 간단한 Python 스크립트가 포함된 코드 셀입니다.

✓
0초 [1] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400

위 셀의 코드를 실행하려면 셀을 클릭하여 선택한 후 코드 왼쪽의 실행 버튼을 누르거나 단축키 'Command/Ctrl+Enter'를 사용하세요. 셀을 클릭하면 코드 수정을 바로 시작할 수 있습니다.

특정 셀에서 정의한 변수를 나중에 다른 셀에서 사용할 수 있습니다.

✓
0초 seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week

604800

Jupyter Notebook 화면

실습 환경 구성

Colab

본 과정에서의 실습은 구글에서 제공하는 코랩(Colab)을 이용
구글 코랩은 주피터 노트북기반의 데이터 분석을 할 수 있도록 만든 서비스

특징

- 구글에서 무료로 제공하는 GPU를 적용할 수 있는 개발 환경
- 대부분의 데이터 분석용 라이브러리는 자동으로 설치되어 있어 별도 설치 불필요
- Google Drive와 jupyter Notebook(.ipynb)을 사용하고, 클라우드에서 동작
- Notebook 파일과 학습 데이터 모두 구글 드라이브 상에서 관리

제약사항

- 상업적 용도 불가
- 공용 클라우드 서버를 사용으로 자원 (RAM, Memory, 리소스 등) 사용 제한이 있음
- 세션 유지 시간이 12시간, 오랜 딥러닝 학습에는 부적합
(유료 버전Colab Pro 에서는 연장 가능)

※ 대용량 빅데이터 분석을 위해 구글드라이브 용량 업그레이드(유료), Colab 프로버전(유료)이 필요
인공지능 모델 구현 프로세스를 학습용으로 빠르고 안정적으로 실습을 할 수 있는 구글 코랩 환경이 최적

실습 환경 구성

Colab 시작하기

<https://colab.research.google.com/> 에 직접 접속 가능하나,
실습파일 및 데이터 공유를 위해 구글 드라이브 인터페이스 사용을 추가 설명

1) 구글 드라이브 (<https://www.google.com/drive/>) 에 접속해서 구글 계정 로그인

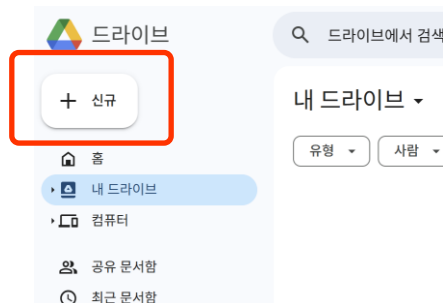
2) (사전에 배포된) 실습 데이터 업로드

코드의 공유 등 실습의 편의를 위해 가급적이면 최상위 폴더에 위치하도록 한다. ('내 드라이브' 바로 밑)

※ 분석용 데이터는 프로젝트 당 최대 5GB로 구성되어, 현재 용량이 5GB 이상 여유가 있는지 확인 필요함.

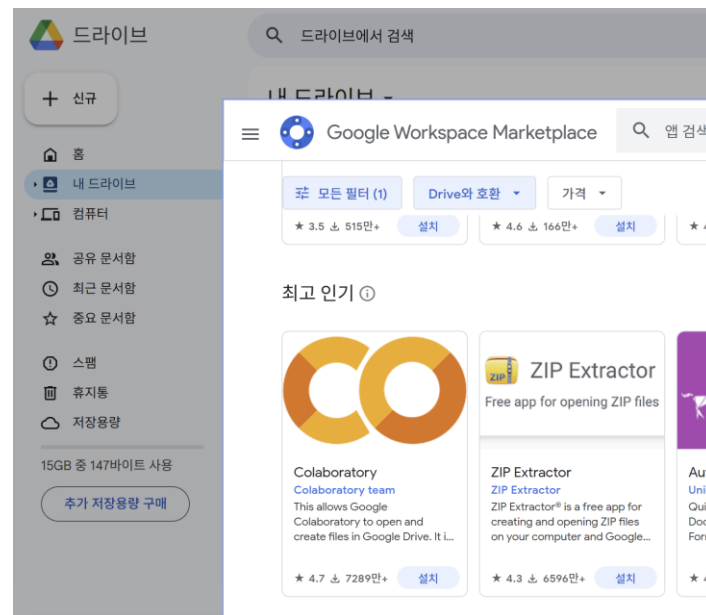
3) 주피터 노트북(Colab 노트북) 생성

• 왼쪽 상단 [+ 신규] 버튼 > [더보기] > 하단 [+ 연결할 앱 더보기] 클릭 > Google Colaboratory 선택

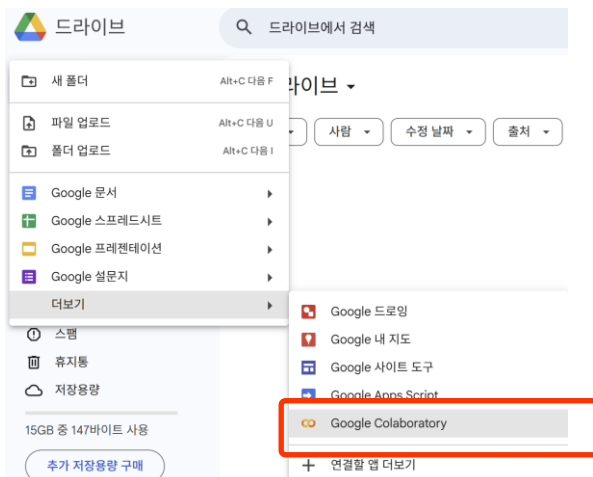


실습 환경 구성

Colab 시작하기



Colab 최초 사용시 Colab 설치 필요

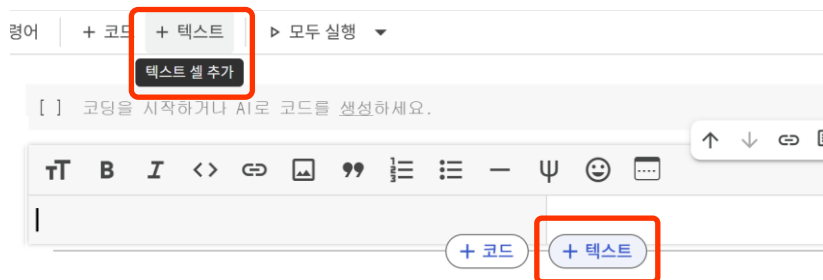


실습 환경 구성

Colab Notebook 시작하기

1) 텍스트 셀 생성

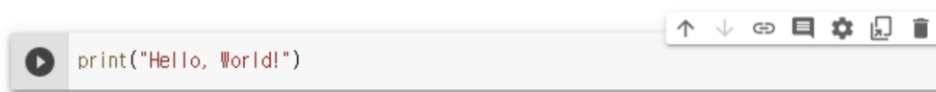
- 아래에 설명을 추가하고 싶은 셀을 클릭 하고 상단 툴바의 [+ 텍스트] 버튼 클릭
 - 아래에 설명을 추가하고 싶은 셀 하단에 나타나는 [+ 텍스트] 버튼 클릭
- 텍스트 셀에서 Markdown 언어 사용 가능, 텍스트 앞에 #, ##, ### 붙여 Header 지정.



2) 코드 셀 생성

코드를 추가하고 싶은 셀 위치에서 [+코드] 버튼을 클릭하여 코드 셀 생성

▼ Colab Notebook 사용하기



실습 환경 구성

Colab Notebook 시작하기

3) 셀 단위 실행 : print("Hello, World!") 출력해보기

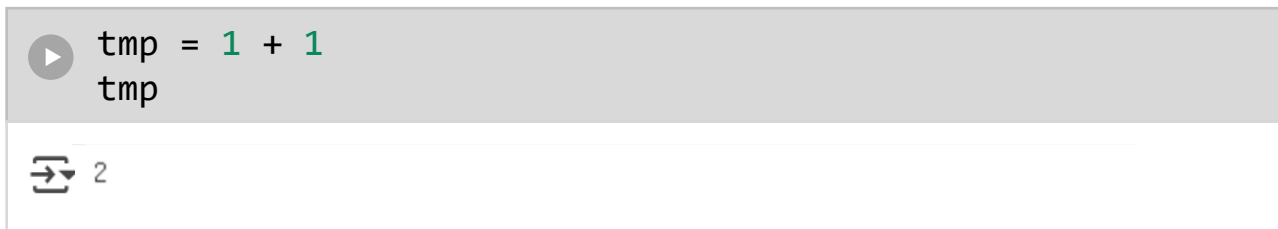
- 셀 좌측 실행버튼 클릭
- 단축키를 사용하여 실행 (ex: Shift + Enter : 셀 실행 후 커서가 다음 셀로 넘어감)



A screenshot of a Google Colab notebook cell. The top bar is grey and contains a play button icon and the code `print("Hello, World!")`. The bottom bar is white and contains a double arrow icon and the output `Hello, World!`.

4) 불필요한 셀 삭제

테스트 용도나 변수 값 확인 등으로 임시로 사용했던 셀은 언제든지 삭제하여 코드 정리 가능



A screenshot of a Google Colab notebook cell. The top bar is grey and contains a play button icon and the code `tmp = 1 + 1` followed by `tmp` on the next line. The bottom bar is white and contains a double arrow icon and the output `2`.

실습 환경 구성

주요 파이썬 라이브러리

파이썬은 풍부한 머신러닝 라이브러리와 프레임워크가 제공

파이썬의 라이브러리 생태계를 활용하여 다양한 라이브러리를 개발에 활용가능

※ <https://github.com/Harksu71/DataScienceAnalytics/Lecture/SeSAC> 의 ipynb 파일 활용한다.

대표적인 라이브러리 목록이다.

3 주요 라이브러리

주요 라이브러리

NumPy

머신 러닝에서 자주 사용되는 벡터, 행렬 및 텐서와 같은 데이터 구조에 대한 효율적인 연산을 지원하는 파이썬 라이브러리, SciPy나 pandas의 베이스 객체로도 사용되며 NumPy에서 사용되는 다양한 코드 표현법을 PyTorch와 TensorFlow에 사용하는 경우가 많은 기본 라이브러리



배열 생성

배포파일 중 '1_주요라이브러리.ipynb'를 열어 NumPy 기본 사용법을 익히자

```
# 라이브러리 로드
import numpy as np
# 다차원 배열 생성
# 벡터: 1차원 배열, 행렬: 2차원 배열, 그 이후는 다차원 배열
mat = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
mat
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

▶ # 랜덤 값으로 채워진 행렬 생성
많은 머신러닝 학습 알고리즘에서 초기 파라미터를 랜덤으로 생성한 후 학습을 통해
조정해 나가게 되는 데 그 과정에서 랜덤 값으로 행렬을 생성하는 경우가 많다.
`np.random.random((3, 4))`

⇒ `array([[0.24122572, 0.05592428, 0.7650242 , 0.79621841],
[0.81368552, 0.06087247, 0.07159139, 0.33514965],
[0.52566476, 0.02139824, 0.16501757, 0.52800433]])`

▶ # 특정한 값으로 채워진 행렬 생성
행렬 연산에 필요
`np.zeros()`: 영행렬, `np.ones()`, `np.full()`, `np.eye()`: 단위행렬
영행렬 생성
`np.zeros((3, 4))`

⇒ `array([[0., 0., 0., 0.],
[0., 0., 0., 0.],
[0., 0., 0., 0.]])`

배열 형태 확인

▶ # shape 확인
mat.shape

⇒ (3, 4)

▶ # 차원 확인
mat.ndim

⇒ (3, 4)

▶ # size 확인
= element 수 확인
mat.size

⇒ 12

▶ # 데이터 타입 확인
mat.dtype

⇒ dtype('int64')

배열 구성요소 선택

▶ # 특정 행, 열의 원소
mat[2, 2]

⇒ np.int64(11)

▶ # 특정 행, 열의 원소 변경하기
mat[2, 2] = 99
mat

⇒ array([[1, 2, 3, 4],
 [5, 6, 7, 8],
 [9, 10, 99, 12]])

▶ # 슬라이싱 (행)
x[start:stop:step]
mat[1, :]

⇒ array([5, 6, 7, 8])

▶ # 슬라이싱 (열)
mat[:, 3]

⇒ array([4, 8, 12])

배열 형태 변환

▶ # 배열 형태 변환 (형태 명시적 지정)
신경망 다층 레이어 구성 시 배열의 차원을 변경할 경우들이 존재한다.
재구성된 배열의 크기가 초기 배열의 크기와 일치해야 한다.
`mat.reshape(4, 3)`

⇒ `array([[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9],
 [10, 99, 12]])`

▶ # 배열 형태 변환 (마지막 차원 자동 완성)
`reshape`의 유용한 인수 중 하나는 `-1`이다.
이는 효과적으로 "필요한 만큼"을 의미하므로
예를 들어 `reshape(2, -1)`은 2개의 행과 필요한 만큼의 열을 의미한다.
`mat.reshape(2, -1)`

⇒ `array([[1, 2, 3, 4, 5, 6],
 [7, 8, 9, 10, 99, 12]])`

배열 형태 변환

▶ # 1차원 벡터로 변환
reshape(1, -1)과 동일
mat.flatten()

⇒ array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 99, 12])

▶ # 행렬 전치
전치는 선형 대수학에서 각 요소의 열과 행 인덱스가 바뀌는 일반적인 연산이다.
mat.T

⇒ array([[1, 5, 9],
[2, 6, 10],
[3, 7, 99],
[4, 8, 12]])

Aggregation (집계)

▶ # aggregation 기능은 데이터 탐색 단계에서 기초 통계량을 추출할 때 사용된다.
mat.min(), mat.max(), mat.mean(), mat.std()
mat.min()과 np.min(mat)는 동일
각 요소의 합
mat.sum()

⇒ np.int64(166)

▶ # 상위 25% (= 75백분위수의 값)
np.percentile(mat, 75, interpolation='nearest')

⇒ np.int64(9)

▶ # np.any(): 모든 요소 중 하나라도 True가 있으면 True
np.all(): 모든 요소가 True이면 True
모든 요소가 유효한 값인지 확인
mat.all()

⇒ np.True_

Boolean Array 활용법

▶ # 비교
mat > 5

```
↔ array([[False, False, False, False],
        [False,  True,  True,  True],
        [ True,  True,  True,  True]])
```

▶ # 비교 (논리 연산자 복합적 사용)
(mat > 5) & (mat < 10)

```
↔ array([[False, False, False, False],
        [False,  True,  True,  True],
        [ True, False, False, False]])
```

▶ # Boolean Array를 이용한 element 선택 - 1
mat[mat>5]

```
↔ array([ 6,  7,  8,  9, 10, 99, 12])
```

Boolean Array 활용법

▶ # Boolean Array를 이용한 element 선택 - 2
`mat[(mat > 5) & (mat < 10)]`

⇒ `array([6, 7, 8, 9])`

▶ # Boolean Array를 이용한 element 선택 후 aggregation(집계) - 1
 # True = 1, False = 0으로 계산
`np.sum(mat > 5)`

⇒ `np.int64(7)`

▶ # Boolean Array를 이용한 element 선택 후 aggregation(집계) - 2
`np.sum(mat[mat>5])`

⇒ `np.int64(151)`

그 밖에, 정렬, 행렬 연산, 역행렬 계산 등 NumPy는 행렬연산에 대한 많은 기능을 제공하지만, 이러한 부분은 머신러닝 라이브러리나 프레임워크에서 내부적으로 처리해주기 때문에 본 과정에서는 직접 다루지 않는다.

주요 라이브러리



pandas(판다스)는 데이터프레임(DataFrame)과 시리즈(Series)라는 자료형과 데이터 분석을 위한 다양한 기능을 제공하는 파이썬 라이브러리로, 데이터 분석에 프로세스 전반적으로 광범위하게 사용된다. Series와 DataFrame은 NumPy의 구조화된 배열이라고 볼 수 있는데, 예를 들어 DataFrame은 다차원 배열에 index와 column과 같은 Label 정보가 추가로 붙어있는 데이터 구조를 띄고 있다. 따라서 NumPy와 같이 단순히 행/열 포지션에 의해 접근하는 것이 아닌 행/열 Label로 접근 가능하다는 차이점이 있다. 이러한 차이점으로 인해 다양한 데이터 분석 관련 기능이 파생되어 제공된다.

Series 자료형

```
# 라이브러리 로드
import pandas as pd

# @title 기본 제목 텍스트
# Series 자료형 데이터 생성
data = pd.Series([1.4, 1.4, 4.7])
data
```

```
0      1.4
1      1.4
2      4.7
```

```
dtype: float64
```

```
# 데이터 타입 확인
type(data)
```

```
pandas.core.series.Series
```

Series 자료형

```
# Series는 Index라는 추가적인 정보를 가진다.
data = pd.Series([1.4, 1.4, 4.7],index=['샘플0', '샘플1', '샘플2'])
data
```

```
↔ 샘플0    1.4
    샘플1    1.4
    샘플2    4.7
```

dtype: float64

```
# Series 요소 선택
# 배열의 포지션으로도 접근 가능하지만, Index Label로도 접근 가능하다.
print(data[1])
print(data['샘플1'])
```

```
↔ 1.4
   1.4
```

DataFrame 자료형

```
# DataFrame 자료형 데이터 생성
df = pd.DataFrame()
df['샘플명'] = ['샘플0', '샘플1', '샘플2', '샘플3', '샘플4', '샘플5']
df['꽃잎 길이'] = [1.4, 1.4, 4.7, 4.5, 6, 5.1]
df['꽃잎 넓이'] = [0.2, 0.2, 1.4, 1.5, 2.5, 1.9]
df['종류'] = ['세토사', '세토사', '버시칼라', '버시칼라', '버지니카', '버지니카']
df
```

	↔	샘플명	꽃잎 길이	꽃잎 넓이	종류
0		샘플0	1.4	0.2	세토사
1		샘플1	1.4	0.2	세토사
2		샘플2	4.7	1.4	버시칼라
3		샘플3	4.5	1.5	버시칼라
4		샘플4	6.0	2.5	버지니카
5		샘플5	5.1	1.9	버지니카

DataFrame 자료형

```
# 인덱스 지정
df = df.set_index(df['샘플명'])
df = df.drop(['샘플명'], axis=1)
df
```

↔ 샘플명	꽃잎 길이	꽃잎 넓이	종류
샘플0	1.4	0.2	세토사
샘플1	1.4	0.2	세토사
샘플2	4.7	1.4	버시칼라
샘플3	4.5	1.5	버시칼라
샘플4	6.0	2.5	버지니카
샘플5	5.1	1.9	버지니카

```
# 데이터 타입 확인
type(df)
```

```
↔ pandas.core.frame.DataFrame
```

DataFrame 자료형

▶ # 기본 정보 확인
df.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, 샘플0 to 샘플5
Data columns (total 3 columns):
#   Column    Non-Null Count  Dtype
---  -
0   꽃잎 길이   6 non-null      float64
1   꽃잎 넓이   6 non-null      float64
2   종류       6 non-null      object
dtypes: float64(2), object(1)
memory usage: 192.0+ bytes
```

▶ # DataFrame은 Index라는 추가적인 정보를 가진다.
df.index

```
Index(['샘플0', '샘플1', '샘플2', '샘플3', '샘플4', '샘플5'],
      dtype='object', name='샘플명')
```


DataFrame 자료형

▶ # DataFrame은 Column이라는 추가적인 정보를 가진다.
df.columns

↪ Index(['꽃잎 길이', '꽃잎 넓이', '종류'], dtype='object')

▶ # DataFrame 요소 선택
배열의 포지션으로도 접근 가능하지만, Index/Column Label로도 접근 가능하다.
print(df.iloc[4,2])
print(df.loc['샘플4', '종류'])

↪ 버지니카
버지니카

조건에 맞는 데이터 추출

▶ # 열단위 추출
1개의 열만 추출하면 시리즈를 얻을 수 있고, 2개 이상의 열을 추출하면 데이터프레임을 얻는다.
df[['']] 대괄호 안의 대괄호는 추출하려는 컬럼이 복수개(리스트 형태)이기 때문이다.
df[['꽃잎 길이', '꽃잎 넓이']].head(2)

↩ 샘플명 꽃잎 길이 꽃잎 넓이

샘플0	1.4	0.2
샘플1	1.4	0.2

▶ # 행단위 추출
loc 인덱스 Label을 기준으로 행 데이터 추출
iloc 행 번호를 기준으로 행 데이터 추출
loc 의 경우 slicing의 end 포지션을 포함하는 것에 유의
print(df.loc['샘플2':'샘플4',:])
print(df.iloc[2:4,:])

↩ 샘플명 꽃잎 길이 꽃잎 넓이 종류

샘플2	4.7	1.4	버시칼라
샘플3	4.5	1.5	버시칼라
샘플4	6.0	2.5	버지니카
샘플명	꽃잎 길이	꽃잎 넓이	종류
샘플2	4.7 1.4	버시칼라	
샘플3	4.5 1.5	버시칼라	

조건에 맞는 데이터 추출

▶ # 조건에 따른 Boolean Array 생성
df_filter = (df['종류']=='세토사')
df_filter

↔

샘플명	종류
샘플0	True
샘플1	True
샘플2	False
샘플3	False
샘플4	False
샘플5	False

dtype: bool

▶ # Boolean Array를 이용한 데이터 추출
df[df_filter]
df[df['종류']=='세토사'] 와 동일

↔

샘플명	꽃잎 길이	꽃잎 넓이	종류
샘플0	1.4	0.2	세토사
샘플1	1.4	0.2	세토사

Aggregation

▶ # DataFrame에서 1개의 열만 추출하면 Series가 된다.
type(df['꽃잎 길이'])

↔ pandas.core.series.Series

▶ # NumPy와 동일하게 평균, 분산, 총합 등의 Aggregation(집계) 가능
df['꽃잎 길이'].mean()

↔ np.float64(3.85)

▶ # 유일한 값 확인
df['종류'].unique()

↔ array(['세토사', '버시칼라', '버지니카'], dtype=object)

▶ # 유일한 값 별 카운트 확인
df['종류'].value_counts()

↔

종류	count
세토사	2
버시칼라	2
버지니카	2

dtype: int64

GroupBy

▶ # Column 기준으로 GroupBy (복수개의 열 가능)
df.groupby(['종류'])

↪ <pandas.core.groupby.generic.DataFrameGroupBy object at 0x7c060ed3e8d0>

▶ # GroupBy만 하면 GroupBy 객체 생성됨. 생성 후 Aggregation 적용
df.groupby(['종류']).mean()

↪ 종류	꽃잎 길이	꽃잎 넓이
버시칼라	4.60	1.45
버지니카	5.55	2.20
세토사	1.40	0.20

주요 라이브러리

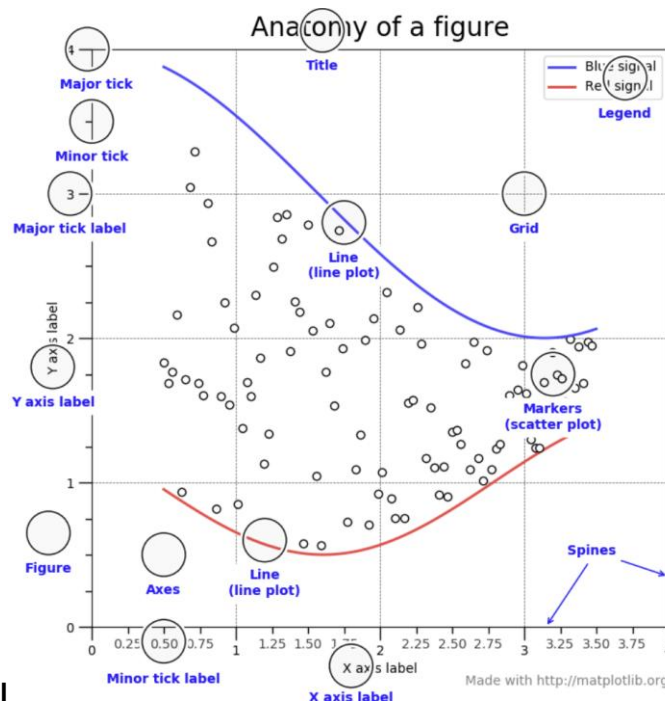


Matplotlib(맷플롯)은 파이썬 데이터 시각화 라이브러리로 다음과 같은 특징을 가진다.

- 데이터 시각화를 위한 라이브러리로 NumPy 배열을 기반으로 한다.
- 다양한 운영체제와 그래픽 백엔드와 호환이 되는 범용 라이브러리이다.
- seaborn, pandas의 그래프 기능도 matplotlib의 Wrapper 모듈이다.

데이터 분석에 있어 EDA (Exploratory Data Analysis, 데이터 탐색) 단계에 활용된다.

matplotlib 그래프의 기본 구성요소



출처: <https://matplotlib.org/3.1.1/gallery/showcase/anatomy.html>

주요 라이브러리

그래프 구성요소의 용어 정리

- Figure: 전체 캔버스. 여러 Axes를 subplot으로 포함할 수 있음
- Axes: 실제 data가 plotting 되는 공간 (좌표평면)
- X axis / Y axis : X축, Y축
- Legend: 범례

```
#라이브러리 로드
import numpy as np
import matplotlib.pyplot as plt
# plotting하기 위한 샘플데이터 생성
X = np.linspace(0, 100, 20) # 0~100까지 균일하게 20point
Y1 = X
Y2 = X**2
```

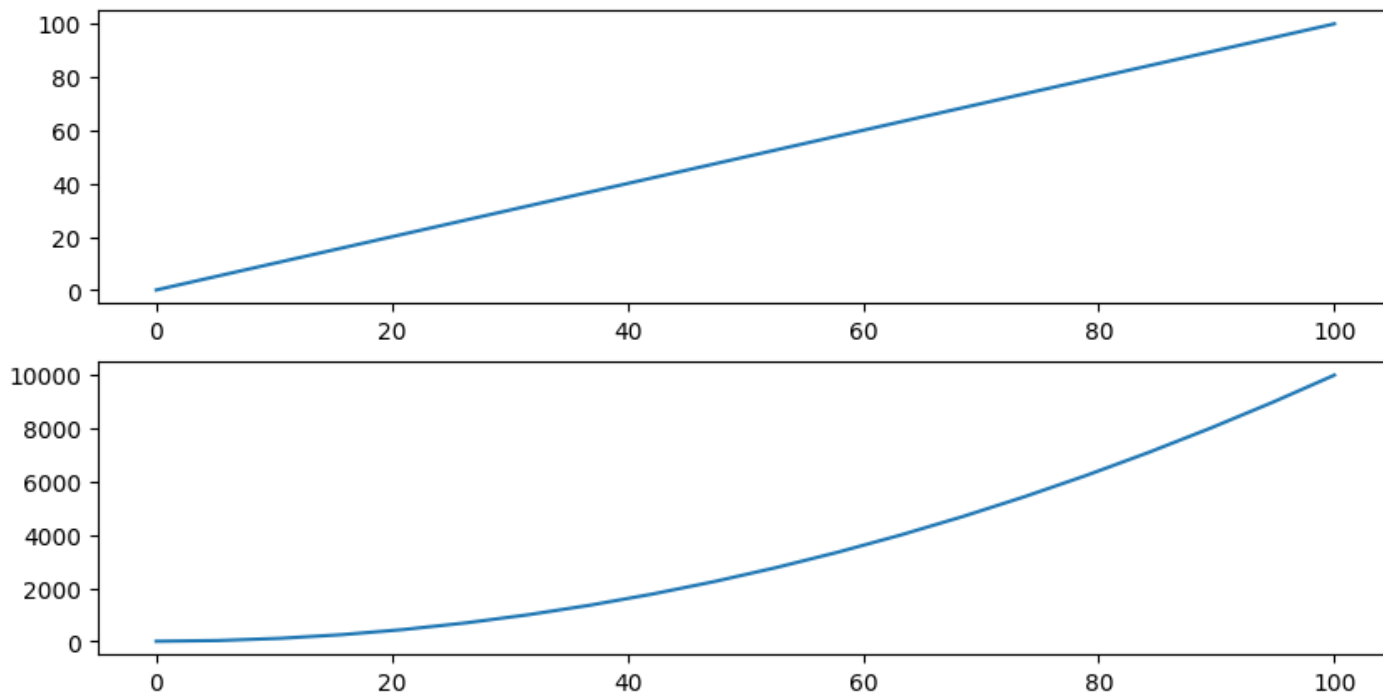


그래프를 그리는 두가지 방법 - stateless 방법

자신이 지정한 figure, ax에 그래프를 그리는 방법, 즉 figure와 ax를 직접 생성하는 방법

```
# stateless 하게 plotting
fig, axes = plt.subplots(2, 1, figsize=(10,5))
axes[0].plot(X, Y1) # 2행 1열의 2개의 그래프 중 첫번째
axes[1].plot(X, Y2) # 2행 1열의 2개의 그래프 중 두번째
```

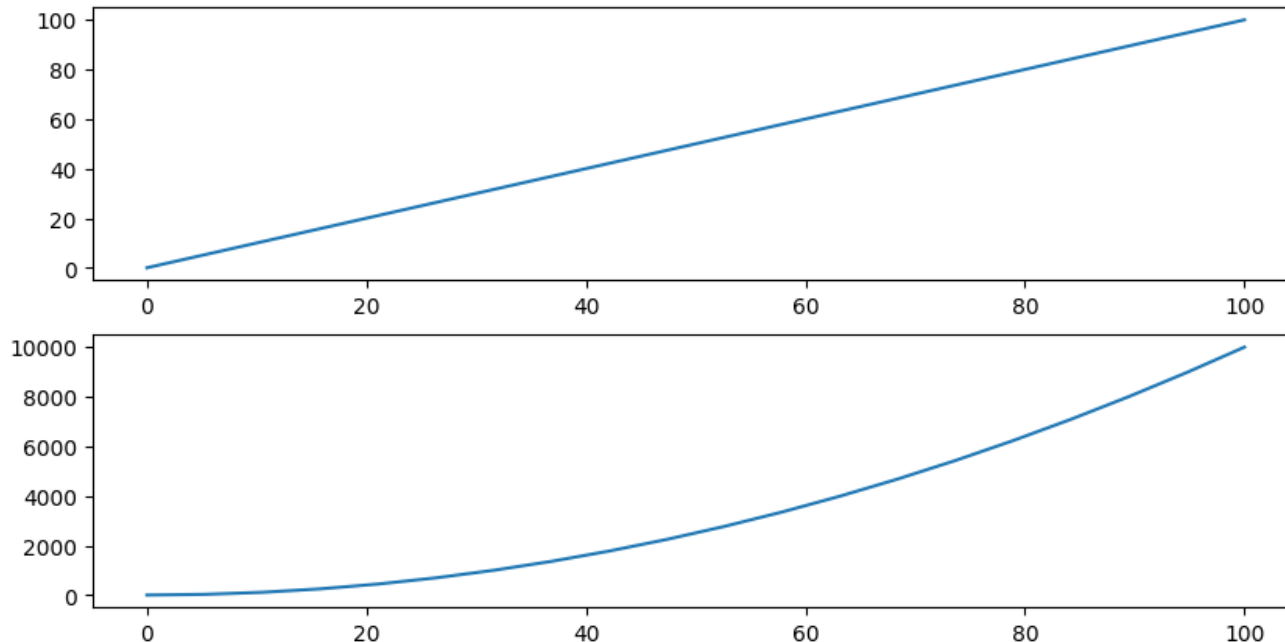
➡ [<matplotlib.lines.Line2D at 0x7c06111055d0>]



그래프를 그리는 두가지 방법 - stateful 방법

현재의 figure, 현재의 ax에 그래프를 그리는 방법: 현재의 figure와 ax를 자동으로 찾아 plotting하는 방법 (pyplot wapper 모듈을 사용)

```
# stateful 하게 plotting
plt.figure(figsize=(10,5))
plt.subplot(2, 1, 1) # 2행 1열의 2개의 그래프 중 첫번째로 state 이동
plt.plot(X, Y1)
plt.subplot(2, 1, 2) # 2행 1열의 2개의 그래프 중 두번째로 state 이동
plt.plot(X, Y2)
```



R 각종 스타일 관련

지원하는 차트의 종류도 많고, 차트마다 셋팅해야 하는 파라미터도 조금씩 상이함

원하는 차트유형의 레퍼런스 문서(API와 예제 등)를 참고하여 시각화에 필요한 구성요소 포함

🔍 Search the docs ...

`plot(x, y)`
`scatter(x, y)`
`bar(x, height) / barh(y, width)`
`stem(x, y)`
`step(x, y)`
`fill_between(x, y1, y2)`
`imshow(Z)`
`pcolormesh(X, Y, Z)`
`contour(X, Y, Z)`
`contourf(X, Y, Z)`
`barbs(X, Y, U, V)`
`quiver(X, Y, U, V)`
`streamplot(X, Y, U, V)`
`hist(x)`
`boxplot(X)`

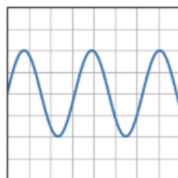
Plot types

Overview of many common plotting commands in Matplotlib.

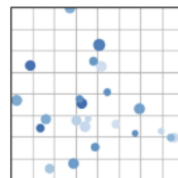
Note that we have stripped all labels, but they are present by default. See the [gallery](#) for many more examples and the [tutorials](#) page for longer examples.

Basic

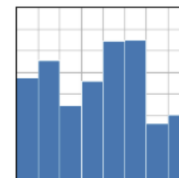
Basic plot types, usually y versus x.



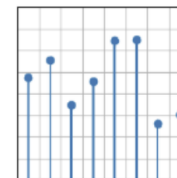
`plot(x, y)`



`scatter(x, y)`

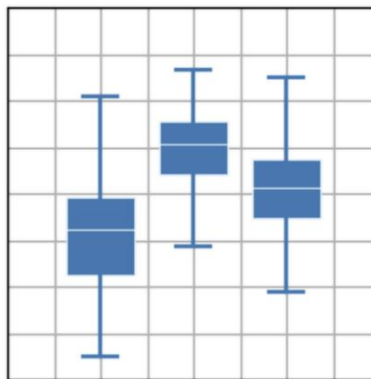


`bar(x, height) / barh(y, width)`



`stem(x, y)`

R – boxplot(X)



```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make data:
np.random.seed(10)
D = np.random.normal((3, 5, 4), (1.25, 1.00, 1.25), (100, 3))

# plot
fig, ax = plt.subplots()
VP = ax.boxplot(D, positions=[2, 4, 6], widths=1.5, patch_artist=True,
                showmeans=False, showfliers=False,
                medianprops={"color": "white", "linewidth": 0.5},
                boxprops={"facecolor": "C0", "edgecolor": "white",
                          "linewidth": 0.5},
                whiskerprops={"color": "C0", "linewidth": 1.5},
                capprops={"color": "C0", "linewidth": 1.5})

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

주요 라이브러리

scikit-learn



scikit-learn(사이킷런)은 NumPy, SciPy, matplotlib을 기반으로 하는 파이썬 대표 머신러닝 라이브러리이다. 데이터 분석에 있어 전처리 일부, 모델링과 모델평가 등 머신러닝 프로세스 전반에 활용된다.

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
Feature 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 기능 제공 (원핫 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 Feature를 우선순위로 선택 작업 수행하는 다양한 기능 제공
	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 Feature를 추출 하는데 활용됨 (Count Vectorizer, TF-IDF Vectorizer 등)
Feature 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련한 알고리즘을 지원하는 모듈 (PCA, NMF, Truncated SVD 등)

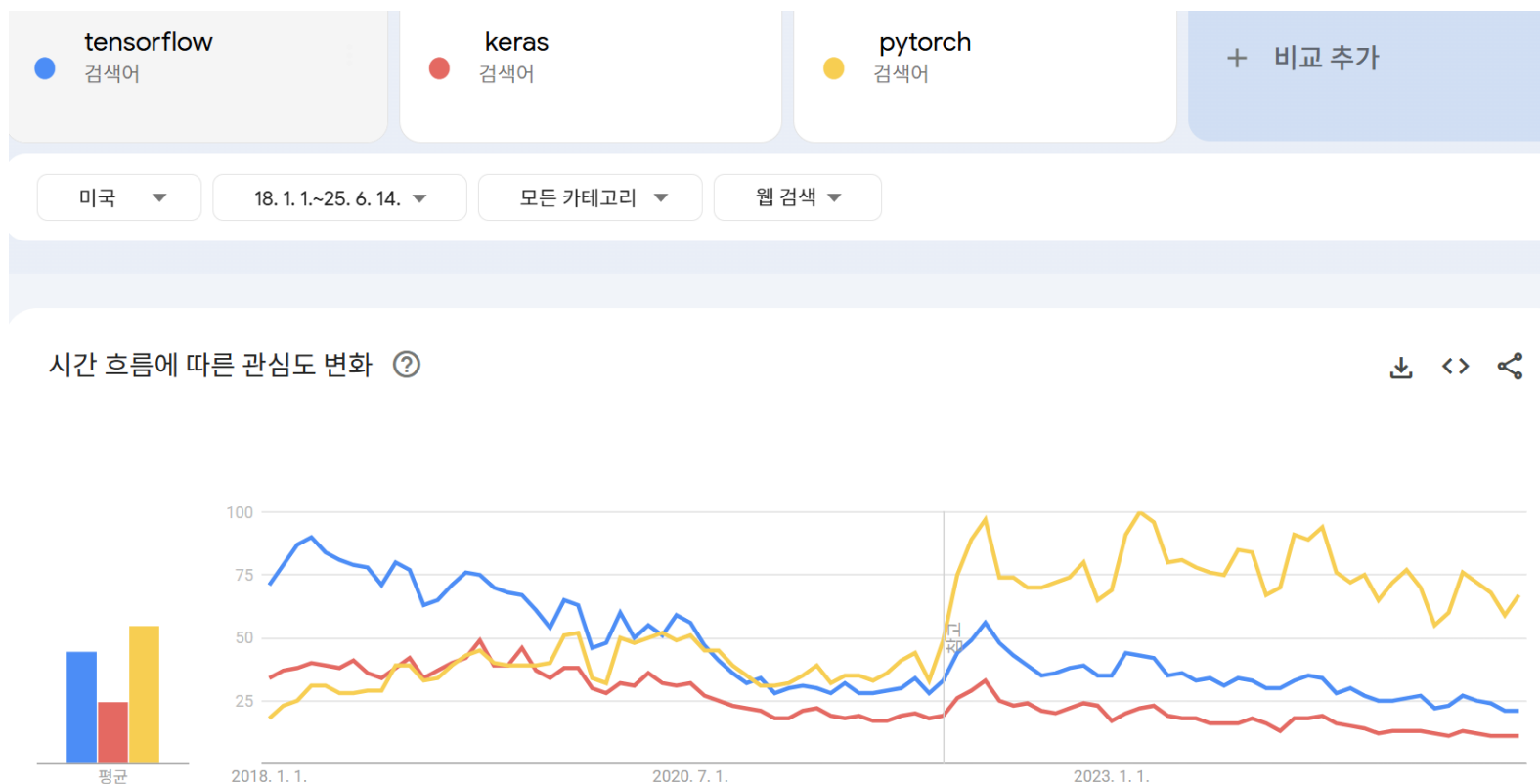
분류	모듈명	설명
데이터 분리, 검증&파라 미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드 서치로 최적 파라미터 추출 등의 API 제공
평가	sklearn.metrics	분류, 회귀, 클러스터링, Pairwise에 대한 다양한 성능 측정 방법 제공 (Accuracy, Precision, Recall, ROC-AUC, RMSE 등)
ML 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 (랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등)
	sklearn.linear_model	주로 회귀 관련 알고리즘을 지원 (선형 회귀, 릿지(Ridge), 라쏘(Lasso), 로지스틱 회귀 등)
	sklearn.naïve_bayes	나이브 베이즈 알고리즘 제공 (가우시안 NB, 다항분포 NB 등)
평가 (Assess)	sklearn.neighbors	최근접 이웃 알고리즘 제공 (K-NN 등)
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
	sklearn.cluster	비지도 클러스터링 알고리즘 제공 (K-평균, 계층형, DBSCAN 등)
유틸리티	sklearn.pipeline	Feature 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

주요 라이브러리

TensorFlow, Keras, PyTorch

프레임워크(Framework)란 응용 프로그램을 개발하기 위한 여러 라이브러리나 모듈 등을 효율적으로 사용할 수 있도록 하나로 묶어 놓은 일종의 패키지를 말한다.

딥러닝 프레임워크들 마다 각각의 고유한 기능과 특성을 제공한다. 이 중 현재 가장 많이 사용되고 있는 프레임워크로는 TensorFlow, Keras, PyTorch가 있다.



주요 라이브러리



TensorFlow

구글이 개발한 오픈소스 소프트웨어 라이브러리로 데이터 플로우 그래프(DataFlow Graph) 구조를 사용하는 특징을 가진다. 이는 수학 계산식과 데이터의 흐름을 노드(Node)와 엣지(Edge)를 사용한 방향성 그래프로 표현하는 방식이다. 각 노드 사이의 연결이나 다차원 배열 등을 의미하는 텐서 사이의 연결 관계를 풍부하게 표현할 수 있는 유연함을 장점으로 갖지만 딥러닝 프레임워크 중 난이도가 가장 높다.



Keras

보다 단순화된 인터페이스를 제공. Keras의 핵심적인 데이터 구조는 시퀀스 모델이다. Keras에서 제공하는 시퀀스 모델로 원하는 레이어를 쉽게 순차적으로 쌓을 수 있으며, 다중 출력 등 더 복잡한 모델을 구성할 때는 Keras 함수 API를 사용하여 쉽게 구성할 수 있다. 딥러닝 초급자도 손쉽게 딥러닝 모델을 개발하고 활용할 수 있도록 직관적인 API를 제공

주요 라이브러리



PyTorch

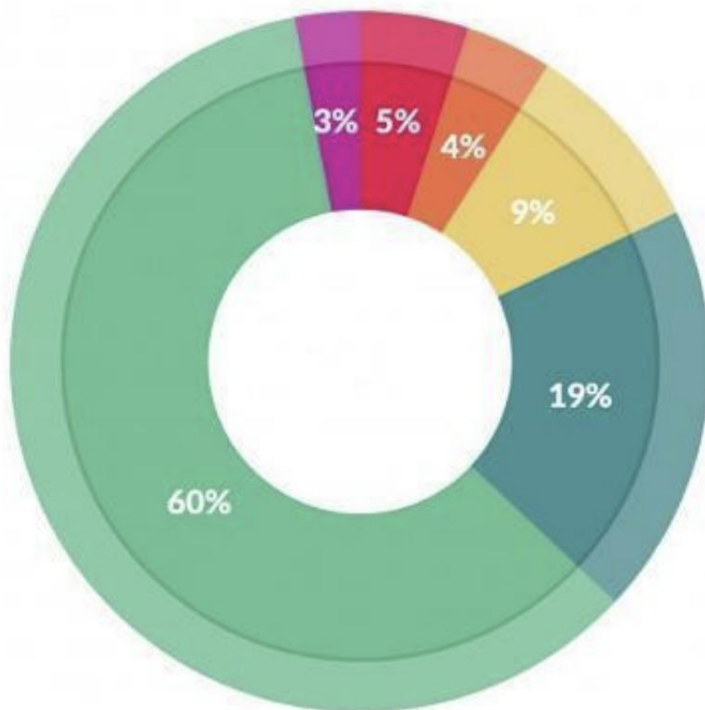
TensorFlow와 PyTorch의 가장 큰 차이점은 딥러닝을 구현하는 패러다임이 다르다는 것. TensorFlow 는 그래프를 먼저 만들어 두고 코드를 실행하는 시점에 데이터를 넣어 실행하는 Define-and-Run 프레임워크인 반면, PyTorch는 그래프를 만들면서 동시에 값을 할당하는 Define-by-Run 방식으로 코드를 깔끔하고 직관적으로 작성할 수 있다.

4 데이터 분석 단계

데이터 분석 단계

실습환경 구축과 주요라이브러리에 대한 준비가 되었다면, 빅데이터 분석 방법론의 '데이터 분석' 단계 중 다음의 3가지 프로세스에 대해 상세히 알아본다.

- 분석용 데이터 준비
- 텍스트 분석
- 탐색적 분석



What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

데이터 분석 단계

분석용 데이터 준비

- 단계에서 정의한 프로젝트 목표에 따라 분석에 필요한 데이터의 범위를 확인한다.
- 데이터 스토어로부터 분석에 필요한 정형, 비정형 데이터를 추출한다.
필요한 경우 적절한 가공을 통하여 입력 데이터로 사용될 수 있도록 한다.

텍스트 분석

- 비정형/반정형 텍스트 데이터의 경우 데이터 스토어에서 필요한 데이터를 추출
- 어휘/구문 분석, 감성 분석, 토픽 분석, 오피니언 분석, 소셜 네트워크 분석 등을 실시
- 텍스트 분석 결과는 모델링 태스크와 연동하여 프로젝트 목적에 부합하는 모델 구축

탐색적 분석

- 다양한 관점으로 평균, 분산 등 기초 통계량을 산출, 데이터의 분포와 변수 간의 관계 등 데이터 자체의 특성과 통계적 특성을 파악
- 시각화를 탐색적 데이터 분석을 위한 도구로 활용, 데이터의 가독성 향상

데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)

분석용 데이터 준비

데이터 분석 실습에서 가장 많이 접하는 데이터셋 중 하나인 IRIS Dataset 을 실습 목적에을 로딩하여 분석 한다.



아이리스 버시칼라
(Northern blue flag)



아이리스 버지니카

식물



부채붓꽃



식물

영어에서 번역됨 -

뽕죽한 붓꽃 인 붓꽃은 붓꽃과의 붓꽃 속의 꽃 식물의 한 종으로, Limniris 아속과 Tripetalae 시리즈에 속합니다. 알래스카, 메인, 캐나다, 러시아, 동북 아시아, 중국, 한국, 남북 일본을 포함한 북극해 전역의 넓은 범위에서 뿌리 중 다년생 식물입니다. 위키백과(영어)

원래 설명 보기 ▾

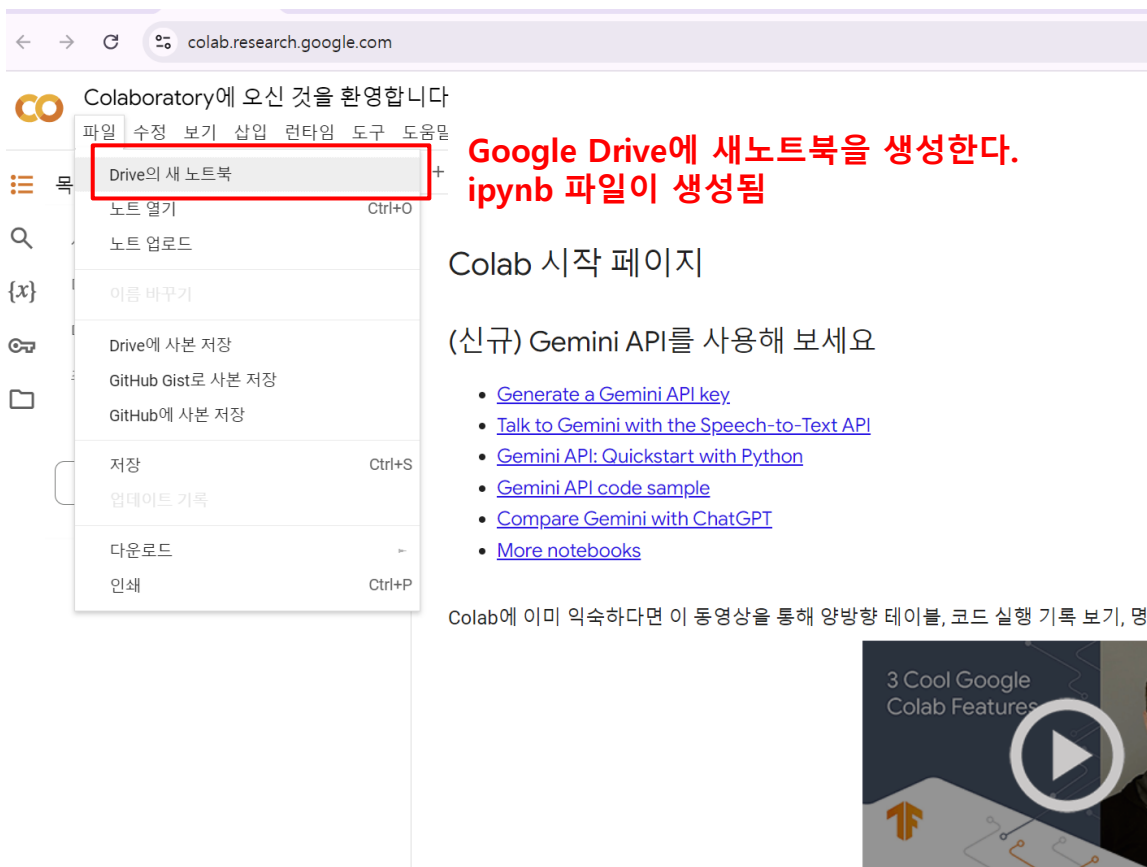
학명: Iris setosa

데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)

Colab의 Drive에서 새노트북

데이터 분석 실습에서 가장 많이 접하는 데이터셋 중 하나인 IRIS Dataset 을 실습 목적에을 로딩하여 분석 한다.



Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말

Drive의 새 노트북

노트 열기 Ctrl+O

노트 업로드

이름 바꾸기

Drive에 사본 저장

GitHub Gist로 사본 저장

GitHub에 사본 저장

저장 Ctrl+S

업데이트 기록

다운로드

인쇄 Ctrl+P

**Google Drive에 새노트북을 생성한다.
ipynb 파일이 생성됨**

Colab 시작 페이지

(신규) Gemini API를 사용해 보세요

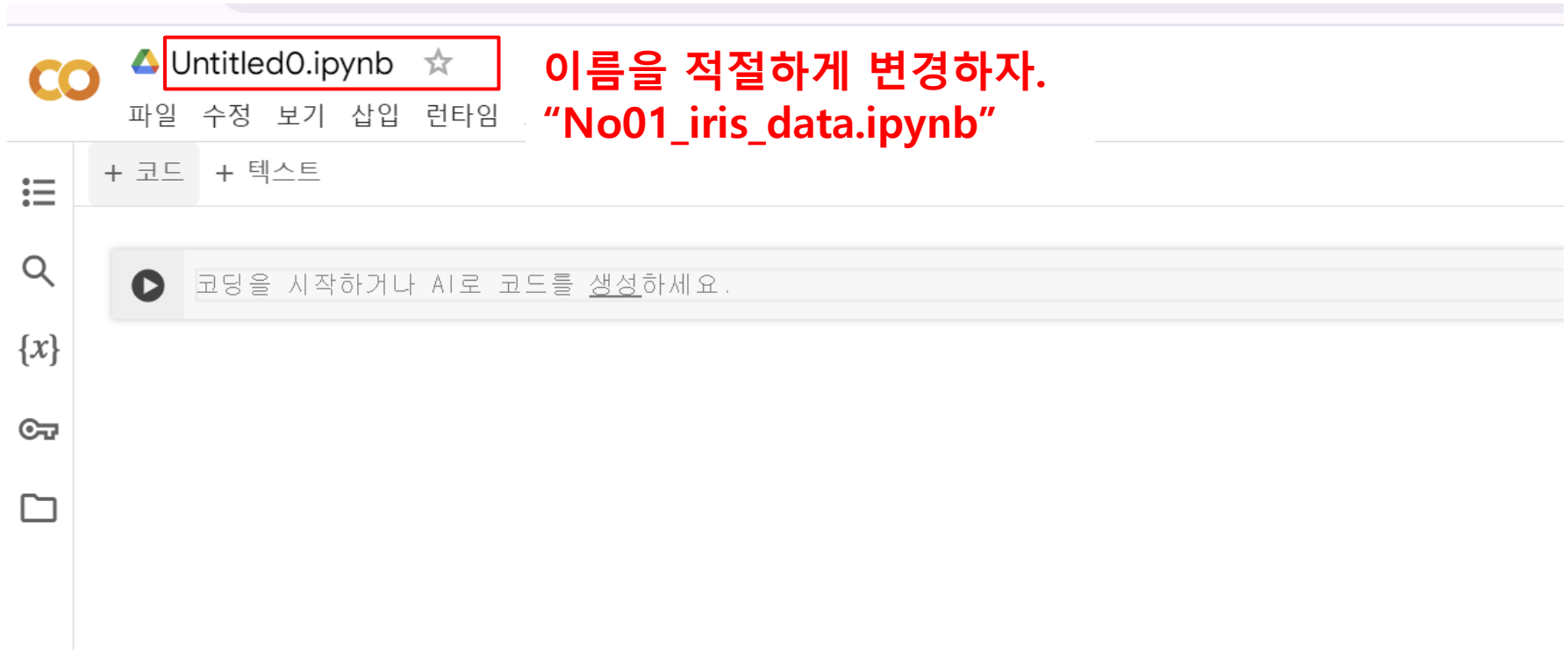
- [Generate a Gemini API key](#)
- [Talk to Gemini with the Speech-to-Text API](#)
- [Gemini API: Quickstart with Python](#)
- [Gemini API code sample](#)
- [Compare Gemini with ChatGPT](#)
- [More notebooks](#)

Colab에 이미 익숙하다면 이 동영상상을 통해 양방향 테이블, 코드 실행 기록 보기, 명

3 Cool Google Colab Features

데이터 분석 단계

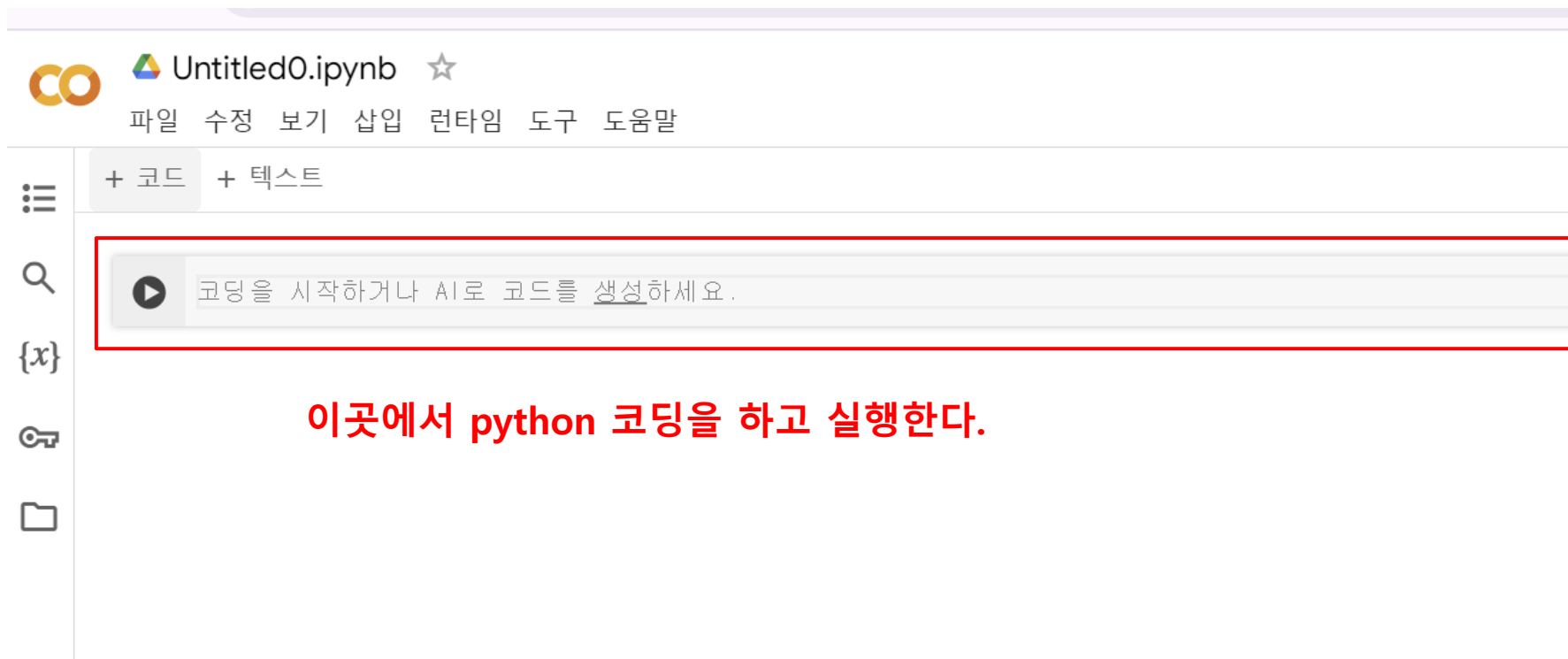
데이터 로드 (코랩 - 구글 드라이브 연동)



The screenshot shows the Google Colab interface. At the top, the file name 'Untitled0.ipynb' is highlighted with a red box. To the right of the box, red text reads: '이름을 적절하게 변경하자. "No01_iris_data.ipynb"'. Below the file name, there are tabs for '파일' (File), '수정' (Edit), '보기' (View), '삽입' (Insert), and '런타임' (Runtime). On the left sidebar, there are icons for a menu, search, code editor, and file explorer. The main area shows a code cell with a play button icon and the text '코딩을 시작하거나 AI로 코드를 생성하세요.'

데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)



이곳에서 python 코딩을 하고 실행한다.

데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)

■ SKlearn에서 데이터를 제공하는 방법은 세가지가 있다.

- 샘플 데이터셋 생성하기

- make_ 함수를 이용하는 방법
- SKLearn 모듈이 함수를 호출할때 파라미터에 맞춰서 데이터를 만들어준다.

예) 분류용 샘플 데이터셋 생성

```
▶ from sklearn.datasets import make_classification

# 2개의 클래스, 4차원 특징 공간의 샘플 데이터셋 생성
X, y = make_classification(n_samples=100, n_features=4, n_classes=2,
n_clusters_per_class=1, random_state=42)

print("Features shape:", X.shape)
print("Target shape:", y.shape)
```

```
⇒ Features shape: (100, 4)
   Target shape: (100,)
```


데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)

- 외부 데이터셋 다운로드 하기
 - fetch_로 시작하는 함수들을 사용하여 외부 데이터셋을 다운로드 할 수 있음

예) 20 Newsgroups 텍스트 데이터셋 다운로드

```

▶ from sklearn.datasets import fetch_20newsgroups

# 20 Newsgroups 데이터셋 다운로드 및 로드
newsgroups = fetch_20newsgroups(subset='train',
categories=['alt.atheism', 'sci.space'])

# 데이터와 타겟 확인
X = newsgroups.data
y = newsgroups.target

print("Number of samples:", len(X))
print("Categories:", newsgroups.target_names)
    
```

```

⇒ Number of samples: 1073
Categories: ['alt.atheism', 'sci.space']
    
```

데이터 분석 단계

데이터 로드 (코랩 - 구글 드라이브 연동)

- 내장 데이터셋 로드하기

- load_iris와 같이 자체 가지고 있는 데이터를 로드해준다.
- SKLearn 모듈이 데이터를 가지고 있다.

예) Iris 데이터셋 로드

```
▶ from sklearn.datasets import load_iris

# Iris 데이터셋 로드
iris = load_iris()

# 데이터와 타겟을 분리
X = iris.data
y = iris.target

print("Features:", X.shape)
print("Target:", y.shape)
```

```
↔ Features: (150, 4)
   Target: (150,)
```

데이터 분석 단계

Iris Dataset 분석

- 아래의 실습파일을 참고하여 데이터를 로드하고 저장하는 실습을 진행한다.

https://github.com/Harksu71/DataScienceAnalytics/blob/main/Lecture/SeSAC/No01_iris_data.ipynb

CO No01_iris_data.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트

```

import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
iris.target_names

```

array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

데이터 분석 단계

붓꽃 데이터 로드

- Iris 데이터셋 로드

```
▶ from sklearn.datasets import load_iris

# Iris 데이터셋 로드
iris = load_iris()

# 데이터와 타겟을 분리
X = iris.data
y = iris.target


print("Features:", X.shape)
print("Target:", y.shape)
```

```
⇒ Features: (150, 4)
   Target: (150,)
```

데이터 분석 단계

붓꽃 데이터 로드

- iris 데이터셋에 대한 설명 조회

 `print(iris.DESCR)`

```

↔ :Number of Instances: 150 (50 in each of three classes)
   :Number of Attributes: 4 numeric, predictive attributes and the class
   :Attribute Information:
       - sepal length in cm
       - sepal width in cm
       - petal length in cm
       - petal width in cm
       - class:
           - Iris-Setosa
           - Iris-Versicolour
           - Iris-Virginica

   :Summary Statistics:
   =====
           Min  Max  Mean  SD  Class Correlation
   =====
sepal length:  4.3  7.9   5.84  0.83    0.7826
sepal width:   2.0  4.4   3.05  0.43   -0.4194

```

데이터 분석 단계

데이터 저장 (로컬 디스크)

- local disk의 데이터는 재구동하면 사라짐

```
# 로컬디스크에 데이터셋 저장
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# 3. 라벨(종)을 숫자에서 이름으로 변환
iris_df['Label'] = iris.target
iris_df['Label'] =
iris_df['Label'].map(dict(enumerate(iris.target_names)))

# 4. DataFrame을 CSV 파일로 저장
file_path = 'iris_dataset.csv'
iris_df.to_csv(file_path, index=False)
```

데이터 분석 단계

데이터 저장 (구글 드라이브)

- Colab 에서 Google Drive를 마운트하고 저장한다

```
▶ from google.colab import drive
   drive.mount('/content/gdrive')

   # Google Drive에 파일 저장
   file_path = '/content/gdrive/MyDrive/Colab Notebooks/iris_dataset.csv'
   iris_df.to_csv(file_path, index=False)
   # Google Drive에서 데이터 로드
   myData = pd.read_csv(file_path)
```

No01_iris_data.ipynb 를 참고하여 데이터 분석을 진행하세요.

Q & A