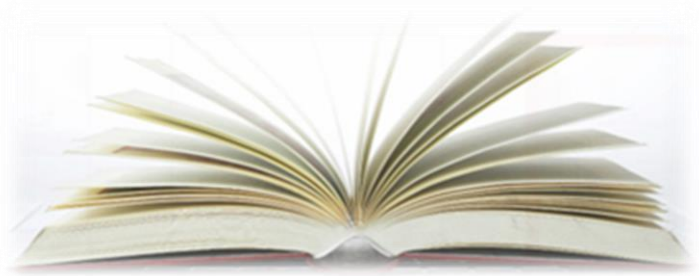
3

미니 프로젝트

1. 미니 프로젝트 개요
2. 채점 기준표

Unit 1



미니 프로젝트 개요

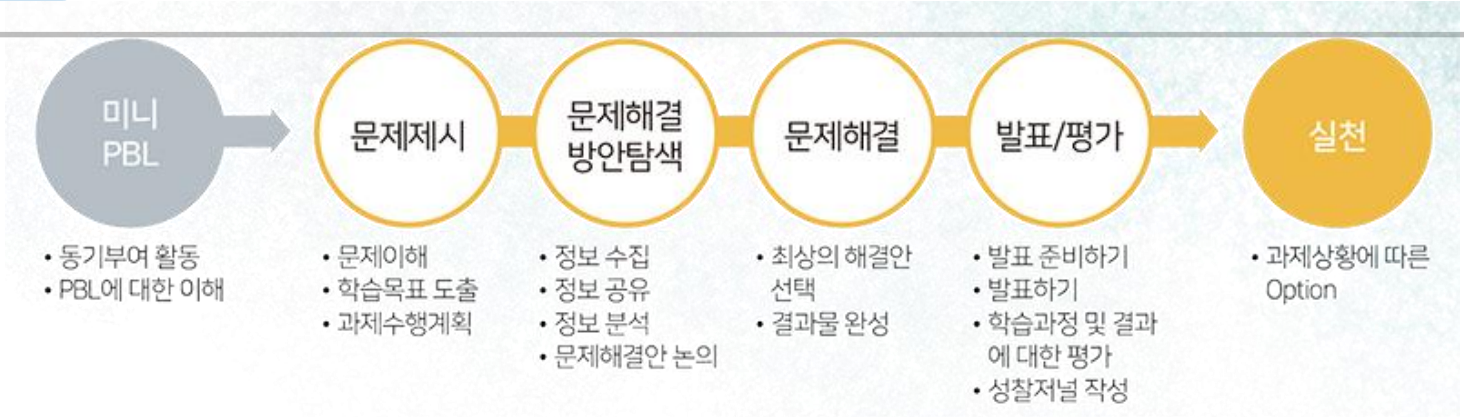
4차 산업혁명 시대에 PBL의 재조명

- PBL의 역사를 언급하자면 두 가지 갈래로 진행할 수 있다.
- 첫째는 '프로젝트 학습'으로서의 PBL의 경우인데, 이때는 Dewey의 경험 중심, 학습자 중심의 교육철학으로부터 기원을 찾게 된다.
- 반면에, '문제기반학습'으로서의 PBL의 경우일 때는 의과대학을 중심으로 '배움과 적용의 일치'를 주장한 Barrows 교수의 연구로부터 시작하여, 비슷한 시기에 등장한 Piaget, Vygotsky의 구성주의 학습이론과 연결지어 생각할 수 있다.
- 이처럼 서로 다른 출발점의 PBL이지만, 지향하는 목표는 '학습자 중심,' '실생활과 연결되는 학습활동' 이라는 점에서 교차점을 발견할 수 있다.
- 초연결되는 사회, 초지능화의 시대에 필요한 것은, 그 학습효과가 검증된 자기주도성, 창의성, 비판적 사고력, 탐구력, 의사소통력, 협력성, 공감력 등이 뛰어난 PBL이 재조명을 받고 있다.

다양한 PBL 모형

Barrows (1985)
문제제시
문제분석
정보나누기
가능한 해결책 생성하기
결과물 제시 및 발표
지식의 일반화 및 평가
McMaster
문제정의
선지식 탐색
가설설정
학습이슈 정의
자가학습
새로운 지식 적용 및 평가
학습에 대한 평가 및 성찰
Aalborg
문제분석
문제해결
보고/문서화
강인애 외 (2007)
문제제시
과제수행
평가 및 발표
IMSA
문제 이해
교육과정 탐색
문제해결
Delisle (1997)
문제와 관련 맺기
구조 설정
문제 탐색하기
문제 재탐색하기
해결책 만들고 수행하기
수행 및 과정 평가하기
SALFORD
그룹으로 문제 탐색
관련된 선지식의 공유
알고 있는 것 분석
기대 결과물 제시
공유된 정보. 학습 내용 확인
발표, 평가, 성찰
Maastricht
컨셉 명시
문제 정의
문제 분석
가능한 해결책 인지
과제, 학습목표 세우기
해결책 탐색
일반화 및 성찰
PBL in Medicine
시나리오 이해 문제 정의
학습목표 인지
조사, 증거 수집
문제 재정의
진단 및 처방 결정
학습요약 및 정리

PBL의 공통된 단계와 학습자 중심



❖ PBL을 실시하고자 할 때 문제의 하나는 PBL의 평가

❖ 보통 PBL 수업의 평가는 학습과정과 학습결과에 대한 평가를 모두 포함하며, 다양한 평가 도구를 활용한다. 가장 대표적인 것이 '성찰저널(일기)'로, 일종의 에세이식 학습일기 쓰기인데, 이를 활용하여 개별 학생들의 학습과정에 대한 평가를 매우 상세히 할 수 있다. 이와 동시에 PBL에서 많이 사용되는 평가도구는 '루브릭'7) 이라는 설명식 평가척도이다. 이때 주목할 점은 루브릭의 평가척도(요소)를 결정할 때 교사가 학생들과 '함께' 루브릭의 평가 준거를 결정한다는 점이다. 이러한 활동을 통해, 학생들은 자신들의 학습과정과 결과에서 자기평가는 물론 동료평가활동에서도 좀더 객관적인 평가로 이어질 수 있다. 결국 학생 참여적 평가활동이 이루어질 때, 학습자 중심 학습 환경으로서의 PBL은 비로소 완성된다고 하겠다.

PBL의 꽃은 실천이다

- ❖ 학생은 자신들에게 부족한 부분이 무엇인지를 '스스로' 도출해내는 과정을 통해 그들의 개별적 이해틀은 조금씩 변화의 움직임을 시도...
- ❖ 교사 스스로도 새로운 역할(촉진자, 동기부여자, 코치, 동료학습자)에 익숙해져야 ...

- ❖ 다시 4차 산업혁명 시대의 도래와 더불어 PBL이 재조명받고 있다. 이제는 더 이상 '이것은 PBL인가, 아닌가?' '저것은 맞고 이것은 아니다'를 잴 때가 아니다, PBL은 학습자 중심 교육이라는 큰 패러다임의 연속선상안에 존재한다는 것을 기억하고, 현재 본인이 다양한 PBL 버전의 일부라도 실천하고 있다면, 이미 '나는 PBL을 실천하고 있다'고 말할 수 있는 것이다.

분석 목표 정의서 확정

분석 기본 정의	분석 명칭	태양광 발전기의 발전량 예측	분석목표 확정일	2021-12-XX
	분석 목적	- 기후데이터를 이용한 월별 평균발전량으로 수익성 예측 필요성	분석 목표 워크숍	2021-12-XX
	분석 우선순위	상	담당 조직명	시장 품질팀
	분석 접근 방안	- 과거 기상자료를 통해 시뮬레이션을 돌려보며 발전량과 수익분석 가능성 파악 - 예측한 정상 발전량 패턴에서 벗어난 이상치들을 감지하여 대처하는 시스템 지원 등 활용성에 대한 정리		
성과 측정	정성적 기준	신규 기법/기술 : 다양한 예측 기법 및 머신러닝 / 딥러닝 활용 외부 데이터: 태양광 발전량 예측을 위한 초고해상도 일사량 분석 정보 신규 데이터 : …		-
	정량적 기준	Best Model 선정		
데이터 정보	내부 데이터	-	데이터 입수 난이도	중
	외부 데이터	기상청 날씨 정보	데이터 입수 난이도	하

프로젝트 헌장 확정

프로젝트 헌장(Project Charter)			
프로젝트 명 (Project Name)	태양광 발전기의 발전량 예측		
프로젝트 설명 (Project Description)	각 팀원이 태양광 발전기의 발전량 예측을 위한 AI·딥러닝 분석모델 선정과 핵심 개념 정리 및 목표 설정 후, 데이터 수집, 준비, 분석 모델 구현, 평가 및 모델 정확도 확인, 결과 분석, 시각화에 대한 개인별 결과를 취합하여 발표자료 작성		
프로젝트 매니저 (Project Manager, PM)	홍길동	승인 날짜 (Date Approved)	
프로젝트 스폰서 (Project Sponsor)		서명 (Signature)	
비즈니스 케이스 (Business Case)		목표(Goals) / 산출물(Deliverables)	
		1) 프로젝트 계획 수립:태양광 발전기의 발전량 예측 기술 필요성 및 배경에 대한 이해 - 기후데이터를 이용한 월별 평균발전량으로 수익성 예측 필요성 - 과거 기상자료를 통해 시뮬레이션을 돌려보며 발전량과 수익분석 가능성 파악 - 예측한 정상 발전량 패턴에서 벗어난 이상치들을 감지하여 대처하는 시스템 지원 등 활용성에 대한 정리 2) 태양광 발전기의 발전량 예측을 위한 AI·딥러닝 분석모델(예: RNN, LSTM 등) 선정과 핵심 개념 정리 및 목표 설정 3) 데이터 수집, 준비(데이터 조합/전처리/환경설정 및 라이브러리 준비) 4) 탐색(데이터 정제하기/데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기/데이터 분포 조정하기) 5) 분석 모델 구현 6) 분석 모델 평가 및 모델 정확도 확인 7) 분석 결과 및 시각화 자료 취합 후 발표자료 작성 8) 발표	
팀 구성원(Team Member)			
이름(Name)	역할(Role)		
홍길동	PM		
박문수	엔지니어		
위험과 제약사항(Risk and Constraints)		주요 일정(Milestones)	

빅 데이터 분석 프로세스 확인

- ❖ 업무 이해 혹은 문제 정의(problem definition)
- ❖ 데이터 이해(data definition)
- ❖ 실험 계획(design of experiment) 혹은 표본화(sampling)
- ❖ 데이터 추출 혹은 취득(data acquisition)
- ❖ 데이터 가공(daa processing, data wragling)
  - PDCA(Plan-Do-Check-Action) 주기에 따라 반복
- ❖ 탐색적 분석과 데이터 시각화(exploratory data analysis, data visualization)
- ❖ 확증적 데이터 분석(Confirmatory Data Analysis, CDA) 혹은 통계적 모델링 혹은 모형화(statistical modeling)
  - 지도 학습 모델
  - 자율 학습 모델
- ❖ 효과 검증
  - A/B 테스트 등
- ❖ 서비스 구현
  - KPI(Key Performance Indicator) 모니터링 가능

수행 예1

문제 개요

❖ <아래> 내용은 분석용 데이터 구축 전 데이터 셋 내용과 분석용 데이터 구축 후 데이터 셋 내용을 보여준다. 직접 준비한 데이터를 활용하거나, 기타 수집한 관찰 데이터를 이용, 포지셔닝 전략 수립을 위한 분석용 데이터 구축을 하고자 한다. 위에서 보여준 예를 참고하여 분석용 데이터 구축을 보이시오.

<아래>

1. 분석용 데이터 구축 전

```
import pandas as pd
```

```
스마트폰=pd.read_excel("phone.xlsx")
```

EDA

```
[ ] 스마트폰.head()
```

	Unnamed: 0	스마트폰 관련	Unnamed: 2	Unnamed: 3	사용자 관련	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	출시년도	구매시기	디스플레이 크기₩n(인치)	연령	키₩n(cm)	몸무게₩n(kg)	평균 스마트폰₩n사용시간₩n(분/일)	평균 컴퓨터₩n사용시간₩n(분/일)	데이터₩n사용량₩n(mb/일)
1	1.0	2015	2015	5	45	173	75	60	500	100
2	2.0	2014	2015	4.5	27	176	59	70	30	50
3	3.0	2015	2015	5	29	183	65	120	300	200
4	4.0	2015	2016	5	28	172	63	80	60	190

2. 분석용 데이터 구축 후

```
[ ] import pandas as pd
    스마트폰=pd.read_csv('스마트폰.csv')
```

```
스마트폰.head()
```

	Unnamed: 0	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
0	0	2	2014	2015	4.5	27	176	59	70	30	50
1	1	3	2015	2015	5.0	29	183	65	120	300	200
2	2	4	2015	2016	5.0	28	172	63	80	60	190
3	3	5	2015	2015	5.0	24	179	65	90	30	500
4	4	6	2016	2016	5.5	34	175	73	80	480	160

```
[ ] 스마트폰=스마트폰.drop(columns=['Unnamed: 0'], axis=1)
```

```
[ ] 스마트폰.head()
```

	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
0	2	2014	2015	4.5	27	176	59	70	30	50
1	3	2015	2015	5.0	29	183	65	120	300	200
2	4	2015	2016	5.0	28	172	63	80	60	190
3	5	2015	2015	5.0	24	179	65	90	30	500
4	6	2016	2016	5.5	34	175	73	80	480	160

수행 예3

개요

❖ Pandas와 Matplotlib 패키지를 이용하여 다음을 연습하고자 한다.

① 엑셀 파일 불러오기
② 데이터 개요 보기
③ 정렬하기
④ 필터링하기
⑤ 집계하기 (합, 평균, 최소값, 최대값)
⑥ 히스토그램 그리기
⑦ 산점도 그리기

수행 예3

사용할 패키지 – PANDAS & MATPLOTLIB

❖ **Pandas는 데이터셋을 불러와 여러 조작할 수 있는 패키지고, Matplotlib은 데이터를 그릴 때 사용**

수행 예3

데이터 불러오기

❖ 확장자에 따라 불러오는 함수가 다르다. csv 확장자라면 pd.read_csv 함수를, xls와 같은 엑셀 확장자라면 pd.read_excel 함수로 불러오면 된다.

데이터 불러오기

```
[4] !rm -rf bigStudy

    !git clone 'https://github.com/looker2zip/bigStudy.git'
```

```
Cloning into 'bigStudy'...
remote: Enumerating objects: 66, done.
remote: Counting objects: 100% (66/66), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 66 (delta 10), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (66/66), done.
```

```
# csv 파일 불러와 csv_data 변수에 저장하기
csv_data = pd.read_csv('bigStudy/dataset/tips.csv')
# 엑셀 파일 불러와  xls_data 변수에 저장하기
xls_data = pd.read_excel('bigStudy/dataset/tips.xlsx')
```

수행 예3

데이터 개요

❖ 전체 데이터 셋을 다 살펴보기엔 양이 많을 수 있으니 몇 줄만 보고 데이터를 파악할 필요가 있다.

```
# 앞 몇 줄만 보기
csv_data.head()
# 뒷 몇 줄만 보기
csv_data.tail()
# 몇 줄을 볼지 지정하여 보기
csv_data.head(3)
```

	total_bill	tip	gender	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3

수행 예3

데이터 개요

❖ 숫자 데이터의 요약을 보고 싶다면 describe() 함수를 사용

```
# 숫자 데이터 요약하기
csv_data.describe()
# 숫자가 아닌 데이터도 같이 보기
csv_data.describe(include='all')
```

	total_bill	tip	gender	smoker	day	time	size
count	244.000000	244.000000	244	244	244	244	244.000000
unique	NaN	NaN	2	2	4	2	NaN
top	NaN	NaN	Male	No	Sat	Dinner	NaN
freq	NaN	NaN	157	151	87	176	NaN
mean	19.785943	2.998279	NaN	NaN	NaN	NaN	2.569672
std	8.902412	1.383638	NaN	NaN	NaN	NaN	0.951100
min	3.070000	1.000000	NaN	NaN	NaN	NaN	1.000000
25%	13.347500	2.000000	NaN	NaN	NaN	NaN	2.000000
50%	17.795000	2.900000	NaN	NaN	NaN	NaN	2.000000
75%	24.127500	3.562500	NaN	NaN	NaN	NaN	3.000000
max	50.810000	10.000000	NaN	NaN	NaN	NaN	6.000000

수행 예3

정렬하기

❖ 정렬은 sort_values(by'column' 함수를 사용

```
# 'total_bill' 열을 기준 잡아 정렬하기
sorted = csv_data.sort_values(by ='total_bill')
sorted.head(10)
```

	total_bill	tip	gender	smoker	day	time	size
67	3.07	1.00	Female	Yes	Sat	Dinner	1
92	5.75	1.00	Female	Yes	Fri	Dinner	2
111	7.25	1.00	Female	No	Sat	Dinner	1
172	7.25	5.15	Male	Yes	Sun	Dinner	2
149	7.51	2.00	Male	No	Thur	Lunch	2
195	7.56	1.44	Male	No	Thur	Lunch	2
218	7.74	1.44	Male	Yes	Sat	Dinner	2
145	8.35	1.50	Female	No	Thur	Lunch	2
135	8.51	1.25	Female	No	Thur	Lunch	2
126	8.52	1.48	Male	No	Thur	Lunch	2

```
# 'day' 열만 출력하기
sorted['day'].head()
```

```
67      Sat
92      Fri
111     Sat
172     Sun
149    Thur
Name: day, dtype: object
```

수행 예3

필터링 하기

❖ 특정 조건을 만족시키도록 필터링을 하고 싶다면 앞서 말한 []를 이용하면 쉽다.

```
# 'total_bill' 이 5.0보다 큰 값만 출력하기
(sorted['total_bill'] > 5.0).head()
# 위의 식이 어떤 값을 갖는지 display
display((sorted['total_bill'] > 5.0).head())
```

```
67     False
92      True
111     True
172     True
149     True
Name: total_bill, dtype: bool
```

수행 예3

필터링 하기

❖ 그러나 그냥 필터를 걸면 boolean 값으로 나온다. 우리가 보기 편한 방식으로 바꾸려면 아래와 같이 취하면 된다.

```
# sorted 데이터에서 total_bill이 5.0보다 큰 값의 헤드 값을 보여준다.
display(sorted[(sorted['total_bill'] > 5.0)].head())

# 만약 5.0 보다 작은 값을 보고 싶다면 부호를 < 로 바꾸는 법도 있고, ~을 이용하는 법도 있다.
display(sorted[(sorted['total_bill'] < 5.0)].head())
display(sorted[~(sorted['total_bill'] > 5.0)].head())
```

	total_bill	tip	gender	smoker	day	time	size
92	5.75	1.00	Female	Yes	Fri	Dinner	2
111	7.25	1.00	Female	No	Sat	Dinner	1
172	7.25	5.15	Male	Yes	Sun	Dinner	2
149	7.51	2.00	Male	No	Thur	Lunch	2
195	7.56	1.44	Male	No	Thur	Lunch	2

	total_bill	tip	gender	smoker	day	time	size
67	3.07	1.0	Female	Yes	Sat	Dinner	1

	total_bill	tip	gender	smoker	day	time	size
67	3.07	1.0	Female	Yes	Sat	Dinner	1

수행 예3

집계하기

❖ 합계, 평균 등 집계 함수는 groupby 함수를 사용하여 간단히 구할 수 있다.

```
# 합계
sorted.groupby('day').sum()

# 평균
sorted.groupby('day').mean()

# 표준편차
sorted.groupby('day').std()

# 최소값
sorted.groupby('day').min()

# 최대값
sorted.groupby('day').max()
```

day	total_bill	tip	gender	smoker	time	size
Fri	40.17	4.73	Male	Yes	Lunch	4
Sat	50.81	10.00	Male	Yes	Dinner	5
Sun	48.17	6.50	Male	Yes	Dinner	6
Thur	43.11	6.70	Male	Yes	Lunch	6

수행 예3

집계하기

❖ 특정 레코드에 개별적으로 접근하고 싶을 때는 for 문을 이용하여 각 값을 별도의 변수에 할당할 수 있다.

```
[20] # tip이 5보다 큰 값 저장하기
     big_tips = csv_data[csv_data['tip'] > 5]

     # 값 확인하기
     display(big_tips.columns)
```

```
Index(['total_bill', 'tip', 'gender', 'smoker', 'day', 'time', 'size'], dtype='object')
```

```
for index, row in big_tips.iterrows():
    print("Record " + str(index) + "has total tip " + str(row['total_bill']))
```

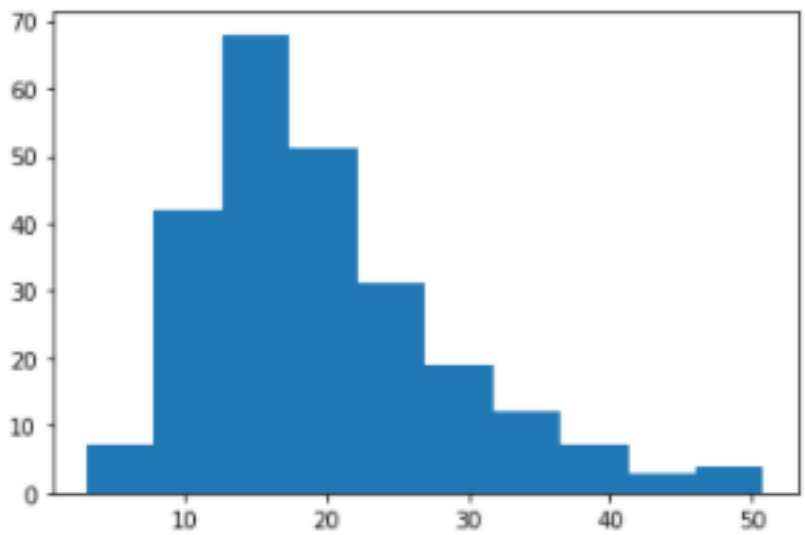
```
Record 23has total tip 39.42
Record 44has total tip 30.4
Record 47has total tip 32.4
Record 52has total tip 34.81
Record 59has total tip 48.27
Record 85has total tip 34.83
Record 88has total tip 24.71
Record 116has total tip 29.93
Record 141has total tip 34.3
Record 155has total tip 29.85
Record 170has total tip 50.81
Record 172has total tip 7.25
Record 181has total tip 23.33
Record 183has total tip 23.17
Record 211has total tip 25.89
Record 212has total tip 48.33
Record 214has total tip 28.17
Record 239has total tip 29.03
```

수행 예3

히스토그램 그리기

❖ matplotlib 패키지를 활용한다. 실제 그래프가 그려지는 것은 show() 함수를 호출할 경우에만 이루어 진다. line(), hist(), bar(), pie(), scatter() 등 이름만으로도 쉽게 유추가 가능

```
# 히스토그램 그리기
plt.hist(csv_data['total_bill'])
plt.show()
```

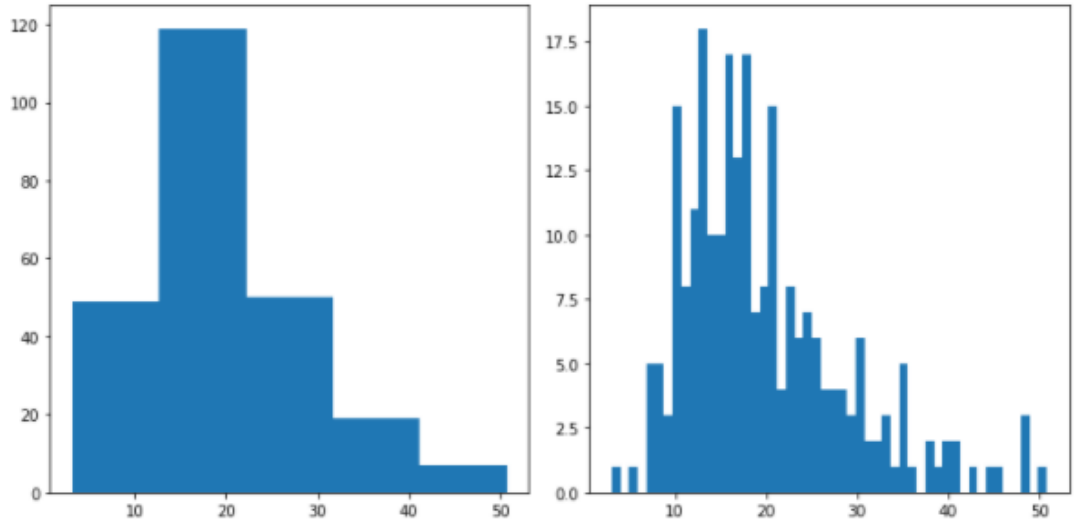


수행 예3

히스토그램 그리기

❖ 여러 그래프를 그리고 싶다면 subplots()를 이용

```
# 행의 개수, 열의 개수, y축을 공유할 것인지 여부, 전체 사이즈를 각각 지정해준다.
fig, axs = plt.subplots(1, 2, sharey=False, figsize=(10,5))

# 첫번째 히스토그램 그리기
axs[0].hist(csv_data['total_bill'], bins =5)

# 두번째 히스토그램 그리기
axs[1].hist(csv_data['total_bill'], bins = 50)

# 그래프의 겹치는 현상 방지하기
plt.tight_layout()
plt.show()
```

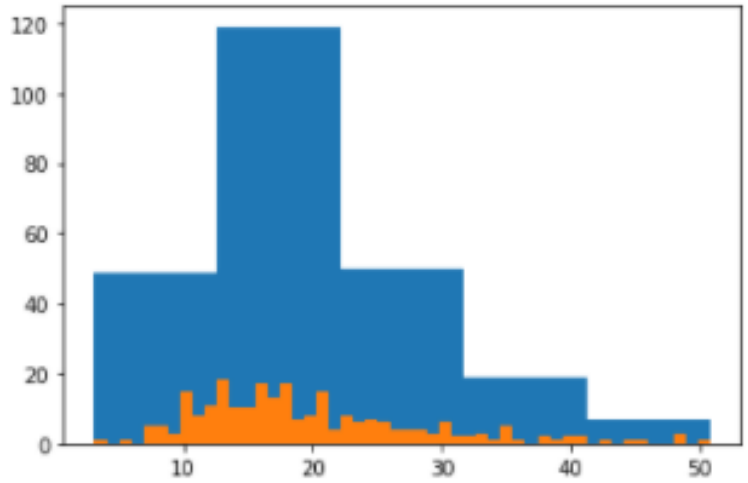


수행 예3

히스토그램 그리기

❖ 여러 그래프를 하나로 겹쳐 그리고 싶다면 다음과 같이 수행

```
# 첫번째 히스토그램 그리기
plt.hist(csv_data['total_bill'], bins =5)

# 두번째 히스토그램 그리기
plt.hist(csv_data['total_bill'], bins = 50)

plt.show()
```

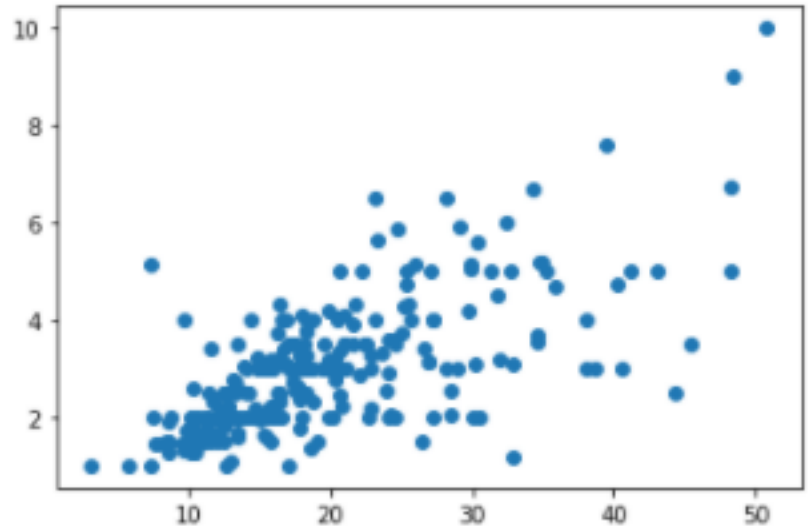


수행 예3

산점도 그리기

❖ 산점도 역시 간단한데 x, y 축만 지정

```
plt.scatter(csv_data['total_bill'], csv_data['tip'])
plt.show()
```

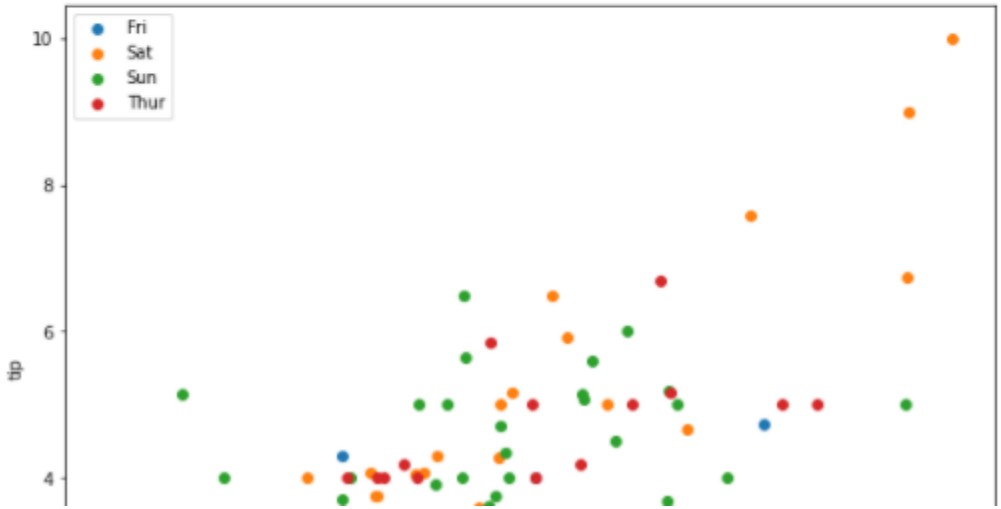


수행 예3

산점도 그리기

❖ 각 점에 대한 범례 표시를 원한다면 annotate()를 사용

```
# 그래프 사이즈 지정하기
plt.figure(figsize = (10, 8))

# day로 그룹화하여 모든 그룹에 반복하기
for name, group in csv_data.groupby('day'):
        # 각 그룹에 맞게 점 그리기
        plt.scatter(group['total_bill'], group['tip'], label = name)

# 축 이름 지정하기
plt.xlabel('total_bill')
plt.ylabel('tip')

# 범례 추가
plt.legend()

plt.show()
```



클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

개관

❖ K-Means와 계층적 클러스터링을 모두 다뤄보는 파이썬 예제이다. 본 예제에서는 학생들의 성적 데이터를 분석하여 유의미한 인사이트를 얻어 보는 과정을 다룬다.

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

학생 데이터 파이썬 분석

❖ 데이터는 학생들의 수강 데이터로, 어떤 학기에 어떤 수업을 들었는지 출석과 성적까지 포함된 데이터이다. 한 학생이 여러 학기에 여러 과목을 들을 가능성이 크므로, 이름으로 그룹핑을 하여 이 학생의 출석과 성적이 어떤지 살펴보자.

```python
# 패키지 불러오기
import pandas as pd

# 데이터 읽기
student_data = pd.read_excel('bigStudy/dataset/ex1.xlsx')

# 데이터 개요 살펴보기
display(student_data.head())
```

	Semester	Name	Course	Mark	Attended
0	FSS2010	Alex Krausche	Database Systems I	1.3	13
1	FSS2010	Tanja Becker	Database Systems I	2.0	12
2	FSS2010	Mariano Selina	Database Systems I	1.7	5
3	FSS2010	Otto Blacher	Database Systems I	2.3	13
4	FSS2010	Frank Fester	Database Systems I	2.0	13

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

학생 데이터 파이썬 분석

❖ 데이터는 학생들의 수강 데이터로, 어떤 학기에 어떤 수업을 들었는지 출석과 성적까지 포함된 데이터이다. 한 학생이 여러 학기에 여러 과목을 들을 가능성이 크므로, 이름으로 그룹핑을 하여 이 학생의 출석과 성적이 어떤지 살펴보자.

```
# 이름으로 그룹핑하여 평균값 구하기
students = student_data.groupby('Name').mean()

# 다시 데이터 살펴보기
display(students.head())
```

Name	Mark	Attended
Alex Krausche	1.325	12.500000
Avid Morvita	3.100	11.333333
Frank Fester	2.200	11.600000
Mariano Selina	1.680	6.200000
Michaela Martke	3.660	7.400000

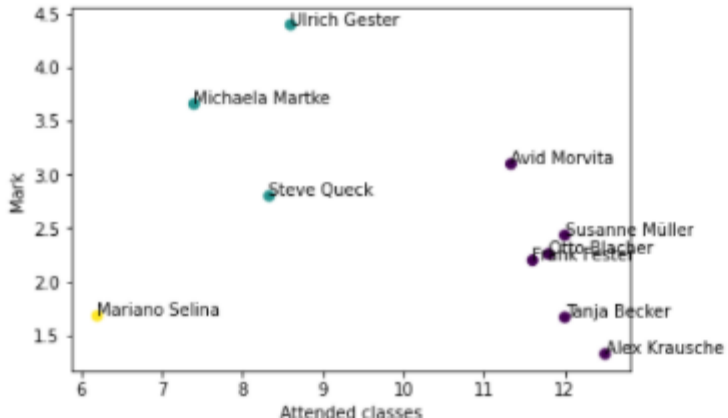
클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

학생 데이터 파이썬 분석

❖ 학생들을 군집으로 나눠보자. 출석과 성적 데이터 기반이니 대략적으로 1) 출석도 좋고 성적도 좋은 학생군 2) 출석은 나쁘지만 성적은 좋은 학생군 3) 출석도 나쁘고 성적도 나쁜 학생군 으로 나눠보도록 하자.

```
from sklearn.cluster import KMeans
```

```
# k=3 클러스터 생성
estimator = KMeans(n_clusters = 3)
cluster_ids = estimator.fit_predict(students)

# 플롯
plt.scatter(students['Attended'], students['Mark'], c=cluster_ids)
plt.xlabel("Attended classes")
plt.ylabel("Mark")

# 범례 달기
for name, mark, attended in students.itertuples():
    plt.annotate(name, (attended, mark))

plt.show()
```



학생 데이터 파이썬 분석

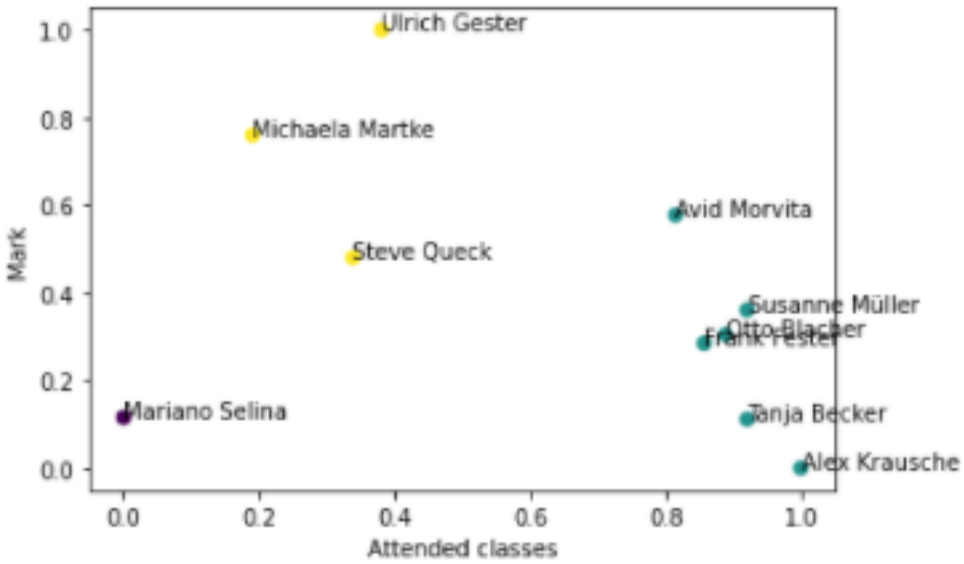
❖ 학생 데이터를 클러스터링했더니 '출석'에 좌우된 것이 보인다. 성적은 0~5점 사이인데 반해 출석은 0~13까지 범위가 더 크기 때문이다. 두 개의 클래스의 단위가 다를 때는 표준화(normalization)를 해주어야 한다. 여기서는 MinMaxScaler()로 표준화를 진행하고자 한다.

```
from sklearn.preprocessing import MinMaxScaler
```

```
# normalizer 생성
min_max_scaler = MinMaxScaler()

# 표준화하기
students[['Mark', 'Attended']] = min_max_scaler.fit_transform(students[['Mark', 'Attended']])

# 클러스터링 생성
estimator = KMeans(n_clusters = 3)
cluster_ids = estimator.fit_predict(students)


plt.scatter(students['Attended'], students['Mark'], c=cluster_ids)
plt.xlabel("Attended classes")
plt.ylabel("Mark")

for name, mark, attended in students.itertuples():
    plt.annotate(name, (attended, mark))

plt.show()
```

학생 데이터 파이썬 분석

❖ 학생 데이터를 클러스터링했더니 '출석'에 좌우된 것이 보인다. 성적은 0~5점 사이인데 반해 출석은 0~13까지 범위가 더 크기 때문이다. 두 개의 클래스의 단위가 다를 때는 표준화(normalization)를 해주어야 한다. 여기서는 MinMaxScaler()로 표준화를 진행하고자 한다.



클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

학생 데이터 파이썬 분석

❖ 같은 데이터로 계층적 클러스터링을 해보자. 어떤 linkage mode(군집 간의 거리를 재는 방식)가 적합한지 반복해서 돌려보고 시각화

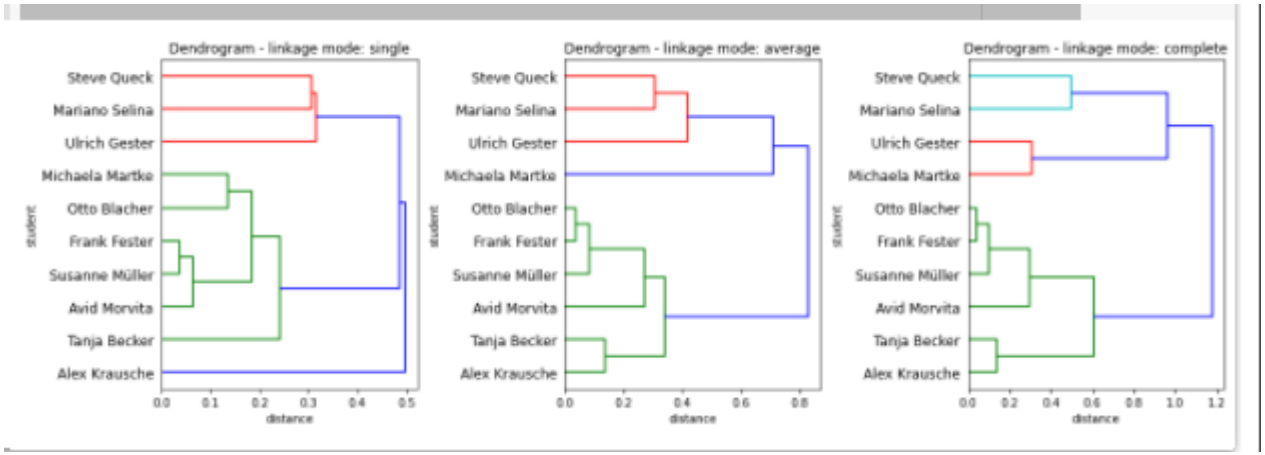
```
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
# 최단연결법은 single, 평균연결법은 average, 최장연결법은 complete으로 표기한다.
modes = ['single', 'average', 'complete']

plt.figure(figsize=(20,5))

# subplot() 함수를 사용하여 서브 플롯을 추가하고 반환 값을 y_axis라는 변수에 할당
# sharey 매개 변수를 사용하여 서브 플롯에 대한 모든 호출에 이 변수를 전달하여 모든 플롯이 동일한
y_axis = None

# 모든 linkage mode 반복 생성
for i, mode in enumerate(modes):
    # 서브플롯 추가, y축은 공유
    y_axis = plt.subplot(1, 4, i + 1, sharey = y_axis)

    # 레이블링
    plt.title('Dendrogram - linkage mode: {}'.format(mode))
    plt.xlabel('distance')
    plt.ylabel('student')

    # 클러스터링
    clustering = linkage(students[['Mark', 'Attended']], mode)

    # 덴드로그램
    dendrogram(clustering, labels=list(students.index), orientation='right')
plt.tight_layout()
plt.show()
```

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

학생 데이터 파이썬 분석

❖ 같은 데이터로 계층적 클러스터링을 해보자. 어떤 linkage mode(군집 간의 거리를 재는 방식)가 적합한지 반복해서 돌려보고 시각화



Unit 2

채점 기준표

채점 기준표

채점 항목	배점	결과	채점 세부항목	
2-1. 분석용 데이터 셋을 구축하는 능력	80		분석용 데이터 적재부터 구축까지 흐름을 잘 파악하고 있다.	80
			분석용 데이터 적재부터 구축까지 흐름을 잘 파악하고 있지는 않지만, 의미가 동일하게 기재하고 있다.	50
			분석용 데이터 적재부터 구축까지 흐름을 일부만 파악하고 있다.	20
			적지 않았을 경우	0
2-2. 분석용 데이터 셋을 식별하는 능력	20		분석용 데이터 셋을 잘 식별하고 있다.	20
			분석용 데이터 셋을 잘 식별하고 있지는 않지만, 의미가 동일하게 기재하고 있다.	15
			분석용 데이터 셋을 일부만 식별하고 있다.	10
			적지 않았을 경우	0
합계	100			

Unit 3

custom dataset으로 YOLOv5 학습하기

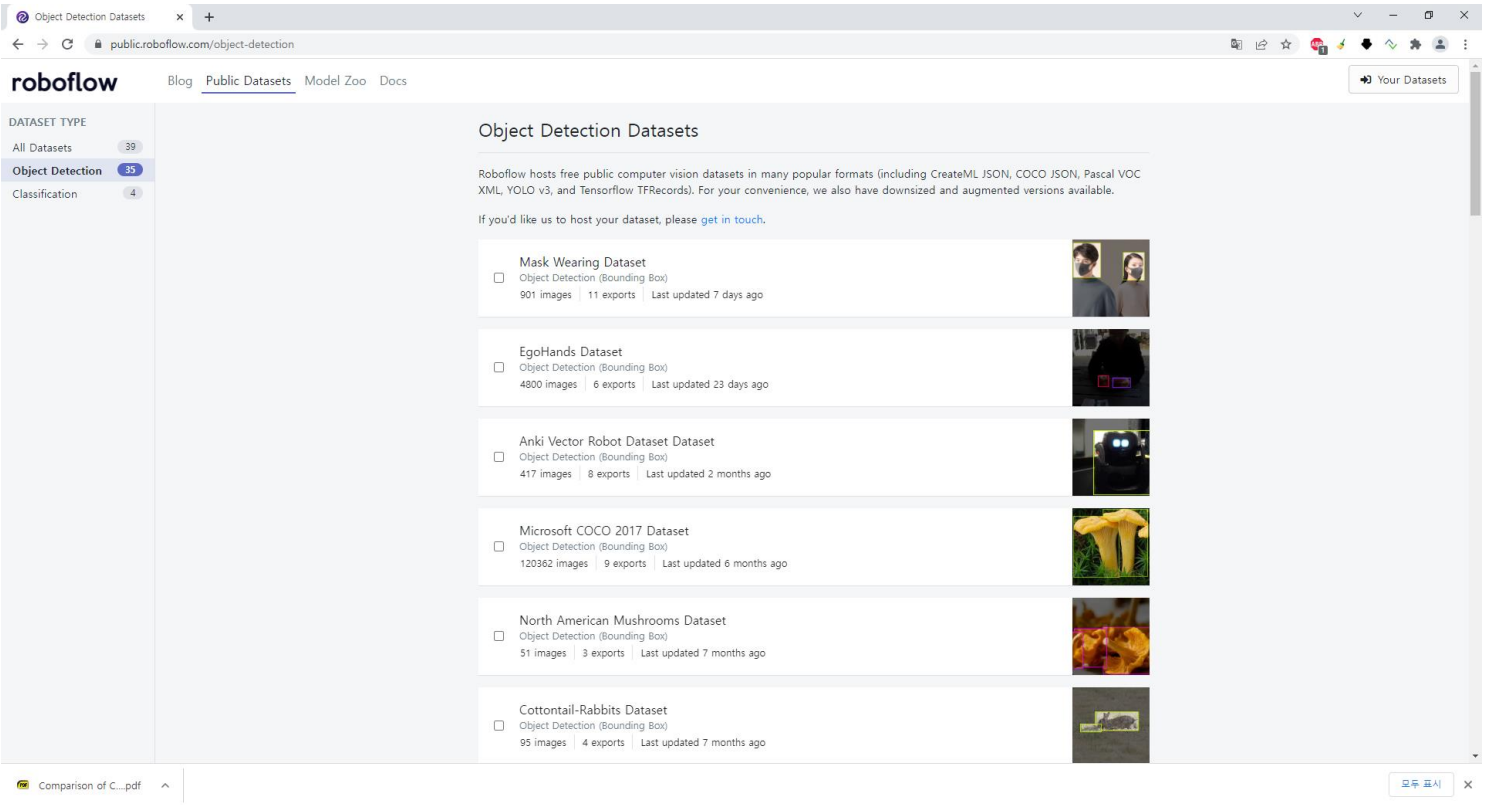
YOLO

❖ YOLO(You Only Look Once)는 널리 쓰이는 object detection 알고리즘이다. 최근에는 YOLOv5 까지 출시되었다. 여기서는 공식 github 계정에 업로드된 YOLOv5 코드로 custom dataset을 학습하는 방법에 대하여 설명한다. google colab 환경에서 진행되었다.

데이터셋 소개

North American Mushrooms Dataset

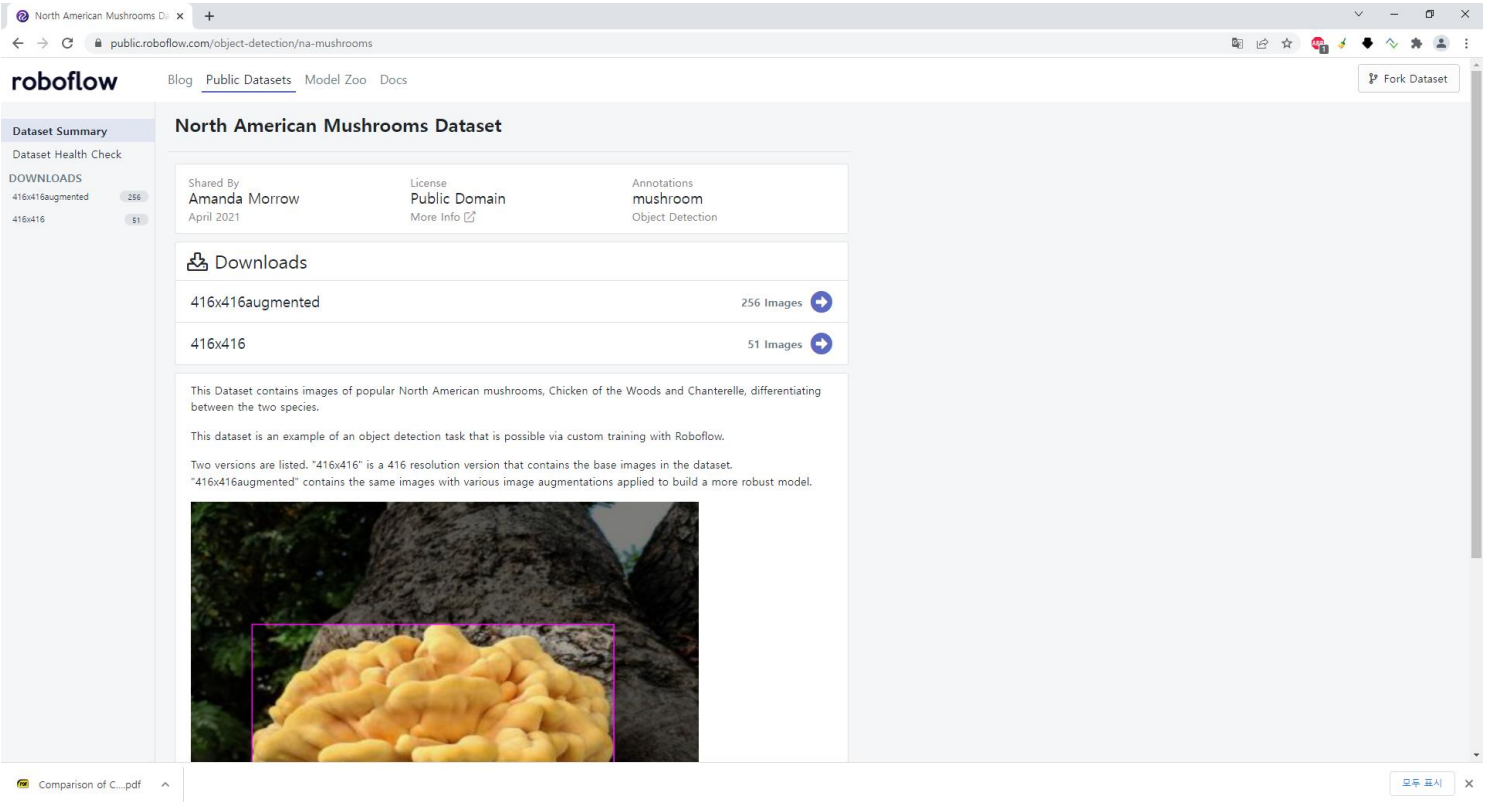
- ❖ 실습에 사용되는 데이터셋은 roboflow에서 제공되는 North American Mushrooms Dataset
- ❖ https://public.roboflow.com/object-detection



데이터셋 소개

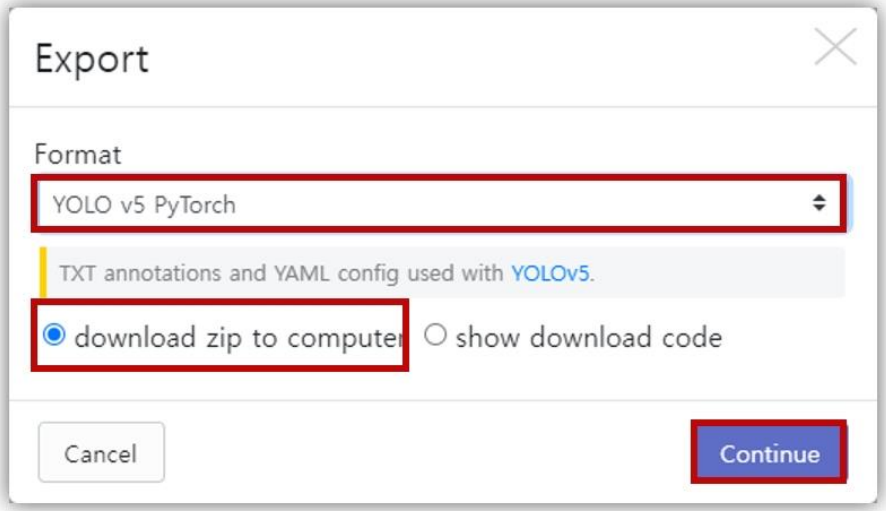
North American Mushrooms Dataset

❖ 416x416 사이즈의 이미지 51장을 다운



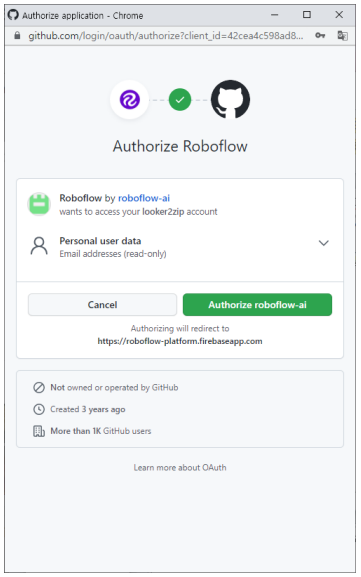
North American Mushrooms Dataset

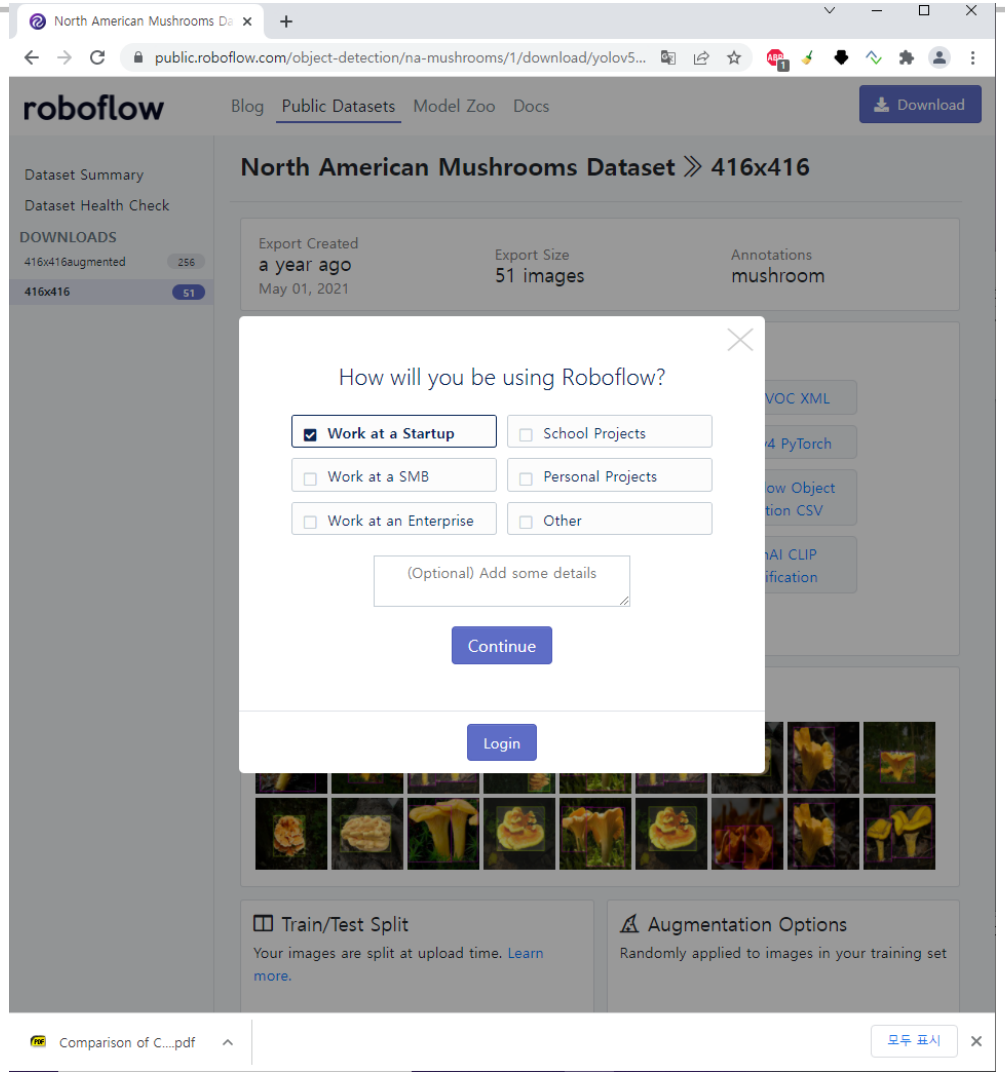
❖ object detection 알고리즘 라이브러리 구현방식에 따라, 그리고 YOLO 버전 별로도 사용하는 레이블링 파일의 포맷이 다르다. roboflow에서는 레이블링 파일 포맷을 선택하여 다운도르 할 수 있다. 우리는 PyTorch로 구현된 공식 계정의 코드를 사용할 예정이므로 YOLO v5 PyTorch를 선택하고 다운로드



North American Mushrooms Dataset

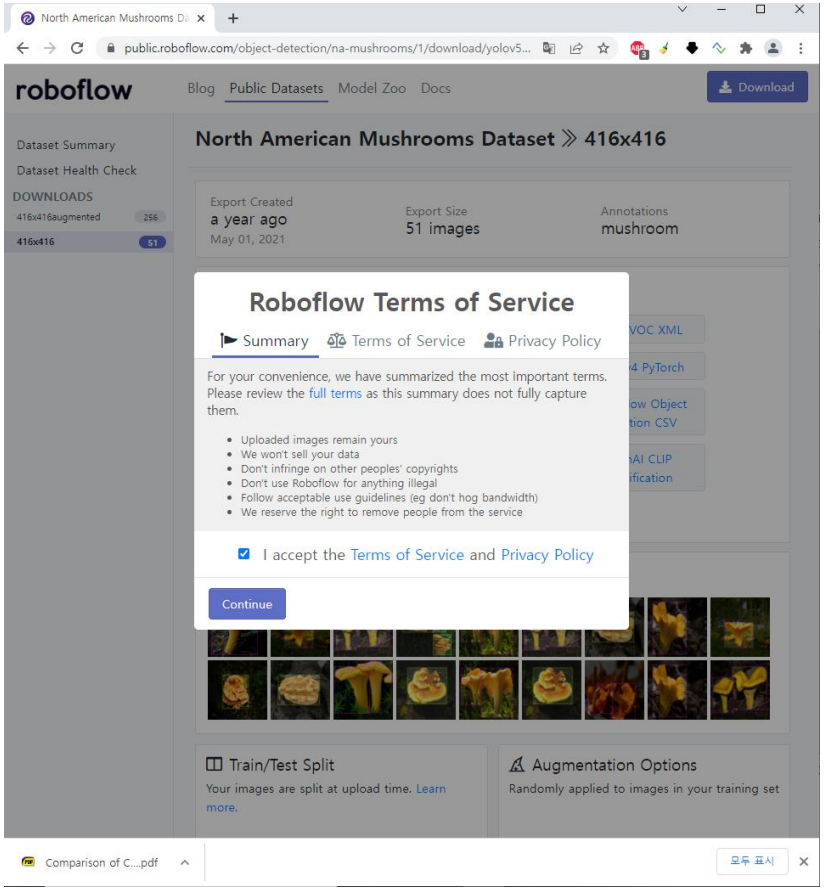
❖ 로그인 필요





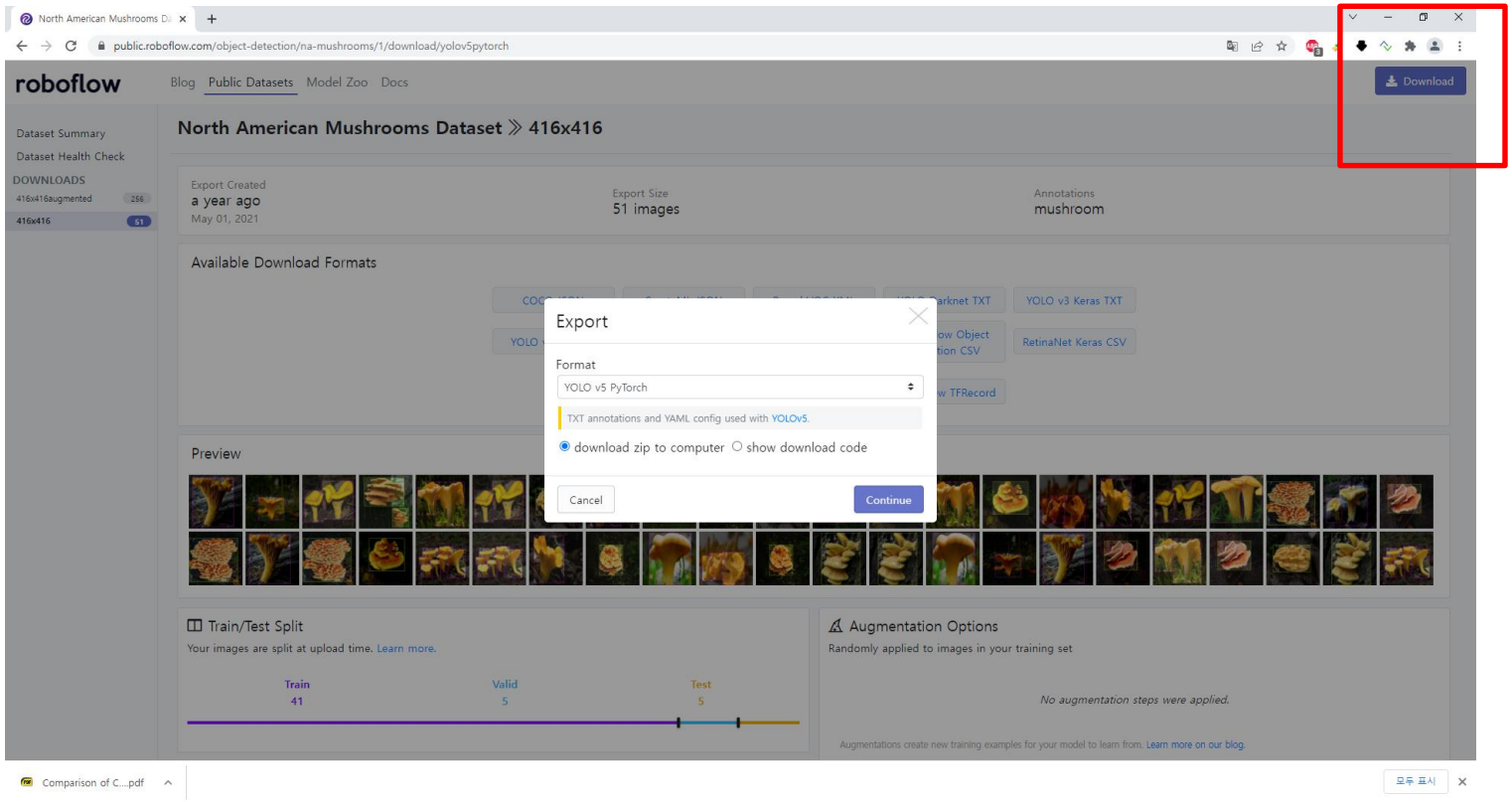
North American Mushrooms Dataset

❖ 동의



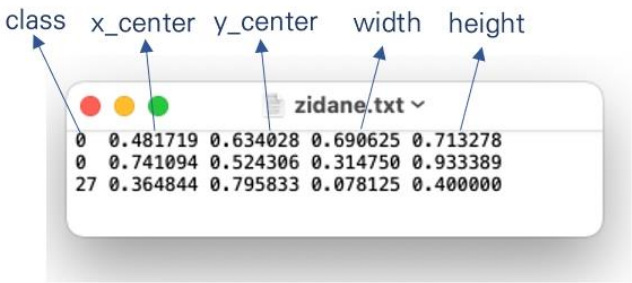
North American Mushrooms Dataset

❖ 다운로드



North American Mushrooms Dataset

❖ YOLOv5 공식계정의 코드는 txt 포맷의 레이블링 데이터를 사용한다. 이 파일은 이미지에서 검출된 object에 대한 클래스와 bounding box 정보를 포함하고 있다. 검출 객체정보 배치는 [class, x_center, y_center, width, height] 형태로 되어있다. bounding box 정보는 이미지 사이즈에 의해 정규화 되어있다. 따라서 0~1 범위의 값을 가진다.



North American Mushrooms Dataset

❖ 편의를 위해 다운로드한 데이터 셋 압축파일의 폴더 이름을 custom_dataset으로 수정한다. 데이터 폴더 구성은 다음과 같다.

```
custom_dataset
│
├── test/
│   ├── images/
│   └── labels/
│
├── train/
│   ├── images/
│   └── labels/
│
├── valid/
│   ├── images/
│   └── labels/
│
├── data.yaml
└── README.dataset.txt
```

데이터셋 소개

North American Mushrooms Dataset

❖ data.yaml파일을 메모장으로 열어보자. 데이터 셋 기본정보가 포함되어 있다. train과 val은 각 데이터 셋의 경로정보이다. 그리고 nc는 class의 수(number of classes)를, names는 각 클래스의 이름이다.

```
train: ../train/images
val: ../valid/images

nc: 2
names: ['CoW', 'chanterelle']
```

colab에서 환경구축하기

환경 구축

❖ google colab에 접속하고 새 노트를 생성한다. 런타임-런타임 유형 변경을 선택하여, 하드웨어 가속기를 GPU로 설정한다.

colab에서 환경구축하기

필수 라이브러리 설치

❖ colab 노트에 공식 github 계정의 파일을 다운로드하고, 필수 라이브러리를 설치하는 명령을 입력

```
!git clone https://github.com/ultralytics/yolov5  # yolov5 코드 clone
%cd yolov5
                                  # clone한 폴더로 진입
%pip install -qr requirements.txt                # 필수 라이브러리 설치
```

Cloning into 'yolov5'... remote: Enumerating objects: 10354, done. remote: Total 10354 (delta 0), reused 0 (delta 0), pack-reused 10354 Receiving objects: 100% (10354/10354), 10.58 MiB | 23.75 MiB/s, done. Resolving deltas: 100% (7149/7149), done. /content/yolov5

|[redacted]| 596 kB 5.4 MB/s

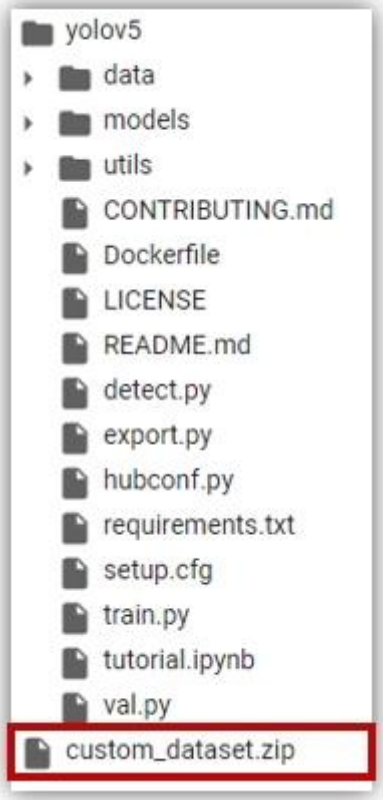
필수 라이브러리 설치

❖ 파일 탐색기에 yolov5 폴더가 생성되었고, 파일들이 다운로드

- yolov5
  - data
  - models
  - utils
  - CONTRIBUTING.md
  - Dockerfile
  - LICENSE
  - README.md
  - detect.py
  - export.py
  - hubconf.py
  - requirements.txt
  - setup.cfg
  - train.py
  - tutorial.ipynb
  - val.py

다운로드 데이터셋 업로드

❖ 다운로드한 데이터셋을 업로드 한다. 파일 탐색기의 업로드 아이콘을 클릭하여 custom_dataset.zip 파일을 업로드 한다. 업로드가 완료되면 탐색기에 해당 파일이 표시



다운로드 데이터셋 업로드

❖ unzip 명령으로 데이터 셋 파일의 압축을 해제

```
!unzip ../custom_dataset.zip
```

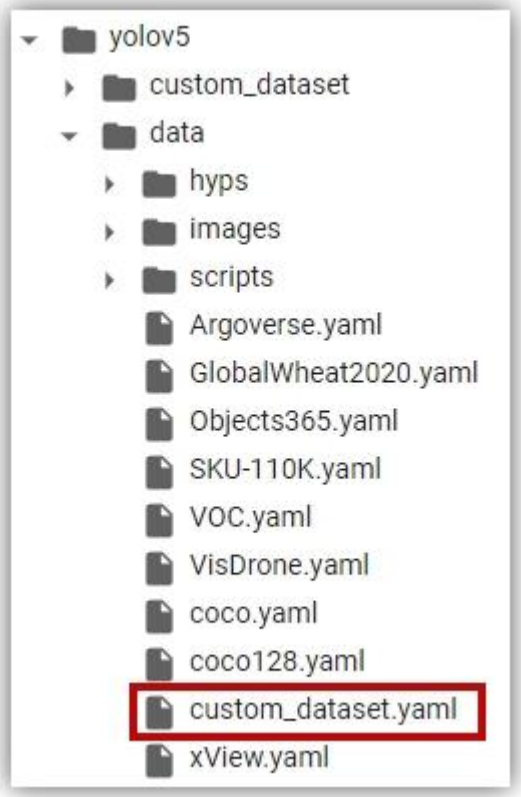
Archive: ../custom_dataset.zip inflating: custom_dataset/data.yaml inflating: custom_dataset/README.dataset.txt inflating: custom_dataset/README.roboflow.txt creating: custom_dataset/test/ creating: custom_dataset/test/images/ inflating: custom_dataset/test/images/chanterelle_02_jpg.rf.f7a48494b7393c532f641585d99a57be.jpg inflating: custom_dataset/test/images/chanterelle_03_jpg.rf.580f8d787af6a8050c21c065bf016f20.jpg inflating: custom_dataset/test/images/chanterelle_03_jpg.rf.cd892d2f06d228ba20d194fc360320fc.jpg --- (생 략)

다운로드 데이터셋 업로드

❖ 완료되면 yolov5/custom_dataset/ 경로에 데이터 셋이 위치하게 된다. (현 작업 디렉토리가 yolov5이기 때문)

다운로드 데이터셋 업로드

데이터 셋 설정파일을 작성한다.
` yolov5/data/ 폴더에 custom_dataset.yaml`이라는 이름의 파일을 생성



다운로드 데이터셋 업로드

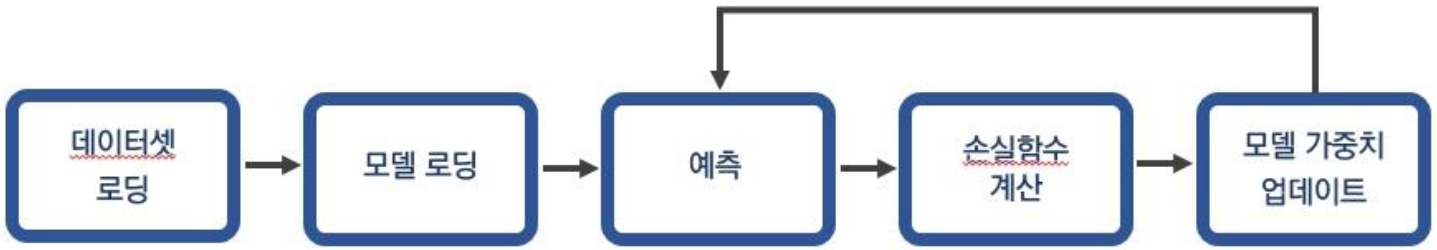
다음과 같이 설정정보를 입력

path: /content/yolov5/custom_dataset #root 디렉토리
train: train/images # 학습데이터 경로
val: valid/images
test: test/images

nc: 2 # 클래스 수
names: ['CoW', 'chanterelle'] # 클래스 이름

세한 내용은 다음 링크의 **1.1 Create dataset.yaml** 항목을 참고
https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

모델 학습 순서



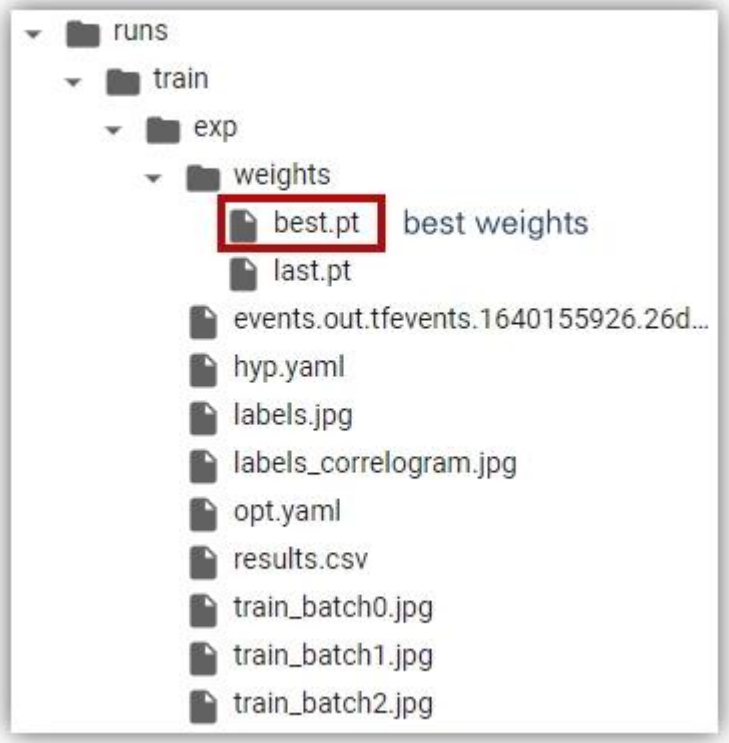
위와 같은 일련의 과정은 train.py 파일 실행을 통해 가능하다. 인자로 학습 데이터 경로와 epoch 수를 입력하고 학습을 진행하자.

모델 학습 순서

```
!python train.py --data "data/custom_dataset.yaml" --epochs 100 #epoch 100회
 Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
 train: weights=yolov5s.pt, cfg=, data=data/custom_dataset.yaml, hyp=data/hyps/hyp.scratch.yaml, epochs=100, batch_size=16,
imgsz=640, '''

 ---(생략)

 Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]

 ---(생략)

 Logging results to runs/train/exp
 Starting training for 100 epochs...

       Epoch    gpu_mem       box       obj       cls    labels  img_size
        0/99      3.23G    0.1252   0.03226   0.02699        28       640: 100% 3/3 [00:05<00:00,  1.95s/it]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,  2.48it/s]
                   all          5         14    0.00695      0.311    0.00395     0.0011

 ---(생략)

 Validating runs/train/exp/weights/best.pt...
 Fusing layers...
 Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,  4.48it/s]
                   all          5         14       0.95      0.996      0.973      0.697
                   CoW          5          5          1      0.991      0.995      0.688
           chanterelle          5          9      0.899          1      0.951      0.706
 Results saved to runs/train/exp
```

모델 학습 순서

❖ 학습이 완료되면 runs/train/exp경로에 학습 결과가 저장된다. 학습을 반복하면 runs/train경로에 exp1, 2, 3... 같은 형태로 폴더가 생성되면서 학습 결과가 기록된다.



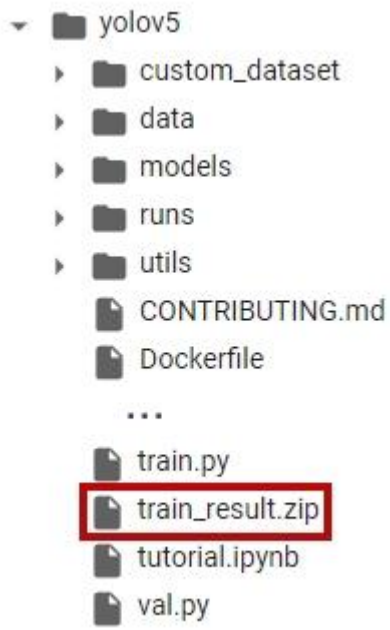
모델 학습 순서

❖ 학습 결과를 다운로드 하고 싶다면 zip 명령을 압축한 뒤, 저장한다 . 예를 들어 train_result.zip이라는 이름으로 압축하고 싶다면 다음과 같이 입력한다.

```
!zip -r train_result.zip /content/yolov5/runs/train/exp
```

모델 학습 순서

❖ 탐색기에 train_result.zip가 표시되면 정상으로 압축



모델 검증

❖ 검증순서는 앞의 학습 절차에서 모델 가중치 업데이트 과정이 생략된 것이다.

모델 검증

❖ 모델 검증은 val.py 파일 실행을 통해 진행한다. 다양한 인자가 있지만 데이터 경로(--data), 모델 가중치(--weights) 정도만 입력해서 실행해보자. 앞에서 학습한 모델 가중치는 runs/train/exp/weights/best.pt에 저장

```
!python val.py --data "data/custom_dataset.yaml" --weights
"/content/yolov5/runs/train/exp/weights/best.pt"
val: data=data/custom_dataset.yaml, weights=['/content/yolov5/runs/train/exp/weights/best.pt'],
batch_size=32, imgsz=640, conf_thres=0.001, iou_thres=0.6, task=val, device=, workers=8,
single_cls=False, augment=False, verbose=False, save_txt=False, save_hybrid=False, save_conf=False,
save_json=False, project=runs/val, name=exp, exist_ok=False, half=False, dnn=False
YOLOv5 🚀 v6.0-155-gdc54ed5 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)

Fusing layers...
Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning '/content/yolov5/custom_dataset/valid/labels.cache' images and labels... 5 found, 0 missing,
0 empty, 0 corrupted: 100% 5/5 [00:00<?, ?it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,
2.58it/s]
                   all          5         14      0.909      0.982      0.961      0.686
                   CoW          5          5          1      0.965      0.995      0.672
           chanterelle          5          9      0.818          1      0.926        0.7
Speed: 0.6ms pre-process, 29.3ms inference, 3.0ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp
```

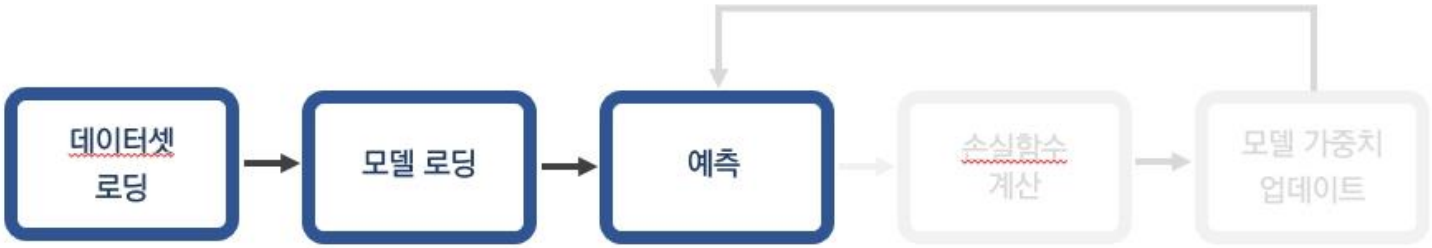
모델 검증

❖ 검증결과는 runs/val/exp에 저장된다. 앞에서와 마찬가지로 다운로드 받고 싶다면 폴더를 압축하자.

```
!zip -r val_result.zip /content/yolov5/runs/val
```

❖ exp 폴더 안에는 confusion matrix, F1 curve 등 성능과 관련된 차트가 저장되어 있다.

모델 예측 절차



❖ 예측 과정은 detect.py 파일을 사용한다. 단순 이미지 뿐만 아니라 웹캠, 비디오 파일 등에서도 실행 가능하다. --source인자에 다음과 같이 설정 해주면 된다.

모델 예측

예측 과정은 detect.py 파일을 사용한다. 단순 이미지 뿐만 아니라 웹캠, 비디오 파일 등에서도 실행 가능하다. --source인자에 다음과 같이 설정해주면 된다.

```
!python detect.py --source 0  # webcam
                  img.jpg  # image
                  vid.mp4  # video
                  path/  # directory
                  path/*.jpg  # glob
                  'https://youtu.be/Zgi9g1ksQHc'  # YouTube
                  'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP stream
```

모델 예측

여기서는 custom_dataset/test/images 경로에 있는 이미지에 대해서 object detection을 실행해본다. 인식 대상 (--source), 모델 가중치(--weights) 경로를 입력해서 실행해보자.

```
!python detect.py --weights "/content/yolov5/runs/train/exp/weights/best.pt" --source
"/content/yolov5/custom_dataset/test/images"
detect: weights=['/content/yolov5/runs/train/exp/weights/best.pt'],
source=/content/yolov5/custom_dataset/test/images, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45,
max_det=1000, device=, view_img=False, save_txt=False, save_conf=False,

---(생략)

Fusing layers...
Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
image 1/5
/content/yolov5/custom_dataset/test/images/chanterelle_02_jpg.rf.f7a48494b7393c532f641585d99a57be.jpg:
640x640 3 chanterelles, Done. (0.034s)
--- (생략)
```

모델 예측

테스트 결과는 /runs/detect/exp 경로에 저장된다. 결과를 다운로드 하고 싶다면 다음과 같이 압축하여 저장한다.

```
!zip -r test_result.zip /content/yolov5/runs/detect/exp
```

폴더를 열어보면 class와 bounding box가 표시된 detection 결과 이미지가 저장되어 있다.

Unit 4

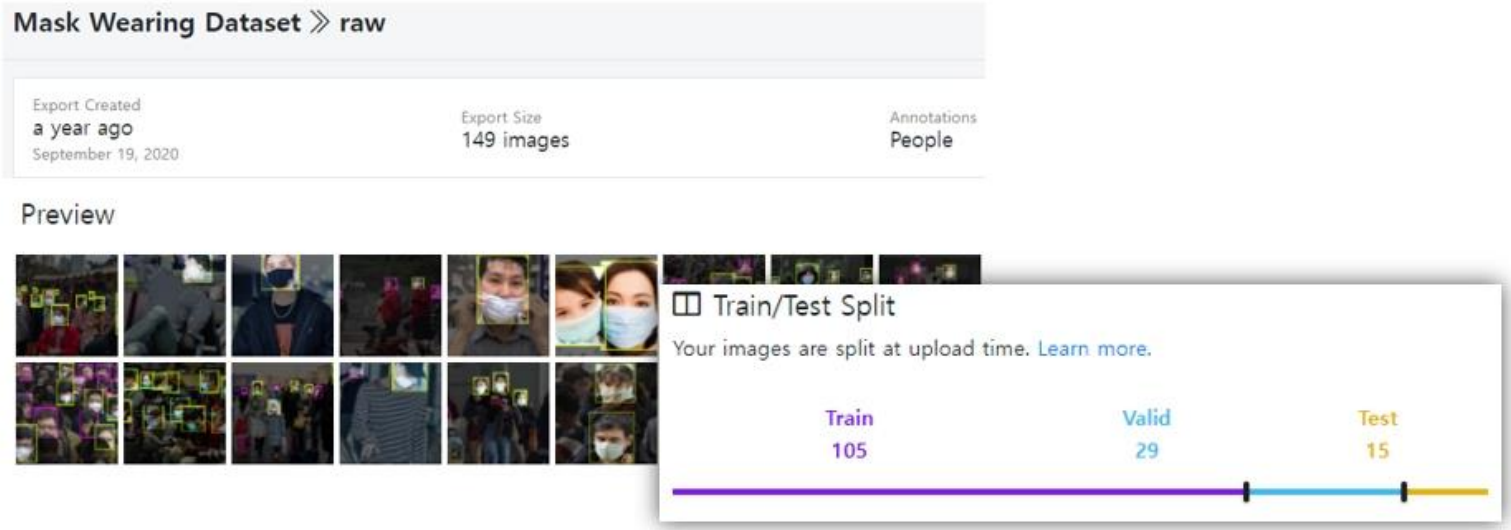
학습 파라미터 설정하기

학습 파라미터 설정

❖ 학습과 관련된 파라미터를 조정하는 방법에 대해서 설명한다. 앞에서와 마찬가지로 실습환경은 google colab이다.

Mask Wearking Dataset

실습에 사용되는 데이터셋은 roboflow에서 제공되는 Mask Wearking Dataset(raw)이다.
https://public.roboflow.com/object-detection/mask-wearing/4
(raw와 416x416으로 변환된 데이터셋을 선택할 수 있는데, 여기서는 raw 데이터셋을 사용한다.)



colab에서 환경구축하기

- google colab에 접속하고 새 노트를 생성
- 런타임-런타임 유형 변경을 선택후, 가속기를 GPU로 설정
- yolov5 파일을 다운로드 및 필수 라이브러리를 설치

```
!git clone https://github.com/ultralytics/yolov5  # yolov5 코드 clone
%cd yolov5
                                  # clone한 폴더로 진입
%pip install -qr requirements.txt                # 필수 라이브러리 설치
```

- custom dataset 업로드 (여기서는 mask_dataset.zip 으로 설명)

- 데이터 셋 파일 압축 해제

```
!unzip ../custom_dataset.zip
```

- ` yolov5/data/ 폴더에 mask_dataset.yaml`파일 작성

```
path: /content/yolov5/mask_dataset
train: train/images
val: valid/images
test: test/images

nc: 2
names: ['mask', 'no-mask']
```

학습 파라미터 살펴보기

파라미터 조정

- 모델을 학습할 때 다음과 같이 데이터셋 관련 경로만 입력하면 가능하다.

```
!python train.py --data "데이터셋.yaml 파일 경로"
```

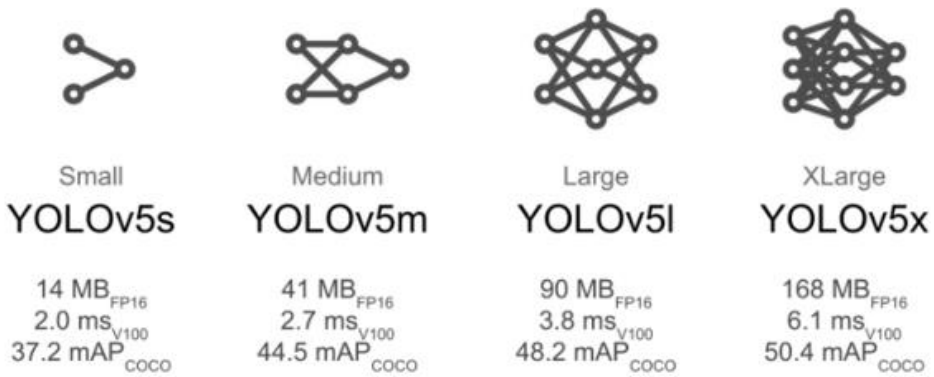
- 나머지 파라미터들은 디펄트 값으로 대체된다. 그러면 학습과 관련된 파라미터에는 어떤 것들이 있을까? yolov5 폴더 안에 있는 train.py 파일을 열어서 440번째 라인 부근에 있는 parse_opt 함수를 살펴보자. 아래와 같이 파라미터들이 정의되어 있다.

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=300)
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')

    --- (생 략)
```

학습 파라미터 살펴보기

모델구조 (–weights)

- 모델 구조와 관련된 파라미터이다. YOLOv5는 다양한 모델 구조를 제공한다. default 값은 YOLOv5로 구조가 제일 간단하다. 모델의 구조가 더 복잡한 것으로 YOLOv5m, YOLOv5l, YOLOv5x 이 있다.



모델구조 (–weights)

- 구조가 복잡할 수록 성능이 높아질 가능성은 높지만 학습할 때 더 많은 시간이 소요되고 많은 리소스가 요구된다. 예를 들어 yolov5m 모델을 학습시키고 싶다면 다음과 같이 입력하면 된다.

--weights "yolov5m.pt"

- 공식 github에는 이외에 새로운 모델 구조가 지속적으로 업로드되고 있다.

- https://github.com/ultralytics/yolov5/releases

배치 사이즈 (–batch-size)

- 학습할 때 한번에 처리할 이미지 수(batch-size)를 지정할 수 있다. default는 16이다. batch size를 32로 입력하고 싶다면 다음과 같이 옵션 설정을 하면된다.

--batch-size 32

학습 파라미터 살펴보기

이미지 크기 (–imgsz, –img, –img-size)

- YOLOv5는 학습할 때 모든 이미지 사이즈를 동일하게 resizing 한다. default 사이즈는 640x640이다. 이미지 사이즈를 크게 설정할수록 모델 성능은 더 좋아실 수 있다. 하지만 학습속도와 리소스 부담은 더 커지게 된다. 이미지 크기를 1280x1280으로 설정하고 싶다면 다음과 같이 입력한다.

--imgsz(or --img or --img-size) 1280

- 검증이나 시험할 때 학습에 사용한 이미지 사이즈와 동일하게 설정해야 한다.

에포크 수 (–epochs)

- 데이터셋으로 학습을 반복할 횟수를 지정하는 에포크의 default 값은 300이다. 100으로 설정하고 싶다면 다음과 같이 입력한다.

--epochs 100

하이퍼 파라미터 (–hyp)

- 하이퍼 파라미터가 정의되어 있는 경로를 지정한다. default 값은 data/hyps/hyp.scratch.yaml이다.

파라미터를 조정하여 모델 학습하기

- colab에서 제공하는 자원을 최대한 사용하여 학습을 진행해보자. 모델 구조는 yolov5m.pt, 입력 이미지 크기는 1280, 배치 사이즈는 8, 에포크 수는 60으로 설정해보자. (모델 구조가 커지고 입력 이미지가 복잡해져서 colab gpu 한계를 맞추기 위해 배치 사이즈와 학습시간을 줄여야만 했다.)

```
!python train.py --data "data/mask_dataset.yaml" --batch-size 8 --img 1280 --weights "yolov5m.pt" --epochs 60
train: weights=yolov5m.pt, cfg=, data=data/mask_dataset.yaml, hyp=data/hyps/hyp.scratch.yaml, epochs=60, batch_size=8, imgsz=1280, rect=False,

--(생 략)

hyperparameters: lr0=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5,
--(생 략)
                 from  n    params  module                                  arguments
  0                -1  1      5280  models.common.Conv                      [3, 48, 6, 2, 2]
  1                -1  1     41664  models.common.Conv                      [48, 96, 3, 2]
  2                -1  2     65280  models.common.C3                        [96, 96, 2]

--(생 략)

Model Summary: 290 layers, 20856975 parameters, 0 gradients, 48.0 GFLOPs
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:06<00:00,  3.12s/it]
                   all         29        162       0.81      0.853      0.861      0.531
                  mask         29        142      0.871      0.805      0.888      0.559
               no-mask         29         20       0.75        0.9      0.835      0.503
Results saved to runs/train/exp3
```

- 여기서는 학습 결과가 /runs/train/exp3에 저장되었다. 사용자 마다 저장 위치가 다를 것이다.

검증하기

검증

- 검증과 관련된 파라미터는 val.py 파일의 306번째 라인의 parse_opt 함수에 정의되어 있다.

- 학습할 때 이미지 사이즈는 1280으로 설정하였고, 모델 가중치는 /runs/train/exp3/weights/best.pt 저장되어 있으므로 다음과 같이 입력하여 검증을 진행하자.

```
!python val.py --data "data/mask_dataset.yaml" --img 1280 --weights "/content/yolov5/runs/train/exp3/weights/best.pt"

val: data=data/mask_dataset.yaml, weights=['/content/yolov5/runs/train/exp3/weights/best.pt'], batch_size=32, imgsz=1280,
conf_thres=0.001, iou_thres=0.6, task=val, device=, workers=8, single_cls=False, augment=False, verbose=False, save_txt=False,
save_hybrid=False, save_conf=False, save_json=False, project=runs/val, name=exp, exist_ok=False, half=False, dnn=False
YOLOv5 🚀 v6.0-159-gdb6ec66 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)

Fusing layers...
Model Summary: 290 layers, 20856975 parameters, 0 gradients, 48.0 GFLOPs
val: Scanning '/content/yolov5/mask_dataset/valid/labels.cache' images and labels... 29 found, 0 missing, 0 empty, 0 corrupted: 100%
29/29 [00:00<?, ?it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:08<00:00,  8.05s/it]
                   all         29        162      0.813      0.851      0.862      0.535
                  mask         29        142      0.878       0.81      0.887      0.556
               no-mask         29         20      0.748      0.892      0.837      0.514
Speed: 1.5ms pre-process, 166.8ms inference, 4.4ms NMS per image at shape (32, 3, 1280, 1280)
Results saved to runs/val/exp
```

- 예측결과가 runs/val/exp에 저장되었다.

예측

- 예측할 때는 기본적으로 모델의 경로(–weights), 입력 데이터 경로(–source)를 지정해줘야 한다. 여기에 추가로 이미지 사이즈(–img)도 지정해주자.

```
!python detect.py --img 1280 --weights "/content/yolov5/runs/train/exp3/weights/best.pt"  --source "/content/yolov5/mask_dataset/test/images"
detect: weights=['/content/yolov5/runs/train/exp3/weights/best.pt'], source=/content/yolov5/mask_dataset/test/images, imgsz=[1280, 1280], conf_thres=0.25,

---(생 략)

Results saved to runs/detect/exp
```

- runs/detect/exp에 예측 결과가 저장된다.

Unit A

참고자료

참고자료

- http://www.ncs.go.kr
- NELLDALE/JOHN LEWIS지음, 조영석/김대경/박찬영/송창근 역, 단계별로 배우는컴퓨터과학, 홍릉과학출판사,2018
- 혼자 공부하는 머신러닝+딥러닝 박해선 지음 | 한빛미디어 | 2020년 12월
- 머신러닝 실무 프로젝트 ,아리가 미치아키, 나카야마 신타, 니시바야시 다카시 지음 | 심효섭 옮김 | 한빛미디어 | 2018년 06월
- 파이썬을 활용한 머신러닝 쿡북 크리스 알본 지음 | 박해선 옮김 | 한빛미디어 | 2019년 09월
- 처음 배우는 머신러닝 김의중 지음 | 위키북스 | 2016년 07월
- 파이썬으로 배우는 머신러닝의 교과서 : 이토 마코토 지음 | 박광수(아크몬드) 옮김 | 한빛미디어 | 2018년 11월
- 기타 서적 및 웹 사이트 자료 다수 참조

나눔고딕 인스톨

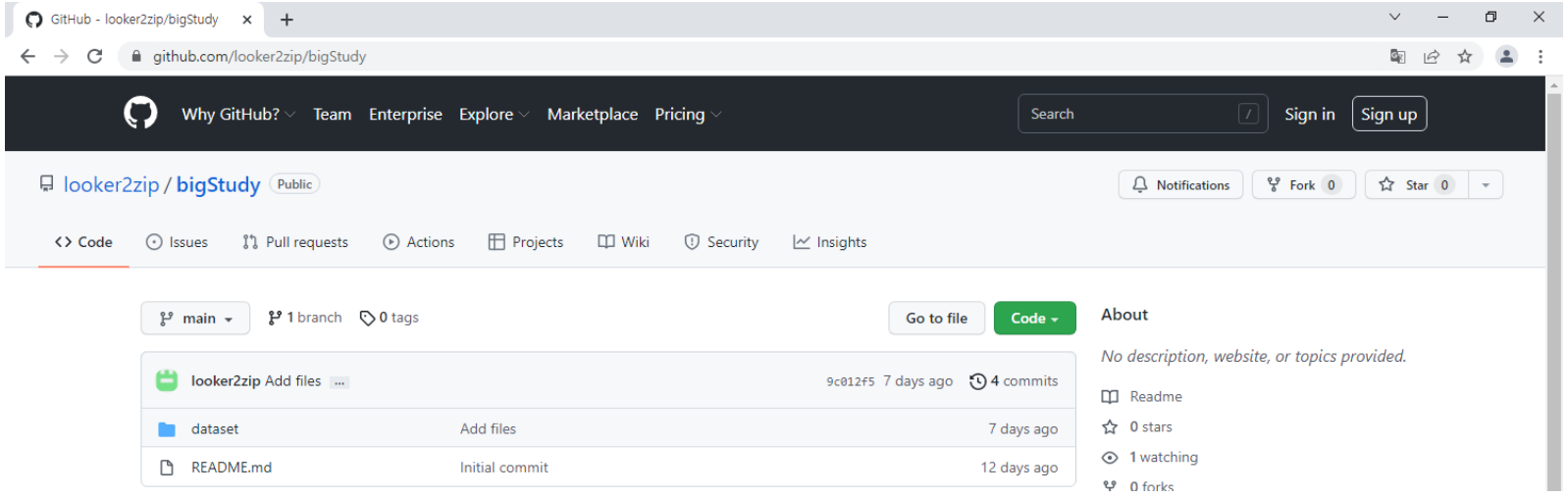
❖ 리눅스 다운로드

!rm -rf bigStudy

!git clone 'https://github.com/looker2zip/bigStudy.git'

❖ 윈도 다운로드

https://github.com/looker2zip/bigStudy



나눔고딕 인스톨

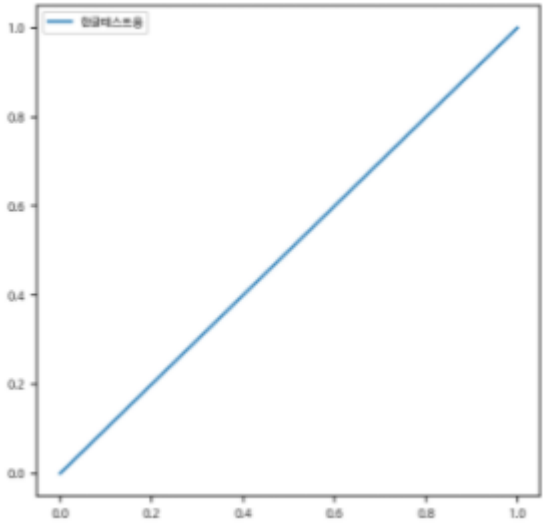
```
%config InlineBackend.figure_format = 'retina'
 !apt -qq -y install fonts-nanum
```

❖ 런타임 다시 시작

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()


plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```

나눔고딕 인스톨

```
plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```



감사합니다.

- Mobile: 010-9591-1401
- E-mail: onlooker2zip@naver.com