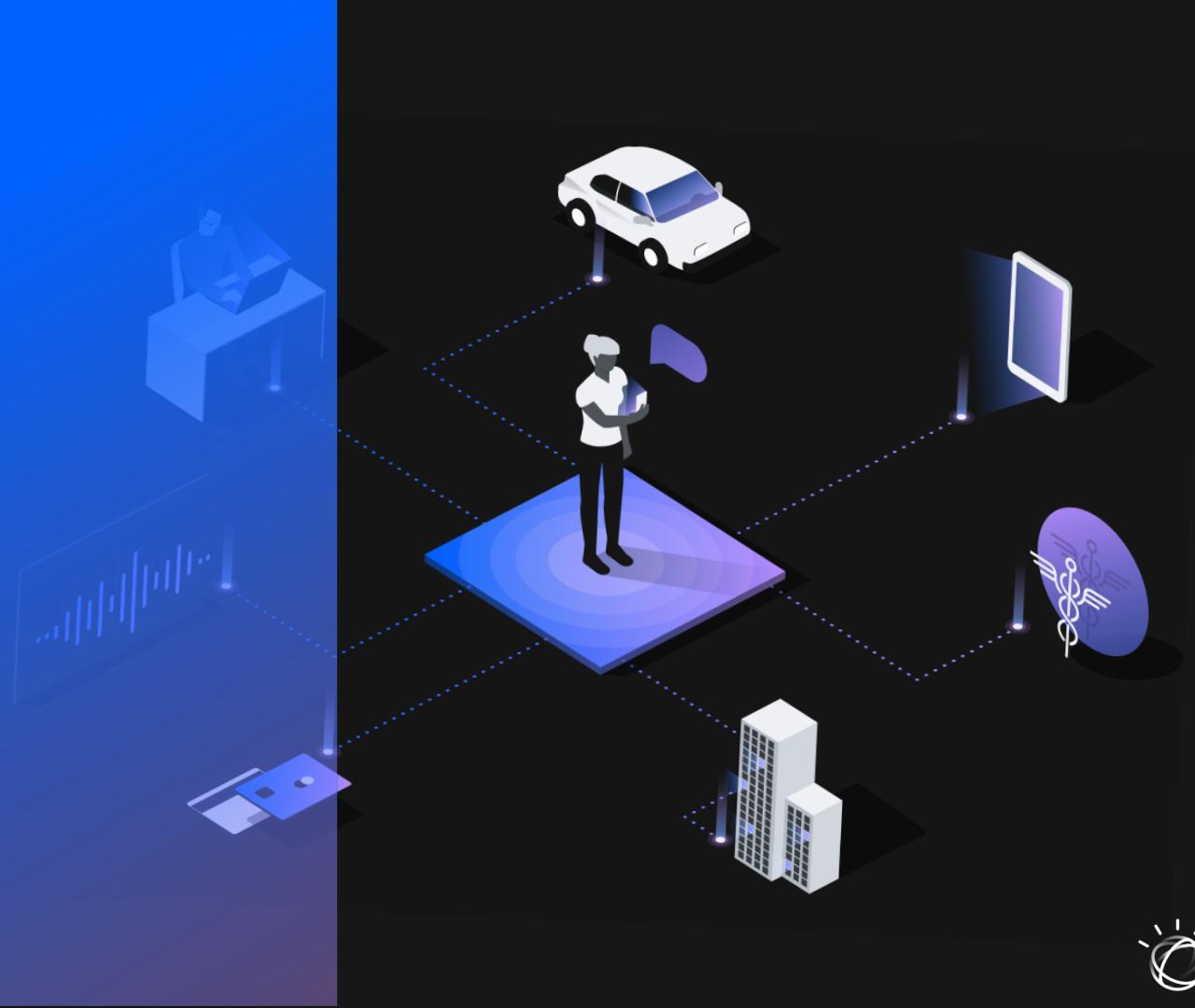


데이터 사이언스

Hands-on

—
Ho-Kyeong Ra, Ph.D.
나호경



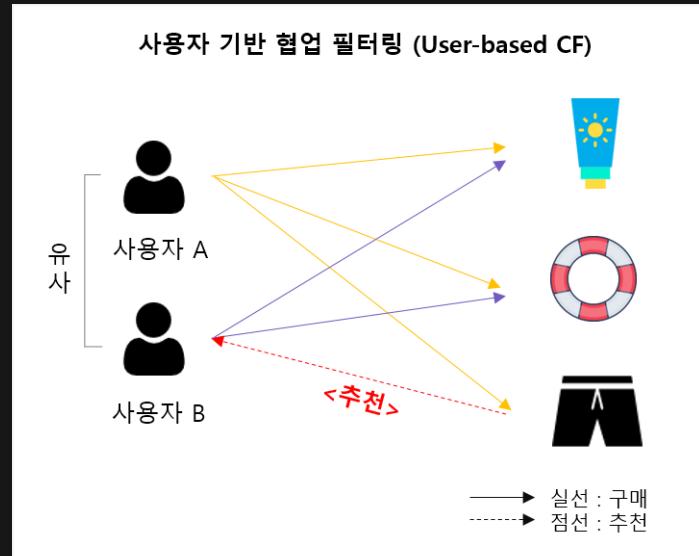
Scenario

시나리오



추천 시스템 – Recommendation System

추천 시스템은 사용자에게 개인 신상, 관심 분야, 선호도 등을 질의 결과를 수집하고, 이 정보를 기반으로 고객의 심리 정보와 선호도 정보에 알맞은 정보 및 상품을 추천하거나 제공하는 방법이다. (예제: 영화, 음악, 뉴스, 책, 연구 주제, 탐색 질의, 상품 등 검색에 이용)

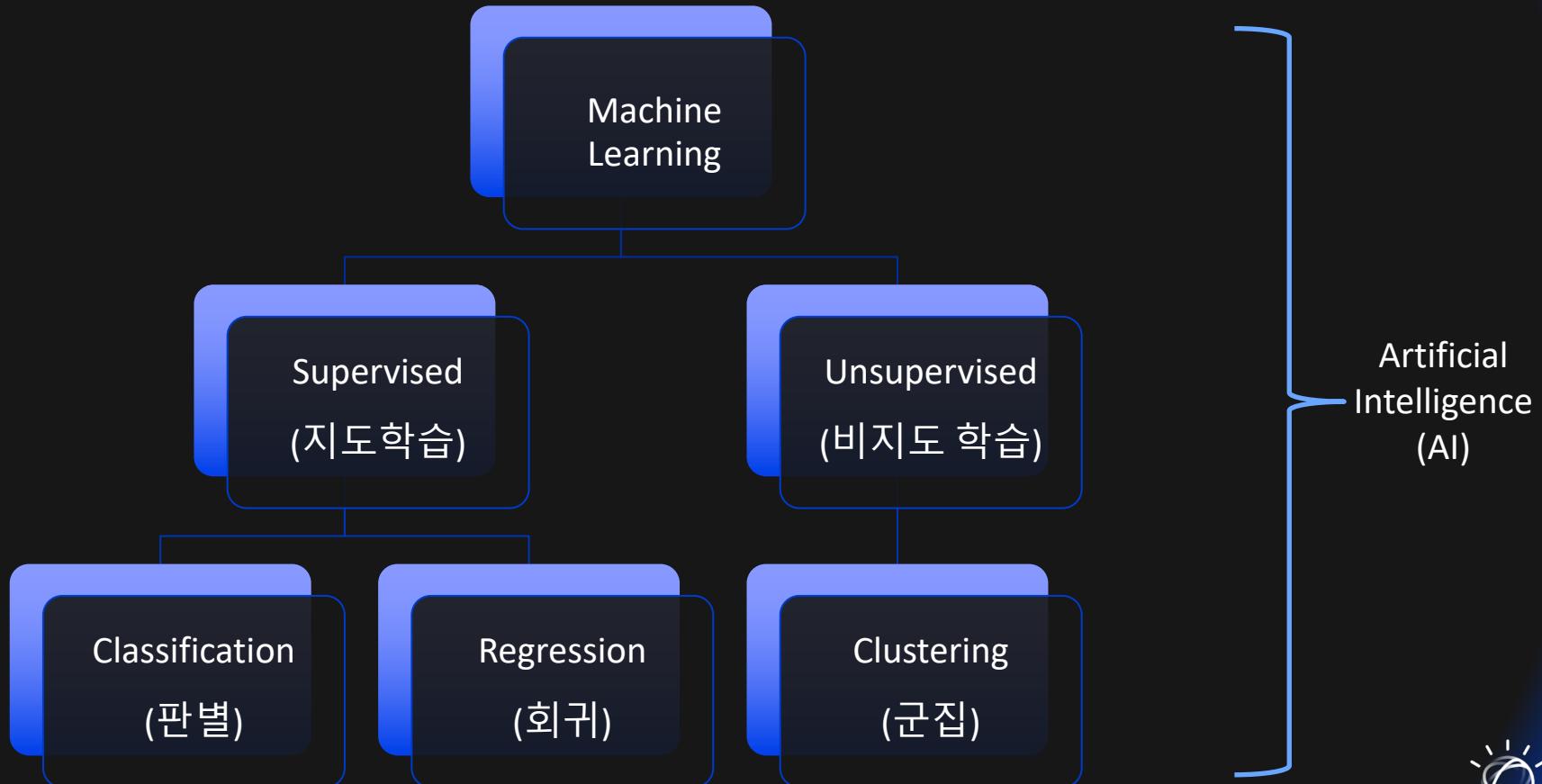


추천 시스템 – Recommendation System

학습/테스트 데이터는 “사용자 정보”, “구매한 상품 항목”이 표시가 되어 있습니다.



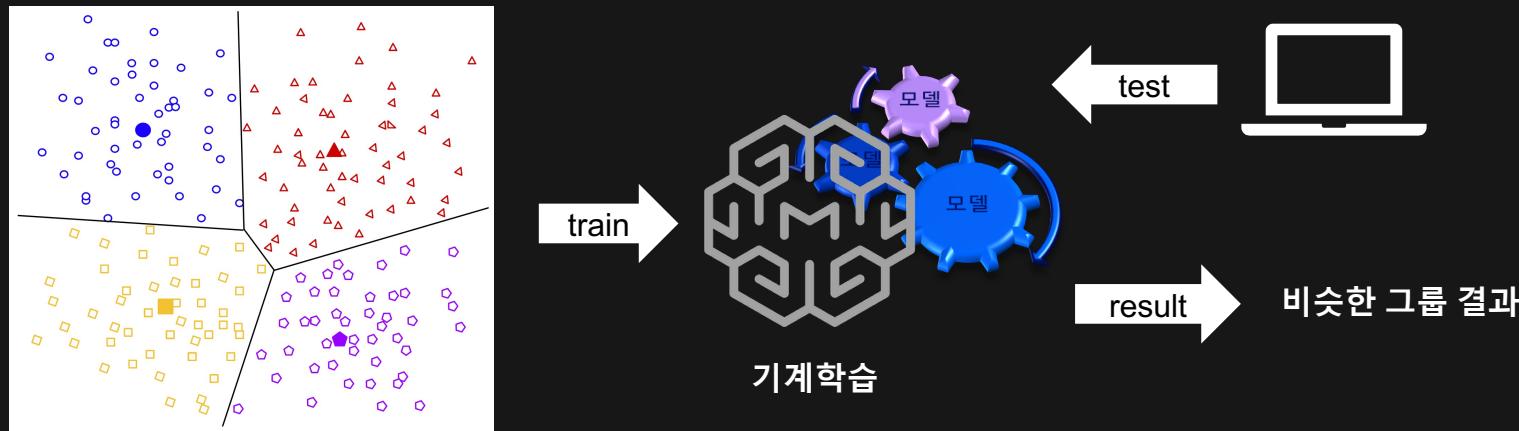
AI 및 기계 학습의 계층적 관점



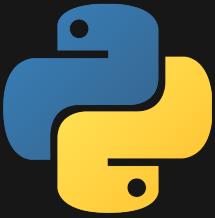
추천 시스템 – Recommendation System

목표:

1. 기계학습을 이용하여 비슷한 사용자끼리 그룹별로 클러스터링
2. 클러스터 기반으로 자동 레이블링 작업
3. 결과 데이터를 학습, 사용자 정보 및 구매 데이터 입력, 해당 그룹 판별
4. 판별된 그룹에서 가장 많이 구매된 상품 조회



개발 흐름 – Recommendation System



파이썬 모델링



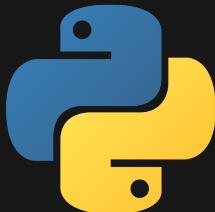
Flask/Django



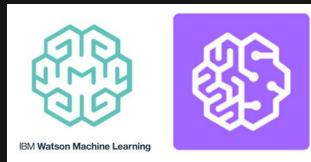
Container
서비스



서비스
호출



파이썬 모델링



모델 배포



서비스
호출

장기적으로 모델 배포와 관리를 자동화 하는 것과 불필요한 서버 리소스 비용에서 큰 차이

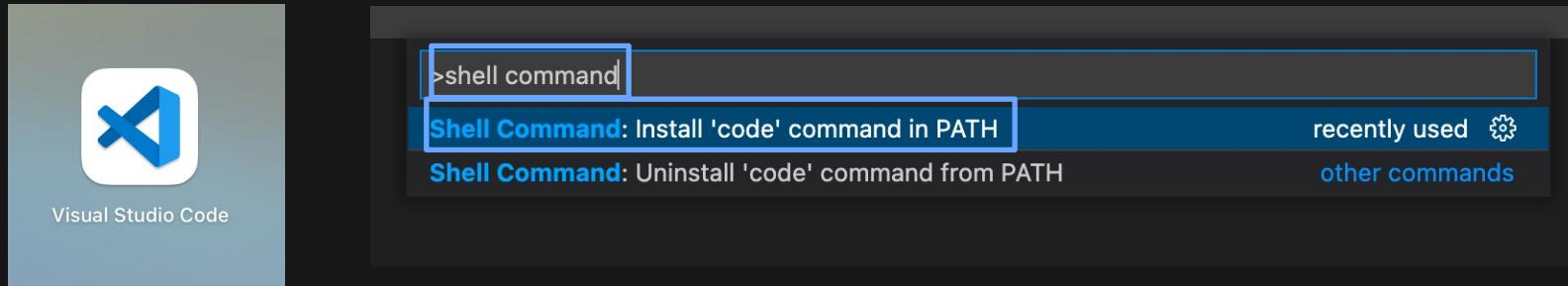


VSCode 설치



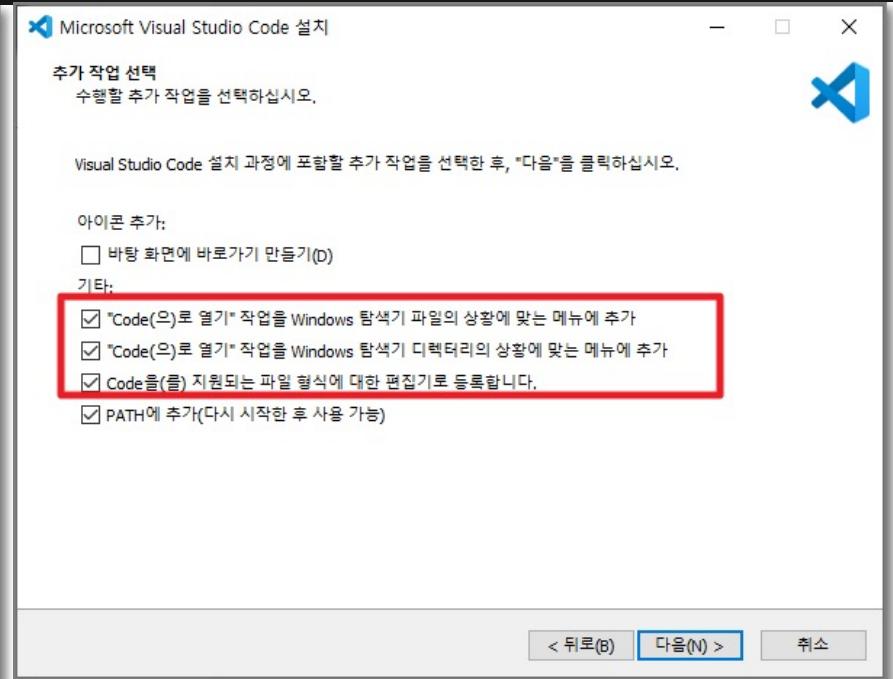
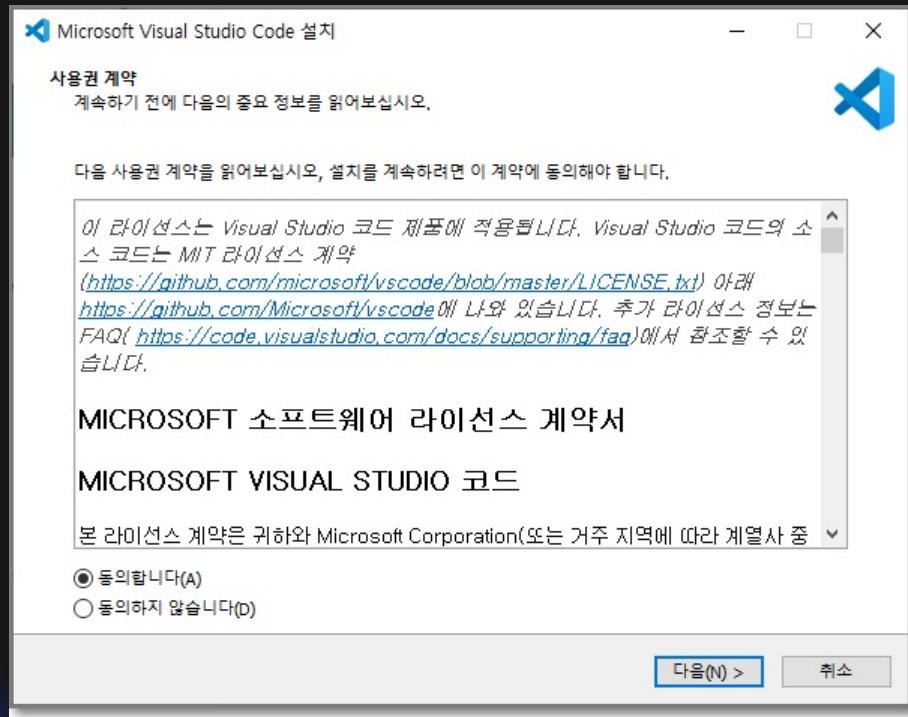
VSCODE 설치 - macOS

- 링크(<https://code.visualstudio.com/docs?dv=osx>)를 클릭하여 VSCODE 다운로드(macOS) 시작합니다.
- 다운로드 된 파일을 더블 클릭하여 압축 해제 합니다.
- “Visual Studio Code.app” 파일을 “응용 프로그램” 폴더로 드래그하여 이동합니다.
- VSCODE는 터미널에서 “code .”를 입력하여 해당 폴더에서 실행도 가능합니다.
- Launcher에서 Visual Studio Code를 실행하고, Cmd+Shift+P를 누릅니다.
- 상단 커맨드 팔레트에서 “shell command”를 입력하고, “Install ‘code’ ...”를 클릭하여 설치를 완료 합니다.



VSCODE 설치 - Windows

- 링크(<https://code.visualstudio.com/docs?dv=win>)를 클릭하여 VSCode 다운로드(Win) 시작합니다.
- 다운로드 된 파일을 더블 클릭하여 설치 시작합니다.
- 동의 절차 이후, 아래의 설정을 따라 설치를 마무리 합니다.



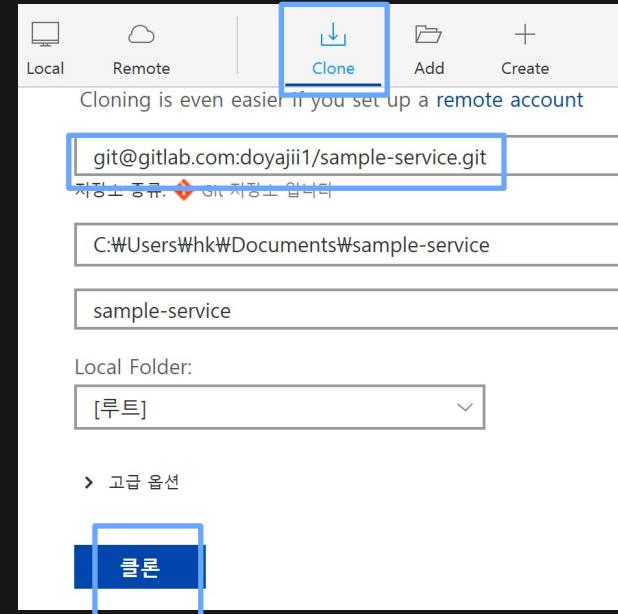
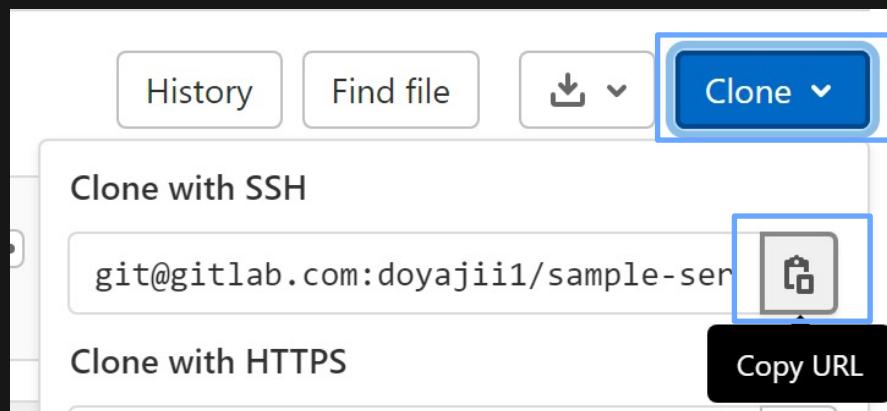
Data Download 학습자료



깃랩에서 데이터/소스 다운로드

<https://gitlab.com/msa2021/machine-learning> 클론 합니다.

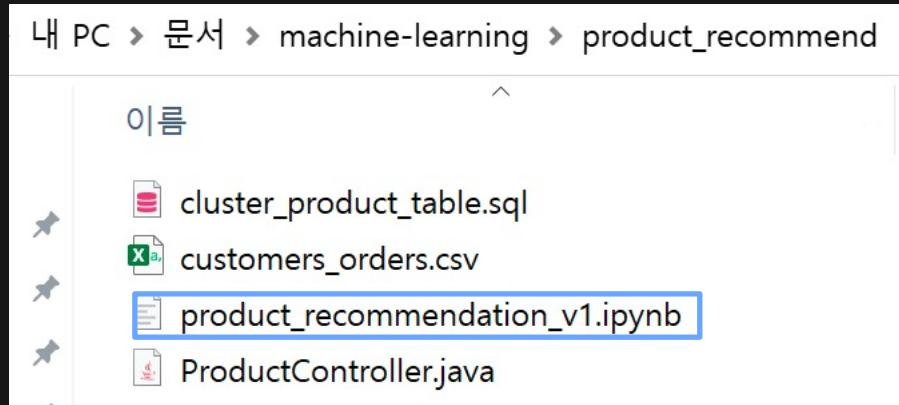
1. 깃랩 링크 > “Clone” 버튼 > Clone with SSH에 “Copy URL”
2. Source tree(소스트리) 또는 선호하는 깃 툴을 이용하여 로컬에 클론



깃랩에서 데이터/소스 다운로드

Clone 디렉토리/machine-learning/product-recommend/

product_recommendation_v1.ipynb파일을 미리 VSCode로 열어 놓습니다.



```
#읽어드린 데이터 객체 이름 변경 및 일부 표시
df = df_data_1
print(df.shape) #크기 (행 x 콜럼)
display(df) #데이터 보기
```

```
#데이터 읽고 크기 및 일부 표시(로컬 주피터 실행시 사용)
# import pandas as pd
# df = pd.read_csv ('r'customers_orders1_opt.csv')
# print(df.shape)
```

IBM

클라우드

계정



계정 생성

1. 링크(<https://cloud.ibm.com/>)를 클릭하여 계정 생성 페이지로 이동합니다.
 2. 이메일 주소 및 암호(영문 대문자, 소문자, 숫자, 특수기호 이용) 입력 및 이메일 인증
 3. 동의 절차 이후 Cloud 콘솔로 로그인
- *추후 재접속은 <https://cloud.ibm.com/>에서 하게 됩니다.

The screenshot shows the IBM Cloud interface. On the left, a modal window titled "Log in to IBM Cloud" is displayed, prompting the user to "Create an account". A blue box highlights the "Create an account" button. Below this, the "1. Account information" step of the sign-up process is shown, requiring an email address and password. A red box highlights the "Email" input field. The main dashboard on the right features a banner for "Build for free on IBM Cloud" and sections for "Create a Kubernetes cluster", "Get started with machine learning + Watson Studio", "Visit the IBM Cloud catalog", and "Create an OpenShift cluster". The bottom of the dashboard includes links for "View all", "Planned maintenance", "View all", and "For you". A "Watson Studio" section is also visible.

**Watson
Studio**

왓슨 스튜디오



Watson Studio 생성 - 1

1. Dashboard에서 “Create resource” 클릭하여 제품 검색 페이지로 이동합니다.
2. 검색 란에 “Watson Studio” 검색 및 선택
3. Lite 플랜, 지역 선택 이후 “Create” 눌러 생성 완료(현재 댈러스 Dallas 선택)
4. “Get Started” 클릭하여 Watson 스튜디오 실행 (2분정도 생성 시간 소요됩니다.)

IBM Cloud cat

watson studio

Watson Studio

IBM Cloud

Select a location

Dallas (us-south)

Select a pricing plan
Displayed prices do not include tax. Monthly prices shown are for country or location: [United States](#)

Plan	Pricing
Lite	Free

1 authorized user
50 capacity unit-hours monthly limit
Environment = # of capacity units required per hour

Summary

Watson Studio Free

Location: Tokyo
Plan: Lite
Service name: Watson Studio-uo
Resource group: Default

Create

Add to estimate

View terms

Welcome to Watson Studio. Let's get started!

Watson Studio

Get Started

Watson Studio 생성 - 2

1. Watson Studio 페이지 처음 열람시
이메일과 전화번호 입력란이 나옵니다.

2. Watson Machine Learning, Cloud
Object Storage 추가 선택 이후
“Continue”

3. 생성 완료 후, “Go to IBM Watson
Studio” 클릭

IBM Watson Studio

Watson Studio

Machine learning and AI made easy! Solve your business problems in a collaborative environment.

Watson Machine Learning

Make smarter decisions, solve tough problems, and improve user outcomes.

[Remove](#) –

Cloud Object Storage

Developer enabled data storage for cloud applications integrated with IBM Cloud services.

[Remove](#) –

위의 항목 오른쪽 리스트에서
선택하여 추가

Continue

Go to IBM Watson Studio

프로젝트 생성

1. 왓슨 스튜디오 Dashboard에서 “Create a project” 클릭하여 프로젝트 생성 페이지로 이동
2. 항목 중 “Create an empty project” 선택 및 이름 부여
3. “Create” 클릭 하여 생성 시작(약 2분 소요)

Watson Machine Learning

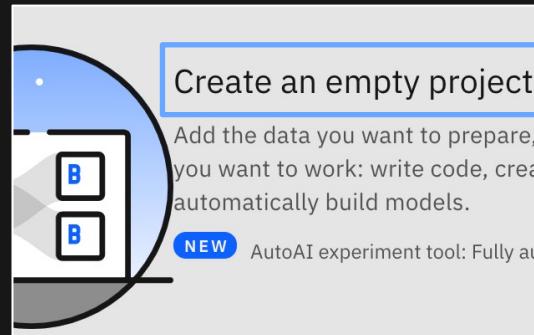
example
gh solving a
siness
a sample

Work with data
Create a project for
your team to prepare
data, find insights, or
build models.

Extend your
capabilities
Add tools,
or other features
creating several
instances.

led tutorial

Create a project



New project

testProject

Description

Project description

Choose project options

Restrict who can be a collaborator ⓘ

Project includes integration with [Cloud Object Storage](#) for storing project assets.

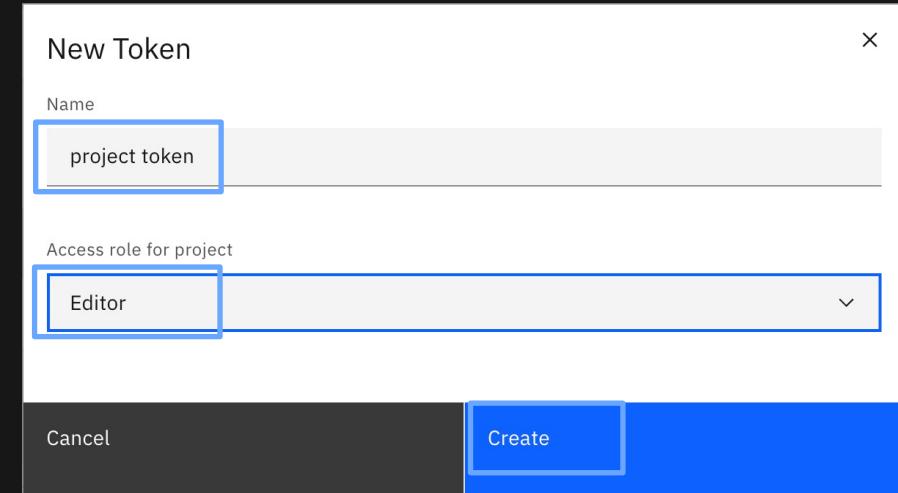
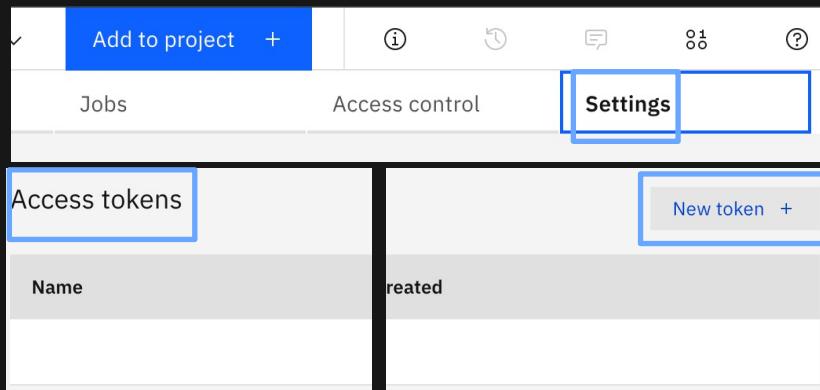
Storage

CloudObjectStorage

Create

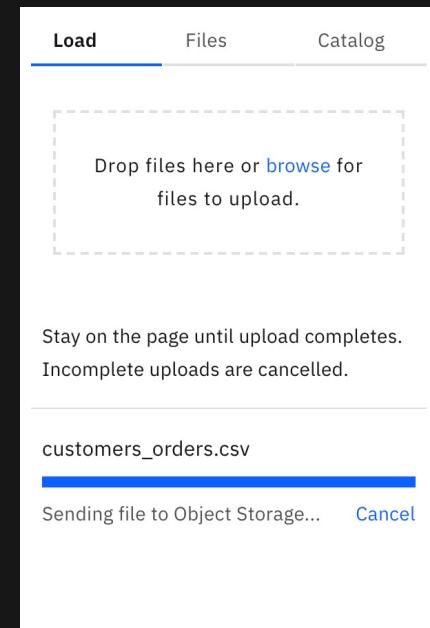
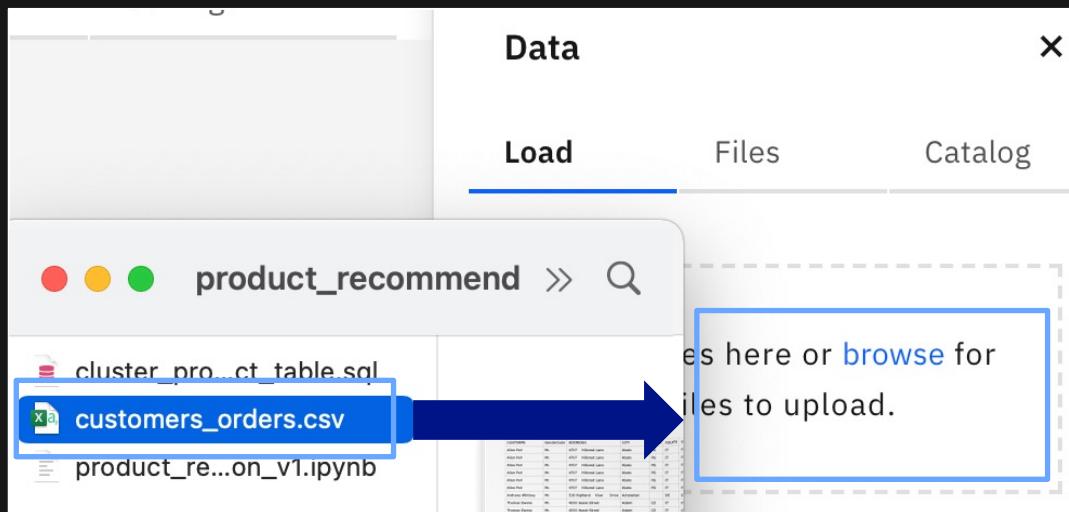
토큰 생성

1. 토큰은 추후 정제된 데이터를 프로젝트에 저장하는데 이용됩니다.
2. 프로젝트의 “Settings” 탭으로 이동
3. 페이지에서 “Access tokens” 부분에서 “New token” 클릭
4. 토큰 이름 입력 및 토큰 룰을 Editor로 설정 이후 “Create” 클릭



학습 데이터 추가

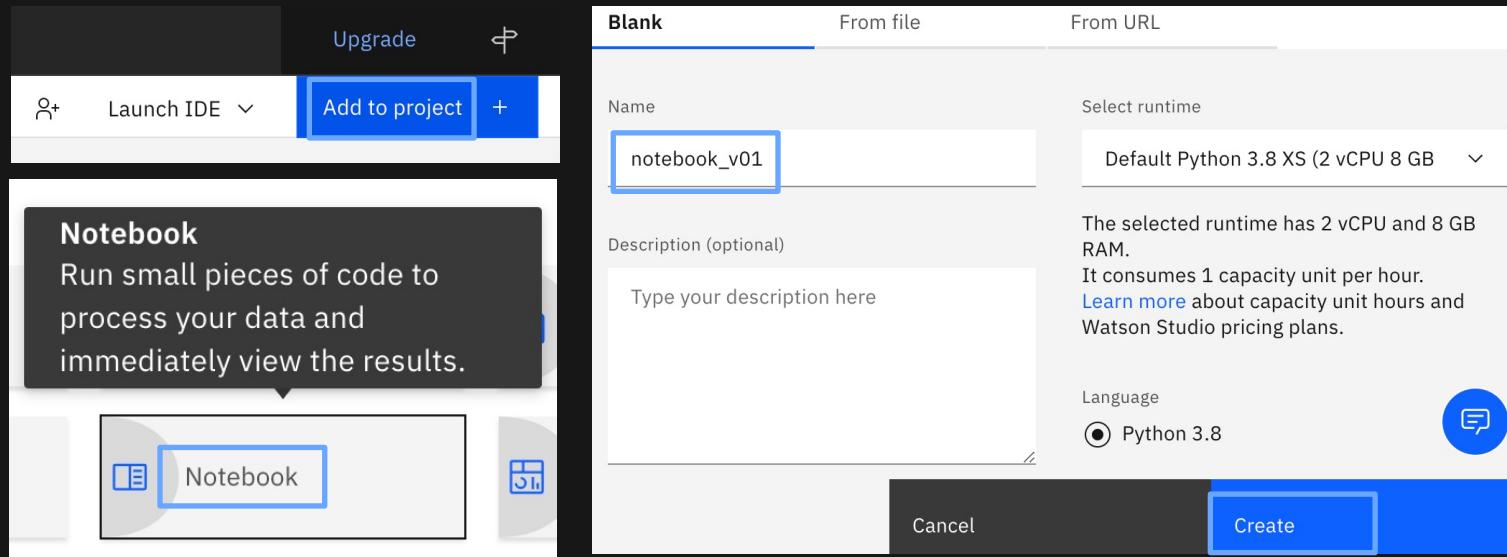
1. 프로젝트의 “Assets” 탭으로 이동
2. deployment/product_recommend/ 폴더에서 “customers_orders.csv” 파일 드래그 하여 Load란으로 이동
3. 처리까지 기다립니다.



노트북 생성

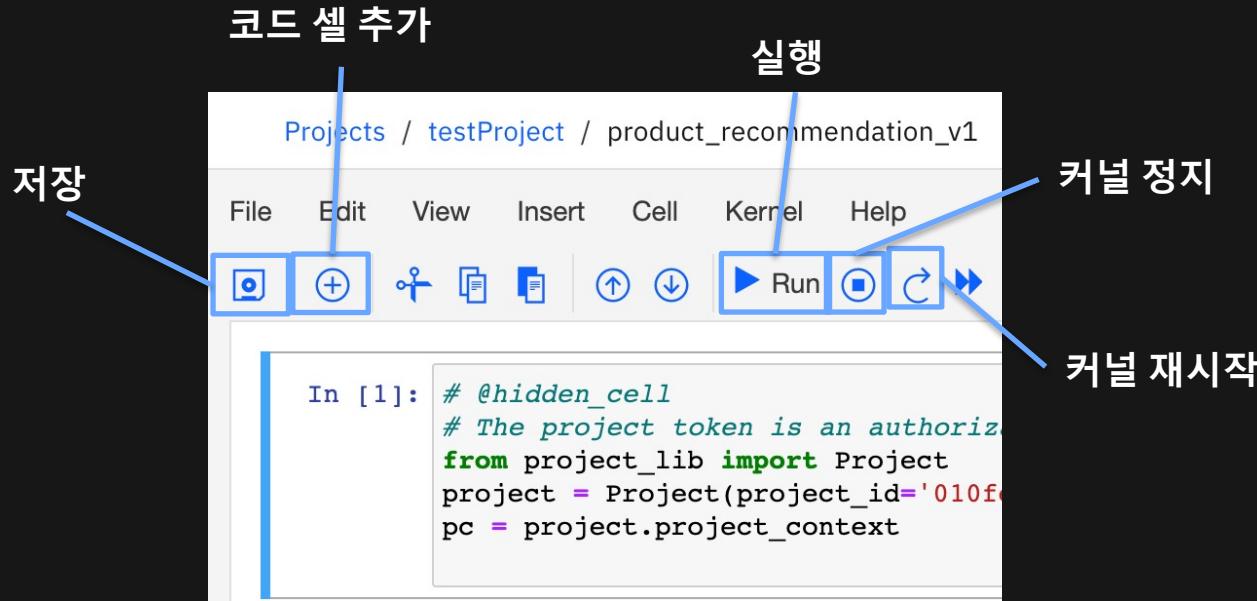
Watson Studio에서 노트북은 Python, R, Spark 언어를 이용한 모델 개발에 이용됩니다.

1. 상단 메뉴에서 “Add to project” > 팝업 메뉴중 “Notebook” 클릭
2. “Default Python” 노트북이 선택 되어 있습니다.
3. 이름을 입력하고 “Create” 클릭하여 노트북 생성합니다.



노트북 메뉴 설명

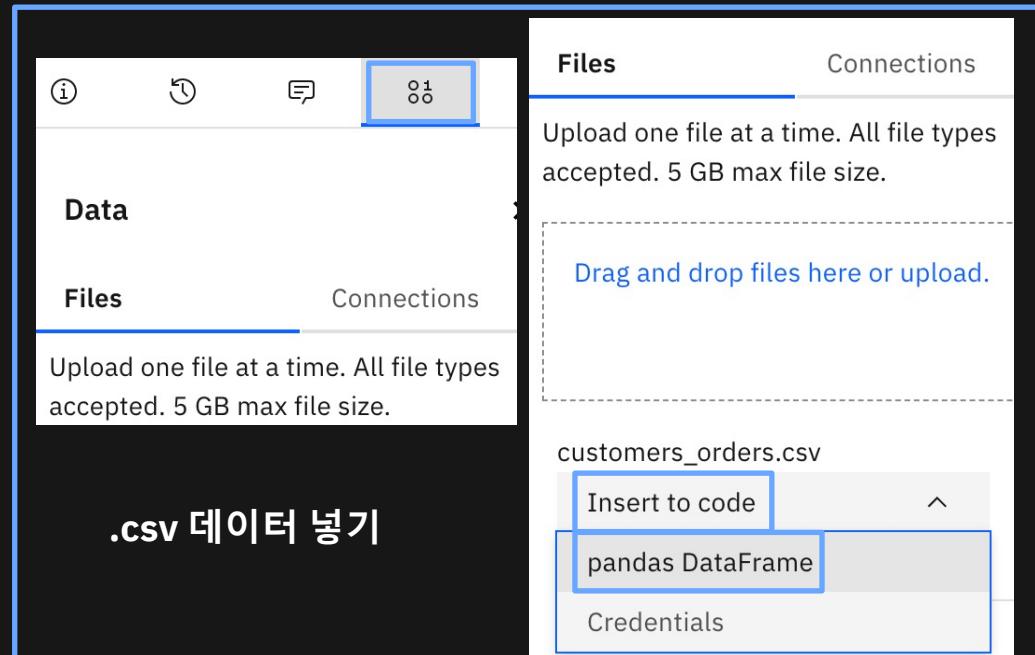
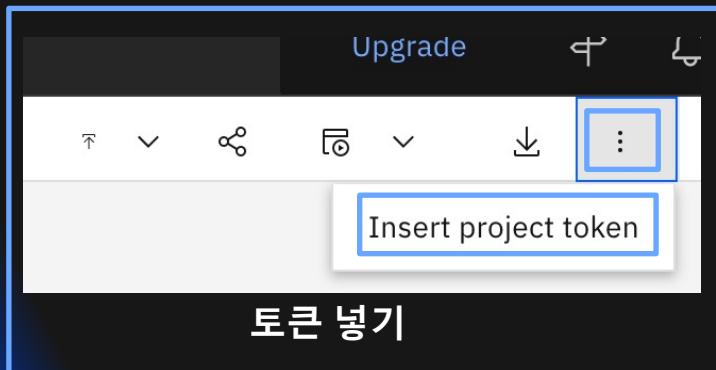
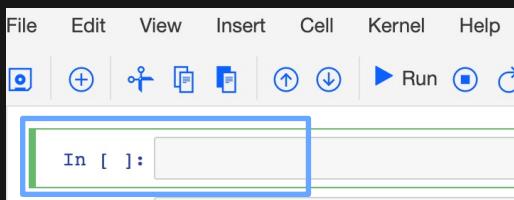
주로 사용하는 메뉴는 아래와 같습니다.



데이터 및 토큰 추가

이미 생성된 토큰 및 업로드된 .csv 데이터를 불러 오겠습니다.

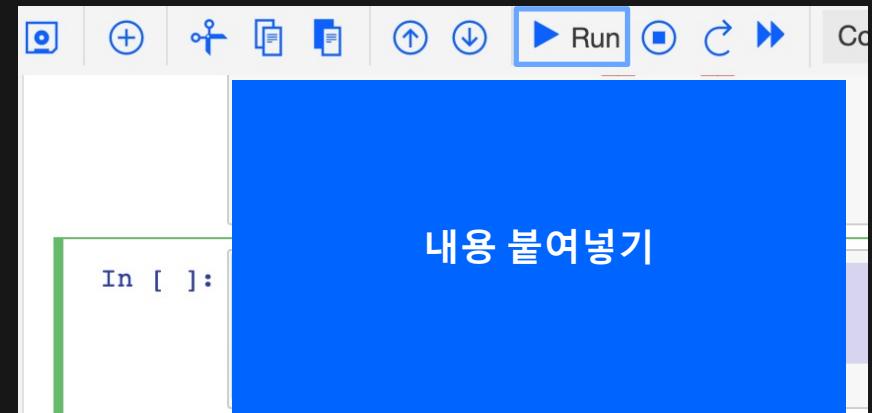
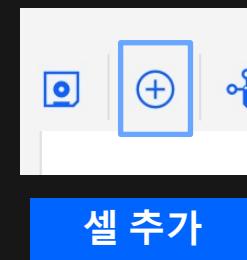
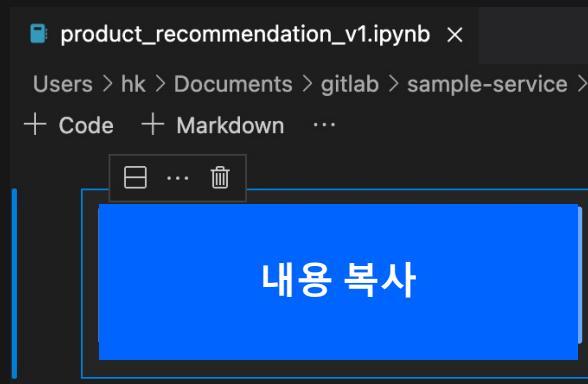
- 처음 공백 셀(cell)을 클릭
- 토큰 넣기: 상단 메뉴에서 “숨긴 메뉴” > 팝업 메뉴중 “Insert project token” 클릭
- 데이터 넣기: 상단 메뉴에서 “0100” 데이터 버튼 클릭
- Files 목록중 업로드한 “customers_orders.csv” 아래에 “Insert to code” > “pandas DataFrame” 클릭



노트북 단계별 실행

노트북 코드를 업로드 하여 한번에 다 실행 할 수 있으나 단계별 실행 하겠습니다.

1. 열람중인 VSCode의 “product_recommendation_v1.ipynb” 파일의 첫번째 셀을 선택 > 복사
2. Watson Studio 노트북에서 + 버튼을 클릭하여 코드 cell을 추가
3. 복사한 내용 붙여넣기
4. 처음 셀(cell) 부터 실행(Run) 버튼을 클릭하여 붙여 넣은 코드 셀(cell) 까지, 셀(cell) 별로 실행 합니다.



노트북 셀(Cell) 실행한 결과

실행한 결과는 print/display/변수명 입력에 따라 결과가 셀(cell) 아래에 표시 됩니다.

1. 아래의 결과가 나오도록 VSCode에서 복사하여 실행합니다.

2. 실행한 결과를 보면 data(m by n) 크기, data 쿨럼명, data를 볼 수 있습니다.

*주요 부분은 설명이 있겠으나, 이외의 부분은 단계별 vscode에서 복사하여 붙이고 실행합니다.

The screenshot shows a Jupyter Notebook interface with a code cell containing Python code and its resulting DataFrame output.

Code Cell Content:

```
df_data_1 = pd.read_csv(body)
df_data_1.head()
#읽어드린 데이터 객체 이름 변경 및 일부 표시
df = df_data_1
print(df.shape) #크기 (행 x 열)
display(df) #데이터 보기
```

Output:

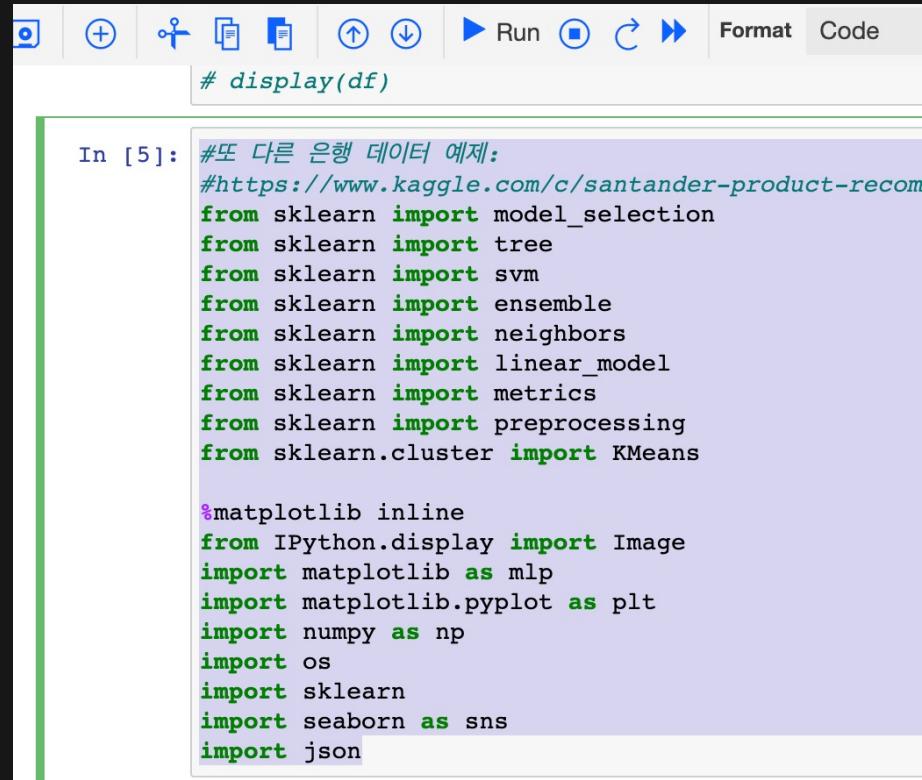
	DRESS1	CITY	STATE	COUNTRY_CODE	POSTAL_CODE	POSTAL_CODE_PLUS4	ADDRESS2		
In []:	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN		
1	Allen Perl	Mr.	Hillcrest Lane	Abeto	PG	IT	6040	0	NaN
2	Allen Perl	Mr.	Hillcrest Lane	Abeto	PG	IT	6040	0	NaN
3	Allen Perl	Mr.	Hillcrest Lane	Abeto	PG	IT	6040	0	NaN
4	Allen Perl	Mr.	Hillcrest Lane	Abeto	PG	IT	6040	0	NaN

노트북 셀(Cell) 실행한 결과 - 라이브러리

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*이 노트북에서 사용할 라이브러리입니다.

```
#또 다른 익행 데이터 예제:  
#https://www.kaggle.com/c/santander-product-recommen...  
from sklearn import model_selection  
from sklearn import tree  
from sklearn import svm  
from sklearn import ensemble  
from sklearn import neighbors  
from sklearn import linear_model  
from sklearn import metrics  
from sklearn import preprocessing  
from sklearn.cluster import KMeans  
  
%matplotlib inline  
from IPython.display import Image  
import matplotlib as mlp  
import matplotlib.pyplot as plt  
import numpy as np  
import os  
import sklearn  
import seaborn as sns  
import json
```



The screenshot shows a Jupyter Notebook cell with the following code:

```
# display(df)  
  
In [5]: #또 다른 익행 데이터 예제:  
#https://www.kaggle.com/c/santander-product-recommen...  
from sklearn import model_selection  
from sklearn import tree  
from sklearn import svm  
from sklearn import ensemble  
from sklearn import neighbors  
from sklearn import linear_model  
from sklearn import metrics  
from sklearn import preprocessing  
from sklearn.cluster import KMeans  
  
%matplotlib inline  
from IPython.display import Image  
import matplotlib as mlp  
import matplotlib.pyplot as plt  
import numpy as np  
import os  
import sklearn  
import seaborn as sns  
import json
```



노트북 셀(Cell) 실행한 결과 – 불필요한 특성 삭제

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습에서 불필요한 특성(feature)은 오히려 판별에 문제를 일으키며, 이에 따라 불필요한 행(column)은 삭제 하겠습니다.(Ex.: 주소, 나이, 전화번호, 등...)

#불필요한 콜럼 지우기

```
df.drop(['CUSTNAME', 'AGE', 'ADDRESS1', 'COUNTRY_CODE',
         'POSTAL_CODE', 'POSTAL_CODE_PLUS4', 'ADDRESS2', 'EMAIL_ADDRESS',
         'PHONE_NUMBER', 'CREDITCARD_TYPE', 'LOCALITY', 'SALESMAN_ID', 'NATIONALITY', 'NATIONAL_ID', 'CREDITCARD_NUMBER',
         'ORDER_ID', 'ORDER_DATE', 'ORDER_TIME', 'FREIGHT_CHARGES',
         'ORDER_SALESMAN', 'ORDER_POSTED_DATE', 'ORDER_SHIP_DATE', 'ORDER_VALUE', 'T_TYPE', 'PURCHASE_TOUCHPOINT', 'PURCHASE_TYPE',
         'GenderCode'], axis=1, inplace=True)
```

#도시명, 지역명, 연령대, 제품 10개 사용

```
print(df.shape)
display(df)
```

	CITY	STATE	CUST_ID	GENERATION	Baby Food	Diapers	Formula	Lotion	Baby wash	Wipes	...	Cleaning Products	Condiments	Frozen Foods	Kitchen Items	Meat	Office Supplies	Per
0	Abeto	PG	10003	Gen_Y	0	0	1	1	0	0	...	0	1	0	0	0	0	
1	Abeto	PG	10003	Gen_Y	0	0	1	0	1	0	...	1	0	0	0	0	0	
2	Abeto	PG	10003	Gen_Y	0	1	0	0	0	0	...	0	0	0	0	0	0	
3	Abeto	PG	10003	Gen_Y	0	0	0	1	0	0	...	0	0	0	0	0	0	
4	Abeto	PG	10003	Gen_Y	0	0	0	0	0	0	...	0	0	0	0	0	0	
...	



노트북 셀(Cell) 실행한 결과 – 불필요한 특성 삭제

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*데이터에 중복 사용자 데이터가 보입니다. 사용자별 구매 내용을 합산하고, 사용자의 아이디 역시 불필요한 정보 및 개인정보로 삭제하겠습니다.

```
#구매자별 구매 리스트 합산
```

```
df = df.groupby(by=[ 'CITY' , 'STATE' , 'GENERATION' , 'CUST_ID' ], as_index=False).sum()
```

```
#고객 아이디 콜럼 삭제
```

```
df.drop([ 'CUST_ID' ], axis=1,inplace=True)
```

```
print(df.shape)
```

```
display(df)
```



노트북 셀(Cell) 실행한 결과 – 정보 취합 이후 삭제

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습에서 문자 데이터는 계산이 안되어, 지역명, 연령대는 숫자로 변환 하겠습니다.

In [9]: #클러스터링을 이용한 레이블링 프로세스
#도시명, 연령대, 지역명 숫자화

```
label_encoder = preprocessing.LabelEncoder()

df['CITY'] = label_encoder.fit_transform(df['CITY'])
df['GENERATION'] = label_encoder.fit_transform(df['GENERATION'])
df['STATE'] = label_encoder.fit_transform(df['STATE'].astype(str))
```

In [10]: df.describe()

Out[10]:

	CITY	STATE	GENERATION	Baby Food	Diapers	Formula	Lotion	Baby wash	Wipes	Fresh Fruits	...	Cleaning Products	Co
count	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	4154.000000	41
mean	857.459316	50.975686	1.342802	0.153346	0.293211	0.163698	0.144439	0.158883	0.153828	0.165383	...	0.295378	
std	349.636961	39.481660	1.204559	0.402651	0.559675	0.411339	0.391130	0.407945	0.403663	0.417929	...	0.571532	



노트북 셀(Cell) 실행한 결과 – K-Means 클러스터링

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*K-Means 클러스터링 > 데이터별 그룹 판별 > 클래스/레이블명 추가

```
In [11]: #스케일링을 해야 하나, 데모상 없이 진행
#k-means cluster
#k-means cluster 생성, 고객 종류 100개
numb_of_cluster=10
kmeans = KMeans(n_clusters=numb_of_cluster)
kmeans.fit(df)

#k-means cluster를 이용한 고객 종류 분류
y_kmeans = kmeans.predict(df)

#각 데이터별 판별된 클러스터 지정 및 기존 데이터에 추가(마지막 콜럼에 추가됨)
df[ "cluster" ] = pd.DataFrame(y_kmeans, columns = { "cluster" })

display(df)
```

GENERATION	Baby Food	Diapers	Formula	Lotion	Baby wash	Wipes	Fresh Fruits	...	Condiments	Frozen Foods	Kitchen Items	Meat	Office Supplies	Personal Care	Pet Supplies	Sea Food	Spices	cluster
3	0	0	1	0	0	1	0	...	0	0	0	0	0	0	0	1	1	9
0	0	0	0	0	0	0	0	...	0	2	0	0	0	3	0	2	1	9



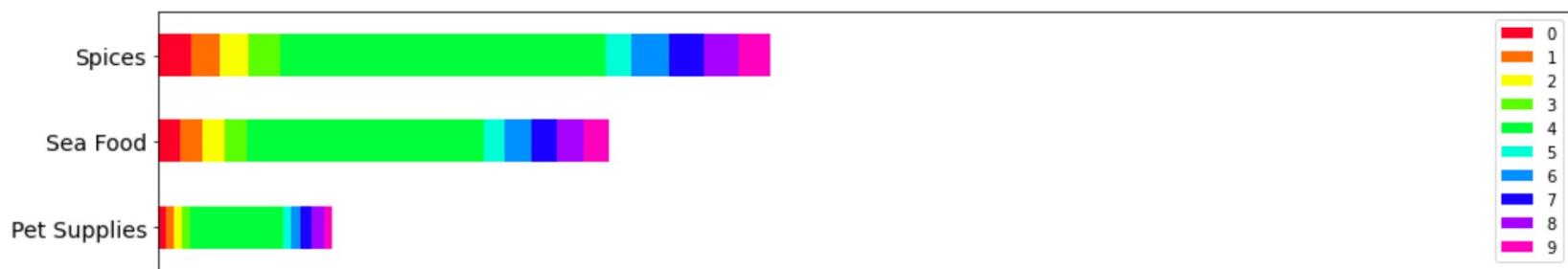
노트북 셀(Cell) 실행한 결과 - 클러스터 결과 리뷰

VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

클러스터별 구매한 아이템을 표기 합니다

```
In [12]: #클러스터된 데이터 리뷰: 각 클러스터별, 연령대별 구매 제품 표시  
grouped_df = df.groupby([ "cluster" ], as_index=False).sum()  
  
#각 줄의 인덱스가 이미 클러스터와 동일하여 클러스터 컬럼 및 불필요한 나머지 컬럼 드롭  
grouped_df.drop(['CITY', 'STATE', 'GENERATION', 'cluster'], axis=1,inplace=True)  
df_a = grouped_df.T  
df_a.plot(kind='barh', stacked=True, fontsize=14, figsize=[16,29], colormap='gist_rainbow')
```

Out[12]: <AxesSubplot:>



노트북 셀(Cell) 실행한 결과 – 많이 판매된 상품

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*클러스터별 많이 구매한 아이템을 수집합니다. 추후에 Mysql database에 넣고 판별된 클러스터에서 많이 판매된 아이템 조회에 이용됩니다.

```
#mysql DB에 넣을 데이터 정리, 아이템 갯수 및 클러스터가 적어서 클러스터별 아이템 랭킹에 많은 차이가 없습니다.  
#상단에 numb_of_cluster 갯수를 30 정도로 높이면 얼마만큼의 차이를 볼 수가 있습니다.  
#목표: values ('cluster_01', 'product 1', 'product 2', 'product 3', 'product 4', 'product 5'),(...);  
result = "values "  
for y in range(0, numb_of_cluster):  
    line = "('cluster_"+str(y)  
    for x in range(1, 6): #가장 많이 구매된 아이템 5개  
        line = line + ', "'+ str(df_a[df_a['rank_'+str(y)] == x].index[0])  
    line = line + ') ,\n"  
    result = result + line  
  
print(result[:-2]+";")  
  
values ('cluster_0', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Wine'),  
('cluster_1', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments'),  
('cluster_2', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments'),  
('cluster_3', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments'),  
('cluster_4', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments'),  
('cluster_5', 'Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments'),
```



노트북 셀(Cell) 실행한 결과 – 클러스터 정보 포함

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*데이터마다 클러스터 클래스/레이블 포함된 데이터를 추후 AutoAI에 사용하기 위해 저장합니다.

```
# 추후 Auto AI를 위해서 프로젝트 폴더에 저장
```

```
project.save_data("product_labeled_data.csv", df.to_csv(index=False), overwrite=True)
```

```
{'file_name': 'product_labeled_data.csv',
'message': 'File saved to project storage.',
'bucket_name': 'testproject-donotdelete-pr-ug7qdyrey7egkg',
'asset_id': '51a4af4f-178d-4177-ae88-439f47337ec1'}
```



Manual Machine Learning 기계학습



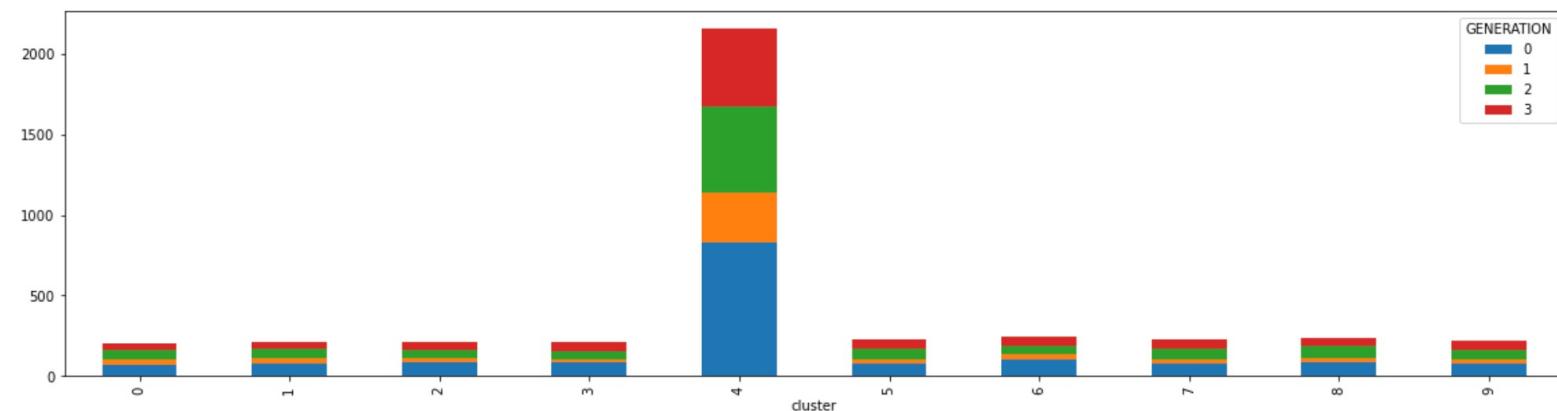
노트북 셀(Cell) 실행한 결과 – 레이블 데이터 리뷰

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*클러스터링을 통하여 자동 레이블된 데이터를 기계학습에 이용하기에 앞서, 데이터를 열람하겠습니다.

```
In [18]: #cluster별 세대 특성(feature) 분포 보기(세대는 별 차이 없는 것으로 보입니다.)  
df_data.groupby(["cluster", "GENERATION"]).size().unstack().plot(kind='bar', stacked=True, figsize=(20,5))
```

```
Out[18]: <AxesSubplot:xlabel='cluster'>
```



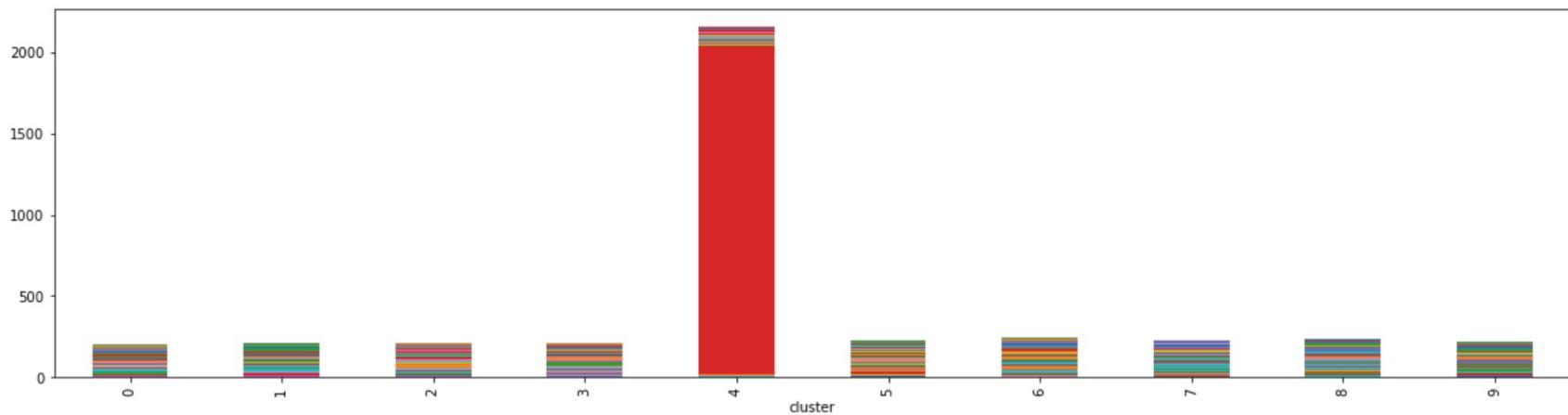
노트북 셀(Cell) 실행한 결과 – 레이블 데이터 리뷰

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*클러스터링을 통하여 자동 레이블된 데이터를 기계학습에 이용하기에 앞서, 데이터를 열람하겠습니다.

```
In [19]: #cluster별 지역 특성(feature) 분포 보기(세대는 별 차이 없는 것으로 보입니다.)  
df_data.groupby(["cluster", "STATE"]).size().unstack().plot(kind='bar', stacked=True, figsize=(20,5), legend=False)
```

```
Out[19]: <AxesSubplot:xlabel='cluster'>
```



노트북 셀(Cell) 실행한 결과 – 레이블 데이터 정리

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*학습/테스트를 위해서 클래스/레이블과 학습에 이용될 데이터를 나누겠습니다.

```
In [20]: #classify 목표(target 또는 y) class/레이블 따로 정리
y = df_data['cluster'].values.astype(np.int)
#학습 데이터에서 레이블 콜럼 삭제
df_data.drop(['cluster'], axis = 1, inplace=True)
print('\n목표(target 또는 y) 레이블:')
print(y)

#골고루 데이터 분포를 위해 스케일링, 결과: x
X = df_data.values.astype(np.float)
print('np float으로 변환:')
print(X)

scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)
print('\n스케일링:')
X.shape
print(X)
```

```
목표(target 또는 y) 레이블:
[9 9 9 ... 2 2 2]
np float으로 변환:
[[0.000e+00 1.250e+02 3.000e+00 ... 0.000e+00 1.000e+00 1.000e+00]
 [1.000e+00 8.600e+01 0.000e+00 ... 0.000e+00 2.000e+00 1.000e+00]
 [2.000e+00 9.400e+01 2.000e+00 ... 0.000e+00 0.000e+00 0.000e+00]]
```



노트북 셀(Cell) 실행한 결과 – 학습/테스트 함수

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*알고리즘별 학습/테스트를 나눠 실행하는 함수를 미리 지정

```
In [21]: #기계학습 알고리즘 별 학습/테스트 데이터를 나눠 학습/테스트 정확성 return 함수 생성
def stratified_cv(X, y, clf_class, shuffle=True, n_folds=2):
    stratified_k_fold = model_selection.StratifiedKFold(n_splits=n_folds, shuffle=shuffle)
    y_pred = y.copy()
    # ii -> train
    # jj -> test indices
    for ii, jj in stratified_k_fold.split(X, y):
        X_train, X_test = X[ii], X[jj]
        y_train = y[ii]
        clf = clf_class
        clf.fit(X_train,y_train)
        y_pred[jj] = clf.predict(X_test)
    return y_pred
```



노트북 셀(Cell) 실행한 결과 – 기계학습 라이브러리

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습 라이브러리 import, 대표적인 알고리즘 4개를 사용하겠습니다.

```
In [22]: # 몇 가지의 기계학습 알고리즘 import
from sklearn.ensemble import GradientBoostingClassifier
gradient_boost = GradientBoostingClassifier()

from sklearn.svm import SVC
svc_model = SVC(gamma='auto')

from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=10)

from sklearn.neighbors import KNeighborsClassifier
k_neighbors = KNeighborsClassifier()

from sklearn.linear_model import LogisticRegression
logistic_regression = LogisticRegression(solver='lbfgs')
```



노트북 셀(Cell) 실행한 결과 – 알고리즘 성능 단순 리뷰

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습 라이브러리 import, 대표적인 알고리즘 4개 단순 정확성 비교를 보겠습니다.

```
#알고리즘별 성능 테스트
print('Gradient Boosting Classifier: {:.2f}'.format(metrics.accuracy_score(y, stratified_cv(X, y, gradient_boost))))
print('Support vector machine(SVM): {:.2f}'.format(metrics.accuracy_score(y, stratified_cv(X, y, svc_model))))
print('Random Forest Classifier: {:.2f}'.format(metrics.accuracy_score(y, stratified_cv(X, y, random_forest))))
print('K Nearest Neighbor Classifier: {:.2f}'.format(metrics.accuracy_score(y, stratified_cv(X, y, k_neighbors))))
print('Logistic Regression: {:.2f}'.format(metrics.accuracy_score(y, stratified_cv(X, y, logistic_regression))))
```

```
Gradient Boosting Classifier: 0.99
Support vector machine(SVM): 0.72
Random Forest Classifier: 0.88
K Nearest Neighbor Classifier: 0.58
Logistic Regression: 0.82
```



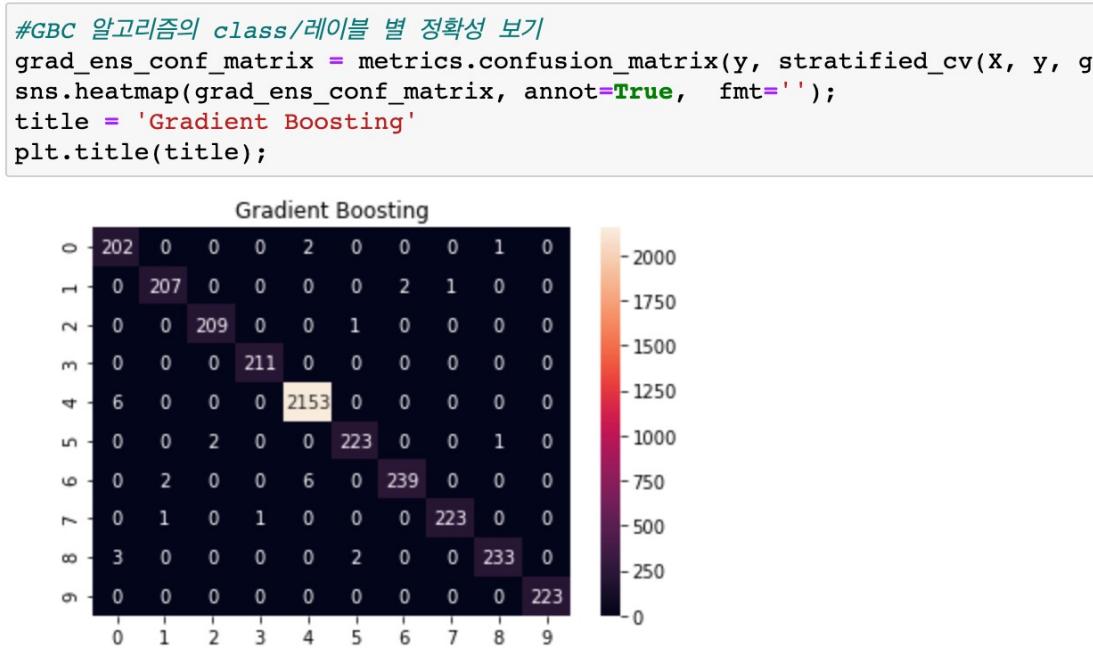
노트북 셀(Cell) 실행한 결과 – 알고리즘 성능 리뷰

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*알고리즘 4개 성능을 확인합니다.

```
#알고리즘별 성능 테스트
print('Gradient Boosting Classifier: {:.2f}'.format(GradientBoostingClassifier().fit(X_train, y_train).score(X_test, y_test)))
print('Support vector machine(SVM): {:.2f}'.format(SVC().fit(X_train, y_train).score(X_test, y_test)))
print('Random Forest Classifier: {:.2f}'.format(RandomForestClassifier().fit(X_train, y_train).score(X_test, y_test)))
print('K Nearest Neighbor Classifier: {:.2f}'.format(KNeighborsClassifier().fit(X_train, y_train).score(X_test, y_test)))
print('Logistic Regression: {:.2f}'.format(LogisticRegression().fit(X_train, y_train).score(X_test, y_test)))

Gradient Boosting Classifier: 0.99
Support vector machine(SVM): 0.73
Random Forest Classifier: 0.86
K Nearest Neighbor Classifier: 0.59
```



레이블 별 판단 성능



노트북 셀(Cell) 실행한 결과 – 알고리즘 성능 리뷰

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습 알고리즘의 성능은 precision과 f1 스코어를 보고 판단합니다.

```
#precision 및 f1 스코어 보기
print('Gradient Boosting Classifier:\n {}' .format(metrics.classification_report(y, str
print('Support vector machine(SVM):\n {}' .format(metrics.classification_report(y, str
print('Random Forest Classifier:\n {}' .format(metrics.classification_report(y, strati
```

Gradient Boosting Classifier:				
	precision	recall	f1-score	support
0	0.97	0.95	0.96	205
1	1.00	0.99	0.99	210
2	1.00	1.00	1.00	210
3	1.00	0.99	0.99	211
4	0.99	1.00	1.00	2159
5	1.00	1.00	1.00	226
6	0.99	0.97	0.98	247
7	0.99	1.00	0.99	225
8	0.99	1.00	0.99	238
9	1.00	1.00	1.00	223
accuracy				0.99
macro avg	0.99	0.99	0.99	4154
weighted avg	0.99	0.99	0.99	4154

레이블 별 판단 성능



노트북 셀(Cell) 실행한 결과 – 알고리즘 선택

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*하나의 알고리즘 선택하여 fit 함수 이용하여 classifier/모델 학습 및 생성 합니다.
생성된 모델은 predict 함수 및 테스트 데이터를 사용하여 결과 도출 가능합니다.

```
In [29]: #gbc 가 정확도가 높아 gbc를 선택하여 모델 생성  
#X 학습 데이터 한줄 마다, y 각 한 줄마다 class/레이블 값으로 학습  
gbc = ensemble.GradientBoostingClassifier()  
gbc.fit(X, y)  
#for large data set use gbc.partial_fit()
```

Out[29]: GradientBoostingClassifier()

```
In [30]: #다중 데이터를 한번에 classify/predict/test/스코어 할 때, 학습 데이터 X와 같이 데이터 생성하여 아래 함수 호출  
#다중 데이터 결과 Array에서 판별 결과 확인  
gbc.predict(X)
```

Out[30]: array([7, 7, 7, ..., 2, 2, 2])



노트북 셀(Cell) 실행한 결과 – 결과 도출

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계학습한 알고리즘의 결과는 클러스터 번호입니다.

각 클러스터 중 가장 많이 판매된 제품 5개를 도출하여 사용자에게 아래와 같이 보여줄 수 있습니다.

```
In [31]: #클라이언트 어플리케이션에서는 받은 클러스터 넘버로 각 클러스터별 많이 가장 판매된 제품 리스트 도출
def get_rank_from_cluster(a):
    result = []
    for x in range(1, 6): #가장 많이 구매된 아이템 5개
        result.append(df_a[df_a['rank_'+str(a)] == x].index[0])
    return result

answer = get_rank_from_cluster(0)
print(answer)

['Club Soda', 'Canned Foods', 'Chips', 'Popcorn', 'Condiments']
```

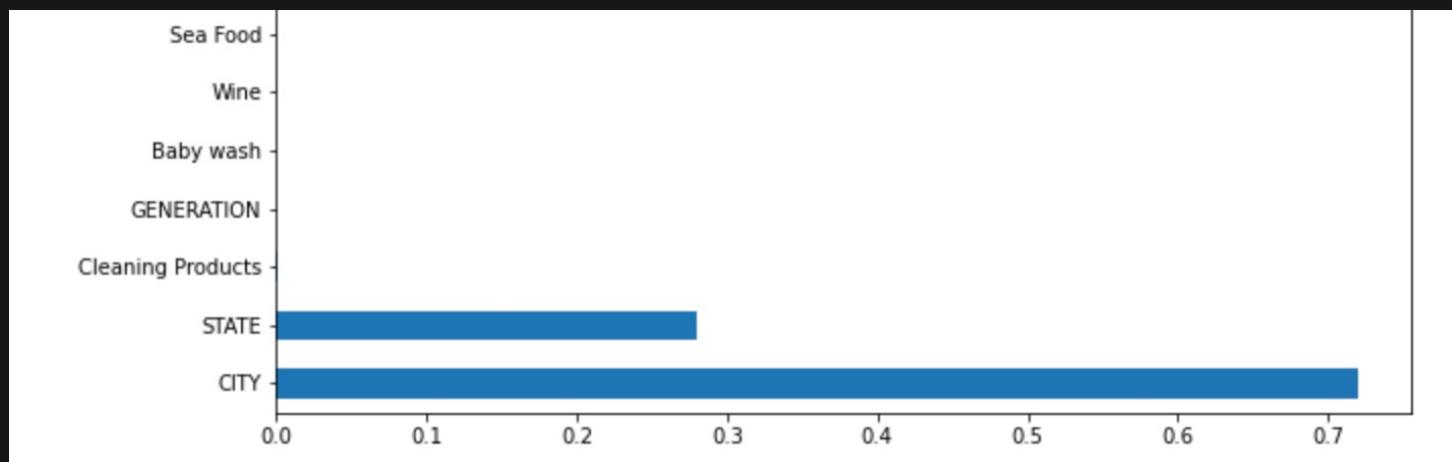


노트북 셀(Cell) 실행한 결과 – 각 특성별 중요도

1. VSCode에서 코드 셀(Cell)을 복사하여 붙여 넣고 코드 셀(Cell)을 실행 합니다.

*기계 학습한 각 특성별 중요도를 보겠습니다.

```
In [32]: #참고: 각 특성(Feature) 별 중요도 보기  
feature_importance = gbc.feature_importances_  
print (gbc.feature_importances_)  
feat_importances = pd.Series(gbc.feature_importances_, index=df_data.columns)  
feat_importances = feat_importances.nlargest(19)  
feat_importances.plot(kind='barh' , figsize=(10,10))
```



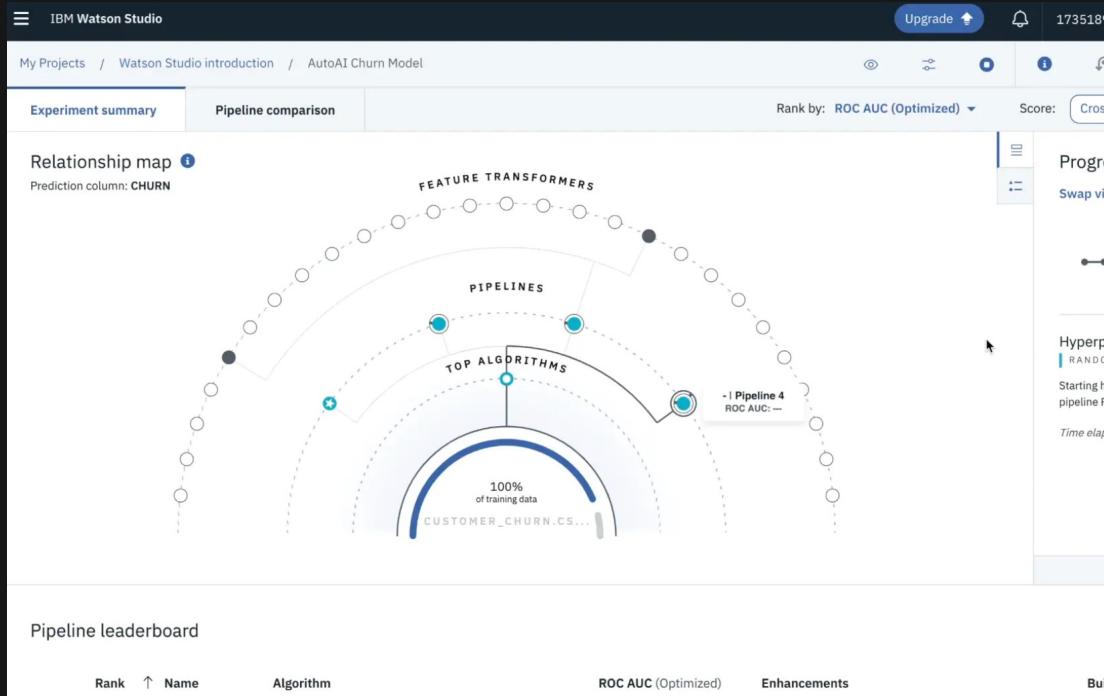
Watson

Auto AI



Watson Auto AI란?

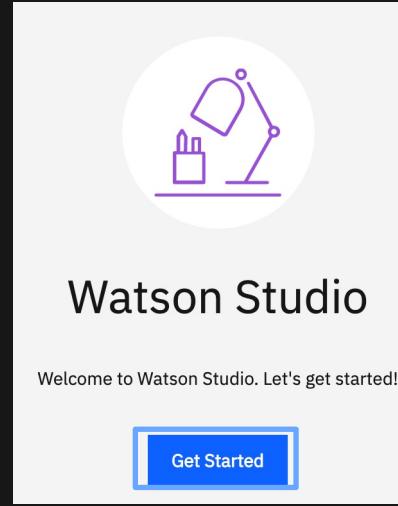
AutoAI는 AutoML의 변형입니다. 전체 AI 라이프사이클로 모델 구축 자동화를 확장합니다. AutoML처럼, AutoAI는 예측적 머신 러닝 모델을 구축하는 단계에 지능형 자동화를 적용합니다. 이러한 단계에는 훈련을 위한 데이터 세트 준비, 분류 또는 회귀 모델과 같이 주어진 데이터에 대한 최상의 모델 유형 식별, 기능 선택으로 알려진 모델이 현재 해결하고 있는 문제를 가장 잘 지원하는 데이터 열 선택 등이 포함됩니다.



Watson Studio 다시 시작

1. 새로운 브라우저 탭에서 <https://cloud.ibm.com>으로 로그인 합니다.
2. Dashboard에서 “Resource list”에서 사전에 생성한 Watson Studio 인스턴트 클릭
3. “Overview”항목 > 이전에 생성한 프로젝트를 클릭합니다.

The screenshot shows the 'Resource list' section of the IBM Cloud dashboard. A blue box highlights the 'Resource list' heading. Below it, there's a search bar labeled 'Filter by name or IP address...' and a collapsed section for 'Name'. Under 'Services and software (12)', a blue box highlights the 'Watson Studio-dev01-lite' entry, which is expanded to show its details. Other entries like 'Watson Machine Learning-dev02' and 'Watson Studio-dev01-lite' are also visible.



Overview

The screenshot shows the 'Recent projects' section of the Watson Studio Overview page. A blue box highlights the 'Recent projects' heading. It lists a single project named 'testProject' with a status indicator 'BK' and a timestamp 'Today at 02:41 PM'.



Watson Auto AI 생성

1. “Assets” 페이지 > “Add to project” 클릭 > “AutoAI experiment” 클릭
2. 현재 Watson Machine Learning이 연결 안되어 있습니다. “Associate a Machine Learning...”을 클릭하여 이미 생성된 Watson Machine Learning과 연결 페이지로 이동합니다.
3. 생성되어 있는 WML 인스턴스 체크(v), “Associate service” 클릭 > AutoAI 생성페이지로 돌아갑니다.
4. Reload 버튼을 클릭 하여 WML 연결 확인 및 이름 입력 후 “Create” 버튼 클릭 합니다.

The screenshot shows the workflow for creating a Watson AutoAI experiment:

- Step 1:** On the "Assets" page, the "Add to project" button is highlighted with a red box and number 1.
- Step 2:** The "Associate a Machine Learning service instance" link is highlighted with a red box and number 2.
- Step 3:** The "Associate service" button in the "Associate services" section is highlighted with a red box and number 4.
- Step 4:** The "Create" button at the bottom right is highlighted with a red box and number 4.

Left Panel (AutoAI experiment details):

- AutoAI experiment: Fully automated approach to building a classification or regression model.
- AutoAI experiment button (highlighted with red box 1).

Top Bar:

- Launch IDE (highlighted with red box 1).
- Add to project (highlighted with red box 1).
- Access control.

Associate services (Main Window):

- Description: Associate a Machine Learning service instance with your project on the project settings page, then click the reload button below to refresh the instances available for associating with your new model builder instance.
- Associate service button (highlighted with red box 4).
- Table:

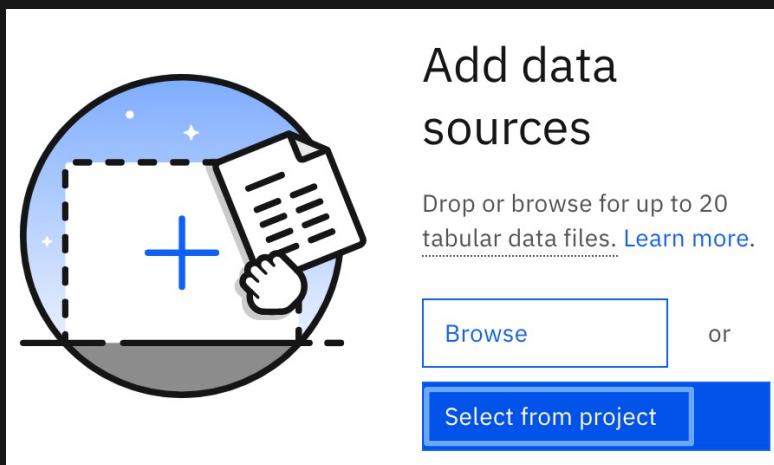
Name	Type	Plan	Location	Status	Group
WatsonMachineLearning	Machine Learning	Lite	Dallas	Not associated	Default

- Define details:
 - Name: Name of AutoAI experiment (highlighted with red box 4).
 - Description: Description of AutoAI experiment.
- Associate services:
 - Watson Machine Learning Service Instance: Watson Machine Learning Service Instance *
 - Associate a Machine Learning service instance with your project on the project settings page, then click the reload button below to refresh the instances available for associating with your new model builder instance.
- Buttons: Reload (highlighted with red box 4) and Create (highlighted with red box 4).

학습/테스트 데이터 추가

학습에 이용될 데이터를 추가 하겠습니다.

1. “Select from project” 클릭
2. 클러스터링 이후 레이블링 맞춘 `product_labeled_data.csv` 데이터 선택
3. “Select asset” 클릭하여 AutoAI에 추가 합니다.



Data assets

Selected assets

All assets details must load before final selection.

Asset name

Asset details

Select asset

Asset	Status
customers_orders.csv	
product_labeled_data.csv	Selected

학습/테스트 데이터 선택

판별에 이용할 데이터를 선택 하겠습니다.

1. 이전에 비슷한 그룹별 클러스터 한 이후 레이블한 클러스터 열(Column)을 선택 합니다.
2. 판별하는 레이블(클래스)가 10개 이여서 Multiclass classification이 선택 된 것을 볼 수 있습니다.

Configure details

What do you want to predict?

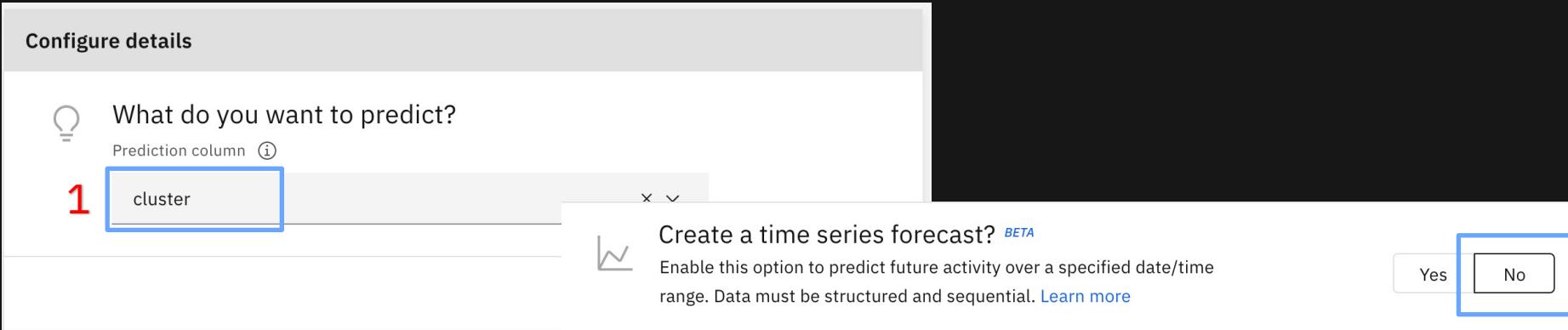
Prediction column [\(i\)](#)

1 cluster

Create a time series forecast? [BETA](#)

Enable this option to predict future activity over a specified date/time range. Data must be structured and sequential. [Learn more](#)

Yes No



Prediction column: cluster

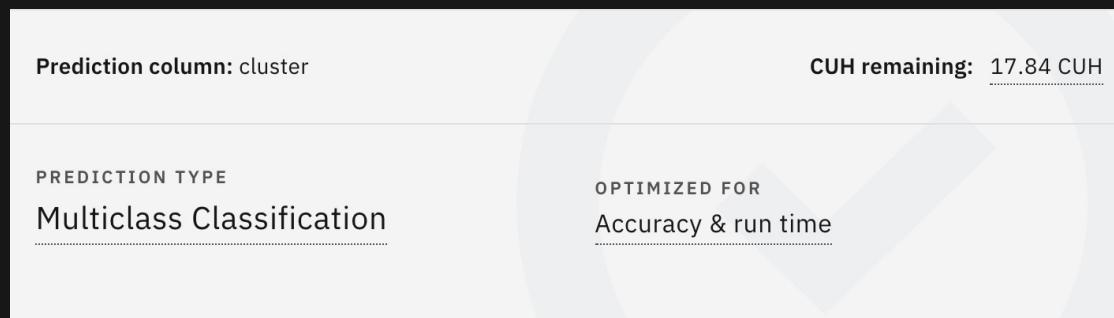
CUH remaining: 17.84 CUH

PREDICTION TYPE

Multiclass Classification

OPTIMIZED FOR

Accuracy & run time



학습/테스트 상세 선택

판별에 이용할 데이터 및 알고리즘 상세 선택 하겠습니다.

1. 하단에 “Experiment Settings”를 클릭합니다.
2. 알고리즘 항목 중 Gradient Boosting Classifier를 체크(v) 합니다.
3. Optimized algorithm 선택을 “Score only”, “Save settings” 클릭 Algorithm 항목에 1 선택
4. “Run experiment” 버튼 클릭하여 학습 시작합니다. (약 15분 정도 소요 되어, 진행 되는 동안 휴식)

The screenshot shows the 'Experiment settings' section. At the top, there's a button labeled 'Experiment settings' with a red number '1' over it. Below it is a section titled 'Algorithms to include (10 / 11)' with a search bar and a table. The table has columns for 'Algorithm name' and checkboxes. Two algorithms are checked: 'Decision Tree Classifier' and 'Extra Trees Classifier'. A third algorithm, 'Gradient Boosting Classifier', is shown with its checkbox also checked, but it is highlighted with a red border and a red number '2' over it, indicating it is the target for step 2.

This is a screenshot of the 'Optimized algorithm selection' dialog box. It contains instructions about how AutoAI selects algorithms. There are two radio button options: 'Score only' (selected) and 'Score and run time'. A red number '3' is placed over the 'Score only' option, indicating it is the target for step 3. Below the dialog is another section titled 'Algorithms to use (1 / 4)' with a slider set to '1'. A red box highlights the '1' on the slider. At the bottom of the dialog are 'Cancel' and 'Save settings' buttons, with the 'Save settings' button being highlighted with a blue border and a red box over it, indicating it is the target for step 4.

멈추세요!! Run experiment 이 버튼은 아직 누르지 마세요!!

AutoAI 학습

1. AutoAI 학습이 시작되며, “Swap view”를 클릭하여 학습 절차를 확인 할 수 있습니다.

The screenshot shows the AutoAI interface with two main views displayed side-by-side:

- Relationship map**: Shows a network graph of data relationships. A file named "product_labeled_data.csv" is listed below it.
- Progress map**: Shows the workflow steps: Read dataset, Split holdout data, Re-training data, Preprocessing, Model selection, Selected algorithm, Hyperparameter optimization, Feature engineering, and Hyperparameter optimization. Nodes P1, P2, P3, and P4 are shown connected to the "Hyperparameter optimization" step.

A blue box highlights the "Swap view" button in both the top navigation bar and the bottom navigation bar. A large red number "1" is overlaid on the Progress map area.

At the bottom right, there is a log window showing:

- Reading data source PRODUCT_LABELED_DA...
- Downloading source data
- Time elapsed: 64 seconds

Buttons for "View log" and "Save code" are also visible.

AutoAI 학습 경과 확인

1. View log를 클릭하여 현재 진행 로그도 확인 가능합니다.

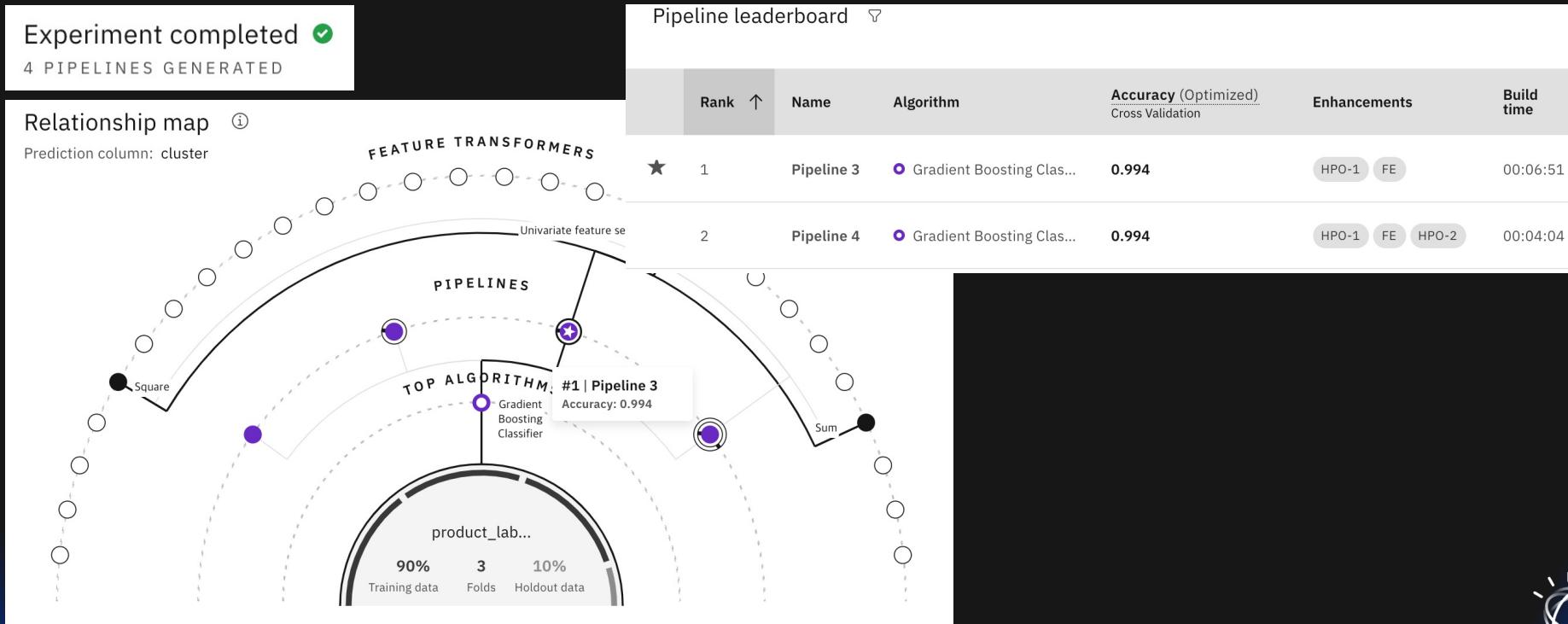
The screenshot shows the AutoAI interface with the following details:

- Progress map:** Displays the workflow stages: Read dataset, Split holdout data, Read training data, Preprocessing, Model selection, Selected algorithm, Hyperparameter optimization (P1), Feature engineering, Hyperparameter optimization (P2, P3, P4). A progress bar indicates the completion of the first four stages.
- Context menu:** A dropdown menu is open at the top right, showing options: "Relationship map", "Swap view", and "View log". The "View log" option is highlighted with a blue box.
- Bottom navigation:** Buttons for "View log" (highlighted with a red box) and "Save code" (with a red number "1" indicating an update).

AutoAI experiment full log				
Date	Start time	End time	Time elapsed	
2021. 9. 28.	오후 2:31:04	...	2 minutes	
<hr/>				
✓	오후 2:31:44	Starting the AutoAI experiment		
✓	오후 2:31:56	Setting default preprocessor parameters		
✓	오후 2:32:01	Selecting an algorithm for pipeline generation using 10% of training data. Discarding underperforming algorithms and keeping the top algorithm		
✓	오후 2:33:01	Completed feature engineering for pipeline P1		
✓	오후 2:33:17	Composing pipeline P1		
✓	오후 2:33:27	Starting model optimization for pipeline P2		

AutoAI 학습 결과 확인

1. AutoAI 학습 완료 되면, Experiment completed가 표시됩니다.
2. Relationship map에 가장 우수한 알고리즘에 별표로 표시 됩니다.
3. 하단 pipeline leaderboard에 우수한 알고리즘 순서로 표기 됩니다.



AutoAI 모델 선택

- 알고리즘 Gradient Boosting Classifier(GBC)의 높은 성능으로 GBC가 선택되어 HPO 작업이 들어간 것을 확인 할 수 있습니다. HPO 작업된 모델보다 Enhancements가 적용되지 않은 모델을 선택 하여 WML을 이용하여 서비스로 배포하겠습니다.
- “Save as”를 클릭합니다 > 모델 선택 > “Create” 버튼을 눌러 생성 합니다.
- “View in project”를 클릭하여 생성된 모델 인스턴스를 확인하려 이동합니다.

The screenshot shows the Watson Studio Pipeline interface. At the top, Pipeline 1 is selected. Below it, Pipeline details show an accuracy of 0.990 and a save button labeled 'Save as' with a red number 2. The Feature summary section displays feature importance for various categories like CITY, STATE, Pet Supplies, etc. The 'Feature summary' tab is currently active.

The screenshot shows the 'Save as' dialog box. It has a 'Select asset type' dropdown set to 'Model' (with a red number 2). The 'Define details' section includes fields for 'Name' (testAutoAI - P1 Gradient Boosting Classifier) and 'Description (optional)' (Enter description here). A success message at the bottom states 'Saved model successfully.' (with a red number 3) and provides a 'View in project' button.

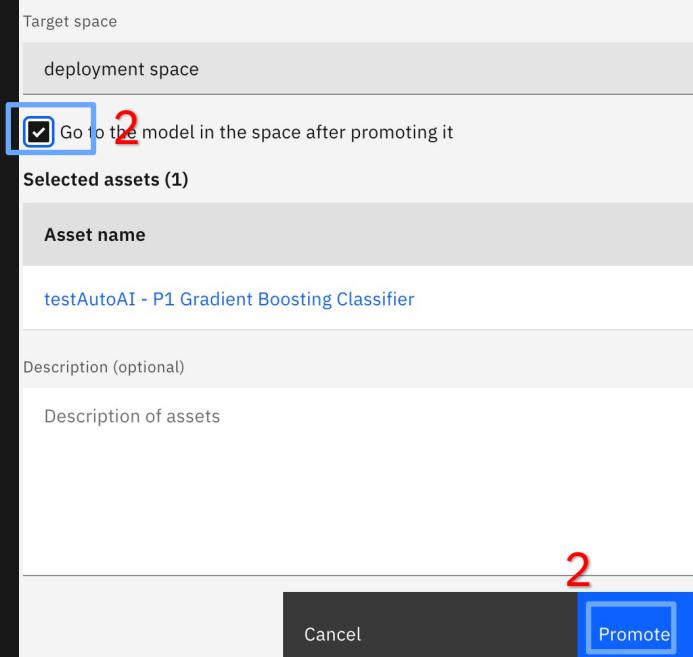
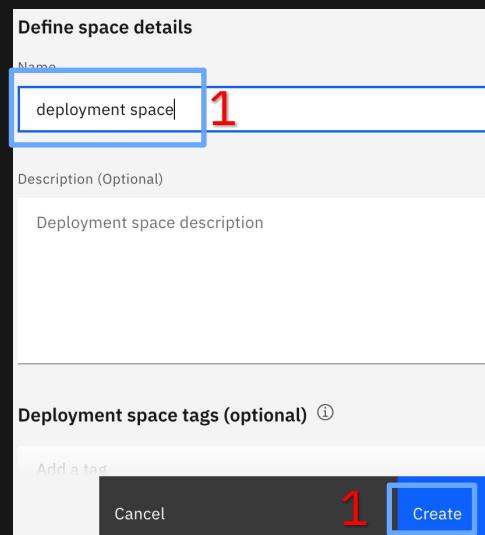
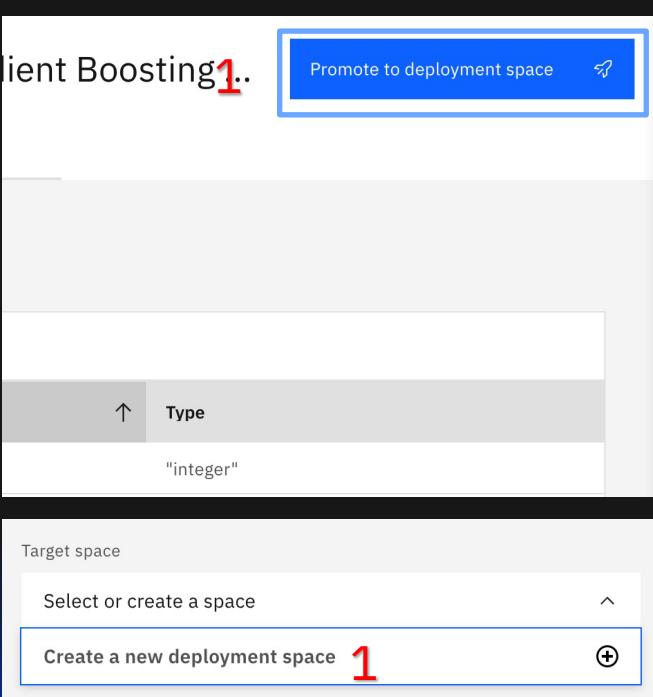
AutoAI 모델 배포 – step 1

선택된 모델을 WML 서비스로 배포하여 REST API 형식으로 판별(classify or predict) 호출 가능합니다.

1. “Promote to deployment space” 클릭 > 타켓 스페이스 항목에 “Create a new deployment space”

선택 후 deployment space 이름 입력 후 “create” 클릭하여 배포 공간 생성 완료합니다.

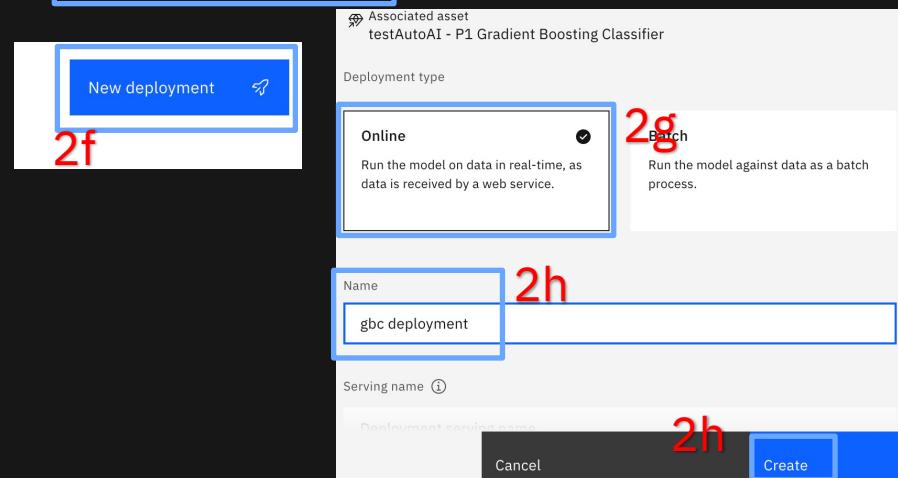
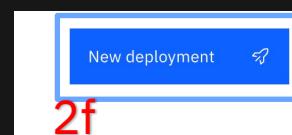
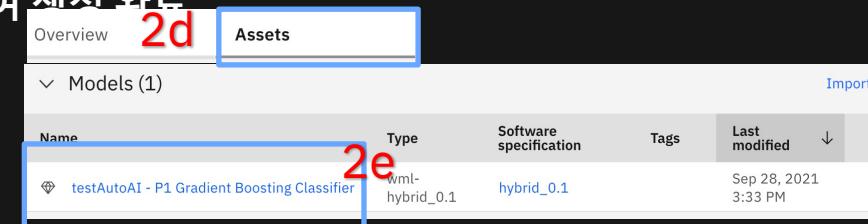
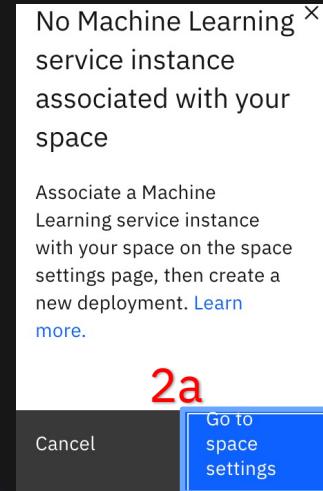
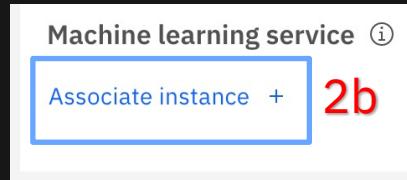
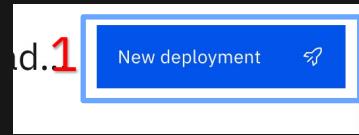
2. “Go to the model ...” 체크(v), “promote”를 클릭하여 배포 공간으로 이동



AutoAI 모델 배포 – step 2

배포 공간으로 이동된 모델을 REST API 서비스로 배포 하여야 합니다.

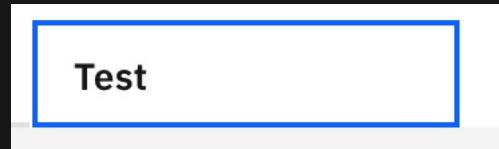
1. “New deployment”를 클릭 > 팝업에 해당 배포 공간에 ML 서비스가 없다는 메시지가 나옵니다.
3. a.“Go to space setting”클릭, b.Machine learning service 항목에 “Associate instance +”를 클릭하여
c. Watson ML 선택 > d. “Assets” 클릭 > e.“생성 된 모델 클릭” > f. 상단 “New deployments” 클릭 >
g.“online” 선택 > h.배포 명 입력 이후 “Create” 클릭하여 생성 완료



AutoAI 모델 배포 – 테스트

- 모델 배포가 완료 되면 초록색 체크(v) 표시가 나옵니다.
- 배포 객체 이름을 클릭하여, 임의 테스트를 진행 하겠습니다.
- 현재 모델의 시, 도 , 연령명은 숫자로 되어 있습니다. 임의 값 72, 124, 0, {임의 제품별 구매 갯수}를 입력 > “Add to list” > 우측 Input list 아래의 Predict 클릭 > 결과 확인 가능합니다.
- 결과는 어느 클러스터에 가장 비슷한 사용자인지 판별된 값입니다.

1 Online Deployment(s)		
Name	Status	Last modified
gbc deployment	Deployed	Sep 28, 2021 3:53 PM



Enter input data

CITY	<input type="text" value="72"/>
STATE	<input type="text" value="124"/>
GENERATION	<input type="text" value="0"/>
Baby Food	<input type="text" value="1"/>

Add to list

A blue rectangular button at the bottom of the input form, which is intended to add the current input data to a list for testing.

Input list (2)

[72, 124, 0, 1, 0, 1, 0, 1, 2, 1, 0]
[72, 124, 0, 1, 0, 1, 0, 1, 1, 2, 0, 0, 0, 1, 1, 0, 0, 1, 0, 2, 2, 3, 1, 0, 1, 0, 1, 0, 3, 2, 0, 1]

Predict (2)

A dark grey rectangular button at the bottom of the input list, which is used to trigger the prediction process.

Result

```
0 {  
1   "predictions": [  
2     {  
3       "fields": [  
4         "prediction",  
5         "probability"  
6       ],  
7       "values": [  
8         [  
9           7,  
10          [  
11            3.208290991049063e-7,  
12            1.2466797099368802e-9,  
13            1.1953129024023906e-9,  
14            1.1954237481178642e-9,  
15            1.1954237481178642e-9  
16          ]  
17        ]  
18      ]  
19    }  
20  ]  
21 }  
22 }  
23 }  
24 }  
25 }  
26 }  
27 }  
28 }  
29 }  
30 }  
31 }  
32 }  
33 }  
34 }  
35 }  
36 }  
37 }  
38 }  
39 }  
40 }  
41 }  
42 }  
43 }  
44 }  
45 }  
46 }  
47 }  
48 }  
49 }  
50 }  
51 }  
52 }  
53 }  
54 }  
55 }  
56 }  
57 }  
58 }  
59 }  
60 }  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }  
101 }  
102 }  
103 }  
104 }  
105 }  
106 }  
107 }  
108 }  
109 }  
110 }  
111 }  
112 }  
113 }  
114 }  
115 }  
116 }  
117 }  
118 }  
119 }  
120 }  
121 }  
122 }  
123 }  
124 }  
125 }  
126 }  
127 }  
128 }  
129 }  
130 }  
131 }  
132 }  
133 }  
134 }  
135 }  
136 }  
137 }  
138 }  
139 }  
140 }  
141 }  
142 }  
143 }  
144 }  
145 }  
146 }  
147 }  
148 }  
149 }  
150 }  
151 }  
152 }  
153 }  
154 }  
155 }  
156 }  
157 }  
158 }  
159 }  
160 }  
161 }  
162 }  
163 }  
164 }  
165 }  
166 }  
167 }  
168 }  
169 }  
170 }  
171 }  
172 }  
173 }  
174 }  
175 }  
176 }  
177 }  
178 }  
179 }  
180 }  
181 }  
182 }  
183 }  
184 }  
185 }  
186 }  
187 }  
188 }  
189 }  
190 }  
191 }  
192 }  
193 }  
194 }  
195 }  
196 }  
197 }  
198 }  
199 }  
200 }  
201 }  
202 }  
203 }  
204 }  
205 }  
206 }  
207 }  
208 }  
209 }  
210 }  
211 }  
212 }  
213 }  
214 }  
215 }  
216 }  
217 }  
218 }  
219 }  
220 }  
221 }  
222 }  
223 }  
224 }  
225 }  
226 }  
227 }  
228 }  
229 }  
230 }  
231 }  
232 }  
233 }  
234 }  
235 }  
236 }  
237 }  
238 }  
239 }  
240 }  
241 }  
242 }  
243 }  
244 }  
245 }  
246 }  
247 }  
248 }  
249 }  
250 }  
251 }  
252 }  
253 }  
254 }  
255 }  
256 }  
257 }  
258 }  
259 }  
260 }  
261 }  
262 }  
263 }  
264 }  
265 }  
266 }  
267 }  
268 }  
269 }  
270 }  
271 }  
272 }  
273 }  
274 }  
275 }  
276 }  
277 }  
278 }  
279 }  
280 }  
281 }  
282 }  
283 }  
284 }  
285 }  
286 }  
287 }  
288 }  
289 }  
290 }  
291 }  
292 }  
293 }  
294 }  
295 }  
296 }  
297 }  
298 }  
299 }  
300 }  
301 }  
302 }  
303 }  
304 }  
305 }  
306 }  
307 }  
308 }  
309 }  
310 }  
311 }  
312 }  
313 }  
314 }  
315 }  
316 }  
317 }  
318 }  
319 }  
320 }  
321 }  
322 }  
323 }  
324 }  
325 }  
326 }  
327 }  
328 }  
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 }  
353 }  
354 }  
355 }  
356 }  
357 }  
358 }  
359 }  
360 }  
361 }  
362 }  
363 }  
364 }  
365 }  
366 }  
367 }  
368 }  
369 }  
370 }  
371 }  
372 }  
373 }  
374 }  
375 }  
376 }  
377 }  
378 }  
379 }  
380 }  
381 }  
382 }  
383 }  
384 }  
385 }  
386 }  
387 }  
388 }  
389 }  
390 }  
391 }  
392 }  
393 }  
394 }  
395 }  
396 }  
397 }  
398 }  
399 }  
400 }  
401 }  
402 }  
403 }  
404 }  
405 }  
406 }  
407 }  
408 }  
409 }  
410 }  
411 }  
412 }  
413 }  
414 }  
415 }  
416 }  
417 }  
418 }  
419 }  
420 }  
421 }  
422 }  
423 }  
424 }  
425 }  
426 }  
427 }  
428 }  
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451 }  
452 }  
453 }  
454 }  
455 }  
456 }  
457 }  
458 }  
459 }  
460 }  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }
```

AutoAI - REST API로 호출

WML에 배포된 모델은 각종 언어를 통하여 호출 가능합니다.
페이지를 닫지 말고 새로운 브라우저 탭을 엽니다.

API reference

Test

Code snippets

cURL

Java

JavaScript

Python

Scala

KEY below using information retrieved from your IBM Cloud account.



```
https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})  
access_token"]
```

```
tion/json', 'Authorization': 'Bearer ' + mltoken}
```

```
he array(s) of values to be scored in the next line
```

```
{"fields": [array_of_input_fields], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]
```

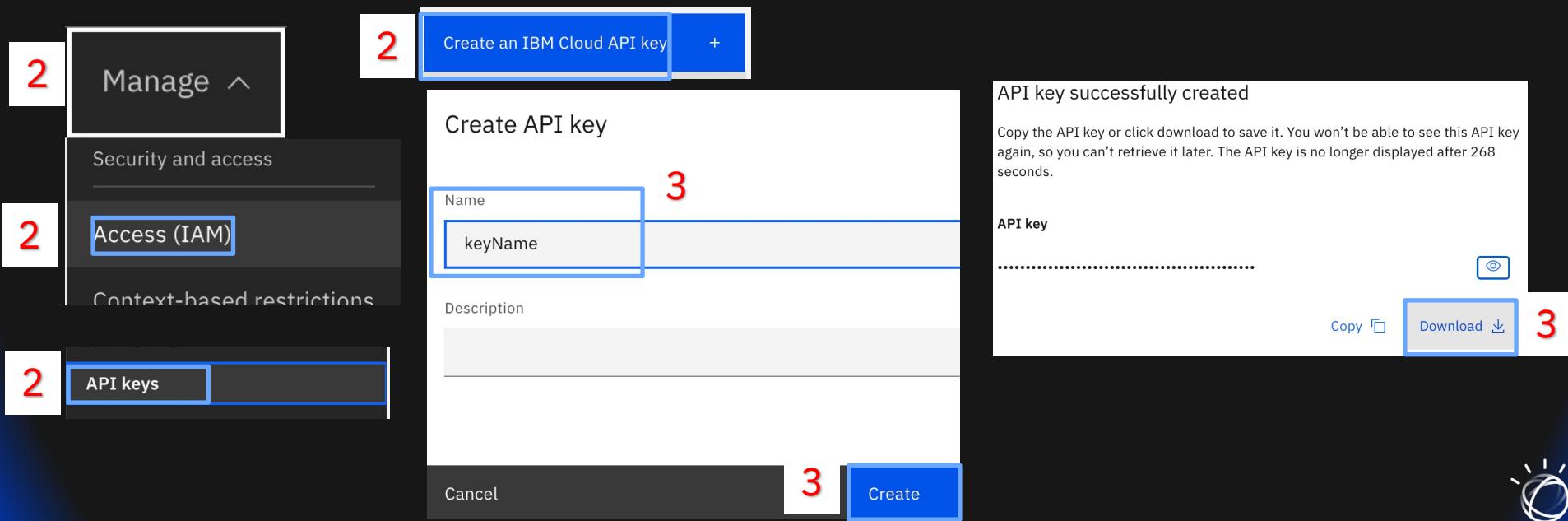
```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/3a6a5d58-0f50-47db-b83c-8b4712ec2272/predictions?version=2021-10-03&version=2021-10-03',
```



AutoAI – IAM API Key 생성

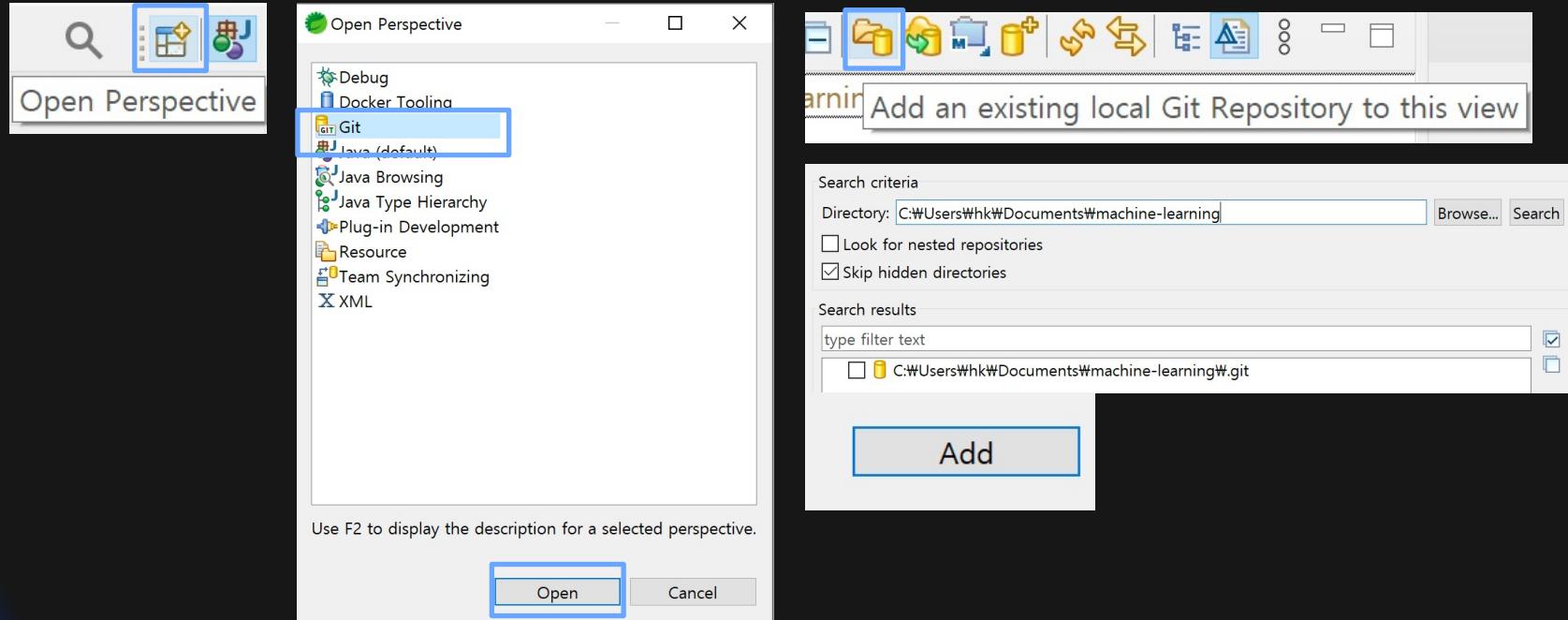
WML에 배포된 모델은 각종 언어를 통하여 호출 가능합니다.

1. 브라우저 새로운 탭/윈도우에서 <https://cloud.ibm.com> 콘솔페이지로 이동합니다.
2. 상단 메뉴중 Manage > Access (IAM) > 좌측 탭에서 “API Keys” > “Create an IBM Cloud API Key”
3. “키 이름” 지정 > “Create” > “Download” 클릭하여 Key 파일 다운로드(추후에 열람하여 사용합니다.)



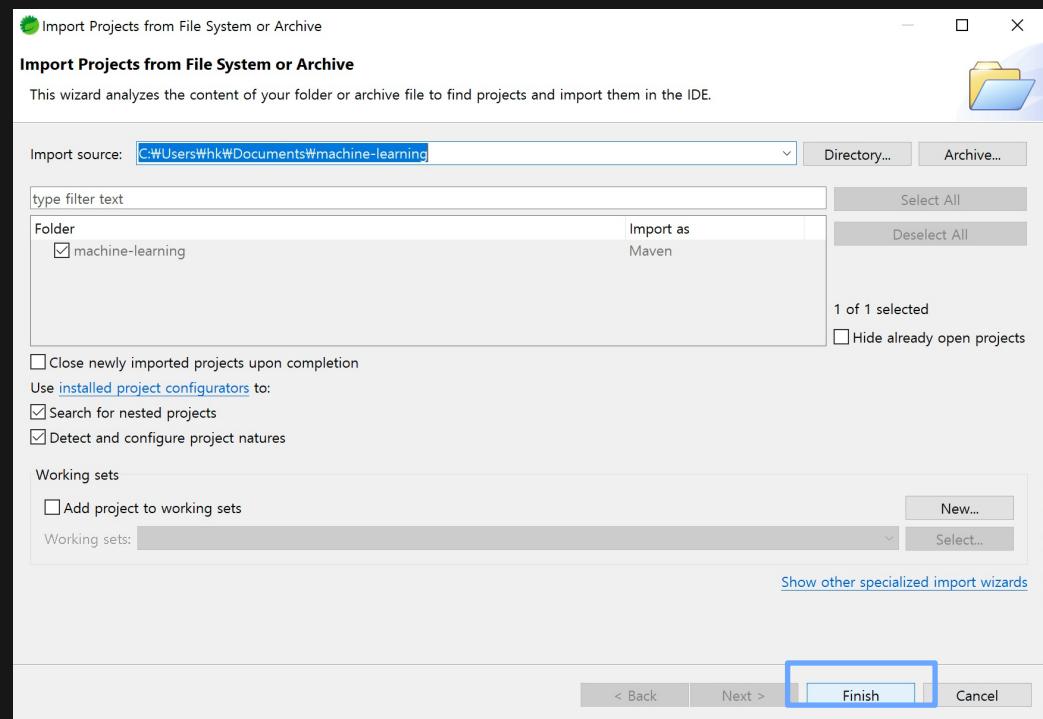
AutoAI – WML 서비스 호출

- STS4를 실행합니다.
- 우측 상단에 “Open Perspective” 클릭 > “Git” 선택 및 “Open” > “Add an existing....” > “Add”



AutoAI – WML 서비스 호출

- Repo가 화면에 보이면, 화면 계층을 한 단계씩 들어가서 “Working Tree”까지 들어갑니다.
- “Working Tree”폴더에서 우클릭 > “Import Projects” > “Finish”
- Java view 클릭 하여 Java 화면으로 이동



AutoAI – WML 서비스 호출

- Product_recommend/ProductController.java를 열람합니다.
- 기존에 생성한 API_KEY와 WML에서 Endpoint를 복사하여 scoringUrl 부분을 업데이트 합니다.

The screenshot shows the IBM Cloud AI interface. On the left, there's a sidebar with "API reference" and "Test" tabs. Below them are sections for "Direct link" and "Endpoint". The "Endpoint" section contains a URL: https://us-south.ml.cloud.ibm.com/ml/v4/deployments/bf20f923-477d-418b-846c-548ec8464420/predictions?version=2021-10-03T07:25:00Z&model_id=product_recommend. There are "Bearer <token>" and "IAM" buttons. On the right, there are tabs for "Code snippets" (with "Java" selected) and "curl". The Java code snippet is:

```
import java.nio.charset.StandardCharsets;
public class HttpClientTest {
    public static void main(String[] args) throws IOException {
        // NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
        String API_KEY = "<your API key>";

        HttpURLConnection tokenConnection = null;
        HttpURLConnection scoringConnection = null;
```

The screenshot shows the Eclipse IDE interface. On the left is the "Package Explorer" view, which lists the project structure: SwaggerConfig.java, com.springboot.microservices.sample.service, src/main/resources, src/test/java, JRE System Library [JavaSE-1.8], Maven Dependencies, target/generated-sources/annotations, target/generated-test-sources/test-annotations, bin, product_recommend (containing cluster_product_table.sql, customers_orders.csv, product_recommendation_v1.ipynb, ProductController.java), src, target, HELP.md, mvnw, mvnw.cmd, pom.xml, and README.md. On the right is the "ProductController.java" editor window. The code is as follows:

```
15  public String predictClusterMethod(String result = "";
16  // NOTE: you must manually set API_KEY
17
18  String API_KEY = "o1ttqzp6wy0zSvzv
19
20  HttpURLConnection tokenConnection
21  HttpURLConnection scoringConnection
22  BufferedReader tokenBuffer = null;
23  BufferedReader scoringBuffer = null
24
25  try {
26      // Getting IAM token
27      URL tokenUrl = new URL("https://
28      tokenConnection = (HttpURLConnection)
29      tokenConnection.setDoInput(true);
30      tokenConnection.setDoOutput(true);
31      tokenConnection.setRequestMethod("POST");
32      tokenConnection.setRequestBody(tokenBuffer =
33          new BufferedReader(new
34              StringBuffer jsonString =
35                  new StringBuffer();
36
37      while ((line = tokenBuffer.readLine()) != null)
```

AutoAI – WML 서비스 호출

- com.springboot.microservices.sample.rest/ProductController.java 파일에 “TODO:” 부분을 작성한 파일에서 복사/맞춰 업데이트 합니다.

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under 'machine-learning [boot] [devtools] [machine-learning]'. It includes 'src/main/java' with packages like com.springboot.microservices.sample, com.springboot.microservices.sample.dao, com.springboot.microservices.sample.model, and com.springboot.microservices.sample.rest, which contains files EnvPropCheck.java, HelloController.java, ProductController.java, and SwaggerConfig.java. It also shows 'src/main/resources', 'src/test/java', JRE System Library [JavaSE-1.8], Maven Dependencies, target/generated-sources/annotations, target/generated-test-sources/test-annotations, bin, and a product_recommend directory containing cluster_product_table.sql, customers_orders.csv, and product recommendation v1.ipynb. On the right, the ProductController.java editor shows code for a REST API endpoint. Lines 80 and 97 both contain a TODO comment: //TODO: product_recommend/ProductController.java 내용을 참고하여 작성하세요. The code uses annotations like @ApiOperation and @ApiImplicitParams to define the API parameters and their required values.

```
78     }
79
80     public String predictClusterMethod(String state_numb, String city_numb, String result = "";
81
82         return result;
83     }
84
85
86
87     @ApiOperation(value="클리스터 판별에 따른 인기 상품 조회(WML Rest API + DB 조회)")
88     @ApiImplicitParams({
89         @ApiImplicitParam(name="state_numb", value="도번호: 0~9", required = true),
90         @ApiImplicitParam(name="city_numb", value="시번호: 0~9", required = true),
91         @ApiImplicitParam(name="age_group", value="연령그룹: 0~9", required = true)
92     })
93     @RequestMapping(value="/product/predict", method= RequestMethod.GET)
94     public String predictProduct(@RequestParam(value="state_numb", required = true),
95         @RequestParam(value="city_numb", required = true),
96         @RequestParam(value="age_group", required = true),
97         String tempResult = "";
98
99
100    return tempResult;
101 }
```

AutoAI – WML 서비스 호출

- 좌측 하단에 **Boot Dashboard**에서 실행 버튼을 클릭하여 실행 합니다.
- 브라우저에서 <http://localhost:8080/swagger-ui.html> 링크를 입력하고 연결
- /product/predict 함수 클릭 > Try it out > 테스트 값 입력 후 > Execute > 결과 확인

The image shows the AutoAI interface with three main components:

- Boot Dashboard:** A sidebar on the left with various icons and a "local" section containing "machine-learning [devtools]".
- IBM Garage REST API:** The central part displays the REST API documentation for the "product-controller". It includes sections for "hello-controller" and "product-controller". Under "product-controller", there are three GET requests:
 - /cluster/predict (설명: 사용자 클러스터 판별 REST API 서비스 조회)
 - /cluster/product (설명: 클러스터 별 제품 랭킹 DB 조회)
 - /product/predict (설명: 클러스터 판별에 따른 인기 상품 조회(WML Rest API + DB 조회))
- Swagger UI:** A modal window for the "/product/predict" endpoint. It shows the following parameters:

Name	Description
age_group * required	연령 그룹: 0~9
city_numb * required	시 번호: 0~9
state_numb * required	도 번호: 0~9

The values entered are: age_group: 0, city_numb: 24, state_numb: 124. At the bottom are "Execute" and "Clear" buttons.



챗봇(Chatbot)

IBM

Watson

Assistant



Chatbot solution

Intent editor Info

An intent represents an action that fulfills a user's request. Intents can have:

- Intents (2) Add Search Sort by name
- FallbackIntent MakeAppointment

Conversation flow Info

A visual flow of the conversation:

```
graph TD; A["I would like to appointment to"] --> B["What type of review like to schedule?"]; B --> C["When should I sch {AppointmentTyp}"]
```

Dialogflow us

InsuranceBot en +

- Intents +
- Entities
- Knowledge [beta]
- Fulfillment
- Integrations
- Training
- Validation
- History

• What do i need to pay for my vehicle?

Contexts ?

Events ?

Training phrases ?

- Add user expression
- What are my monthly charges
- What do i need to pay for my vehicle
- What is the premium for my car insurance

Add node Add child node Add folder

Greetings welcome

1 Responses / 3 Context Set / Does not return

Extract Intents \$intents == null

0 Responses / 2 Context Set / Skip user input / Does not return

Iterate Intents \$intents.size() > 0

0 Responses / 1 Context Set / Skip user input / Does not return

AWS Lex

**Google
Dialogflow**

**IBM Watson
Assistant**

Chatbot development flow

인텐트

엔티티

다이얼
로그

테스트

클라이언트



Watson Assistant 생성 및 시작

- 링크(<https://cloud.ibm.com/>)를 클릭하여 Dashboard 페이지로 이동합니다.
- “Create resource” 클릭 > Watson assistant 검색
- 사용 동의 이후 “Create” 클릭
- Launch Watson Assistant 클릭

The screenshot illustrates the process of creating a Watson Assistant instance through the IBM Cloud catalog:

- Dashboard:** Shows the main dashboard with a "Create resource" button highlighted.
- Search:** The search bar contains "watson assistant". A result card for "Watson Assistant" is displayed, also highlighted.
- Location Selection:** A dropdown menu titled "Select a location" shows "Dallas (us-south)" selected.
- Pricing Plan:** A table shows the "Lite" plan selected, which includes "Everything you need to get started, free for as long as you need it" and "Up to 1,000 unique monthly active".
- Agreement and Create Button:** A checkbox is checked, agreeing to license agreements. A large blue "Create" button is visible.
- Final Step:** A large blue "Launch Watson Assistant" button is shown at the bottom right of the final step panel.

Watson Assistant 시작

IBM Watson Assistant Lite Upgrade

1. Intent: 사용자의 의도 또는 질문 파악
2. Entities: 숫자, 시간, 날짜와 같은 값
3. Dialog: 대화 흐름 정의

My first skill

Intents

Entities

My Entities

System Entities

Dialog

Options

Webhooks

Disambiguation

Autocorrection

Intent Detection

Analytics

Overview

User conversations

Versions

Content Catalog

What is an intent?

An intent is a collection of user statements that have the same meaning. By creating intents, you can teach your bot to understand the variety of ways users ask for the same thing.

You will find some pre-made intents in the Content Catalog. [Browse content catalog](#)

Create intent +



Watson Assistant 다이얼로그 스킬 생성

*현재 초기 Assistant 언어가 영어로 되어 있습니다.

1. 왼쪽위에 “Assistant” 버튼 클릭 > “Create assistant” 클릭
2. “Add an actions or dialog skill” 클릭
3. “Skill name” 입력 > 언어 선택 (한국어) > Dialog 선택
4. “Create skill” 클릭

IBM Watson Assistant Lite Upgrade

Assistants

An assistant helps your customer information faster. It may clarify requests from a knowledge base, and can answer human if needed.

Create assistant

Create assistant

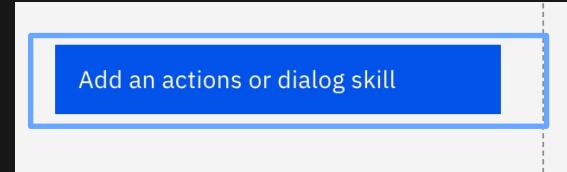
Create an assistant to deploy the skill that addresses your customers' goals.

Name

Description (optional)

Add a description for this assistant

Create assistant



Add Actions or Dialog skill

Add an existing skill or use the sample skill.

Add existing skill **Create skill** Use sample skill Upload skill

Name

Description (optional)

Add a description for this skill

Language ⓘ

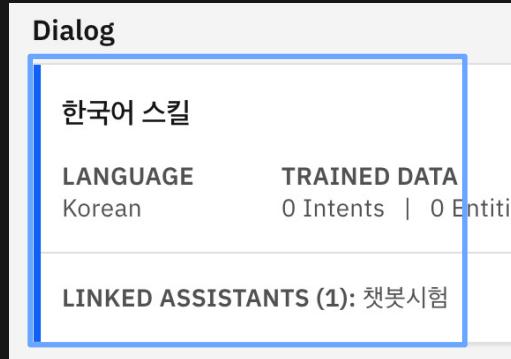
Skill type

Actions Dialog

Create skill

Watson Assistant 인텐트 생성 - 1

1. 생성한 디아일로그 선택
2. Create intent 클릭
3. Intent 이름과 설명 넣고 Create intent 클릭



What is an intent?

An intent is a collection of user statements that have the same meaning. By creating intents, you train your assistant to understand the variety of ways users express a goal. [Learn more](#)

[Create intent](#) + [Upload intents](#)

[←](#) | Create intent

Intent name
#누구인가

Name your intent to match a customer's question

Description (optional)
누구인지 물어보는 질문을 인식합니다

[Create intent](#)



Watson Assistant 인텐트 생성 - 2

Intent(인텐트)는 앞에서 설명과 같이 챗봇 사용자의 의도 파악에 이용됩니다.

여기서는 “누구인지” 물어보는 질문을 파악하는 인텐트를 파악하려 합니다.

1. 비슷한 종류의 많은 질문을 집어 넣겠습니다. 각 예제 문구 입력 후 “Add example”을 클릭

예제)

누구인가?

누구야?

넌 누구야?

이름이 뭐야?

자기 소개해봐

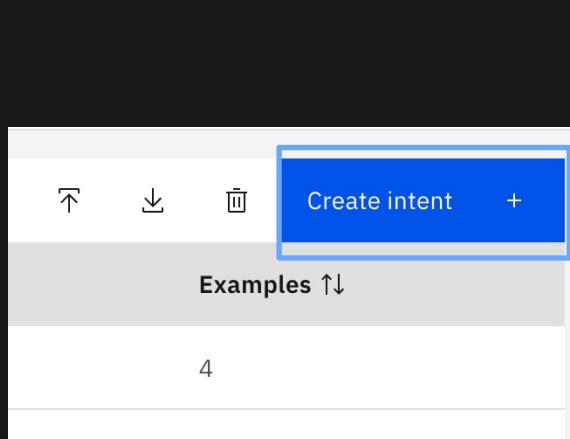
2. 입력 완료 후 상단 뒤로 가기 버튼을 누릅니다.

The screenshot shows the Watson Assistant interface for creating a new intent. The intent name is set to '#누구인가'. The description is optional and reads '누구인지 물어보는 질문을 인식합니다'. In the 'User example' section, there is a text input field containing the Korean phrase '누구나? 뭐하는 놈이야?' followed by a blue 'Add example' button. Below this, there is a summary section titled 'User examples (2)' containing two entries: '넌 누구야?' and '누구야?'. A blue box highlights the back arrow icon at the top left of the screen.

Watson Assistant 인텐트 생성 - 3

예약과 관련된 인텐트도 미리 생성 하겠습니다.

1. 우측 위에 Create Intent를 클릭 > 인텐트 명과 설명 입력
2. 예약 관련 예제 문구 입력 및 Add example 클릭
3. 예제 문구 추가로 더 입력하고, 상단 뒤로가기 버튼 클릭



This is a detailed view of the "Create intent" form. It includes the "Intent name" field with "#예약", the "Description (optional)" field with "예약하려는 의도 파악", and the "Create intent" button at the bottom. The entire form is set against a light grey background.

This screenshot shows the "User example" section of the form. It features a blue-bordered input field labeled "User example" with the placeholder "Type a user example here". Below it is a grey button labeled "Add example". Further down, there is a section titled "User examples (2)" with two checked checkboxes: "예약 하고 싶어" and "예약 해줘". The entire section is set against a light grey background.

Watson Assistant 다이얼로그에서 테스트

다이얼로그에 자기 소개 인텐트 인식 노드를 생성하고 테스트 하겠습니다.
아래의 순서를 따라 노드를 생성하고, 테스트 해보겠습니다.

The screenshot shows the Watson Assistant Dialog interface with various UI elements highlighted by blue boxes and red numbers:

- 1 Entities**: A sidebar menu item.
- 2 Add node**: A button in the top navigation bar.
- 3 누구인지 물어봄**: An intent node name input field.
- 4 인텐트 > 누구 관련 선택**: A large red annotation indicating the step to select the "Who" related intent choice.
- 5 예제 답변 입력**: A large red annotation indicating the step to enter example responses.
- 6 닫기**: A close button for a modal or dialog.
- 7 try it**: A "Try it" button in the top right corner.
- Try it out**: A button in the top right corner.
- Clear**: A button in the top right corner.
- Manage context**: A button in the top right corner.
- 누구야?**: A sample message in the conversation log.
- #누구인가**: A response variation in the conversation log.
- 저도 제가 누군지...제 이름은 뭘까요?**: A sample message in the conversation log.
- #누구인가**: A response variation in the conversation log.
- 제가 누굴까요? ㅋㅋ**: A sample message in the conversation log.
- Use the up key for most recent**: A text instruction at the bottom.
- Enter something to test your assistant**: A text input field at the bottom.

The main workspace displays three intents: 환영 인사 (welcome), 누구인지 물어봄 (#누구인가), and 기타 (anything_else). The "누구인지 물어봄" intent is selected. The "누구인지 물어봄" response variation is set to sequential, with two variations: "저도 제가 누군지...제 이름은 뭘까요?" and "제가 누굴까요? ㅋㅋ".

Watson Assistant 다이얼로그에서 테스트

다이얼로그에 예약 인텐트 인식 노드를 생성하고 응답 테스트 하겠습니다.
아래의 순서를 따라 노드를 생성하고, 테스트 해보겠습니다.

한국어 스킬

Intents

Add node Add child node Add folder

Entities

My Entities System Entities

Dialog

Options

Webhooks Disambiguation

Analytics

Overview User conversations

Versions

1 아래에 노드 추가

2 예약

Try it

Customize X

Try it out Clear Manage context

3 인텐트 > 언제 관련 선택

4 예제 답변 입력

5 try it

6 닫기

예약하고 싶어 #예약 언제 예약 하시겠어요?
날짜와 시간 말씀 주시겠어요?

Enter response variation

Use the up key for most recent

Enter something to test your assistant

Response variations are set to **sequential**. Set to random | multiline

Learn more

Try it out Clear Manage context

Watson Assistant 시스템 엔티티

엔티티는 날짜, 시간, 숫자 또는 음식/물건과 같은 종류를 커스텀으로 사전에 인식 지정할 수 있습니다.
사전에 학습된 시스템 엔티티를 모두 **on**으로 변경해 놓겠습니다.

한국어 스킬 Save new version Try it

Intents

Entities ^

My Entities

System Entities

Dialog

Options ^

Webhooks

Disambiguation

Analytics ^

Overview

User conversations

Versions

Name (5)	Description	Status
@sys-date	Extracts date mentions (금요일)	<input checked="" type="checkbox"/> On
@sys-number	Extracts numbers expressed as digits or written as numbers. (21)	<input checked="" type="checkbox"/> On
@sys-time	Extracts time mentions (아침 10시에)	<input checked="" type="checkbox"/> On
@sys-percentage	Extracts amounts including the number and the % sign. (15%)	<input checked="" type="checkbox"/> On
@sys-currency	Extracts currency values (amount and the unit). (20전)	<input checked="" type="checkbox"/> On

Watson Assistant 날짜/시간 시스템 엔티티 사용 - 1

예약 노드가 날짜/시간을 인식하게 하겠습니다.

1. 다이얼로그 탭 > 예약 노드 메뉴 > Child 노드 추가
2. 노드 명 추가 > Customize 클릭
3. Slot 확인 on / prompt for everything 체크 > Apply

The screenshot shows the Watson Assistant interface with the 'Dialog' tab selected (1). In the center, there are three nodes: '환영 인사' (welcome), '누구인지 물어봄' (#누구인가), and '예약' (#예약). A context menu is open over the '예약' node, with the 'Add child node' option highlighted (3).

The screenshot shows the 'Customize' dialog for the '날짜 시간 받기' node (4). It has tabs for 'Customize node' (selected) and 'Digressions'. Under 'Slots' (6), the 'Prompt for everything' checkbox is checked (7). At the bottom are 'Cancel' and 'Apply' buttons (8).

Watson Assistant 날짜/시간 시스템 엔티티 사용 - 2

예약 노드가 날짜/시간을 인식하게 하겠습니다.

1. 어시스턴트가 인식하는 부분에 날짜(@Sys-date) 및 Any 값 컨디션 선택
2. +를 누르고 or 선택
3. 두번째 인식하는 부분에 시간(@sys-time) 및 Any 값 컨디션 선택
4. 만약에 사용자가 날짜/시간을 입력 안할 경우를 대비하여 재질문 하는 부분을 설정 합니다.
5. “Check for” @sys-time 선택, “시간은요?” 와 같이 시간을 다시 물어보는 질문 설정
6. Add slot+ > “Check for” @sys-date 선택, “날짜는요?” 와 같이 날짜를 다시 물어보는 질문 설정

The screenshot shows the Watson Assistant configuration interface with two main panels. On the left, under 'If assistant recognizes', there are two slots: one for '@sys-date' (marked 1) and one for '@sys-time' (marked 3). A 'or' operator (marked 2) connects them. Below this, an 'Operator' section is set to 'any'. On the right, under 'Then check for', there are two entries. The first entry (marked 4) checks for '@sys-time' and saves it as '\$time', with the question '시간은요?' (Is it time?) and a required type. The second entry (marked 7) checks for '@sys-date' and saves it as '\$date', with the question '날짜는요?' (Is it date?) and a required type. At the bottom center, a button labeled 'Add slot +' (marked 6) is visible. A small circular icon with a gear and a plus sign is in the bottom right corner.

Check for	Save it as	If not present, ask	Type
4 1 @sys-time	\$time	시간은요?	Required
7 7 @sys-date	\$date	날짜는요?	Required

Then check for

0 Manage handle

If assistant recognizes

1 @sys-date 2 or 3 @sys-time +

Operator

any

Add slot +

6

7

8

9

Watson Assistant 날짜/시간 시스템 엔티티 사용 - 3

예약 노드를 완료 하겠습니다.

1. 사용자에게 입력 받은 시스템 엔티티 값은 \$date, \$time에 저장되어 있습니다.
2. 아래와 같이 응답 호출에 이용하겠습니다.
3. 상단에 노드 설정 닫기를 클릭 > Try it out

Assistant responds

Customize 2

Text

1 \$date, \$time 에 예약 하였습니다.

Enter response variation

Response variations are set to **sequential**. Set to [random](#) | [multiline](#)

[Learn more](#)

Try it out Clear Manage context 4

예약 하고 싶어

#예약

언제 예약 하시겠어요?

오후 2시에

Irrelevant

@sys-time:14:00:00

날짜는요?

10월 25일

Irrelevant

@sys-date:2021-10-25

2021-10-25, 14:00:00 에 예약 하였습니다.

Use the up key for most recent

Enter something to test your assistant



Watson Assistant 추가 기능

Demo 영상



