

AutoML 구현

백상현 우현수 백진우 문승현 박태건

목차

01 프로젝트 소개

- 프로젝트 목표
- AutoML 이란?
- 주제 선정 이유

02 프로젝트 과정

- Rule based 모델
- Auto-Sklearn based 모델

03 프로젝트 결과

- 사용한 데이터
- Azure Auto ML과의 성능 비교



1

프로젝트 소개

2

프로젝트 과정

3

프로젝트 결과

프로젝트 목표

=

AutoML 직접 구현



AutoML: 자동화된 + 머신러닝

기계 학습 모델 개발의 시간이 많이 걸리는 반복적인 작업을 자동화하는 프로세스

1. 광범위한 프로그래밍 + 통계 지식 없이 머신러닝 솔루션 구현
2. 시간 및 리소스 절약
3. 데이터 사이언스 모범 사례 활용
4. Agile 문제 제공 - 해결



1. 프로젝트 소개

대표적인 AutoML



Google's AutoML



1. 프로젝트 소개



직접 만들어보자!

직접 구현해보며 AutoML에 대해 제대로 이해하자

단순히 사용하는 user에서



‘**변화**’를 줄 수 있는 **provider**로

1

프로젝트 소개



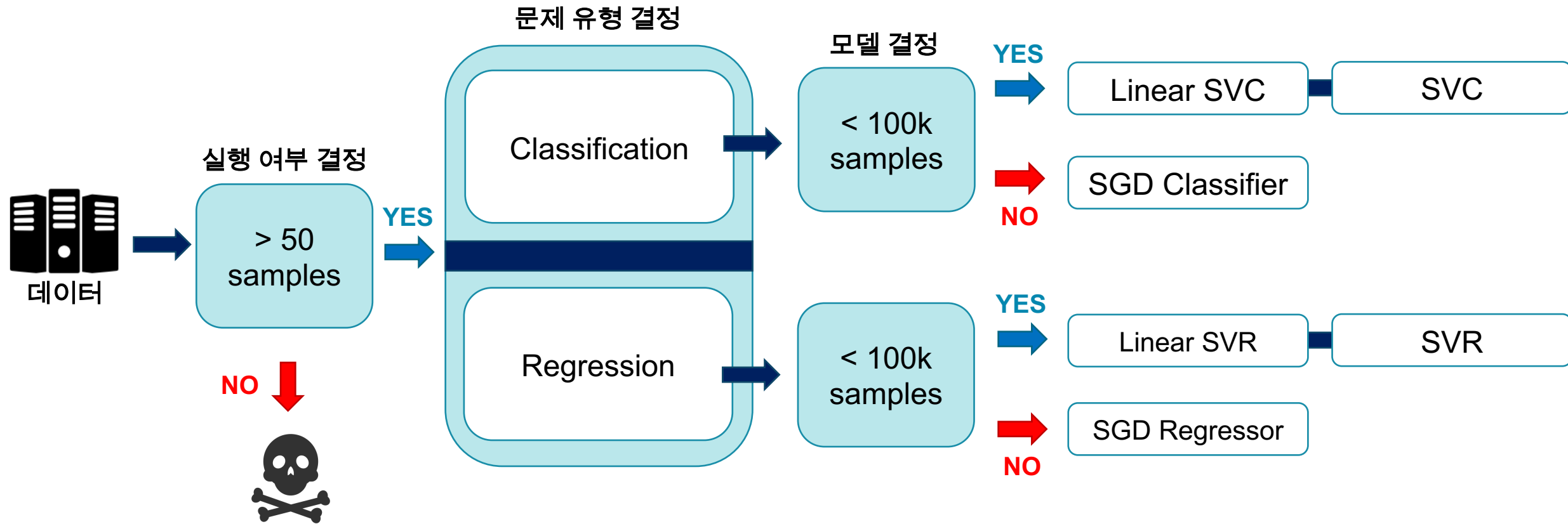
2

프로젝트 과정

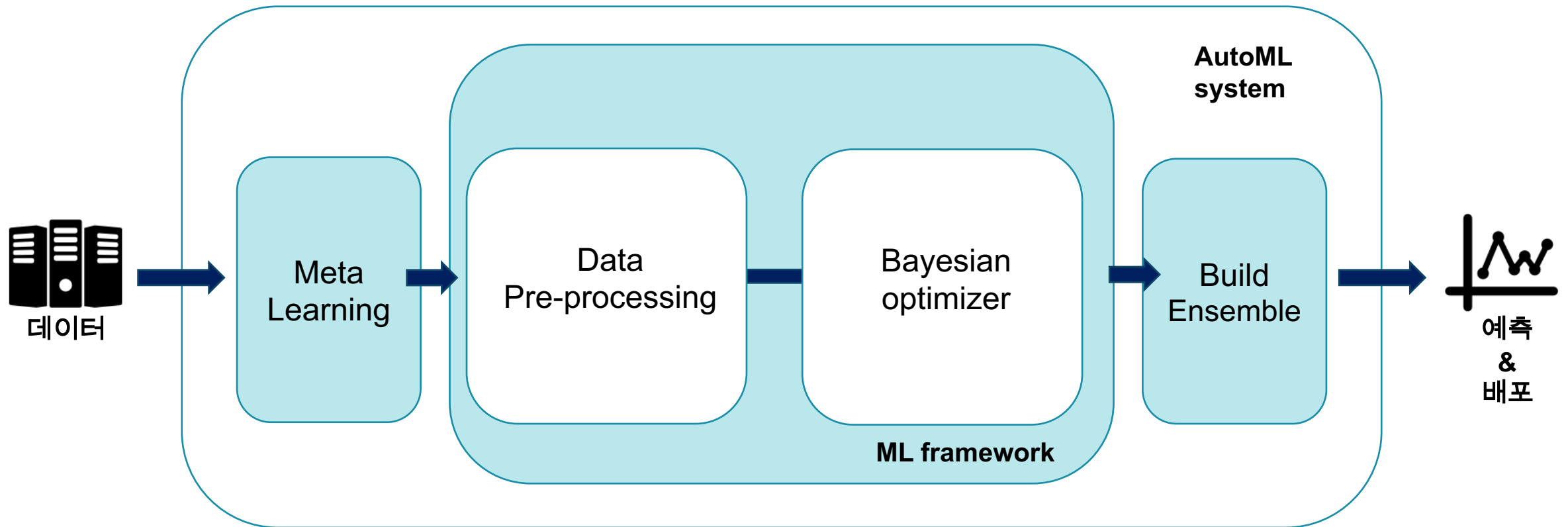
3

프로젝트 결과

Rule based Auto ML Pipeline



Auto-Sklearn based Auto ML Pipeline



Meta Learning



데이터



목표: 최적의 Model 선정



Open ML의 수많은 dataset들의 meta feature 보유 (API 제공)



새로운 데이터의 meta feature 계산하여



Open ML 내 datasets들의 meta feature 비교 (cosine similarity 사용)



데이터에 맞는 최적의 model 5개 선정



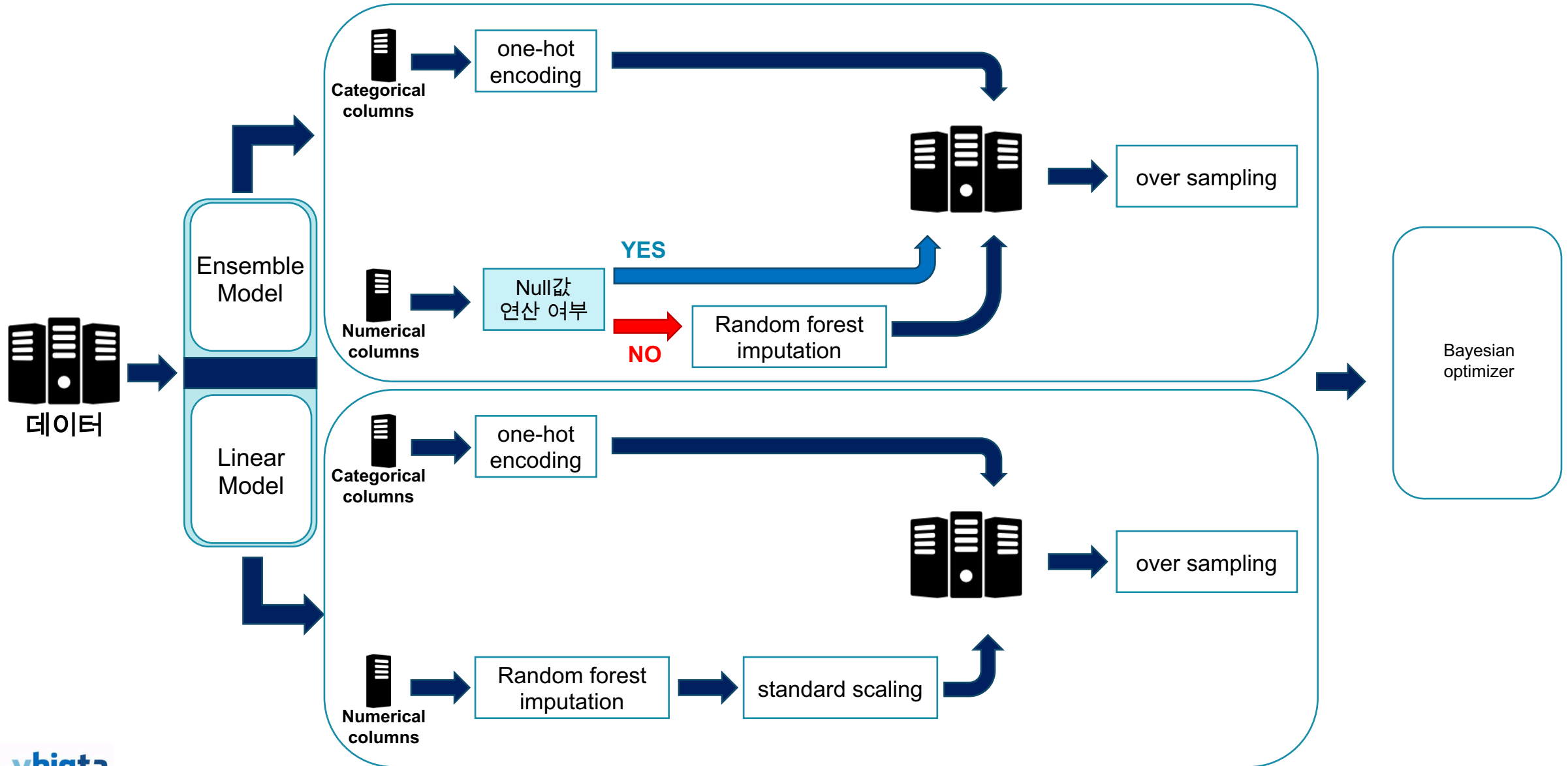
Data
Pre-processing

Meta Learning

코드 스크린샷 추가

2. 프로젝트 과정

Data pre-processing



2. 프로젝트 과정

Data pre-processing

```
def preprocess4ensemble(df):
    print("preprocess4ensemble")
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]
    X_num = X.select_dtypes(include="number")
    X_cat = X.select_dtypes(include="object")

    #Check if all right
    if not(X_num.shape[1]+X_cat.shape[1] == X.shape[1]) or not(X_num.shape[0] == X_cat.shape[0] and X_cat.shape[0] == X.shape[0]):
        print("categorical and numerical seperation operation has a problem")

    for cat_col in X_cat.columns:
        X_cat = pd.concat([X_cat, pd.get_dummies(X_cat[cat_col], dummy_na=False)], axis=1)
        del X_cat[cat_col]

    imputer = MissForest()
    X_imputed = pd.DataFrame(imputer.fit_transform(X_num))
    X_imputed.columns = X_num.columns
    del X_num

    X = pd.concat([X_imputed, X_cat], axis=1)
    ros = RandomOverSampler(random_state=42)
    X_res, y_res = ros.fit_resample(X, y)
    return X_res, y_res

def preprocess4xgb(df):
    print("preprocess4xgb")
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]
    X_num = X.select_dtypes(include="number")
    X_cat = X.select_dtypes(include="object")

    #Check if all right
    if not(X_num.shape[1]+X_cat.shape[1] == X.shape[1]) or not(X_num.shape[0] == X_cat.shape[0] and X_cat.shape[0] == X.shape[0]):
        print("categorical and numerical seperation operation has a problem")

    for cat_col in X_cat.columns:
        X_cat = pd.concat([X_cat, pd.get_dummies(X_cat[cat_col], dummy_na=False)], axis=1)
        del X_cat[cat_col]

    #
    imputer = MissForest()
    #
    X_imputed = pd.DataFrame(imputer.fit_transform(X_num))
    #
    X_imputed.columns = X_num.columns
    #
    del X_num

    def oversampling(X_train, y_train):
        rus = RandomOverSampler(return_indices=True)
        X_resampled, y_resampled, idx_resampled = rus.fit_resample(X_train, y_train)
        X_resampled = pd.DataFrame(X_resampled)
        y_resampled = pd.Series(y_resampled)
        X_resampled.columns = X_train.columns
        return X_resampled, y_resampled

    X = pd.concat([X_imputed, X_cat], axis=1)
    print(type(X))
```

```
ros = RandomOverSampler(random_state=42)
X, y = X.fillna(10000000000), y
X, y = oversampling(X, y)
X, y = X.replace(10000000000, np.nan), y.replace(10000000000, np.nan)

return X, y

def preprocess4normal(df):
    print("preprocess4normal")
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]
    X_num = X.select_dtypes(include="number")
    X_cat = X.select_dtypes(include="object")

    #Check if all right
    if not(X_num.shape[1]+X_cat.shape[1] == X.shape[1]) or not(X_num.shape[0] == X_cat.shape[0] and X_cat.shape[0] == X.shape[0]):
        print("categorical and numerical seperation operation has a problem")

    for cat_col in X_cat.columns:
        X_cat = pd.concat([X_cat, pd.get_dummies(X_cat[cat_col], dummy_na=False)], axis=1)
        del X_cat[cat_col]

    #Impute nan
    imputer = MissForest()
    X_imputed = pd.DataFrame(imputer.fit_transform(X_num))
    X_imputed.columns = X_num.columns
    del X_num

    #Scaling
    scaled_features = StandardScaler().fit_transform(X_imputed.values)
    scaled_features_df = pd.DataFrame(scaled_features, index=X_imputed.index, columns=X_imputed.columns)

    X = pd.concat([scaled_features_df, X_cat], axis=1)
    ros = RandomOverSampler(random_state=42)
    X_res, y_res = ros.fit_resample(X, y)
    return X_res, y_res

def preprocess(df, algo_type):
    if "RandomForest" in str(algo_type).split("(")[0] or "GradientBoosting" in str(algo_type).split("(")[0]:
        return preprocess4ensemble(df)
    elif "XGB" in str(algo_type).split("(")[0]:
        return preprocess4xgb(df)
    else:
        return preprocess4normal(df)
```


Bayesian optimizer



전처리 된
데이터



목표: Model 별 최적의 hyper-parameter 선정



TPE 사용



Meta learning 단계에서 선별 된 5가지 Model 별
최적의 hyper-parameter 선정

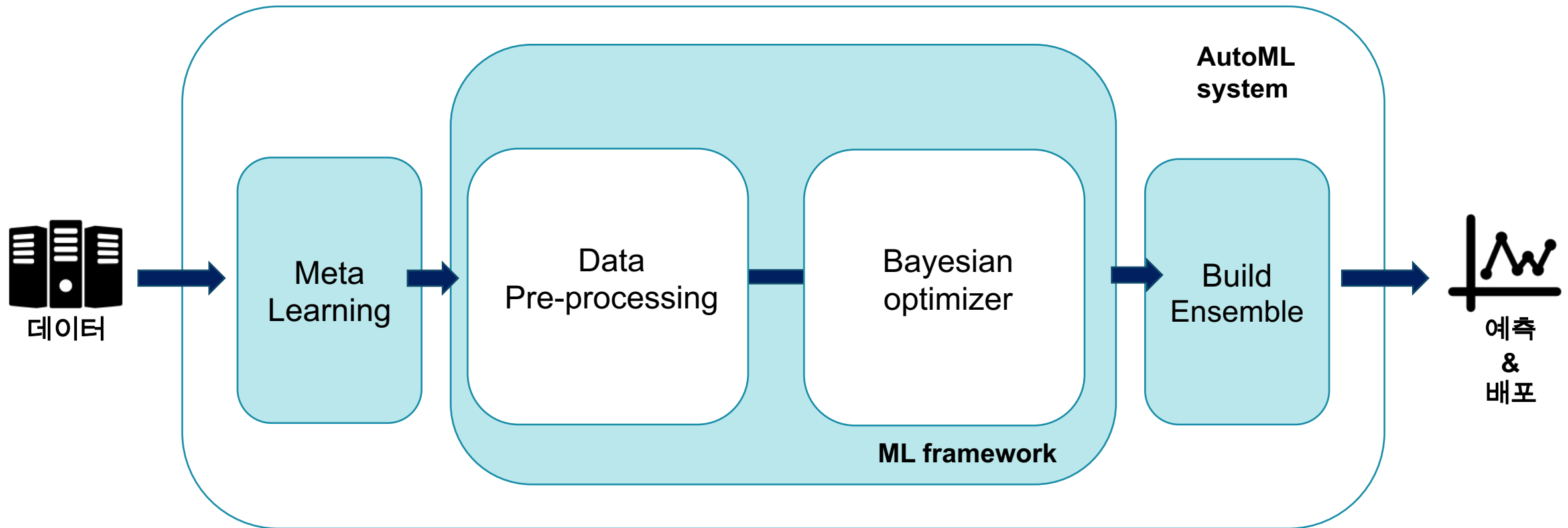


Build
Ensemble

Bayesian optimizer

코드 스크린샷 추가

Auto-Sklearn based Auto ML Pipeline



1

프로젝트 소개

2

프로젝트 과정



3

프로젝트 결과

Azure의 AutoML과의 성능 비교!

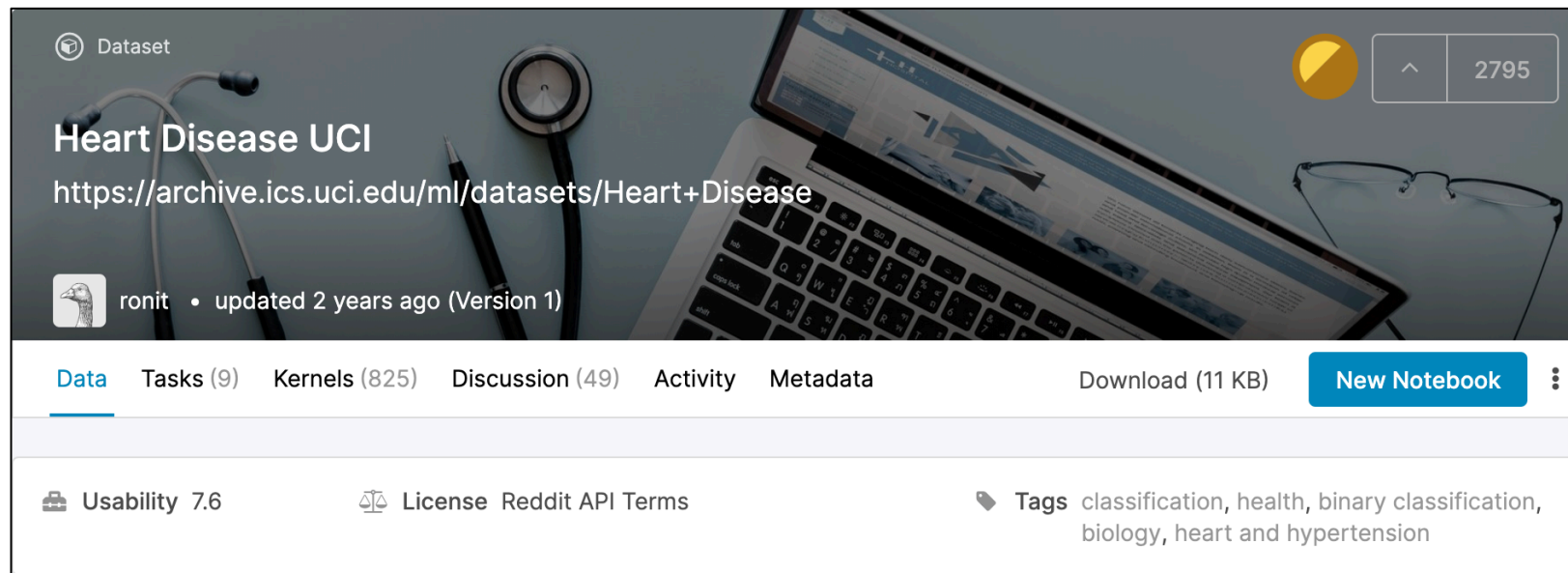


Google's AutoML



1. 프로젝트 소개

Dataset



The screenshot shows the Kaggle page for the 'Heart Disease UCI' dataset. The background image features a laptop, a stethoscope, and a pair of glasses. The dataset title 'Heart Disease UCI' is prominently displayed, along with its URL: <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>. The uploader is 'ronit', and it was updated 2 years ago (Version 1). The page includes navigation tabs for 'Data', 'Tasks (9)', 'Kernels (825)', 'Discussion (49)', 'Activity', and 'Metadata'. A 'Download (11 KB)' link and a 'New Notebook' button are also visible. At the bottom, there is a 'Usability' score of 7.6, a 'License' section with 'Reddit API Terms', and a 'Tags' section with 'classification, health, binary classification, biology, heart and hypertension'.

Dataset

Heart Disease UCI

<https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

ronit • updated 2 years ago (Version 1)

Data Tasks (9) Kernels (825) Discussion (49) Activity Metadata Download (11 KB) [New Notebook](#)

Usability 7.6 License Reddit API Terms Tags classification, health, binary classification, biology, heart and hypertension

Heart Disease Classification

Azure AutoML vs YBIGTA AutoML

Model details	Visualizations	Explanations	Logs	Outputs
Properties				
Algorithm name				
VotingEnsemble				
Primary metric				
AUC weighted				
Score				
0.920180294350186				
Sdk version				
1.0.76				
Deploy status				
No deployment yet				



THANK YOU