

# AutoML 활용한 AI 모델 탐색

[https://github.com/MIRAE-Laboratory/Lectures/tree/main/AIF\\_AutoML](https://github.com/MIRAE-Laboratory/Lectures/tree/main/AIF_AutoML)

## 서 호 건

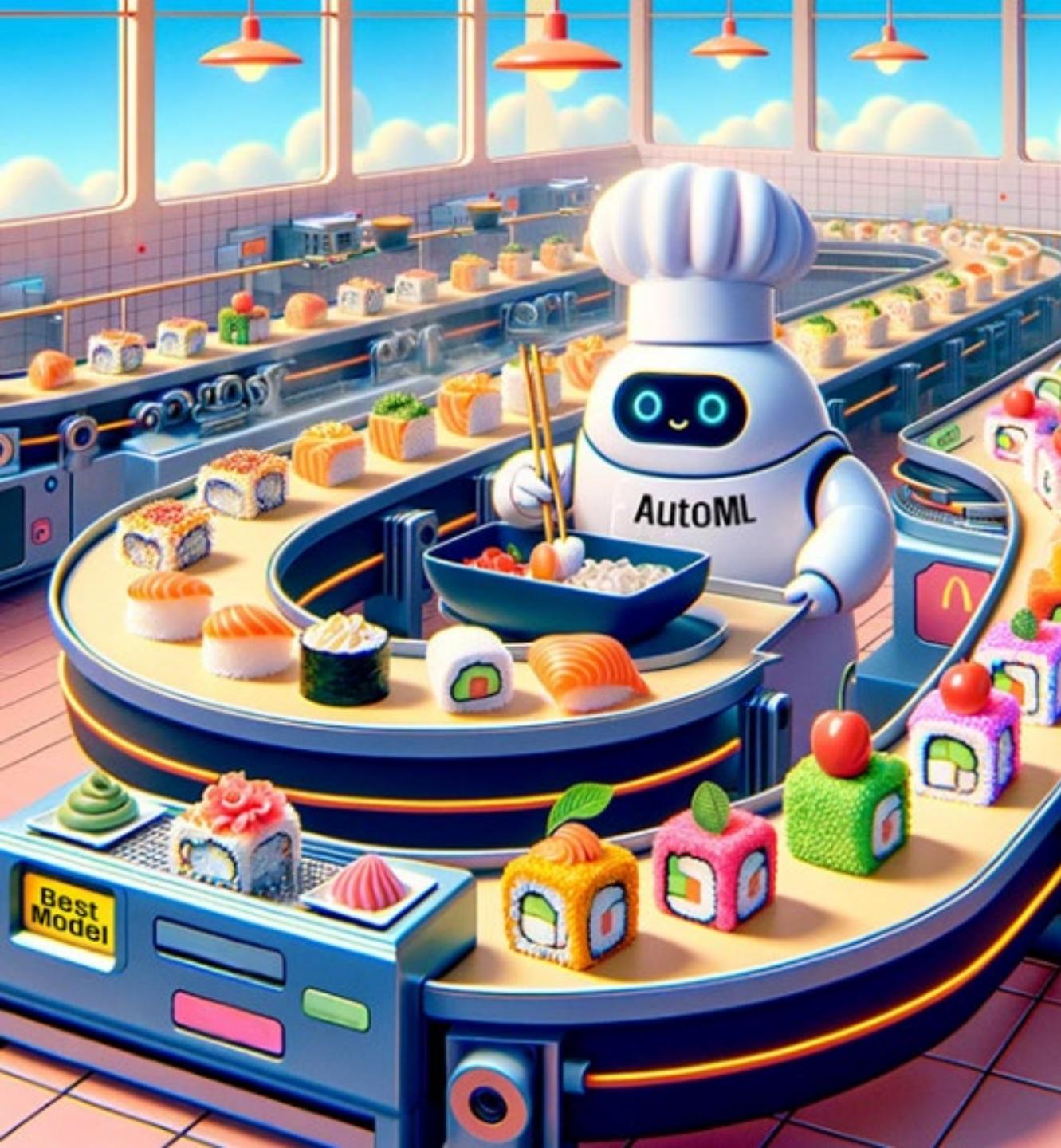
국가연구소대학교(UST) :: 인공지능 전공 :: 부교수

한국원자력연구원(KAERI) :: 인공지능응용연구실 :: 선임연구원

한국비파괴검사학회(KSNT) :: 대외협력이사

hogeony@hogeony.com

# Contents



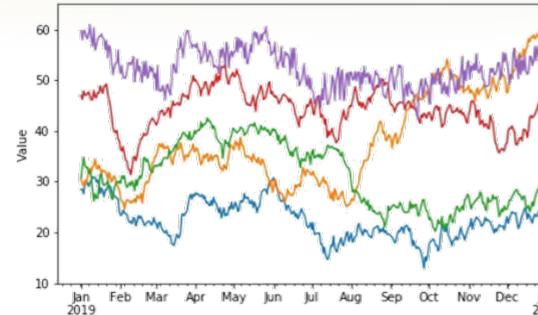
- 0. What is AutoML?**
- 1. 이미지 데이터에 대한 AutoML 활용**
- 2. 신호 데이터에 대한 AutoML 활용**
- 3. AutoML 모델을 활용한  
최적의 입력 데이터 추정**

# Image Datasets vs. Sequential Datasets (a.k.a. Time-series Data, Signal)

그림/사진



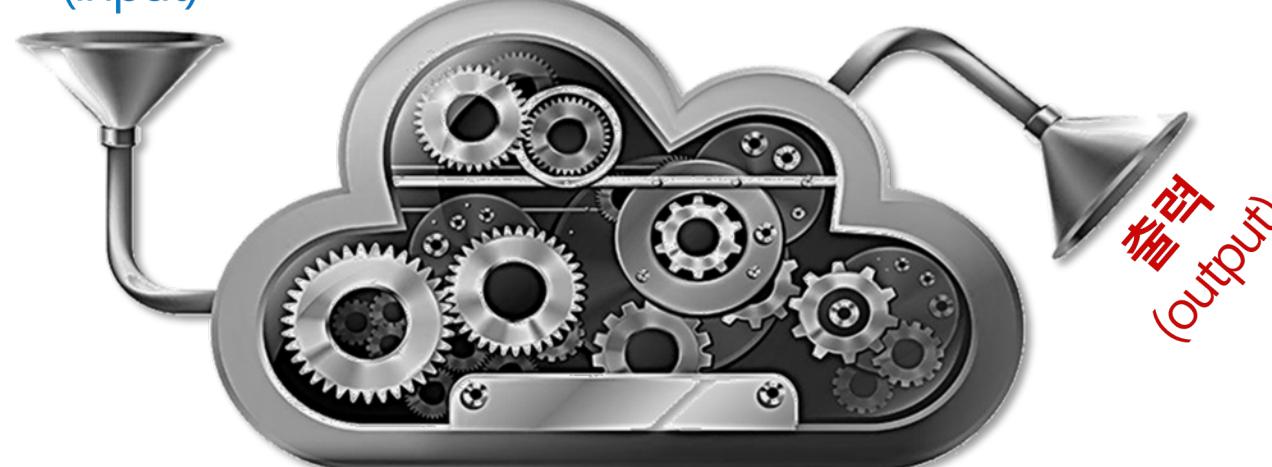
신호



표

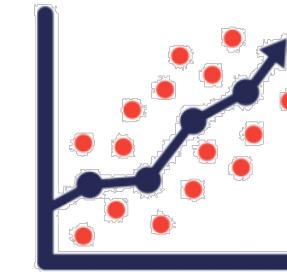
A	B	C	D	E	F
Passenger	Survived	Pclass	Name	Sex	Age
1	0	3	Braund, M	male	22
2	1	1	Cumings, female		38
3	1	3	Heikkinen female		26
4	1	1	Futrelle, female		35
5	0	3	Allen, Mr. male		35
6	0	3	Moran, M	male	
7	0	1	McCarthy, male		54
8	0	3	Palsson, N	male	2
9	1	3	Johnson, female		27
10					

입력  
(input)

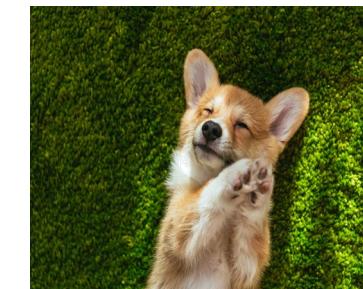


머신러닝 / 딥러닝 모델

회귀 (regression)

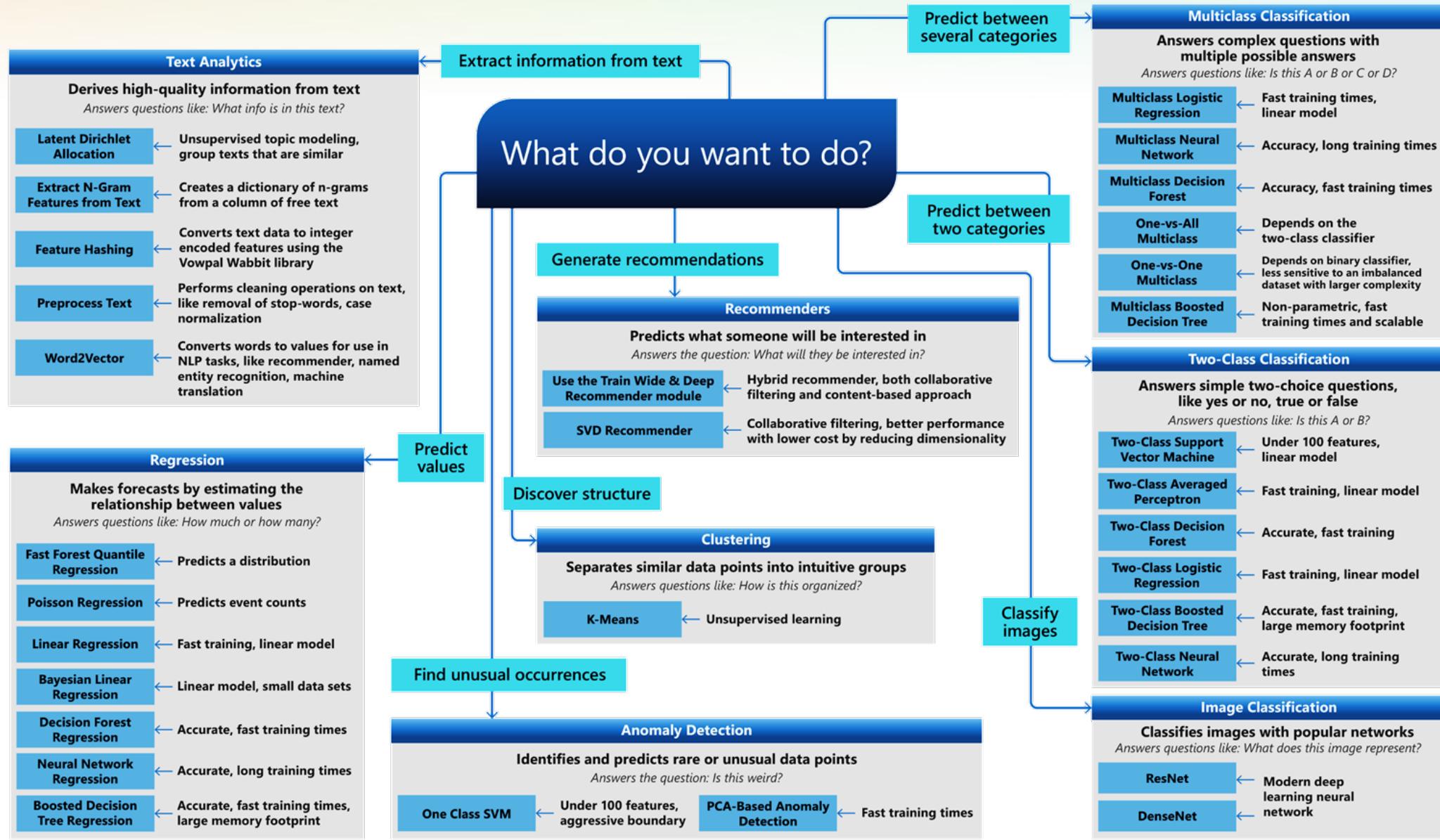


분류 (classification)



Dog:	<div style="width: 96%; background-color: #6aa84f;"></div>	96%
Cat:	<div style="width: 29%; background-color: #d9eaf7;"></div>	29%
Duck:	<div style="width: 2%; background-color: #e9967a;"></div>	2%
Bird:	<div style="width: 0%; background-color: #d9eaf7;"></div>	0%

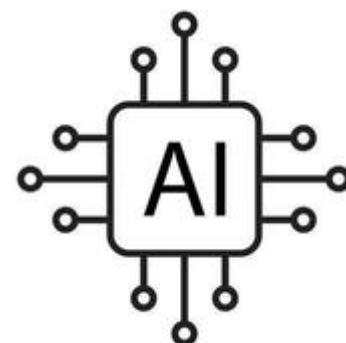
# Various kinds of Machine Learning Models



# What is AutoML (Automated Machine Learning)?

**AutoML :: Automated Machine Learning (자동화된 기계 학습)**

## AI Model Development



AI Model

## ML/DL Pipeline



Data Preprocessing

Feature Engineering

Model Selection

Hyperparameter Tuning



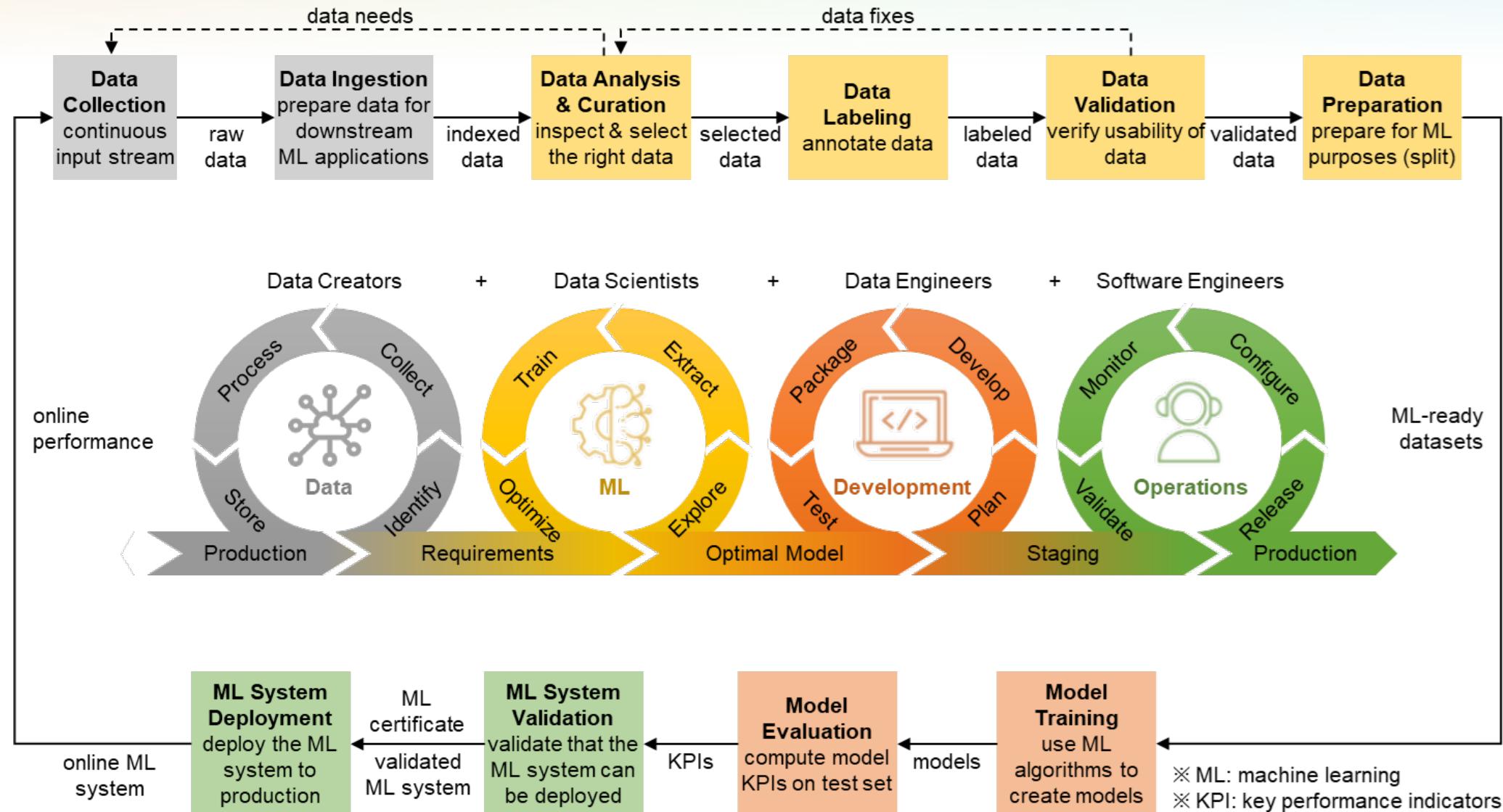
## AutoML



AutoML



# Machine Learning Operations (MLOps)



# Machine Learning / Deep Learning Pipeline

## Data Preprocessing

- Missing Values
- Data Types
- Anomalies
- Text Encoding
- Data Partition

## Feature Extraction

- Numeric
- Discrete
- Textual / Images
- Time-series (sequential)
- Cross-features

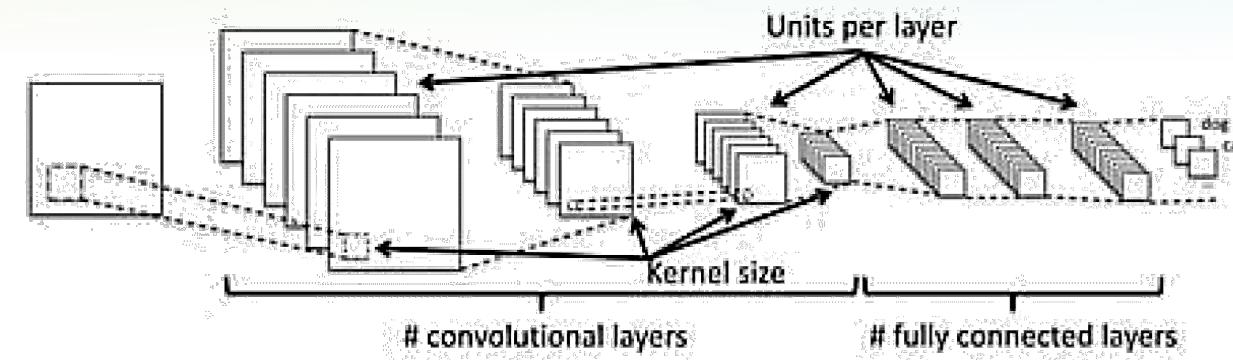
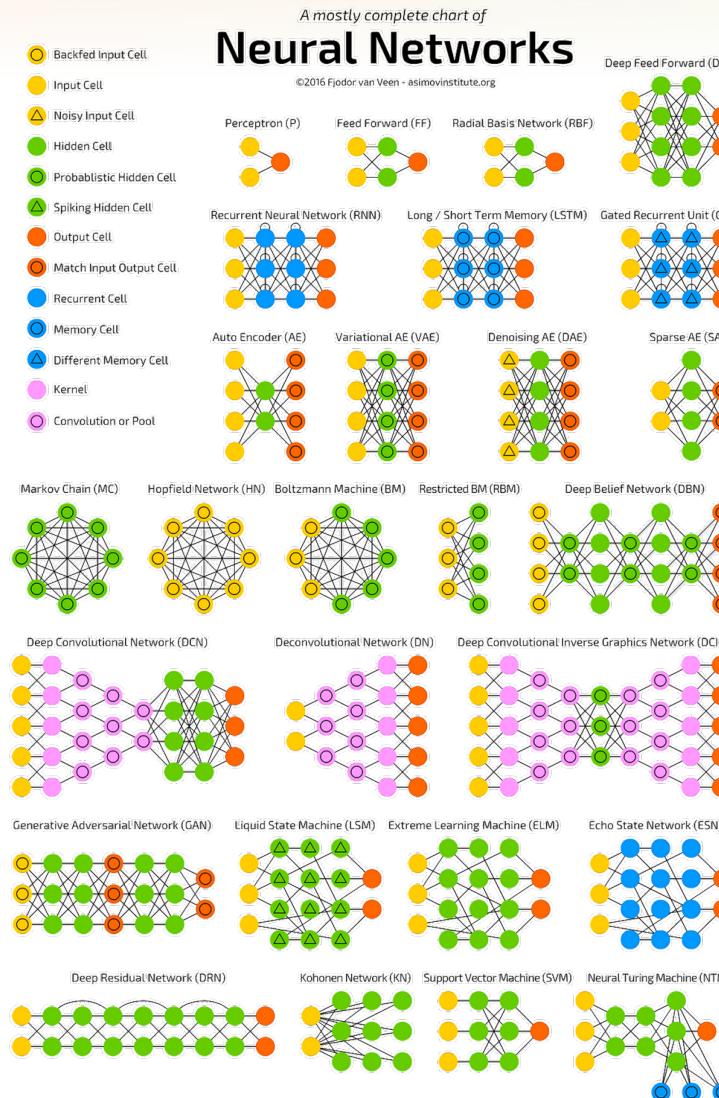
## Feature Selection

- Feature Importance
- Random Projection
- Linear Projection
- Nonlinear Projection
- Reduction

## Model Selection & Hyperparameter Tuning

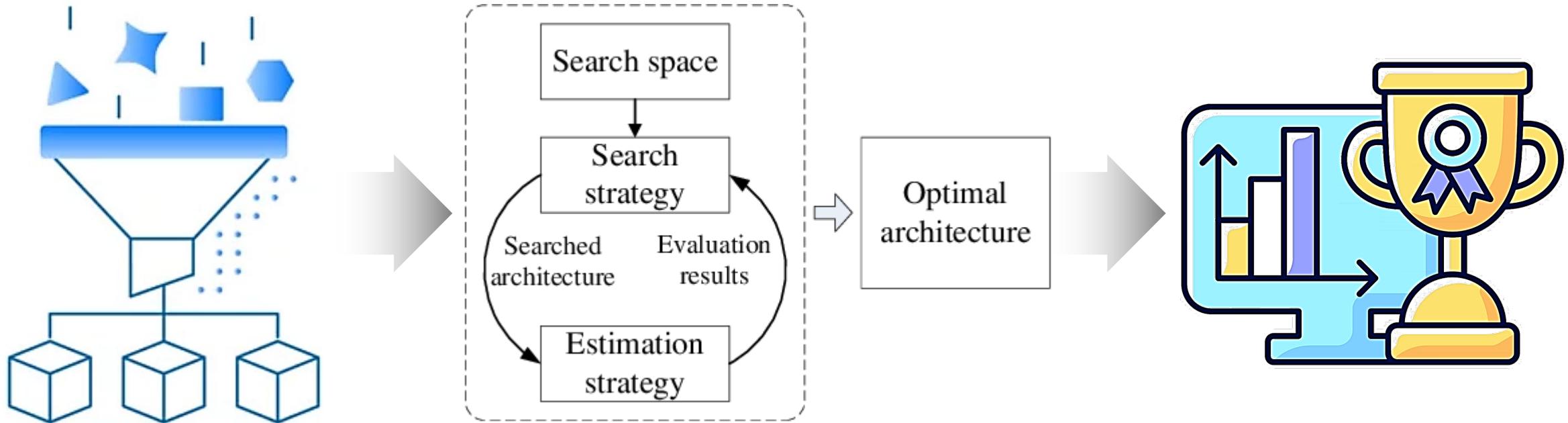
- Grid/Random Search
- Bayesian Optimization
- Reinforcement Learning
- Neural Architecture Search (NAS)

# Neural Architecture Search (NAS)



- Model Selection (ex: FNN, CNN, RNN, Transformer)
- Hyperparameters
  - optimization algorithm, learning rate, ...
  - batch size, epochs, early stopping, ...
  - regularization, dropout, ...
  - data augmentation, ...

# Automated Machine Learning (AutoML)



**Data Preprocessing**

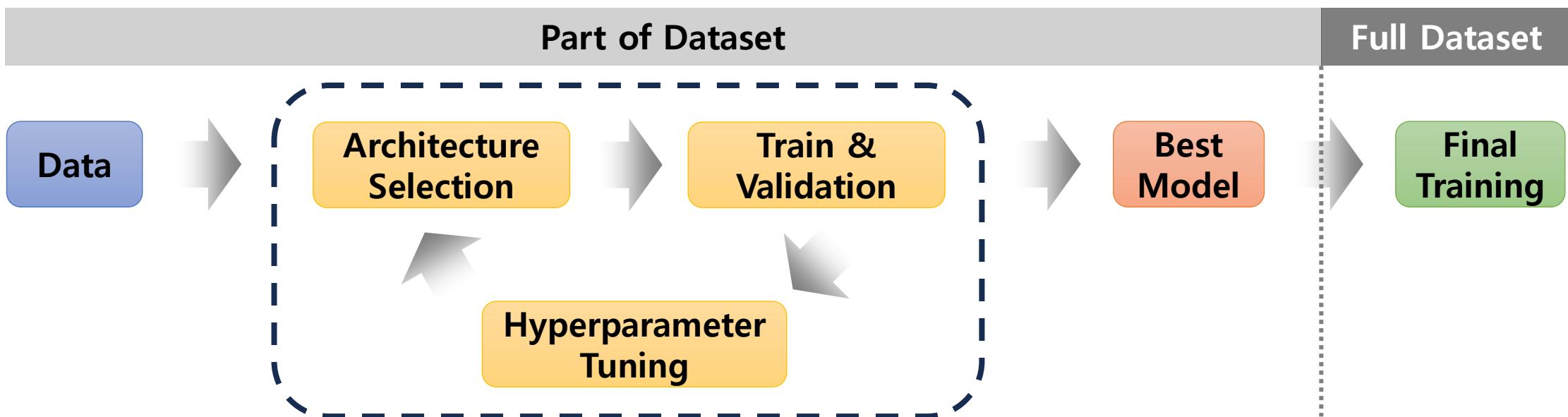
**Model Selection &  
Optimization**

**Best Model Evaluation**

# Various AutoML Frameworks

AutoML Framework	Price	Applicable Dataset Types	Latest Release	Key Features	Link
Google Cloud AutoML	Commercial	Image, Tabular, Text, Time Series	-	Custom models; comprehensive; user-friendly	<a href="https://cloud.google.com/automl">https://cloud.google.com/automl</a>
Azure AutoML	Commercial	Image, Tabular, Text, Time Series	-	Comprehensive; supports various model types; easy deployment	<a href="https://azure.microsoft.com/ko-kr/products/machine-learning/">https://azure.microsoft.com/ko-kr/products/machine-learning/</a>
DataRobot	Commercial	Image, Tabular, Text, Time Series	-	User-friendly; non-expert friendly; customizable	<a href="https://www.datarobot.com/">https://www.datarobot.com/</a>
AWS AutoML	Commercial	Image, Tabular, Text, Time Series	-	Offers binary classification, regression, multiclass classification	<a href="https://aws.amazon.com/ko/machine-learning/automl/">https://aws.amazon.com/ko/machine-learning/automl/</a>
Ludwig	Free	Image, Tabular, Text, Time Series	2023.12.21.	Uber's toolbox; no-code training and testing	<a href="https://github.com/ludwig-ai/ludwig/">https://github.com/ludwig-ai/ludwig/</a>
AutoGluon	Free	Image, Tabular, Text	2023.11.30.	AWS library; versatile; user-friendly	<a href="https://github.com/autogluon/autogluon">https://github.com/autogluon/autogluon</a>
AutoKeras	Free	Image, Tabular, Text	2023.01.29.	Neural networks; Bayesian optimization; hyperparameter search	<a href="https://github.com/keras-team/autokeras">https://github.com/keras-team/autokeras</a>
Auto-PyTorch	Free	Image, Tabular, Text	2022.07.24.	Architecture, hyperparameter search for PyTorch	<a href="https://github.com/automl/Auto-PyTorch">https://github.com/automl/Auto-PyTorch</a>
ML Box	Free	Tabular, Text	2019.07.26.	Fast data processing; feature selection; high accuracy	<a href="https://github.com/AxeldeRomblay/MLBox">https://github.com/AxeldeRomblay/MLBox</a>
MLJAR	Free	Tabular	2024.01.08.	Browser-accessible; supports CUDA, Python, TensorFlow	<a href="https://github.com/mljar/mljar-supervised">https://github.com/mljar/mljar-supervised</a>
TPOT2 Alpha	Free	Tabular	2023.12.22.	Tree-based Pipeline Optimization Tool	<a href="https://github.com/EpistasisLab/tpot2">https://github.com/EpistasisLab/tpot2</a>
Auto-SKLearn	Free	Tabular	2023.02.13.	Sci-kit-learn based; simplifies ML process	<a href="https://github.com/automl/auto-sklearn">https://github.com/automl/auto-sklearn</a>
H2O AutoML	Free	Tabular	2023.01.12.	Gradient-boosted machines; deep learning; automated tasks	<a href="https://github.com/h2oai/h2o-3">https://github.com/h2oai/h2o-3</a>
AutoWeka	Free	Tabular	2022.03.07.	Optimizes WEKA's algorithms; hyperparameter tuning	<a href="https://github.com/automl/autoweka">https://github.com/automl/autoweka</a>
TransmogrifAI	Free	Tabular	2020.06.12.	Scala-based, SparkML; data cleansing, model selection	<a href="https://github.com/salesforce/TransmogrifAI">https://github.com/salesforce/TransmogrifAI</a>
SMAC	Free	General (Flexible to Various Types)	2019.12.05	Hyperparameter optimization; model domain information	<a href="https://github.com/automl/SMAC3">https://github.com/automl/SMAC3</a>
AutoFolio	Free	General (Flexible to Various Types)	2019.10.17.	Algorithm selection; portfolio-based optimization	<a href="https://github.com/automl/AutoFolio">https://github.com/automl/AutoFolio</a>
RoBO	Free	General (Flexible to Various Types)	2016.05.24.	Bayesian Optimization; complex ML tasks	<a href="https://github.com/automl/RoBO">https://github.com/automl/RoBO</a>

# AutoKeras Pipeline



# AutoKeras Tutorials (<https://autokeras.com/tutorial/overview>)

AK AutoKeras

Search

 GitHub  
1.1.0 ⭐ 9k ⚡ 1.4k

AutoKeras

Home

Installation

Tutorials

Overview

Image Classification

Image Regression

Text Classification

Text Regression

Structured Data Classification

Structured Data Regression

TimeSeriesForecaster

Multi-Modal and Multi-Task

Customized Model

Export Model

Load Data from Disk

FAQ

Extensions

Docker

Contributing Guide

Documentation

Benchmarks

About

Table of contents

Learning resources

Installation

Community

Contributing Code

Cite this work

Acknowledgements

AutoKeras: An AutoML system based on Keras. It is developed by [DATA Lab](#) at Texas A&M University. The goal of AutoKeras is to make machine learning accessible to everyone.

## Learning resources

- A short example.

```
import autokeras as ak

clf = ak.ImageClassifier()
clf.fit(x_train, y_train)
results = clf.predict(x_test)
```

- [Official website tutorials](#).
- The book of [Automated Machine Learning in Action](#).
- The LiveProjects of [Image Classification with AutoKeras](#).

Five-Project Series

Image Classification with

# AutoKeras Tutorials ([https://autokeras.com/tutorial/image\\_classification](https://autokeras.com/tutorial/image_classification))

AutoKeras

Search

GitHub 1.1.0 9k 1.4k

AutoKeras

Home

Installation

Tutorials

- Overview
- Image Classification**
- Image Regression
- Text Classification
- Text Regression
- Structured Data Classification
- Structured Data Regression
- TimeSeriesForecaster
- Multi-Modal and Multi-Task
- Customized Model
- Export Model
- Load Data from Disk

FAQ

Extensions

- Docker
- Contributing Guide

Documentation

Benchmarks

About

## Image Classification

[View in Colab](#) [GitHub source](#)

```
!pip install autokeras
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist

import autokeras as ak
```

## A Simple Example

The first step is to prepare your data. Here we use the MNIST dataset as an example

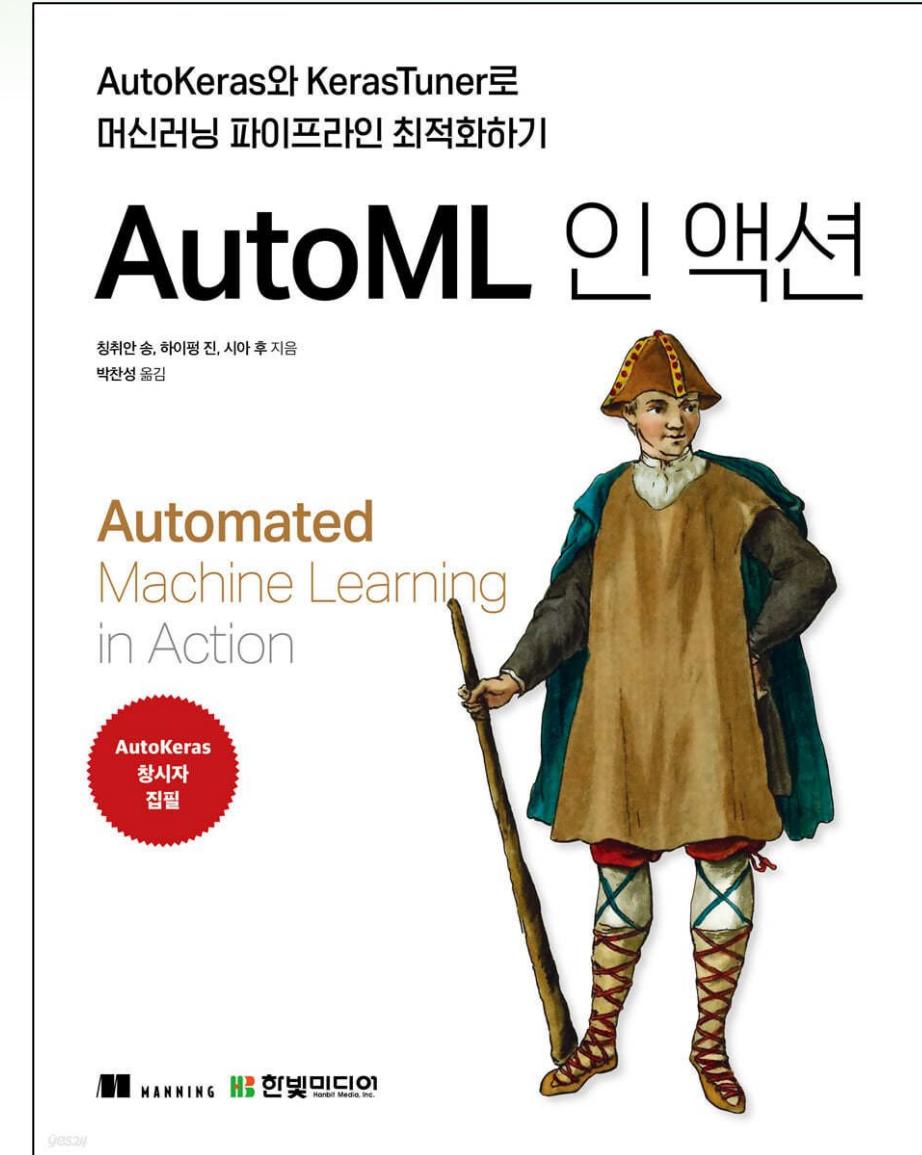
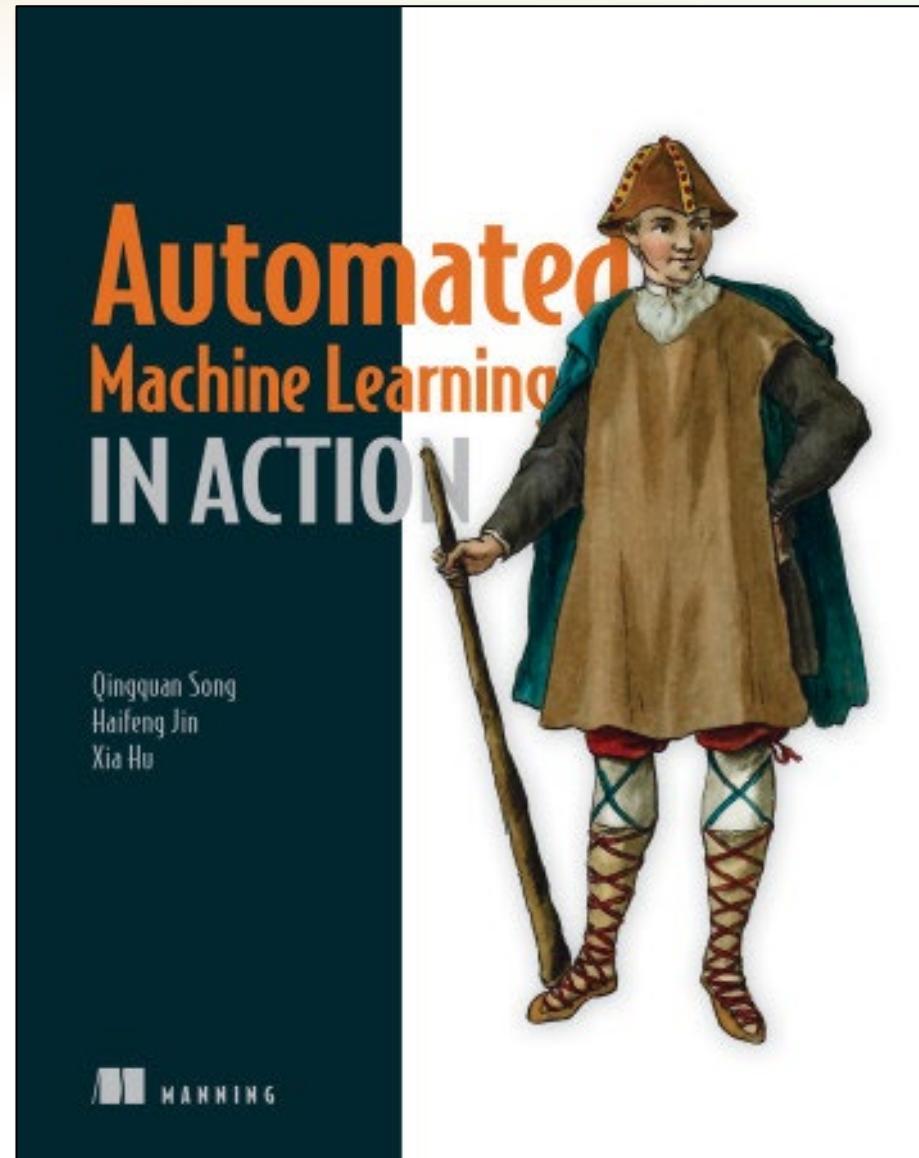
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape) # (60000, 28, 28)
print(y_train.shape) # (60000,)
print(y_train[:3]) # array([7, 2, 1], dtype=uint8)
```

The second step is to run the ImageClassifier. It is recommended have more trials for more complicated datasets. This is just a quick demo of MNIST, so we set max\_trials to 1. For the same reason, we set epochs to 10. You can also leave the epochs unspecified for an adaptive number of epochs.

Table of contents

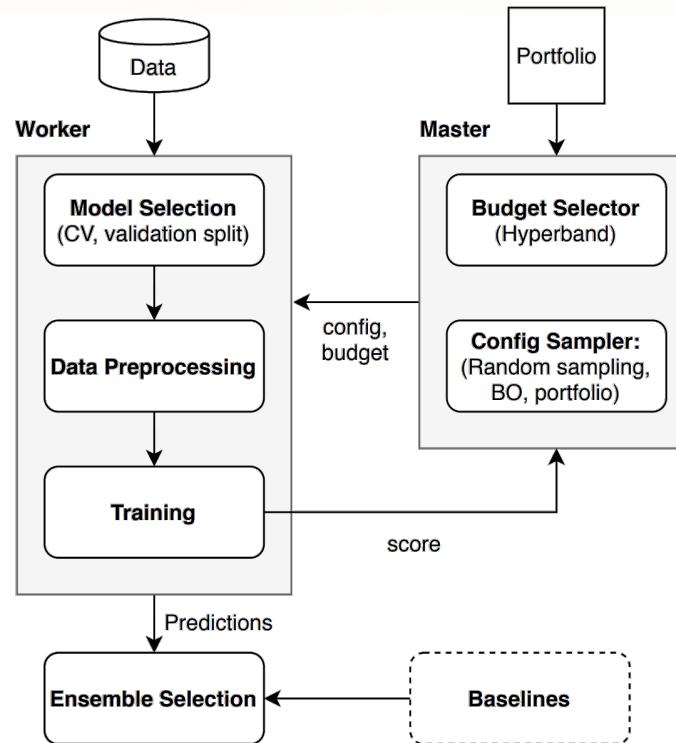
- A Simple Example
- Validation Data
- Customized Search Space
- Data Format
- Reference

# Automated Machine Learning in Action

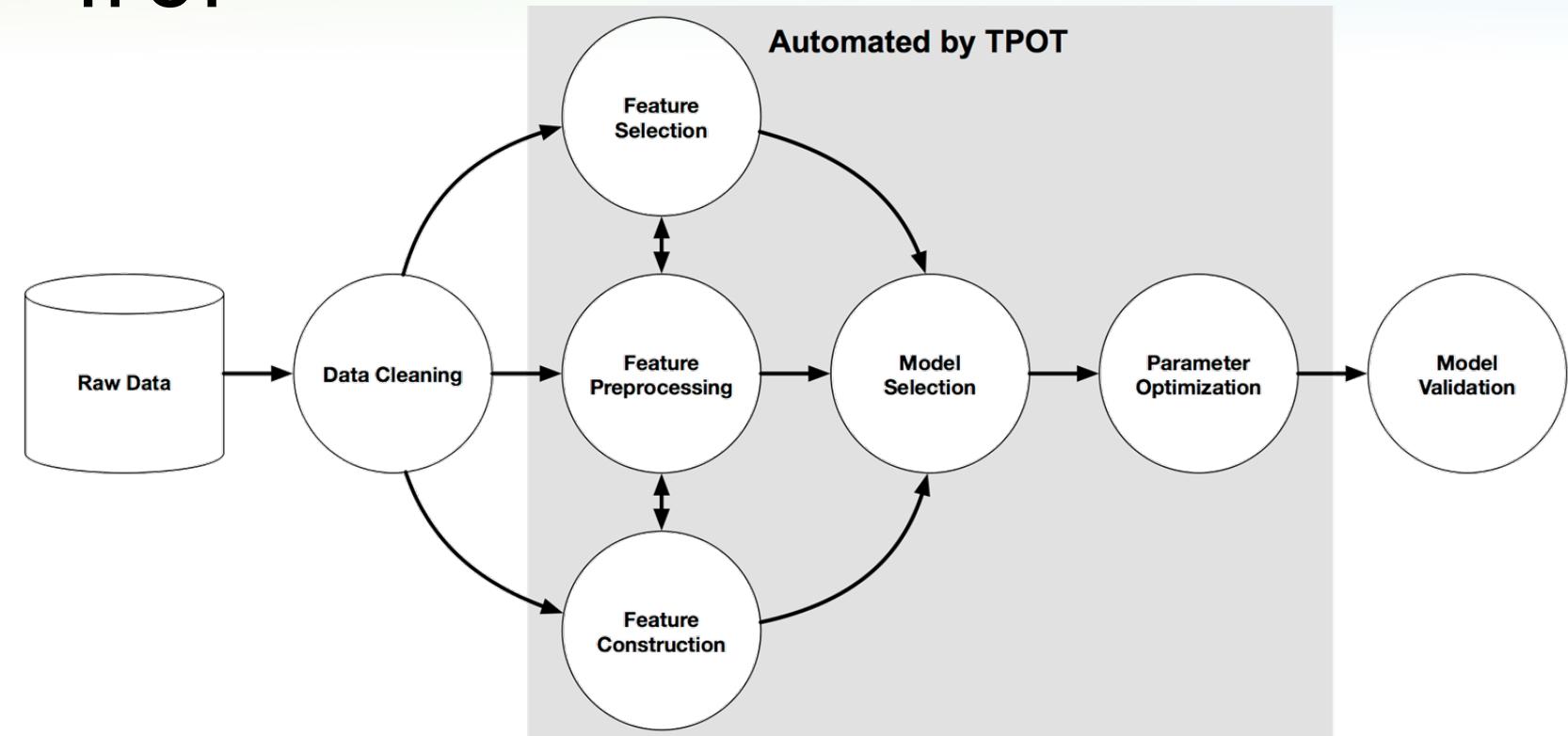


# Various AutoML Frameworks & Pipelines

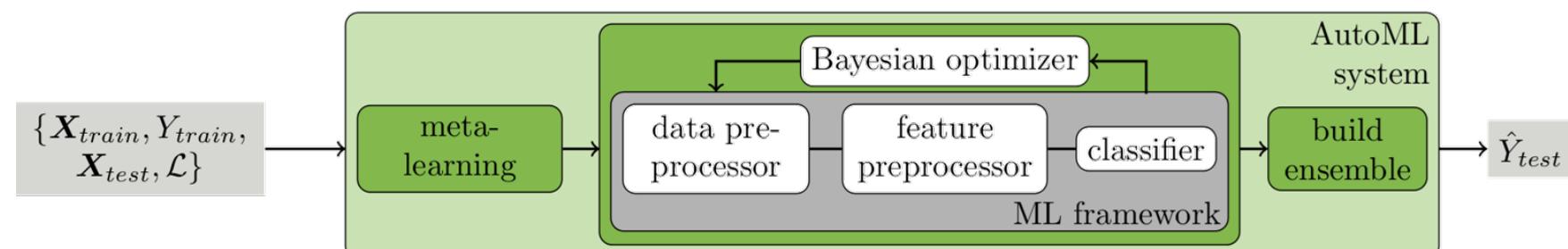
## Auto-PyTorch



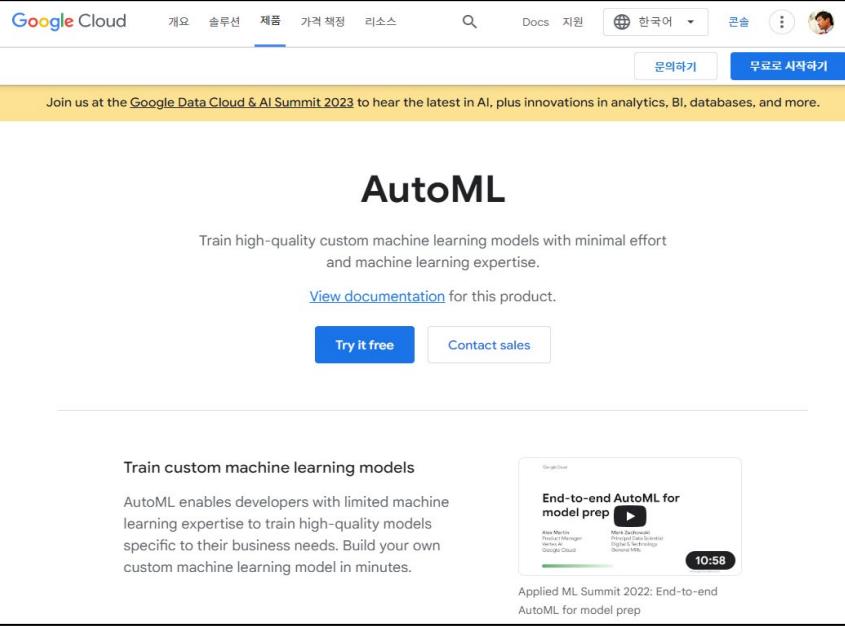
## TPOT



## Auto-SKLearn



# Google Cloud AutoML

	<p><b>Sight</b></p> <p><b>AutoML Image</b></p> <p>Derive insights from object detection and image classification, in the cloud or at the edge. <a href="#">Try it now.</a></p> <ul style="list-style-type: none"> <li>✓ Use REST and RPC APIs</li> <li>✓ Detect objects, where they are, and how many</li> <li>✓ Classify images using custom labels</li> <li>✓ Deploy ML models at the edge</li> </ul> <p><b>AutoML Video</b></p> <p>Enable powerful content discovery and engaging video experiences. <a href="#">Try it now.</a></p> <ul style="list-style-type: none"> <li>✓ Annotate video using custom labels</li> <li>✓ Streaming video analysis</li> <li>✓ Shot change detection</li> <li>✓ Object detection and tracking</li> </ul>	<p><b>Structured data</b></p> <p><b>AutoML Tabular</b></p> <p>Automatically build and deploy state-of-the-art machine learning models on structured data. <a href="#">Try it now.</a></p> <ul style="list-style-type: none"> <li>✓ Handles wide range of tabular data primitives</li> <li>✓ Easy to build models</li> <li>✓ Easy to deploy and scale models</li> </ul> <p><b>Vertex AI Tabular Workflows</b></p> <p>A glassbox and managed AutoML pipeline that lets you see and interpret each step in the model building and deployment process.</p> <ul style="list-style-type: none"> <li>✓ The ease of AutoML is now combined with greater control</li> <li>✓ Pick and choose parts of the process AutoML handles versus what you engineer</li> <li>✓ Comfortably train datasets over a terabyte, without sacrificing accuracy</li> </ul> <p><b>Language</b></p> <p><b>AutoML Text</b></p> <p>Reveal the structure and meaning of text through machine learning. <a href="#">Try it now.</a></p> <ul style="list-style-type: none"> <li>✓ Integrated REST API</li> <li>✓ Custom entity extraction</li> <li>✓ Custom sentiment analysis</li> <li>✓ Large dataset support</li> </ul> <p><b>AutoML Translation</b></p> <p>Dynamically detect and translate between languages. <a href="#">Try it now.</a></p> <ul style="list-style-type: none"> <li>✓ Integrated REST and gRPC APIs</li> <li>✓ Supports 50 language pairs</li> <li>✓ Translate with custom models</li> </ul>
--	--	---

# Google Cloud AutoML

 Google Cloud 무료로 사용해 보기

## 총 2단계 중 1단계 계정 정보



Hogeon Seo

hogeony@hogeony.com

[계정 전환](#)



좋은 소식입니다. 무료 체험판 크레딧 \$100.00를 추가로 받아 총 \$400.00를 사용할 수 있습니다. 가입 완료 후 24시간 내에 크레딧이 지급됩니다.

국가

대한민국



### 모든 Google Cloud 제품에 액세스

Firebase, Google Maps API 등을 포함해 앱, 웹사이트, 서비스를 구축하고 실행하는 데 필요한 모든 기능을 이용할 수 있습니다.

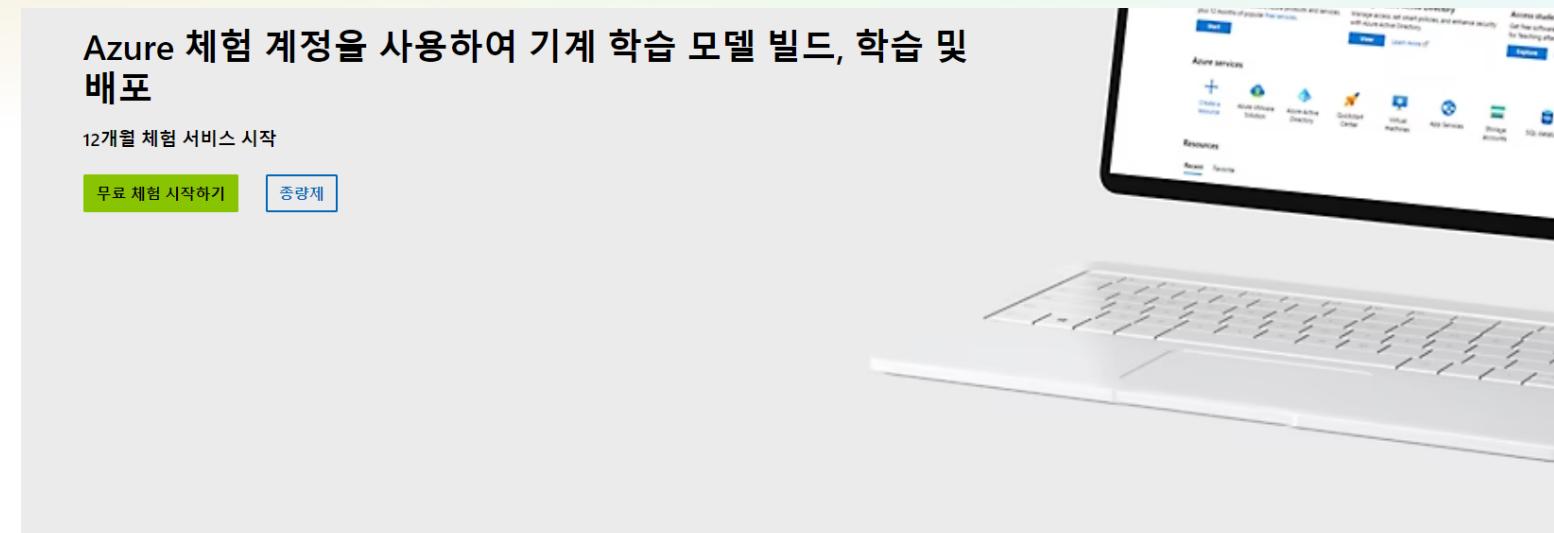
### \$300의 무료 크레딧

90일간 사용할 수 있는 \$300 크레딧으로 Google Cloud를 실제 작업에 활용할 수 있습니다.

### 무료 체험판 종료 후 자동 청구되지 않음

신용카드를 요청하는 이유는 자동 가입을 방지하기 위해서입니다. 신용카드나 체크카드를 사용하는 경우 일반 계정을 직접 활성화하지 않는 한 요금이 청구되지 않습니다.

# Azure AutoML



## Azure 체험 계정을 사용하여 기계 학습 모델 빌드, 학습 및 배포

12개월 체험 서비스 시작

무료 체험 시작하기

증량제

12개월 동안 별도의 비용  
없이 인기 서비스 제공

[모든 서비스 보기](#)

상시 무료로 제공되는 다  
른 55가지 이상의 서비스

[모든 서비스 보기](#)

최소 USD200 Azure 크레  
딧부터

[체험 서비스 외에 30일간 크레딧을 사용할 수 있습니다.](#)

### Azure로 수행할 수 있는 작업



#### 자동화된 기계 학습

적합한 알고리즘과 하이퍼 매개 변수를 더 빠르게 식별하세요.



#### 기계 학습을 위한 DevOps

통합형 CI/CD, 기계 학습 파이프라인 및 모델 관리를 사용하여 생산  
성을 높이세요.



#### 도구 중립적 Python SDK

원하는 Python 환경 및 기계 학습 프레임워크를 사용하세요.

# Azure AutoML

## AWS 제품 요금

AWS 제품 및 서비스 검색

필터 지우기

1-7 (7)

정렬 기준  ▼

프리 티어 유형	기계 학습	무료 평가판	기계 학습	12개월 무료	기계 학습
<input checked="" type="checkbox"/> 12개월 무료	<b>Amazon Augmented AI</b>	ML 예측의 인적 검토를 손쉽게 구현	<b>Amazon Comprehend</b>	텍스트에서 인사이트 및 관계 파악	<b>Amazon Lex</b>
<input checked="" type="checkbox"/> 언제나 무료					음성 및 텍스트 챗봇 구축
<input type="checkbox"/> 평가판					
제품 카테고리	기계 학습	12개월 무료	기계 학습	12개월 무료	기계 학습
<input type="checkbox"/> 분석	<b>Amazon Polly</b>	텍스트를 생생한 음성으로 변환	<b>Amazon Rekognition</b>	이미지 및 비디오 분석	<b>Amazon Transcribe</b>
<input type="checkbox"/> 애플리케이션 통합					자동 음성 인식
<input type="checkbox"/> 블록체인					
<input type="checkbox"/> 클라우드 금융 관리					
<input type="checkbox"/> 컴퓨팅					
<input type="checkbox"/> 컨테이너					
<input type="checkbox"/> 데이터베이스					
<input type="checkbox"/> 개발자 도구					
<input type="checkbox"/> 최종 사용자 컴퓨팅					
<input type="checkbox"/> 프런트 엔드 웹 및 모바일					
<input type="checkbox"/> 사물 인터넷(IoT)					
<input checked="" type="checkbox"/> 기계 학습	<b>Amazon Translate</b>	자연스럽고 유창한 언어 번역			

# Key Point for Successful AI Application / R&D

## DATA

- **Type & Shape & Count**
- **Label Distribution**
- **Preprocessing**
  - : Imputation, Encoding, Sampling, Scaling, Filtering, Noise Reduction
- **Data Split**
  - : Train:Valid:Test Ratio, Method

## INPUT & OUTPUT

- **Feature**
  - : Engineering, Importance
- **Augmentation**
  - : Flip, Rotation, CutMix, MixUp, SMOTE, ADASYN
- **Modality Transformation**
  - : signal  $\Leftrightarrow$  image

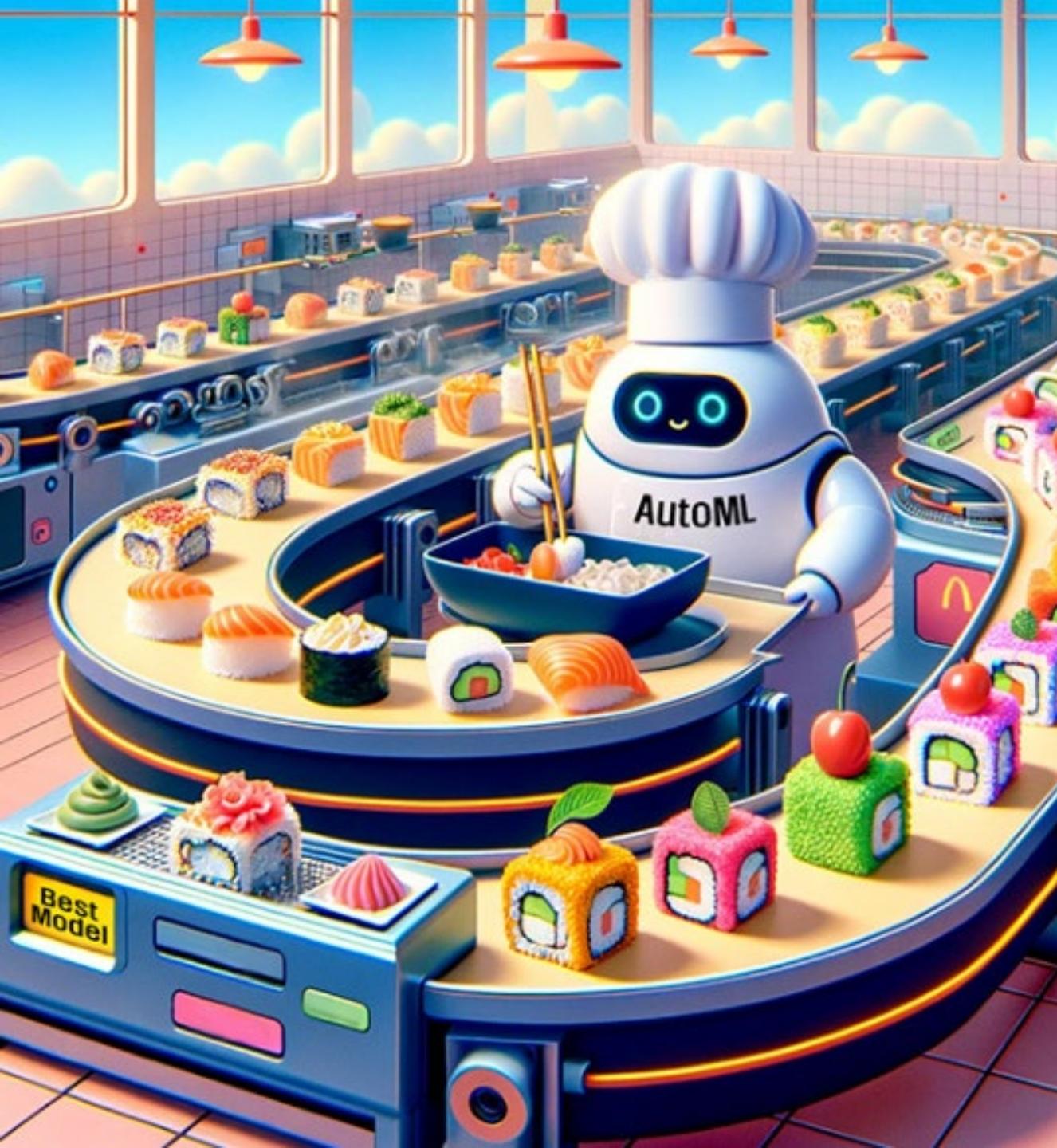
## RESULT

- **Loss Function**
- **Evaluation Metrics**
  - : Accuracy, MSE, mAP, IoU
- **Cross Validation**
  - : K-fold Cross Validation
- **Explanation (XAI)**
- **Inverse Problem Solving**
  - : Bayesian Optimization

# Key Point for Successful AutoML

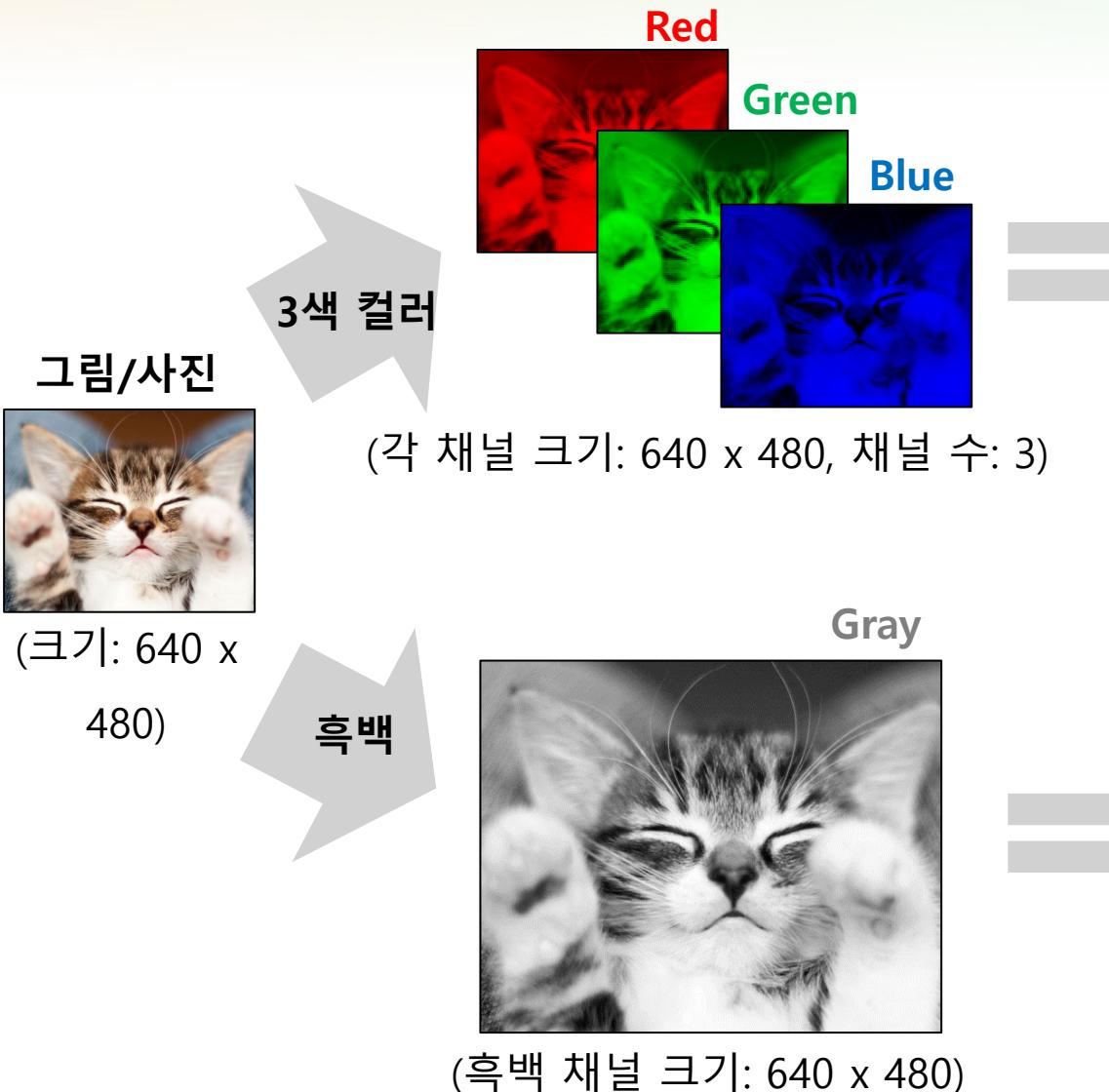
- 1) 학습을 위한 충분한 데이터양
- 2) 데이터 전처리, 입력, 출력, 평가에 대한 정의
- 3) 고성능 컴퓨팅 파워 or 넉넉한 시간
- 4) 성능 향상을 위한 도메인 지식
- 5) 도메인 지식을 반영하기 위한 코딩 실력

# Contents



0. What is AutoML?
1. 이미지 데이터에 대한 AutoML 활용
2. 신호 데이터에 대한 AutoML 활용
3. AutoML 모델을 활용한  
최적의 입력 데이터 추정

# Image Data Characteristics



Red	Green	Blue							
180	178	177	177	177	178	181	182	...	183
179	179	177	177	177	179	182	184	...	181
178	179	177	177	177	179	182	184	...	181
177	177	176	177	177	179	182	184	...	181
176	177	175	177	177	179	182	184	...	181
178	177	174	174	174	174	174	174	...	151
176	174	176	174	174	174	174	174	...	148
174	174	178	174	174	174	174	174	...	146
...	...	...	...	...	...	...	...	...	146
178	180	179	178	178	178	178	178	...	145
162	159	161	161	161	161	161	161	...	145
160	160	161	161	161	161	161	161	...	139
...	...	...	...	...	...	...	...	...	127
162	165	16	16	16	16	16	16	...	112
142	142	146	148	150	151	146	133	...	112
...	...	...	...	...	...	...	...	...	111
145	147	147	149	150	147	143	136	126	140

(배열 크기: 3 x 640 x 480)

Gray										
186	185	185	185	185	185	185	188	188	...	189
185	186	185	185	185	185	186	188	190	...	187
185	185	185	185	185	185	187	188	189	...	186
184	184	184	184	184	184	187	189	189	...	186
184	184	183	183	183	185	188	188	188	...	186
185	185	182	183	183	187	188	188	187	...	182
184	182	184	187	187	188	188	188	184	...	172
182	182	185	187	188	189	186	176	176	...	154
...	...	...	...	...	...	...	...	...	...	141
183	185	187	188	189	189	184	173	153	141	

(배열 크기: 1 x 640 x 480)

# AutoML Pipeline for Image Classification

## Load Dataset

- Dataset  
(train & test folders)
- Plot Image Data

## Preprocessing

- rgb\_to\_grayscale
- resize\_and\_fill
- resize\_and\_crop
- flip
- rotate
- mixup
- cutmix

## AutoML

- epoch
- validation\_split
- augmentation\_block
- retraining

## Evaluation

- save best model
- load best model

# AutoML Pipeline for Image Classification

 **AutoKeras**

Search GitHub

Table of contents

- Load Images from Disk
- Load Texts from Disk
- Load Data with Python Generators
- Reference

**Load Data from Disk**

[View in Colab](#) [GitHub source](#)

```
!pip install autokeras
```

```
import os
import shutil

import numpy as np
import tensorflow as tf

import autokeras as ak
```

**Load Images from Disk**

If the data is too large to put in memory all at once, we can load it batch by batch into memory from disk with `tf.data.Dataset`. This [function](#) can help you build such a `tf.data.Dataset` for image data.

First, we download the data and extract the files.

```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
local_file_path = tf.keras.utils.get_file(
    origin=dataset_url, fname="image_data", extract=True
)
# The file is extracted in the same directory as the downloaded file.
local_dir_path = os.path.dirname(local_file_path)
# After check manually, we know the extracted data is in 'flower_photos'.
data_dir = os.path.join(local_dir_path, "flower_photos")
print(data_dir)
```

The directory should look like this. Each folder contains the images in the same class.

```
flower_photos/
daisy/
dandelion/
roses/
sunflowers/
tulips/
```

The directory should look like this. Each folder contains the images in the same class.

```
flower_photos/
daisy/
dandelion/
roses/
sunflowers/
tulips/
```

We can split the data into training and testing as we load them.

```
batch_size = 32
img_height = 180
img_width = 180

train_data = ak.image_dataset_from_directory(
    data_dir,
    # Use 20% data as testing data.
    validation_split=0.2,
    subset="training",
    # Set seed to ensure the same split when loading testing data.
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
)

test_data = ak.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
```

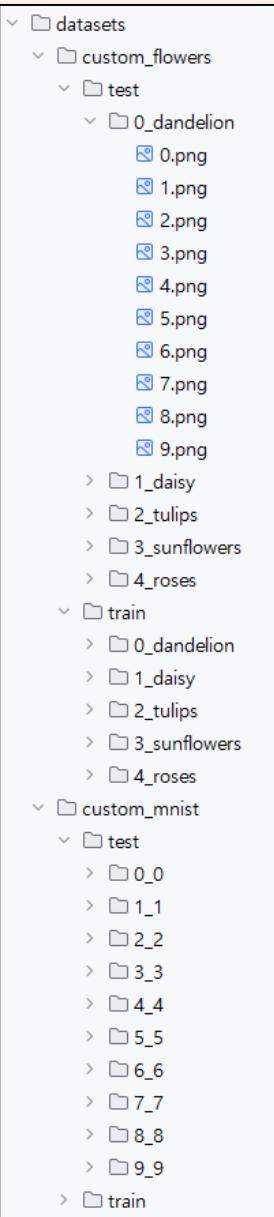
# Image Data Characteristics

File Explorer view showing image datasets:

- datasets
  - face
  - flower
    - test
      - 0\_dandelion
      - 1\_daisy
      - 2\_tulips
    - train
      - 0\_dandelion
      - 1\_daisy
      - 2\_tulips
    - sigmean
    - sigpeak

Category	Image Type	File Name	Description
flower	Dandelion	18999743619_c ec3f39bee.jpg	Close-up of a dandelion seed head.
		19004688463_1 2a8423109.jpg	Close-up of a dandelion seed head.
		19064700925_b 93d474e37.jpg	Close-up of a dandelion seed head.
		19067907051_1 6d530c7d2.jpg	Close-up of a dandelion seed head.
		19397467530_1 e8131a7cf.jpg	Close-up of a dandelion seed head.
		19426575569_4 b53c0b726.jpg	Close-up of a dandelion seed head.
flower	Dandelion	19551343814_4 8f764535f_m.jpg	Close-up of a dandelion seed head.
		19551343954_8 3bb52f310_m.jpg	Close-up of a dandelion seed head.
		19586799286_b eb9d684b5.jpg	Close-up of a dandelion seed head.
		19593576916_f 5a083d7fe_n.jpg	Close-up of a dandelion seed head.
		19599413676_fc 9ee2640e.jpg	Close-up of a dandelion seed head.
		19600096066_6 7dc941042.jpg	Close-up of a dandelion seed head.
flower	Dandelion	19812060274_c 19961979110_fc	Close-up of a dandelion seed head.
		20165867412_fc	Close-up of a dandelion seed head.
		20456824132_b	Close-up of a dandelion seed head.
		20754920332_5	Close-up of a dandelion seed head.
		20983660733_0	Close-up of a dandelion seed head.
		sigpeak	

# Load Dataset :: Image Datasets



```

def load_image_data(dir_base_path='drive/MyDrive/KIRD/datasets/mnist', split='train', shuffle=True, random_seed=42, verbose=True):
    try:
        X, Y = [], []
        for dir_name in tqdm(os.listdir(dir_base_path)):
            dir_split_path = os.path.join(dir_base_path, dir_name)

            indices_label_names = sorted([x for x in os.listdir(dir_split_path) if '!' not in x])
            if len(indices_label_names) > 0:
                indice = [int(x.split('_')[0]) for x in indices_label_names]
                label_names = [x.split('_')[1] for x in indices_label_names]

            for index_label_name, index in zip(indices_label_names, indice):
                dir_image_path = f'{dir_split_path}/{index_label_name}'
                path_image_list = sorted(glob.glob(f'{dir_image_path}/*'))
                for path_image in path_image_list:
                    X.append(np.array(Image.open(path_image)))
                    Y.append(tf.keras.utils.to_categorical(index, num_classes=len(label_names), dtype='float32'))
                    print('Loaded %d images from %s' % (len(path_image_list), dir_image_path))
    else:
        filename_list = sorted(os.listdir(dir_split_path))
        for filename in tqdm(filename_list):
            X.append(np.array(Image.open(os.path.join(dir_split_path, filename))))
            Y.append(float(filename.split('_')[0]))
            label_names = ""

    info = {'base_path': dir_base_path, 'split': split, 'label_names': label_names}

    if shuffle:
        dataset = list(zip(X, Y))
        np.random.seed(random_seed)
        np.random.shuffle(dataset)
        X, Y = zip(*dataset)

    if verbose:
        if label_names == "":
            print(dir_base_path, f'X length (shape): {len(X)} ({X[0].shape}), Y length (shape): {len(Y)} ({Y[0].shape})')
        else:
            print(dir_base_path, f'X length (shape): {len(X)} ({X[0].shape}), Y length (shape): {len(Y)} ({Y[0].shape})')

    return X, Y, label_names, info

except Exception as e:
    print(f'Error: {e}')

```

```

100% [██████████] 100/100 [00:00<00:00, 2086.01it/s]
100% [██████████] 100/100 [00:00<00:00, 2163.21it/s]
Loaded 100 images from datasets\custom_mnist\train\2_2\99.png
100% [██████████] 100/100 [00:00<00:00, 2691.43it/s]
Loaded 100 images from datasets\custom_mnist\train\3_3\99.png
100% [██████████] 100/100 [00:00<00:00, 2605.35it/s]
Loaded 100 images from datasets\custom_mnist\train\4_4\99.png
100% [██████████] 100/100 [00:00<00:00, 2639.34it/s]
Loaded 100 images from datasets\custom_mnist\train\5_5\99.png
100% [██████████] 100/100 [00:00<00:00, 1756.68it/s]
Loaded 100 images from datasets\custom_mnist\train\6_6\99.png
100% [██████████] 100/100 [00:00<00:00, 2646.13it/s]
0% | 0/100 [00:00<?, ?it/s] Loaded 100 images from datasets\custom_mnist\train\7_7\99.png
100% [██████████] 100/100 [00:00<00:00, 2623.57it/s]
0% | 0/100 [00:00<?, ?it/s] Loaded 100 images from datasets\custom_mnist\train\8_8\99.png
100% [██████████] 100/100 [00:00<00:00, 2625.18it/s]
Loaded 100 images from datasets\custom_mnist\train\9_9\99.png
roses X length (shape): 1100 ((28, 28)), Y length (shape): 1100 ((10,))
100% [██████████] 10/10 [00:00<00:00, 181.24it/s]
0% | 0/10 [00:00<?, ?it/s] Loaded 10 images from datasets\custom_flowers\test\0_dandelion\9.png
100% [██████████] 10/10 [00:00<00:00, 271.68it/s]
0% | 0/10 [00:00<?, ?it/s] Loaded 10 images from datasets\custom_flowers\test\1_daisy\9.png
100% [██████████] 10/10 [00:00<00:00, 244.75it/s]
0% | 0/10 [00:00<?, ?it/s] Loaded 10 images from datasets\custom_flowers\test\2_tulips\9.png
100% [██████████] 10/10 [00:00<00:00, 295.64it/s]
0% | 0/10 [00:00<?, ?it/s] Loaded 10 images from datasets\custom_flowers\test\3_sunflowers\9.png
100% [██████████] 10/10 [00:00<00:00, 262.05it/s]
0% | 0/100 [00:00<?, ?it/s] Loaded 10 images from datasets\custom_flowers\test\4_roses\9.png
100% [██████████] 100/100 [00:00<00:00, 263.25it/s]
Loaded 100 images from datasets\custom_flowers\train\0_dandelion\99.png
100% [██████████] 100/100 [00:00<00:00, 292.77it/s]
Loaded 100 images from datasets\custom_flowers\train\1_daisy\99.png
100% [██████████] 100/100 [00:00<00:00, 280.40it/s]
0% | 0/100 [00:00<?, ?it/s] Loaded 100 images from datasets\custom_flowers\train\2_tulips\99.png
100% [██████████] 100/100 [00:00<00:00, 265.88it/s]
0% | 0/100 [00:00<?, ?it/s] Loaded 100 images from datasets\custom_flowers\train\3_sunflowers\99.png
100% [██████████] 100/100 [00:00<00:00, 277.99it/s]
Loaded 100 images from datasets\custom_flowers\train\4_roses\99.png
roses X length (shape): 550 ((240, 320, 3)), Y length (shape): 550 ((5,))

```

# Load Dataset :: Public Datasets

```
def load_data(name='mnist', split='train', verbose=True):
    try:
        ds, info = tfds.load(name=name, split=split, with_info=True)
        label_names = info.features['label'].names

        X, Y = [], []
        for xy in tqdm(tfds.as_numpy(ds)):
            X.append(xy['image'])
            Y.append(tf.keras.utils.to_categorical(xy['label'], num_classes=len(label_names), dtype='float32'))

    except Exception as e:
        print(f'Error: {e}')

    if verbose:
        try:
            print(f'X length (shape): {len(X)} ({X[0].shape}), Y length (shape): {len(Y)} ({Y[0].shape})')
        except Exception as e:
            print(f'X length: {len(X)}, Y length: {len(Y)})')

    return X, Y, label_names, info

for name in ['mnist', 'tf_flowers']:
    X, Y, label_names, info = load_data(name, split='train')
    print(info)
```

[https://www.tensorflow.org/datasets/api\\_docs/python/tfds/load](https://www.tensorflow.org/datasets/api_docs/python/tfds/load)



```
Downloading and preparing dataset 11.06 MiB (download: 11.06 MiB, generated: 21.00 MiB, total: 32.06 MiB)
  0%|          | 0/5 [00:01<00:00, 2.16 file/s]
  100%|██████████| 60000/60000 [00:40<00:00, 1454.16it/s] X length (shape): 60000 ((28, 28, 1)), Y length (shape): 60000 ((10,))

tfds.core.DatasetInfo(
    name='mnist',
    full_name='mnist/3.0.1',
    description='',
    The MNIST database of handwritten digits.
    ...
    homepage='http://yann.lecun.com/exdb/mnist',
    data_path='/root/tensorflow_datasets/mnist/3.0.1.incomplete.tgz',
    file_format='record',
    download_size=11.06 MiB,
    dataset_size=21.00 MiB,
    features=FeaturesDict({
        'image': Image(shape=(28, 28, 1), dtype=uint8),
        'label': ClassLabel(shape=(), dtype=int64, num_classes=10),
    }),
    supervised_keys=('image', 'label'),
    disable_shuffling=False,
    splits={
        'test': <SplitInfo num_examples=10000, num_shards=1>,
        'train': <SplitInfo num_examples=60000, num_shards=1>,
    },
    citation="""
@article{lecun2010mnist,
    title={MNIST handwritten digit database},
    author={LeCun, Yann and Cortes, Corinna and Burges, CJ},
    journal={[ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist]},
    volume={2},
    year={2010}
}""",
)
```

# Load Dataset :: Plot Image Data

```

def plot_images(images, labels, label_names, i_list=[]):
    try:
        images = np.array(images, dtype=np.uint8)
        labels = np.array(labels)
    except:
        pass
    i_list = range(len(images)) if i_list == [] else i_list
    grayscale = False if images[0].shape[-1] == 3 else True
    fig = plt.figure(figsize=(3 if grayscale else 3 * 4, 3 * len(i_list)))

    ax_list = []
    for i_ax, i_image in enumerate(i_list):
        x = images[i_image]
        y = labels[i_image].astype(np.float16)

        try:
            y = f'{label_names[np.argmax(y)]} {y}'
        except:
            pass

        if grayscale:
            ax = fig.add_subplot(len(i_list), 1, i_ax + 1)
            ax.imshow(x, cmap='gray')
            ax.set_title(f"Image #{i_image}\nLabel: {y}\nShape: {x.shape}")
            ax.set_axis_off()
        else:
            for i_ax_column in range(4):
                if i_ax_column == 0:
                    ax = fig.add_subplot(len(i_list), 4, i_ax * 4 + 1)
                    ax.imshow(x)
                    ax.set_title(f"Image #{i_image}\nLabel: {y}\nShape: {x.shape}", loc='left')
                    ax.set_axis_off()
                else:
                    i_channel = i_ax_column - 1
                    x_channel = x.copy()
                    x_channel[:, :, [x for x in range(3) if x != i_channel]] = 0
                    ax = fig.add_subplot(len(i_list), 4, i_ax * 4 + 1 + i_ax_column)
                    ax.imshow(x_channel)
                    ax.set_title(f"[Red, 'Green', 'Blue'][i_channel] Channel")
                    ax.set_axis_off()
            ax_list.append(ax)

    fig.tight_layout()
    fig.show()

```

Image #1  
Label: 1 [0. 1. 0. 0. 0. 0. 0. 0.]  
Shape: (28, 28, 1)

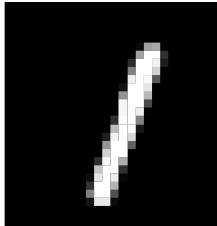


Image #2  
Label: 0 [1. 0. 0. 0. 0. 0. 0. 0.]  
Shape: (28, 28, 1)

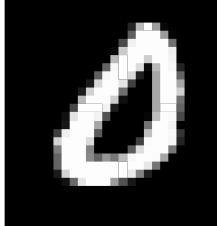


Image #3  
Label: 7 [0. 0. 0. 0. 0. 0. 1. 0.]  
Shape: (28, 28, 1)

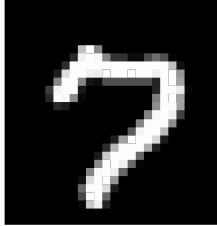
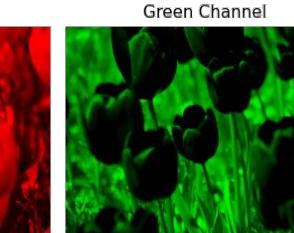


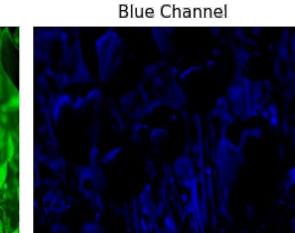
Image #0  
Label: tulips [0. 0. 1. 0. 0.]  
Shape: (333, 500, 3)



Red Channel



Green Channel

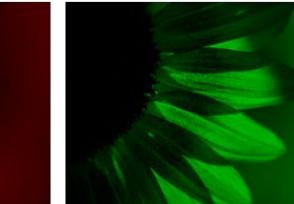


Blue Channel

Image #1  
Label: sunflowers [0. 0. 0. 1. 0.]  
Shape: (212, 320, 3)



Red Channel



Green Channel



Blue Channel

Image #5  
Label: dandelion [1. 0. 0. 0. 0.]  
Shape: (212, 320, 3)



Red Channel



Green Channel



Blue Channel

# Preprocessing

```
def preprocessing(X, Y, i_list=[]):
    XX, YY = [], []
    for i in tqdm(i_list if len(i_list) > 0 else range(len(X))):
        x, y = X[i], Y[i]

        x = resize_and_fill_image(x, target_size=(100, 100), fill_color=(255, 255, 255))
        x = resize_and_crop_image(x, target_size=(100, 100))
        x = rgb_to_grayscale_image(x)
        x = flip_image(x, mode='lr')
        x = flip_image(x, mode='ud')
        x = rotate_image(image=x, angle_list=[])

        XX.append(x)
        YY.append(y)

    XX, YY = mixup_images(XX, YY)
    XX, YY = cutmix_images(XX, YY)

    return XX, YY

XX, YY = preprocessing(X, Y)
plot_images(images=XX, labels=YY, label_names=label_names, i_list=[0, 1, 5])
```

# Preprocessing :: rgb\_to\_grayscale

```
# prompt: python code to convert rgb image to grayscale image based on opencv
def rgb_to_grayscale_image(image):
    try:
        if image.shape[-1] == 3:
            return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        else:
            print('Error: The input image is not a RGB image')
            return image
    except Exception as e:
        print(f'Error: {e}')
        return image
```

Image #0  
Label: tulips [0. 0. 1. 0. 0.]  
Shape: (333, 500)



Image #1  
Label: sunflowers [0. 0. 0. 1. 0.]  
Shape: (212, 320)



# Tip :: How to create Preprocessing Code via ChatGPT?



python code to convert rgb image to grayscale image based on opencv



Certainly! Here's an example of Python code that uses OpenCV library to convert an RGB image to grayscale:

python



Copy code

```
import cv2

# Read the RGB image
rgb_image = cv2.imread('input_image.jpg')

# Convert the RGB image to grayscale
gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2GRAY)

# Save the grayscale image
cv2.imwrite('output_image.jpg', gray_image)
```

# Tip :: How to create Preprocessing Code via ChatGPT?



```
def rgb_to_grayscale_image(image):
    try:
        if image.shape[-1] == 3:
            return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        else:
            print('Error: The input image is not a RGB image')
            return image

    except Exception as e:
        print(f'Error: {e}')
        return image
```

I made the above code to convert rgb image to grayscale image based on opencv.

Write the python code for resize\_and\_fill\_image based on the function's input and output.



Certainly! Here's an example of Python code for the `resize\_and\_fill\_image` function based on your input and output requirements:

```
python

import cv2
import numpy as np

def resize_and_fill_image(image, target_size):
    try:
        # Resize the image while maintaining the aspect ratio
        height, width = image.shape[:2]
        aspect_ratio = width / height
        target_width, target_height = target_size

        if aspect_ratio > 1:
            new_width = target_width
            new_height = int(new_width / aspect_ratio)
        else:
            new_height = target_height
            new_width = int(new_height * aspect_ratio)

        resized_image = cv2.resize(image, (new_width, new_height))

        # Create a blank canvas of the target size
        canvas = np.zeros((target_height, target_width, image.shape[2]), dtype=np.uint8)

        # Calculate the position to place the resized image in the canvas
        x = (target_width - new_width) // 2
        y = (target_height - new_height) // 2

        # Fill the canvas with the resized image
        canvas[y:y+new_height, x:x+new_width] = resized_image

    return canvas

    except Exception as e:
        print(f'Error: {e}')
        return image
```

# Tip :: How to create Preprocessing Code via ChatGPT?

```
python

import cv2
import numpy as np

def resize_and_fill_image(image, target_size):
    try:
        # Resize the image while maintaining the aspect ratio
        height, width = image.shape[:2]
        aspect_ratio = width / height
        target_width, target_height = target_size

        if aspect_ratio > 1:
            new_width = target_width
            new_height = int(new_width / aspect_ratio)
        else:
            new_height = target_height
            new_width = int(new_height * aspect_ratio)

        resized_image = cv2.resize(image, (new_width, new_height))

        # Create a blank canvas of the target size
        canvas = np.zeros((target_height, target_width, image.shape[2]), dtype=np.uint8)

        # Calculate the position to place the resized image in the canvas
        x = (target_width - new_width) // 2
        y = (target_height - new_height) // 2

        # Fill the canvas with the resized image
        canvas[y:y+new_height, x:x+new_width] = resized_image

        return canvas

    except Exception as e:
        print(f'Error: {e}')
        return image
```

simplify and make the code faster

Certainly! Here's a simplified and optimized version of the `resize\_and\_fill\_image` function:

```
python

import cv2
import numpy as np

def resize_and_fill_image(image, target_size):
    try:
        # Resize the image while maintaining the aspect ratio
        resized_image = cv2.resize(image, target_size)

        # Create a blank canvas of the target size
        canvas = np.zeros((target_size[1], target_size[0], image.shape[2]),

        # Calculate the position to place the resized image in the canvas
        x = (target_size[0] - resized_image.shape[1]) // 2
        y = (target_size[1] - resized_image.shape[0]) // 2

        # Fill the canvas with the resized image
        canvas[y:y+resized_image.shape[0], x:x+resized_image.shape[1]] = resized_image

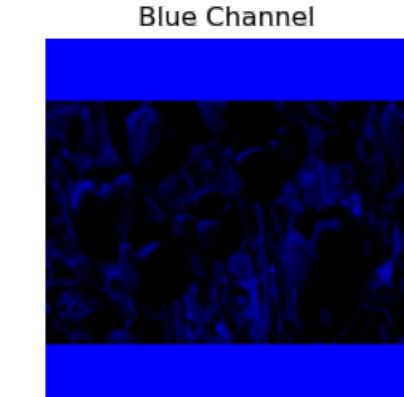
        return canvas

    except Exception as e:
        print(f'Error: {e}')
        return image
```

# Preprocessing :: resize\_and\_fill\_image

```
def resize_and_fill_image(image, target_size, fill_color=(0, 0, 0)):  
    try:  
        if image.shape[-1] == 1:  
            new_image = np.full((target_size[0], target_size[1]), fill_color[0], dtype=np.uint8)  
        else:  
            new_image = np.full((target_size[0], target_size[1], 3), fill_color, dtype=np.uint8)  
  
        scale = min(target_size[0] / image.shape[0], target_size[1] / image.shape[1])  
        resized_image = cv2.resize(image, None, fx=scale, fy=scale, interpolation=cv2.INTER_AREA)  
        center = (target_size[0] // 2, target_size[1] // 2)  
        position = (center[0] - resized_image.shape[0] // 2, center[1] - resized_image.shape[1] // 2)  
        new_image[position[0]:position[0] + resized_image.shape[0], position[1]:position[1] + resized_image.shape[1]] = resized_image  
    return new_image  
  
except Exception as e:  
    print(f'Error: {e}')  
    return image
```

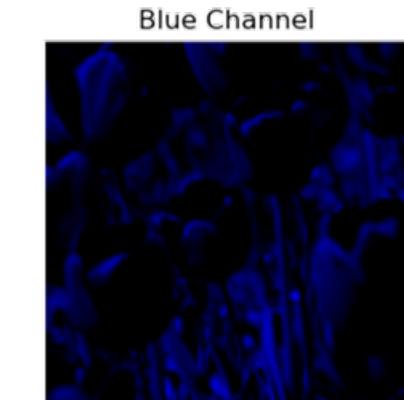
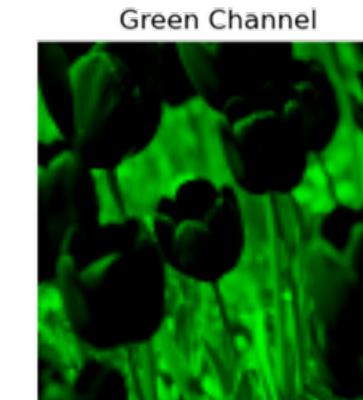
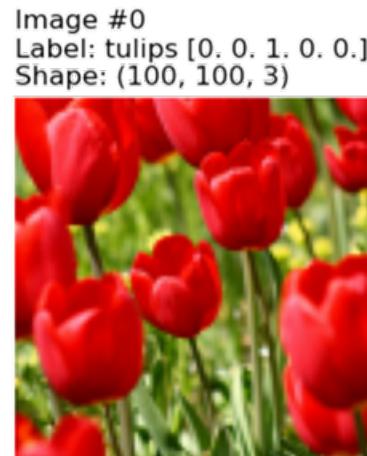
Image #0  
Label: tulips [0. 0. 1. 0. 0.]  
Shape: (100, 100, 3)



# Preprocessing :: resize\_and\_crop\_image

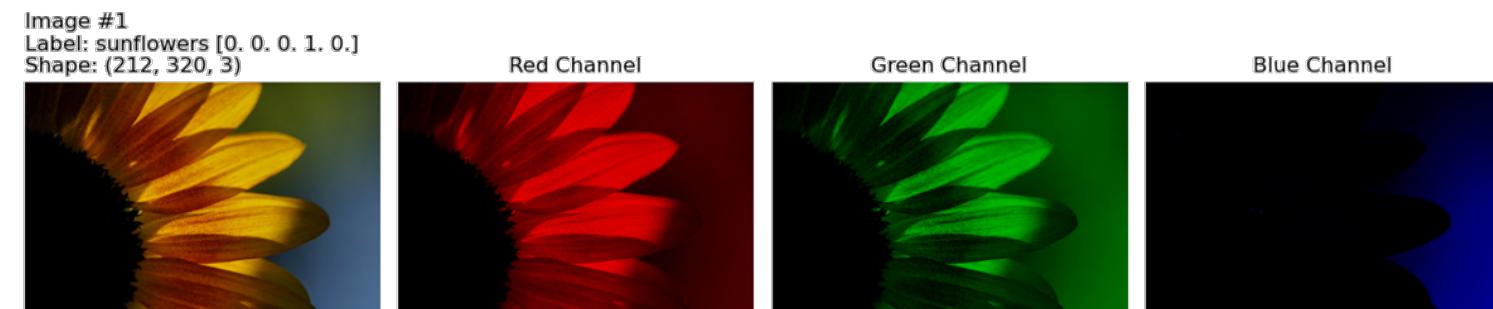
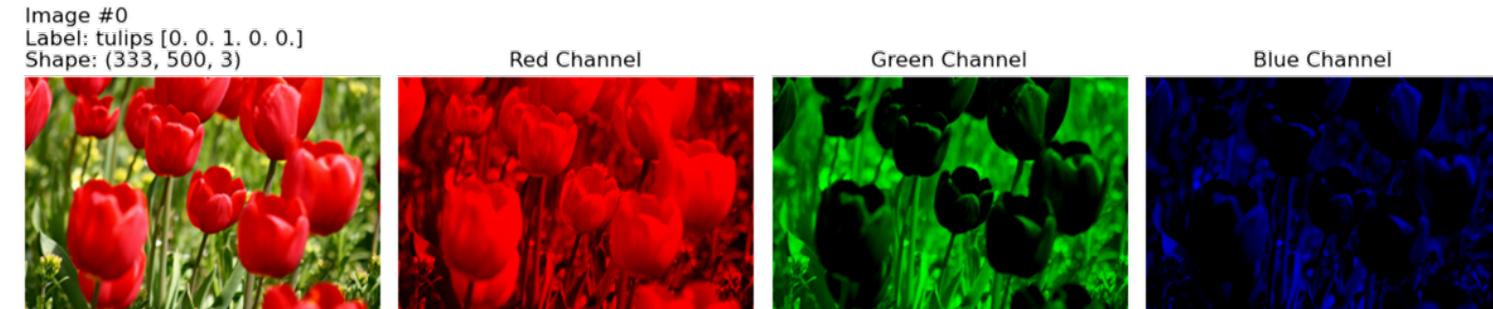
```
def resize_and_crop_image(image, target_size):
    try:
        scale = max(target_size[0] / image.shape[0], target_size[1] / image.shape[1])
        resized_image = cv2.resize(image, None, fx=scale, fy=scale, interpolation=cv2.INTER_AREA)
        start_x = (resized_image.shape[1] - target_size[1]) // 2
        start_y = (resized_image.shape[0] - target_size[0]) // 2
        end_x = start_x + target_size[1]
        end_y = start_y + target_size[0]
        cropped_image = resized_image[start_y:end_y, start_x:end_x]
        return cropped_image

    except Exception as e:
        print(f'Error: {e}')
        return image
```



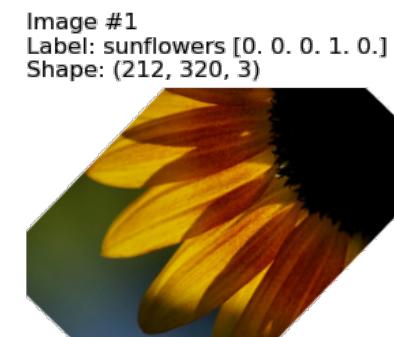
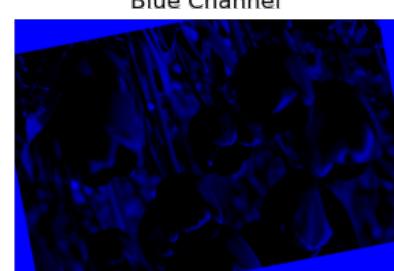
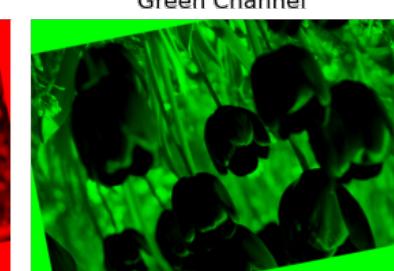
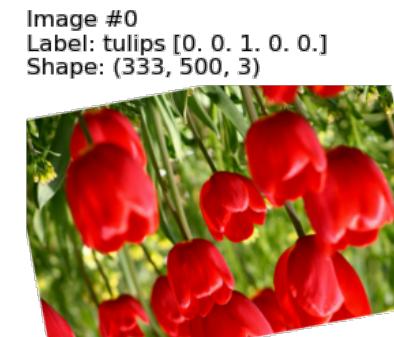
# Preprocessing :: flip\_image

```
def flip_image(image, mode='lr'): # lr: left-right, ud: up-down
    try:
        if mode == 'lr':
            flipped_image = np.fliplr(image)
        elif mode == 'ud':
            flipped_image = np.flipud(image)
        else:
            print('Error: The input mode is not valid (lr for left-right, ud for up-down)')
            return image
        return flipped_image
    except Exception as e:
        print(f'Error: {e}')
        return image
```



# Preprocessing :: rotate\_image

```
# prompt: python code to rotate image based on opencv
def rotate_image(image, angle_list=[], fill_color=(255, 255, 255)):
    try:
        center = (image.shape[1] // 2, image.shape[0] // 2)
        angle = np.random.randint(0, 360) if angle_list == [] else -angle_list[np.random.randint(0, len(angle_list))]
        rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
        rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1], image.shape[0]), borderValue=fill_color)
        return rotated_image
    except Exception as e:
        print(f'Error: {e}')
        return image
```



# Preprocessing :: mixup & cutmix

```

def preprocessing(X, Y, i_list=[]):
    XX, YY = [], []
    for i in tqdm(i_list if len(i_list) > 0 else range(len(X))):
        x, y = X[i], Y[i]

        x = resize_and_crop_image(x, target_size=(100, 100))

        XX.append(x)
        YY.append(y)

    XX, YY = mixup_images(XX, YY)
    XX, YY = cutmix_images(XX, YY)

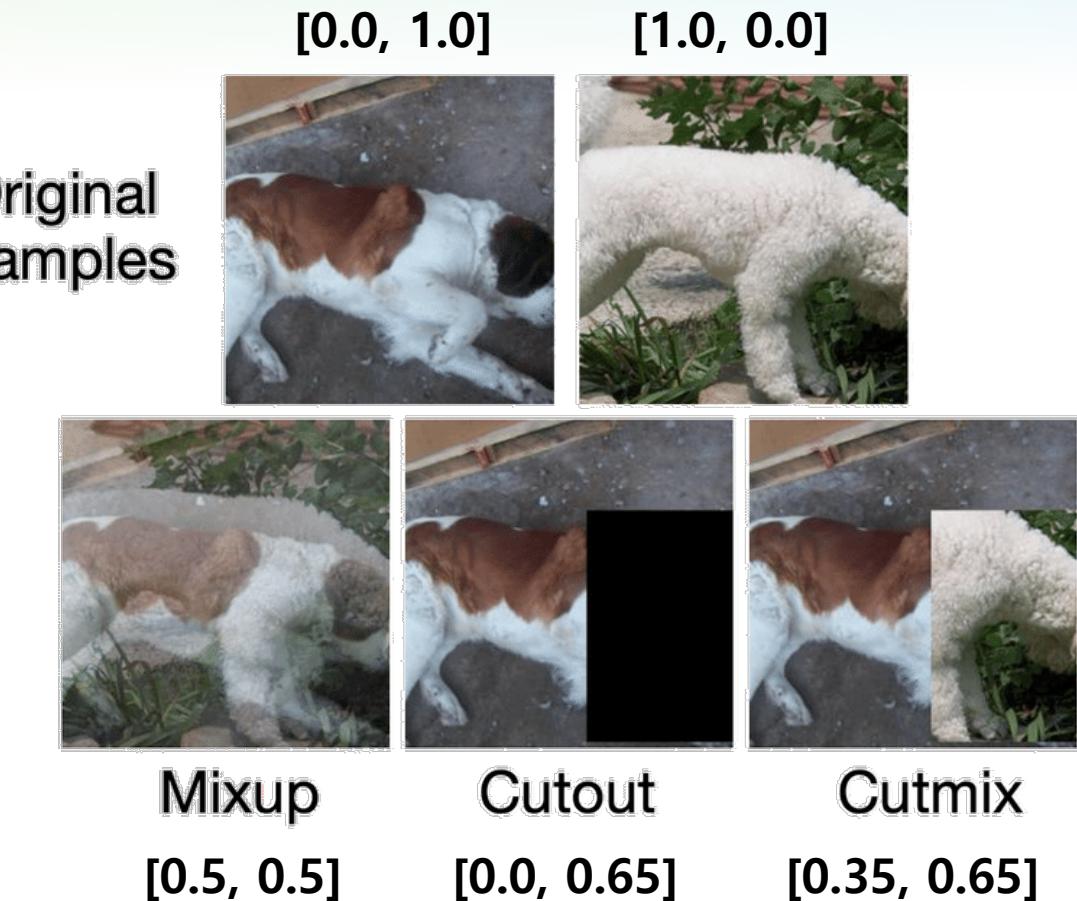
    return XX, YY

```

```

XX, YY = preprocessing(X, Y)
plot_images(images=XX, labels=YY, label_names=label_names, i_list=[0, 1, 5])

```



# Preprocessing :: mixup\_images

```
def mixup_images(images, labels, alpha=0.5):
    try:
        images, labels = np.array(images), np.array(labels)
    except:
        assert False, 'The input images should be same shape'

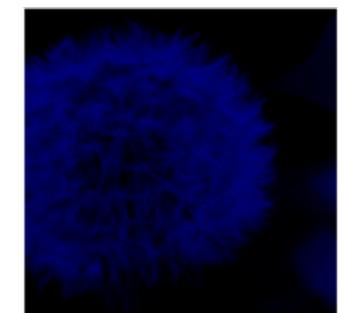
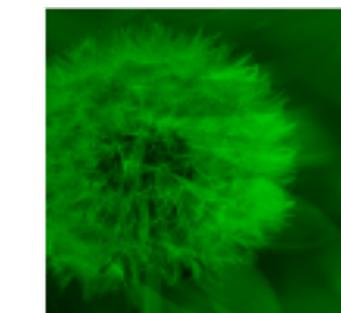
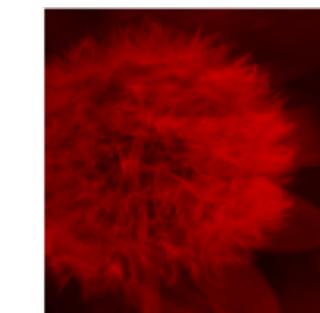
    indices_random = np.random.permutation(images.shape[0])
    images_shuffled, labels_shuffled = images[indices_random], labels[indices_random]

    ratio = np.random.beta(alpha, alpha, images.shape[0]) if alpha > 0 else 1.0

    ratio = ratio.reshape(-1, 1, 1, 1)
    images_mixed = ratio * images + (1 - ratio) * images_shuffled

    ratio = ratio.reshape(-1, 1)
    labels_mixed = ratio * labels + (1 - ratio) * labels_shuffled

    return images_mixed, labels_mixed
```



# Preprocessing :: cutmix\_images

```
def cutmix_images(images, labels, beta=0.5):
```

```
try:
```

```
    images, labels = np.array(images), np.array(labels)
```

```
except:
```

```
    assert False, 'The input images should be same shape'
```

```
indices_random = np.random.permutation(len(images))
```

```
target_a, target_b = labels, labels[indices_random]
```

```
h, w = images[0].shape[:2]
```

```
ratio_cut = np.sqrt(1. - np.random.beta(beta, beta))
```

```
w_cut, h_cut = int(w * ratio_cut), int(h * ratio_cut)
```

```
x_cut, y_cut = np.random.randint(w), np.random.randint(h)
```

```
bbx1, bby1 = np.clip(x_cut - w_cut // 2, 0, w), np.clip(y_cut - h_cut // 2, 0, h)
```

```
bbx2, bby2 = np.clip(x_cut + w_cut // 2, 0, w), np.clip(y_cut + h_cut // 2, 0, h)
```

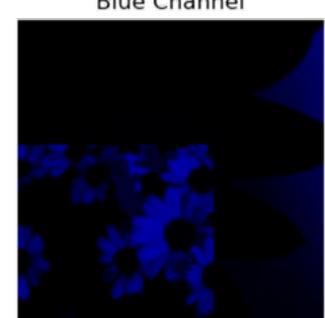
```
images_mixed = images.copy()
```

```
images_mixed[:, bbx1:bbx2, bby1:bby2, :] = images[indices_random, bbx1:bbx2, bby1:bby2, :]
```

```
ratio = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (h * w))
```

```
labels_mixed = target_a * ratio + target_b * (1. - ratio)
```

```
return images_mixed, labels_mixed
```



# Preprocessing :: custom function

```
def preprocessing(X, Y, i_list=[]):
```

```
    XX, YY = [], []
```

```
    for i in tqdm(i_list if len(i_list) > 0 else range(len(X))):
```

```
        x, y = X[i], Y[i]
```

```
x = cv2.blur(x, (5, 5))
```

```
XX.append(x)
```

```
YY.append(y)
```

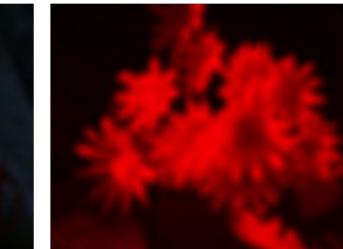
```
XX, YY = np.array(XX), np.array(YY)
```

```
return XX, YY
```

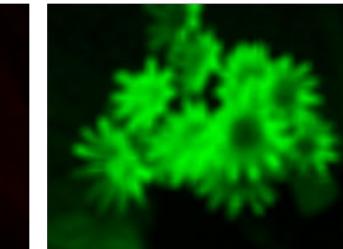
Image #0  
Label: daisy [0. 1. 0. 0. 0.]  
Shape: (240, 320, 3)



Red Channel



Green Channel



Blue Channel

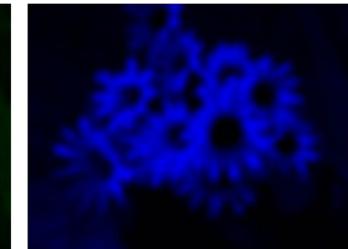


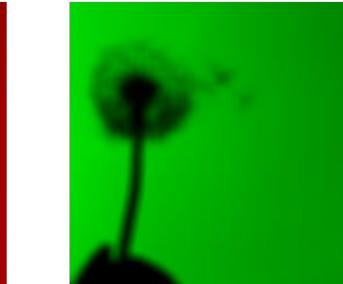
Image #1  
Label: dandelion [1. 0. 0. 0. 0.]  
Shape: (240, 240, 3)



Red Channel



Green Channel



Blue Channel



```
XX, YY = preprocessing(X, Y)
```

```
plot_images(images=XX, labels=YY, label_names=label_names, i_list=[0, 1, 5])
```

# AutoML (w/o Augmentation) for Custom Dataset with Preprocessing

```
trial_max = 5 # model search trials
epoch_max = 100 # epochs for each model
project_name = 'AutoML_img_clf'
train_validation_split = 0.2
train_validation_split_random_state = 42
augment = False
# augment = ak.ImageAugmentation(horizontal_flip=True, vertical_flip=True, rotation_factor=1, zoom_factor=0.5, contrast_factor=0.5)
```

```
XX_train, XX_val, YY_train, YY_val = train_test_split(XX, YY, test_size=train_validation_split, random_state=train_validation_split_random_state)
```

```
input_node = ak.ImageInput()
output_node = ak.ImageBlock(augment=augment)(input_node)
output_node = ak.ClassificationHead()(output_node)
model = ak.AutoModel(inputs=input_node, outputs=output_node, project_name=project_name, overwrite=True, max_trials=trial_max)
model.fit(XX_train, YY_train, validation_data=(XX_val, YY_val), epochs=epoch_max)
```

```
model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))
model_best.summary()
```

# AutoML (w Augmentation) for Custom Dataset with Preprocessing

Trial 1 Complete [00h 00m 57s]  
val\_loss: 0.0474175363779068

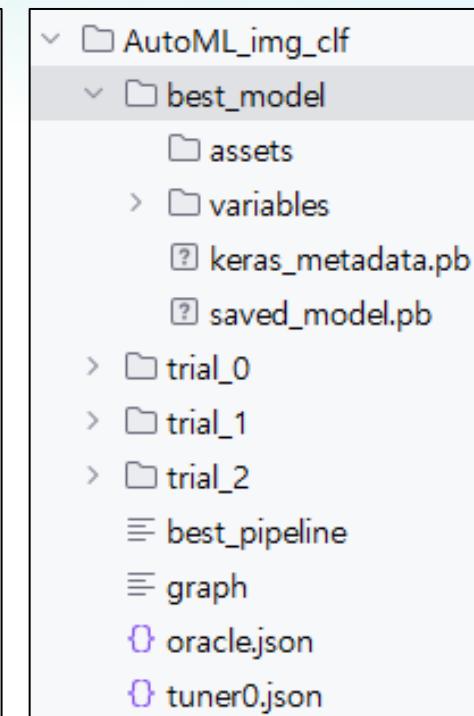
Best val\_loss So Far: 0.0474175363779068  
Total elapsed time: 00h 00m 57s

Search: Running Trial #2

Value	Best Value So Far	Hyperparameter
resnet	vanilla	image_block_1/block_type
True	True	image_block_1/normalize
True	False	image_block_1/augment
True	None	image_block_1/image_augmentation_1/horizontal_flip
True	None	image_block_1/image_augmentation_1/vertical_flip
0	None	image_block_1/image_augmentation_1/contrast_factor
0	None	image_block_1/image_augmentation_1/rotation_factor
0.1	None	image_block_1/image_augmentation_1/translation_factor
0	None	image_block_1/image_augmentation_1/zoom_factor
False	None	image_block_1/res_net_block_1/pretrained
resnet50	None	image_block_1/res_net_block_1/version
True	None	image_block_1/res_net_block_1/imagenet_size
global_avg	flatten	classification_head_1/spatial_reduction_1/reduction_type
0	0.5	classification_head_1/dropout
adam	adam	optimizer
0.001	0.001	learning_rate

Epoch 1/5  
291/1500 [====>.....] - ETA: 6:16 - loss: 1.4451 - accuracy: 0.5483

Model: "model"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[None, 28, 28]	0
cast_to_float32 (CastToFloat32)	(None, 28, 28)	0
expand_last_dim (ExpandLastDim)	(None, 28, 28, 1)	0
normalization (Normalization)	(None, 28, 28, 1)	3
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dropout_1 (Dropout)	(None, 9216)	0
dense (Dense)	(None, 10)	92170
classification_head_1 (Softmax)	(None, 10)	0
<hr/>		
Total params: 110,989		
Trainable params: 110,986		
Non-trainable params: 3		



# Evaluation of Best Model searched by AutoML

```
model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))
```

```
X_test, Y_test, label_names, info = load_image_data(path_dataset, split='test')
```

```
XX_test, YY_test = preprocessing(X_test, Y_test)
```

```
YY_predicted = model_best.predict(XX_test)
```

```
YY_probability = softmax(YY_predicted, axis=1)
```

```
YY_test_index, YY_predicted_index = np.argmax(YY_test, axis=1), np.argmax(YY_predicted, axis=1)
```

```
accuracy = accuracy_score(YY_test_index, YY_predicted_index)
```

```
print(f'{path_dataset} :: Test Dataset :: Accuracy = {accuracy:.3f}')
```

```
cm = confusion_matrix(YY_test_index, YY_predicted_index, normalize='true')
```

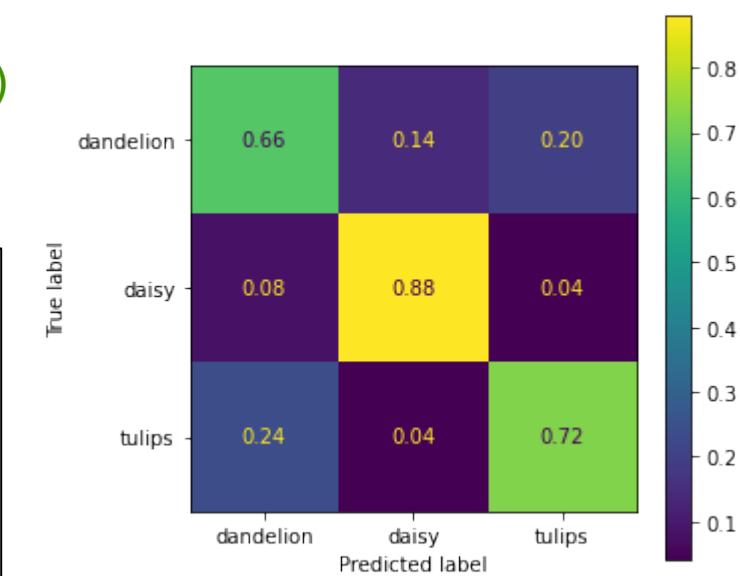
```
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=label_names)
```

*fig, ax = plt.subplots(figsize=(5, 5)) # Adjust the size (width, height) as needed*

```
disp.plot(ax=ax, values_format='.2f')
```

```
plt.show()
```

```
Loaded 50 images from datasets/flower\test/0_dandelion
Loaded 50 images from datasets/flower\test/1_daisy
Loaded 50 images from datasets/flower\test/2_tulips
100%|██████████| 150/150 [00:00<00:00, 2128.10it/s]
datasets/flower\test X count (shape): 150 ((240, 320, 3)), Y count (shape): 150 ((3,))
datasets/flower :: Test Dataset :: Accuracy = 0.753
```



# Extra Training for Best Model

```
epoch_max = 20
```

```
path_model_best = 'model_best.h5'
```

```
model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))  
model_best.fit(XX_train, YY_train, validation_data=(XX_val, YY_val), epochs=epoch_max)
```

```
model_best.save(path_model_best)
```

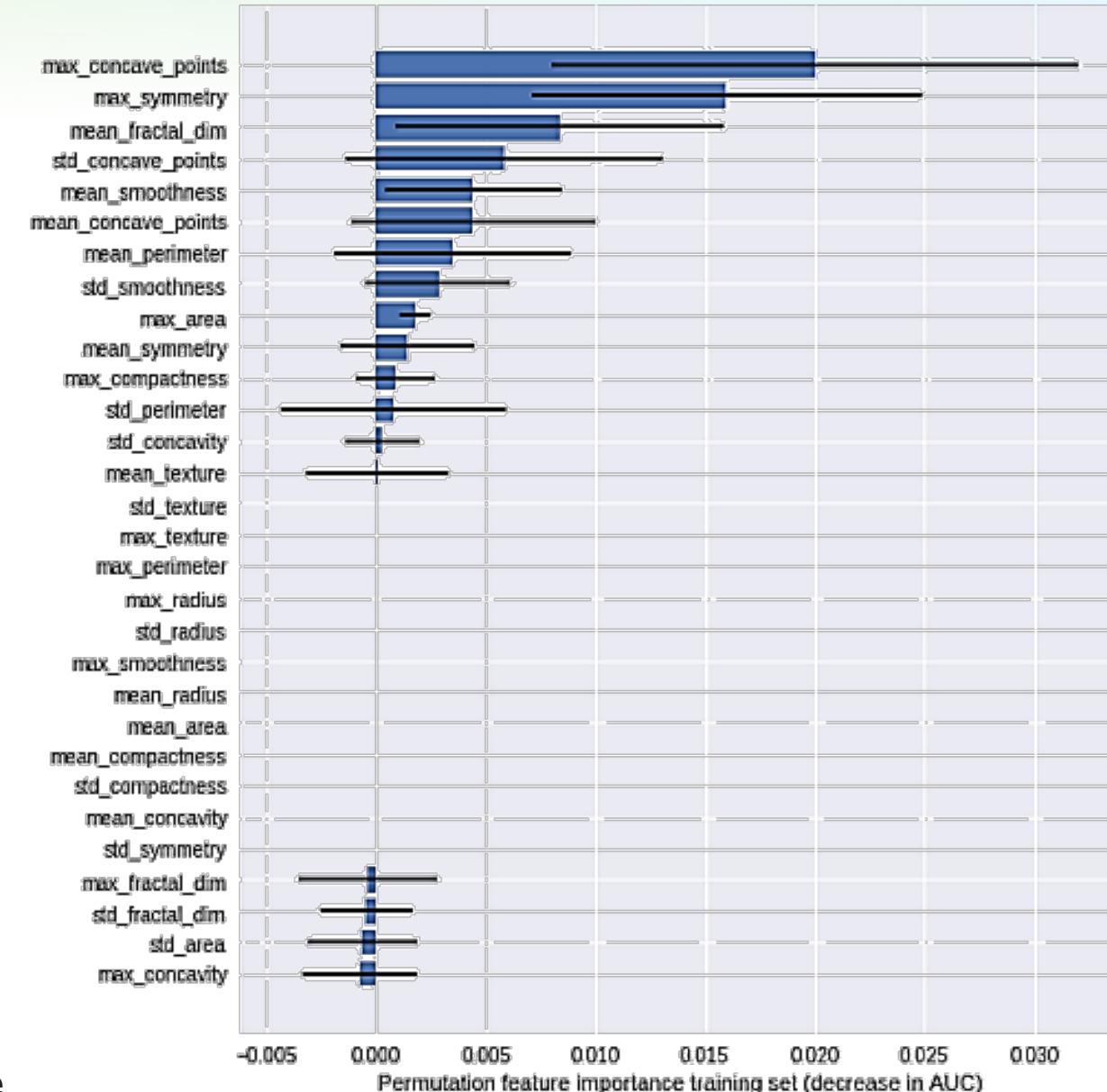
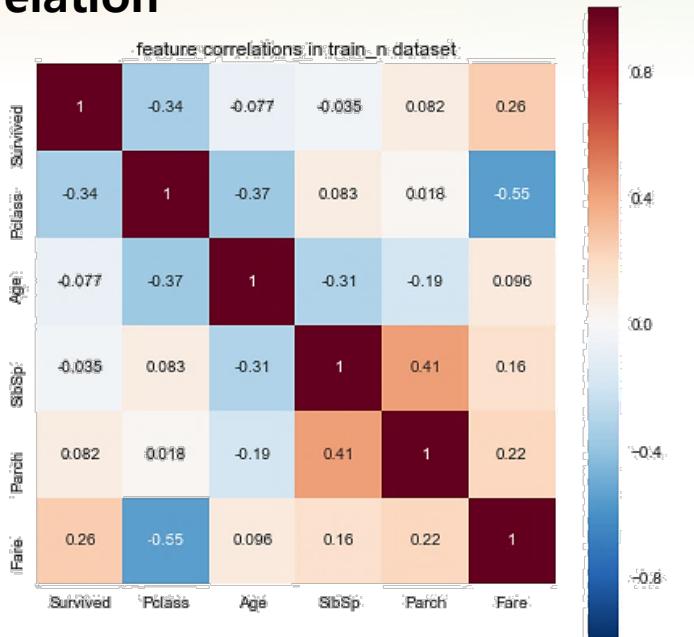
```
model_best = tf.keras.models.load_model(path_model_best)
```

```
print('## Model loaded:', path_model_best)
```

```
Epoch 14/20  
34/34 [=====] - 2s 71ms/step - loss: 0.1706 - accuracy: 0.9463 - val_loss: 5.1009 - val_accuracy: 0.6074  
Epoch 15/20  
34/34 [=====] - 2s 67ms/step - loss: 0.0765 - accuracy: 0.9750 - val_loss: 1.4722 - val_accuracy: 0.7222  
Epoch 16/20  
34/34 [=====] - 2s 66ms/step - loss: 0.0473 - accuracy: 0.9870 - val_loss: 1.6399 - val_accuracy: 0.6593  
Epoch 17/20  
34/34 [=====] - 2s 65ms/step - loss: 0.0570 - accuracy: 0.9824 - val_loss: 2.7498 - val_accuracy: 0.6889  
Epoch 18/20  
34/34 [=====] - 2s 64ms/step - loss: 0.0612 - accuracy: 0.9806 - val_loss: 1.4744 - val_accuracy: 0.7037  
Epoch 19/20  
34/34 [=====] - 2s 66ms/step - loss: 0.0358 - accuracy: 0.9898 - val_loss: 1.4762 - val_accuracy: 0.7370  
Epoch 20/20  
34/34 [=====] - 2s 69ms/step - loss: 0.0567 - accuracy: 0.9815 - val_loss: 2.3584 - val_accuracy: 0.6815  
## Model loaded: model_best.h5
```

# Exploratory Data Analysis (EDA) & eXplainable AI (XAI)

## Feature Correlation



## Feature Permutation Importance

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

# XAI :: Local Interpretable Model-agnostic Explanations (LIME)

## “Why Should I Trust You?” Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro  
University of Washington  
Seattle, WA 98105, USA  
[marcotcr@cs.uw.edu](mailto:marcotcr@cs.uw.edu)

Sameer Singh  
University of Washington  
Seattle, WA 98105, USA  
[sameer@cs.uw.edu](mailto:sameer@cs.uw.edu)

Carlos Guestrin  
University of Washington  
Seattle, WA 98105, USA  
[guestrin@cs.uw.edu](mailto:guestrin@cs.uw.edu)



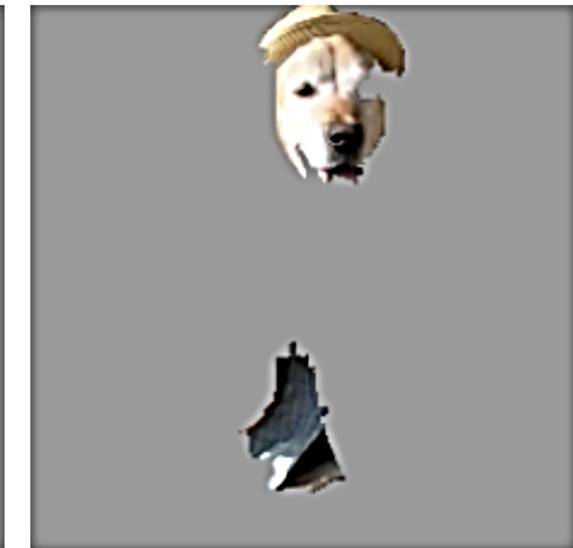
(a) Original Image



(b) Explaining *Electric guitar*



(c) Explaining *Acoustic guitar*

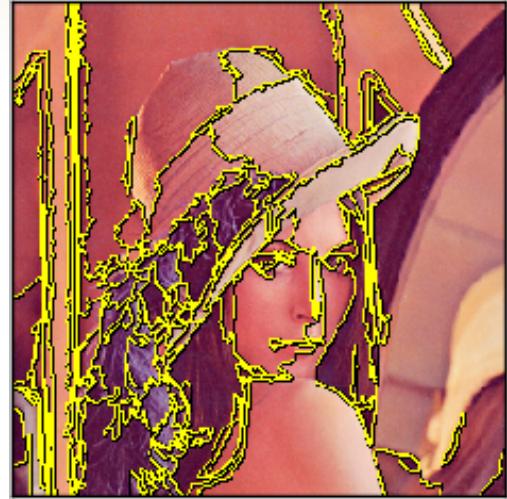


(d) Explaining *Labrador*

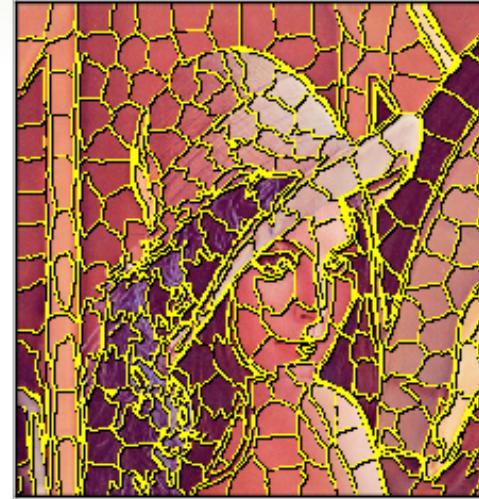
Figure 4: Explaining an image classification prediction made by Google’s Inception neural network. The top 3 classes predicted are “Electric Guitar” ( $p = 0.32$ ), “Acoustic guitar” ( $p = 0.24$ ) and “Labrador” ( $p = 0.21$ )

# Segmentation Options

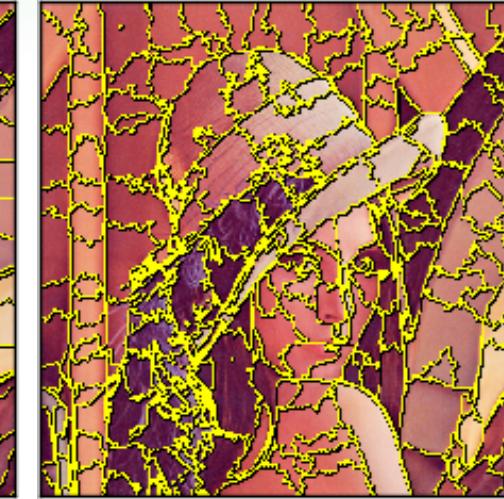
Felzenszwalbs's method



SLIC



Quickshift



Felzenszwalbs's method



SLIC



Quickshift



# XAI :: Local Interpretable Model-agnostic Explanations (LIME)

```
i_check = 0
num_segments = 100
num_samples = 1000
seg_type = 'quickshift' # algo_type: quickshift, felzenszwalb, slic

model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))

X_test, Y_test, label_names, info = load_image_data(path_dataset, split='test')
XX_test, YY_test = preprocessing(X_test, Y_test)
YY_predicted = model_best.predict(XX_test)
YY_probability = softmax(YY_predicted, axis=1)
print(YY_test[i_check], YY_predicted[i_check], YY_probability[i_check])

explainer = lime_image.LimeImageExplainer()

segmenter = SegmentationAlgorithm(algo_type=seg_type, n_segments=num_segments, compactness=1, sigma=1)
test_image = XX_test[i_check]

explanation = explainer.explain_instance(test_image, model_best.predict, top_labels=len(label_names), hide_color=0,
                                         num_samples=num_samples, segmentation_fn=segmenter)
i_label_predicted = explanation.top_labels[0]
```

```
[0. 1. 0.] [6.6214677e-13 1.0000000e+00 2.6636424e-11] [0.21194156 0.5761169 0.21194156]
100%  1000/1000 [00:09<00:00, 117.54it/s]
```

# XAI :: SHapley Additive exPlanations (SHAP)

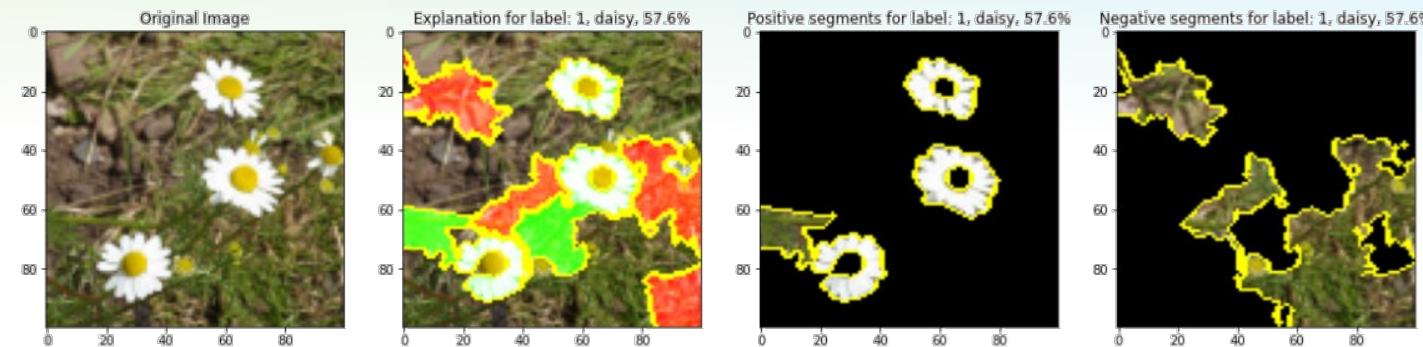
```
num_features = 5
```

```
fig, ax_list = plt.subplots(1, 4, figsize=(20, 10))
```

```
ax = ax_list[0]
```

```
ax.imshow(test_image)
```

```
ax.set_title('Original Image')
```



```
ax = ax_list[1]
```

```
temp, mask = explanation.get_image_and_mask(i_label_predicted, positive_only=False, num_features=num_features * 2, hide_rest=False)
ax.imshow(mark_boundaries(temp / 255.0, mask))
```

```
ax.set_title('Explanation for label: {}, {}, {:.1f}%'.format(i_label_predicted, label_names[i_label_predicted], YY_probability[i_test][i_label_predicted] * 100))
```

```
ax = ax_list[2]
```

```
temp, mask = explanation.get_image_and_mask(i_label_predicted, positive_only=True, num_features=num_features, hide_rest=True)
ax.imshow(mark_boundaries(temp / 255.0, mask))
```

```
ax.set_title('Positive segments for label: {}, {}, {:.1f}%'.format(i_label_predicted, label_names[i_label_predicted], YY_probability[i_test][i_label_predicted] * 100))
```

```
ax = ax_list[3]
```

```
temp, mask = explanation.get_image_and_mask(i_label_predicted, positive_only=False, negative_only=True, num_features=num_features, hide_rest=True)
ax.imshow(mark_boundaries(temp / 255.0, mask))
```

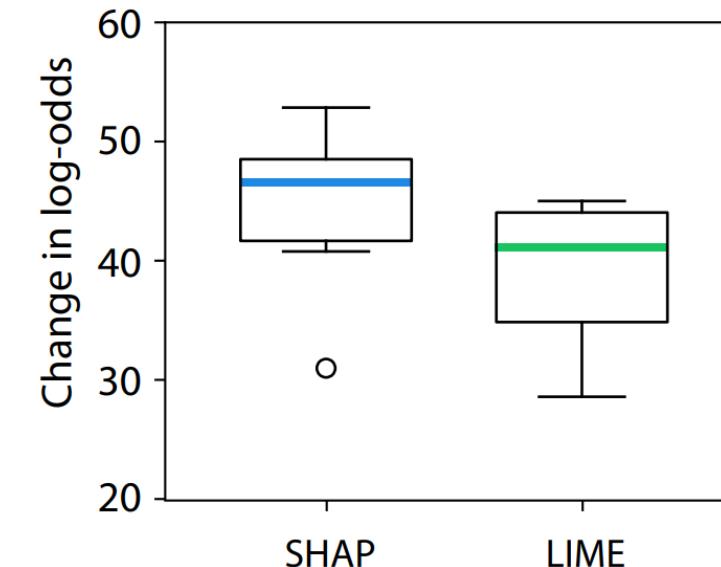
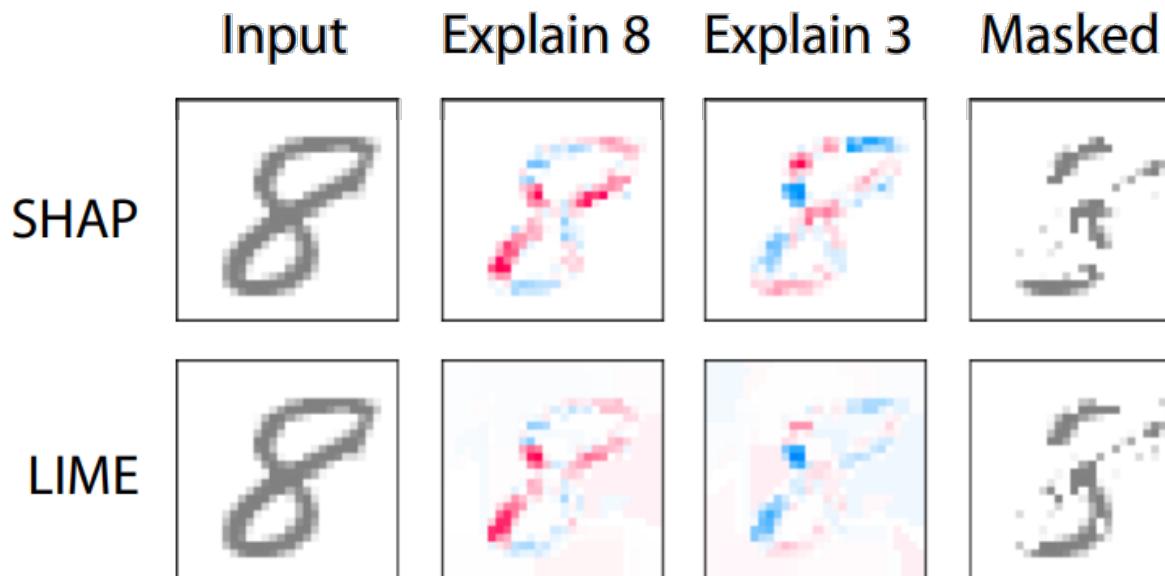
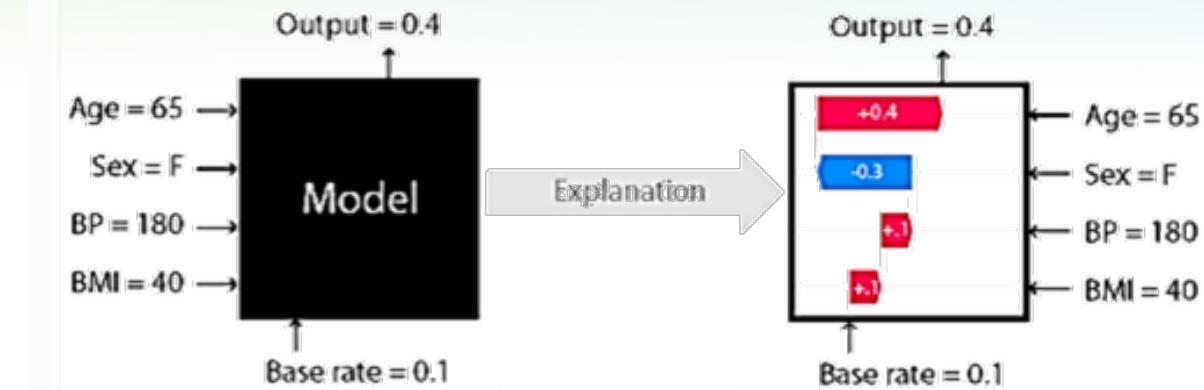
```
ax.set_title('Negative segments for label: {}, {}, {:.1f}%'.format(i_label_predicted, label_names[i_label_predicted], YY_probability[i_test][i_label_predicted] * 100))
```

# XAI :: SHapley Additive exPlanations (SHAP)

## A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg  
 Paul G. Allen School of Computer Science  
 University of Washington  
 Seattle, WA 98105  
 slund1@cs.washington.edu

Su-In Lee  
 Paul G. Allen School of Computer Science  
 Department of Genome Sciences  
 University of Washington  
 Seattle, WA 98105  
 suinlee@cs.washington.edu



# XAI :: SHapley Additive exPlanations (SHAP)

```

import shap
import numpy as np

i_check = 0
max_evals = 500
batch_size = 50

model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))

X_test, Y_test, label_names, info = load_image_data(path_dataset, split='test')
XX_test, YY_test = preprocessing(X_test, Y_test)
YY_predicted = model_best.predict(XX_test)
YY_probability = softmax(YY_predicted, axis=1)

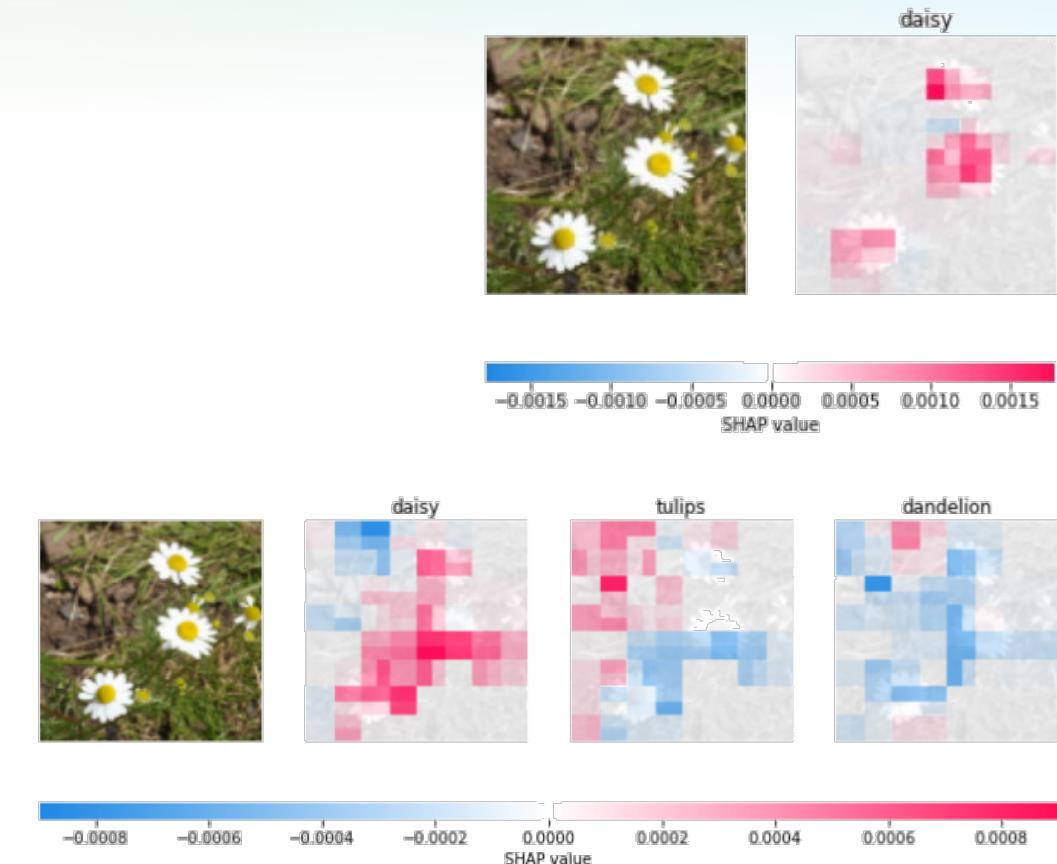
i_sort = np.argsort(YY_probability[i_check])[::-1]
for i in i_sort:
    print('%d, %s, %.1f%%' % (i, label_names[i], YY_probability[i_check][i] * 100))

masker = shap.maskers.Image("inpaint_teela", XX_test[i_check].shape)
explainer = shap.Explainer(model_best, masker, output_names=label_names)

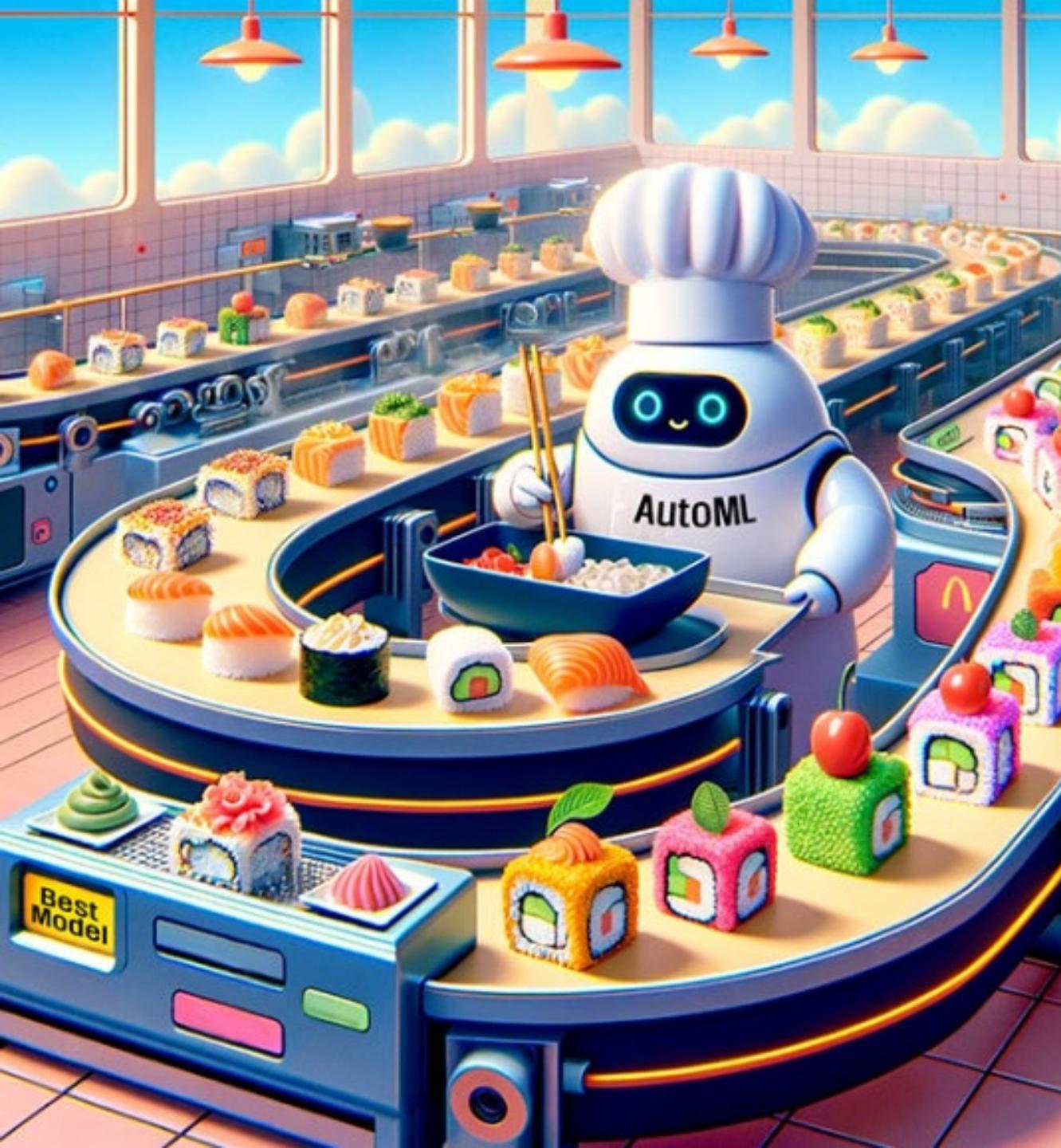
shap_values = explainer(XX_test[i_check:i_check + 1], max_evals=max_evals, batch_size=batch_size, outputs=shap.Explanation.argsort.flip[:1])
# shap_values = explainer(XX_test[i_check:i_check + 1], max_evals=max_evals, batch_size=batch_size, outputs=shap.Explanation.argsort.flip)

shap.image_plot(shap_values)

```

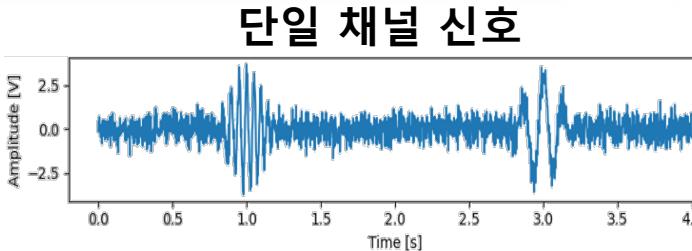


# Contents



0. What is AutoML?
1. 이미지 데이터에 대한 AutoML 활용
2. 신호 데이터에 대한 AutoML 활용
3. AutoML 모델을 활용한  
최적의 입력 데이터 추정

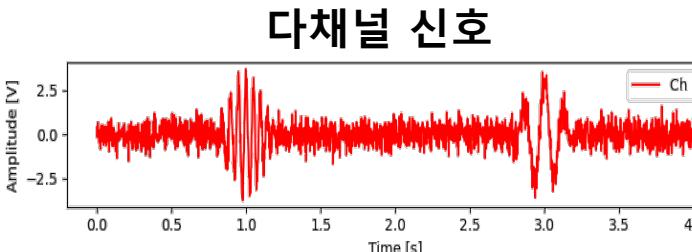
# Signal / Sequence Data → Feature Extraction → Tabular Data

**1차원 배열**

0.40	-0.16	0.02	0.68	0.33	-0.12	...	0.23
------	-------	------	------	------	-------	-----	------

**특징값 추출**

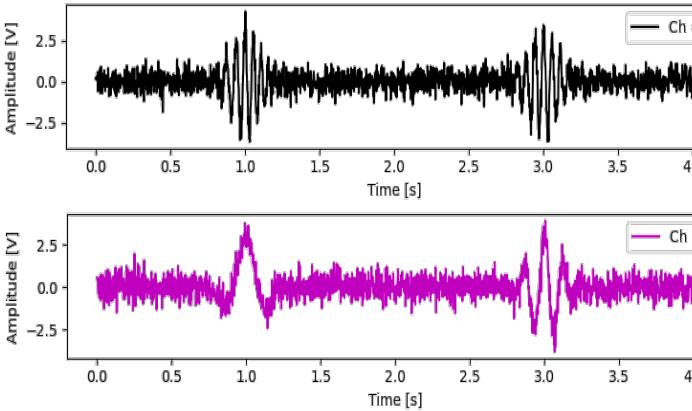
최소	최대	평균	표준편차
-0.45	0.68	0.11	0.32

**2차원 배열**

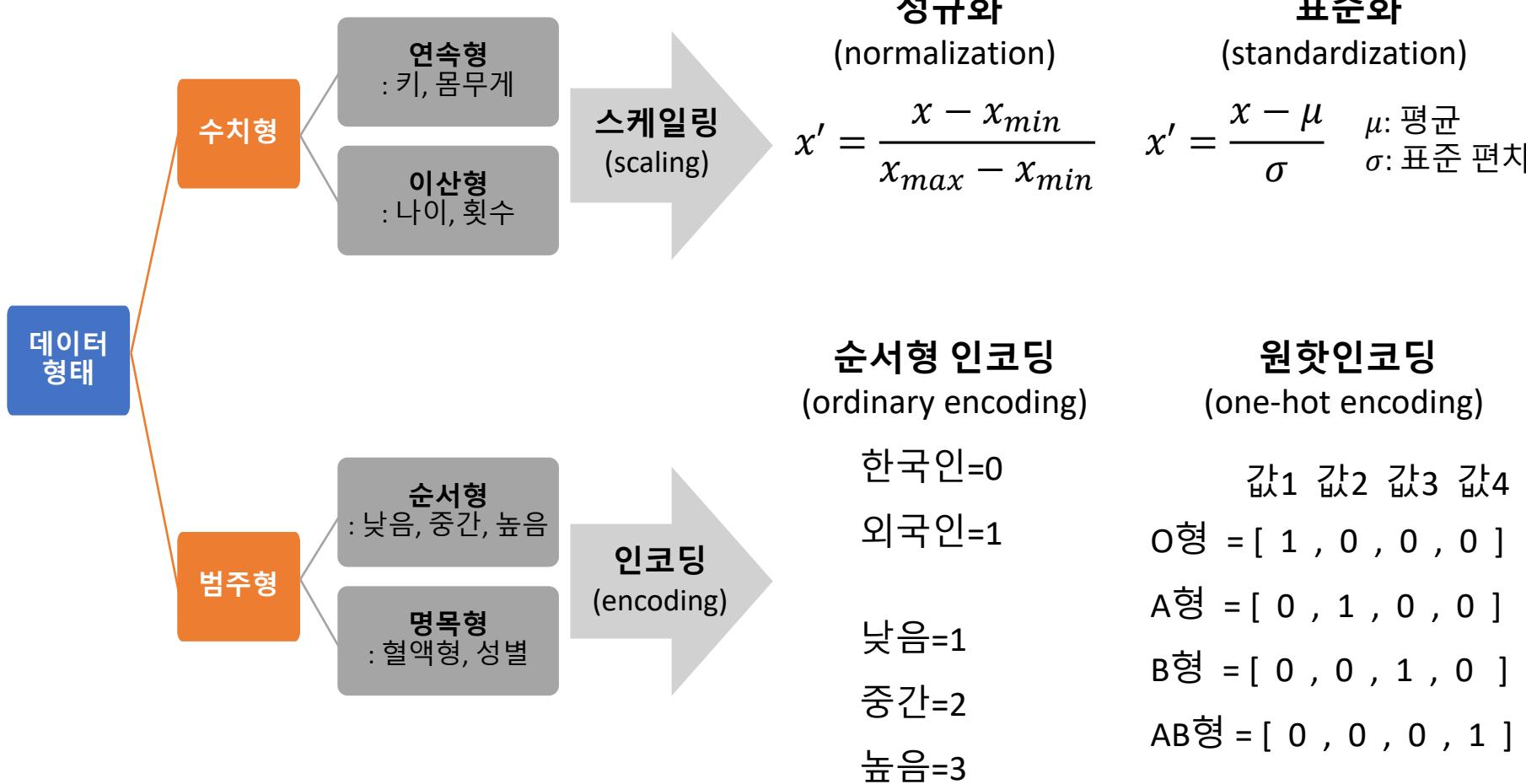
0.40	-0.16	0.02	0.68	0.33	-0.12	...	0.23
0.56	0.44	0.42	0.50	-0.07	-0.37	...	0.59
-0.14	0.01	-0.78	0.75	-0.15	-0.36	...	-0.56

**특징값 추출**

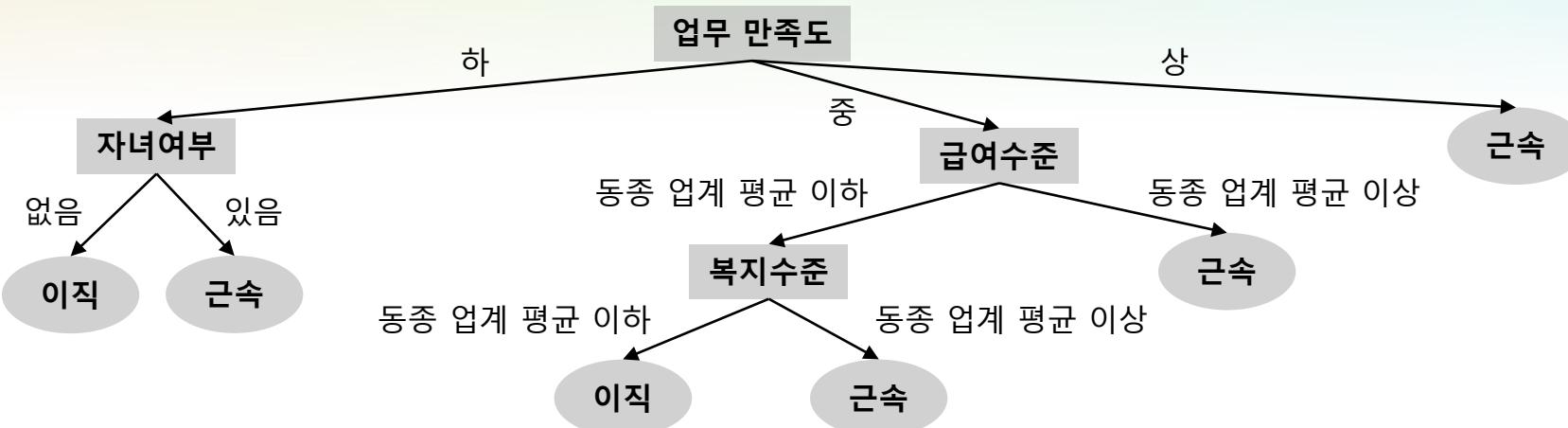
최소	최대	평균	표준편차
-0.45	0.68	0.11	0.32
-0.62	0.59	0.20	0.42
-0.78	0.75	-0.13	0.52



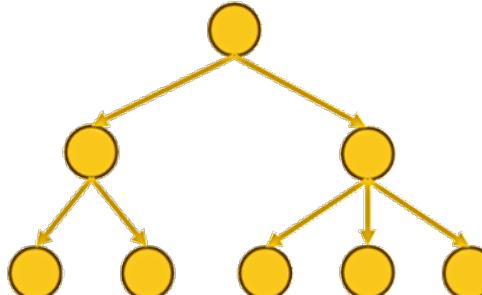
# Tabular Data Characteristics



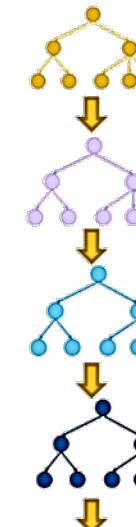
# Machine Learning :: Tree Models



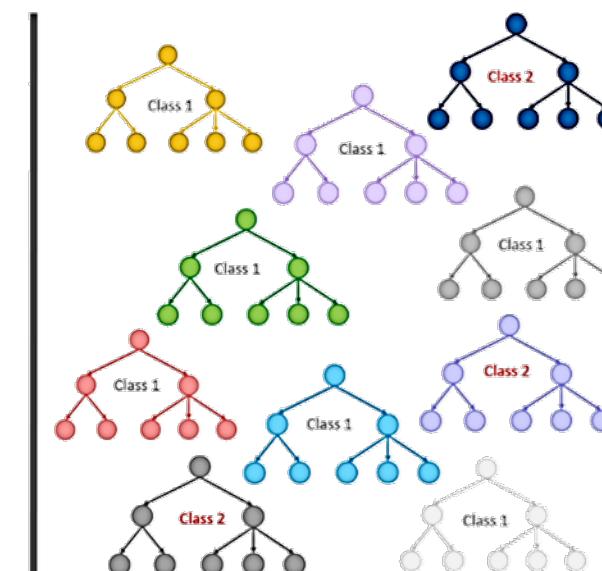
Decision Tree



Gradient Boosted Trees



Random Forest

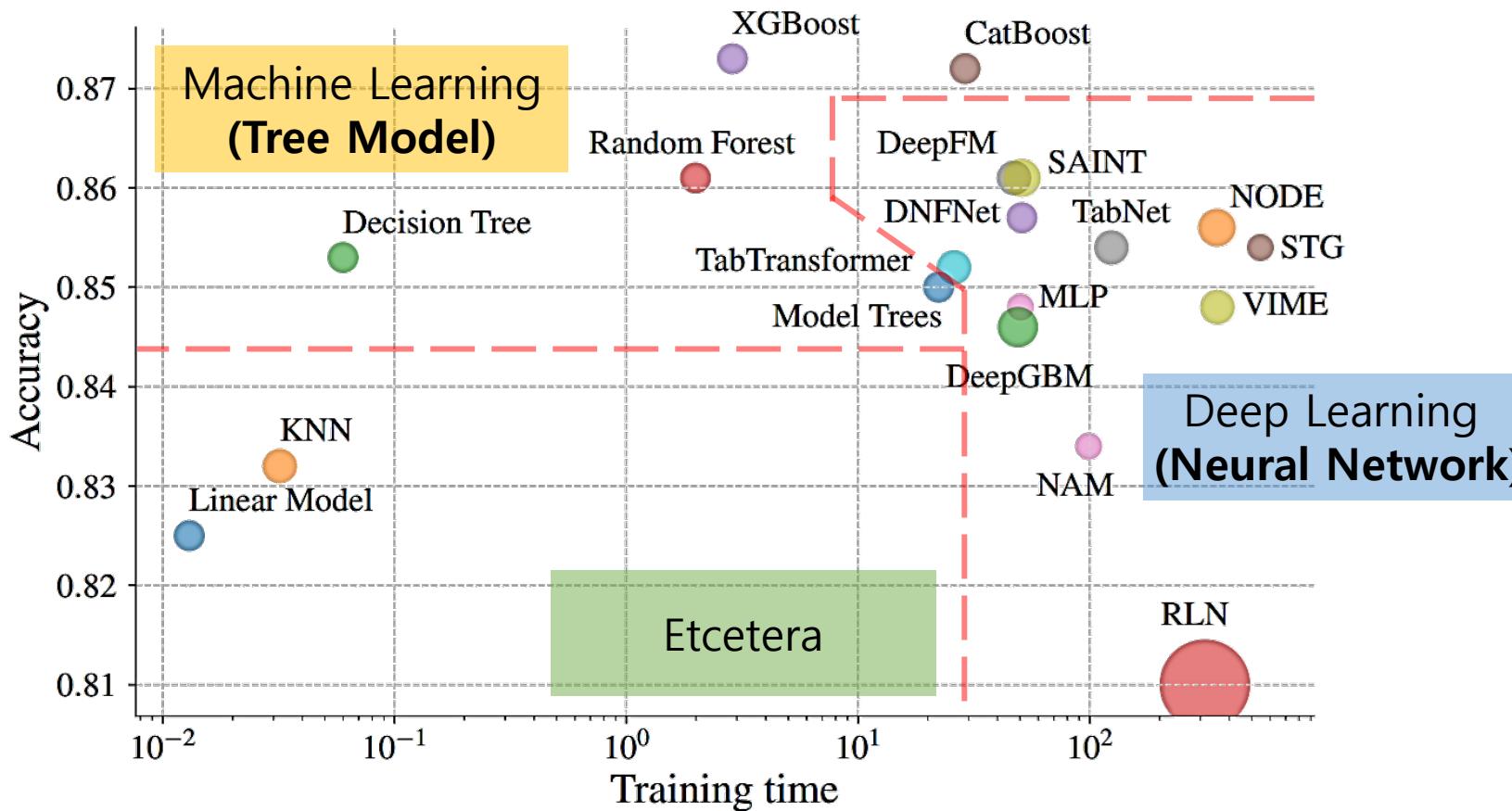


# ML > DL on Tabular Data

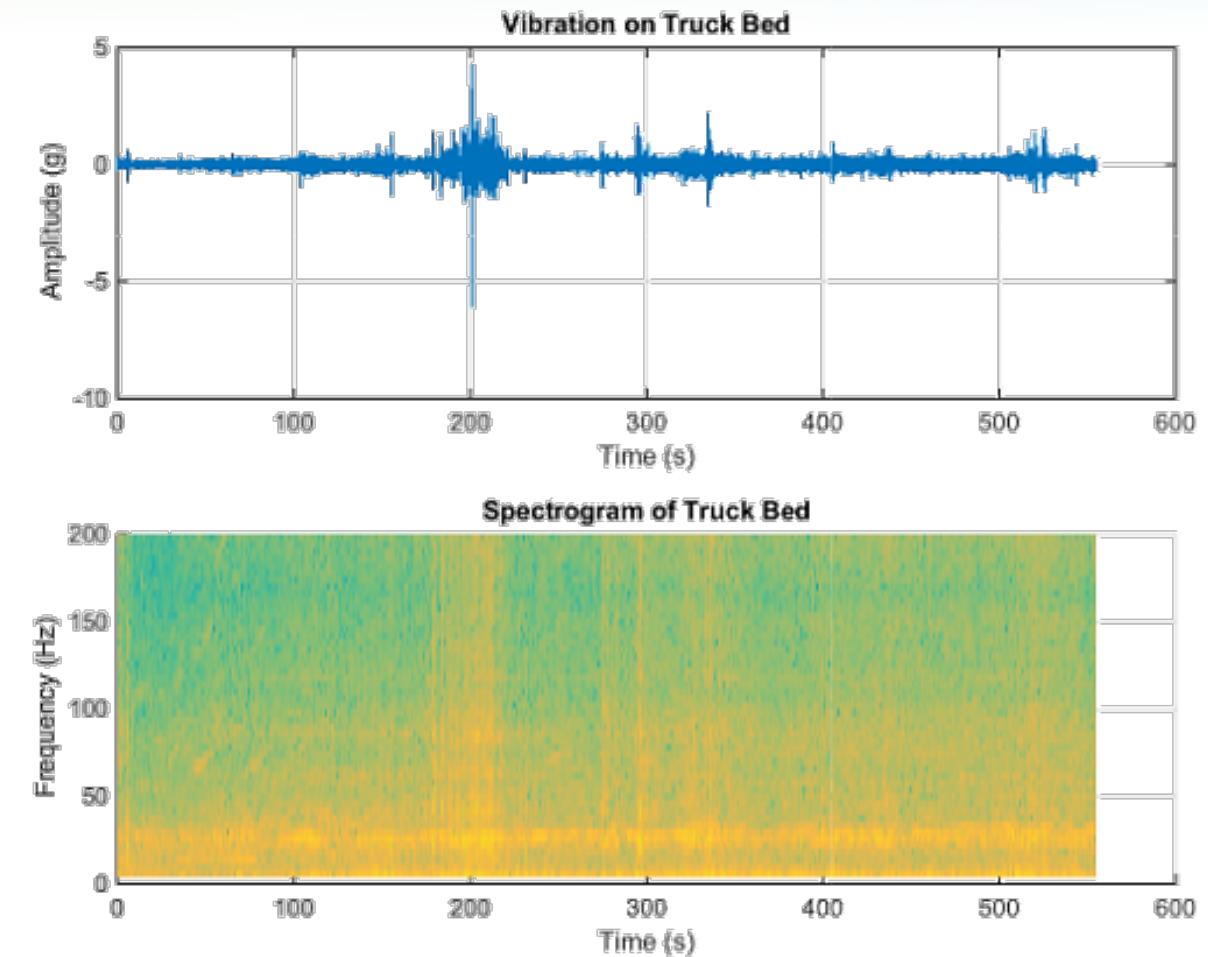
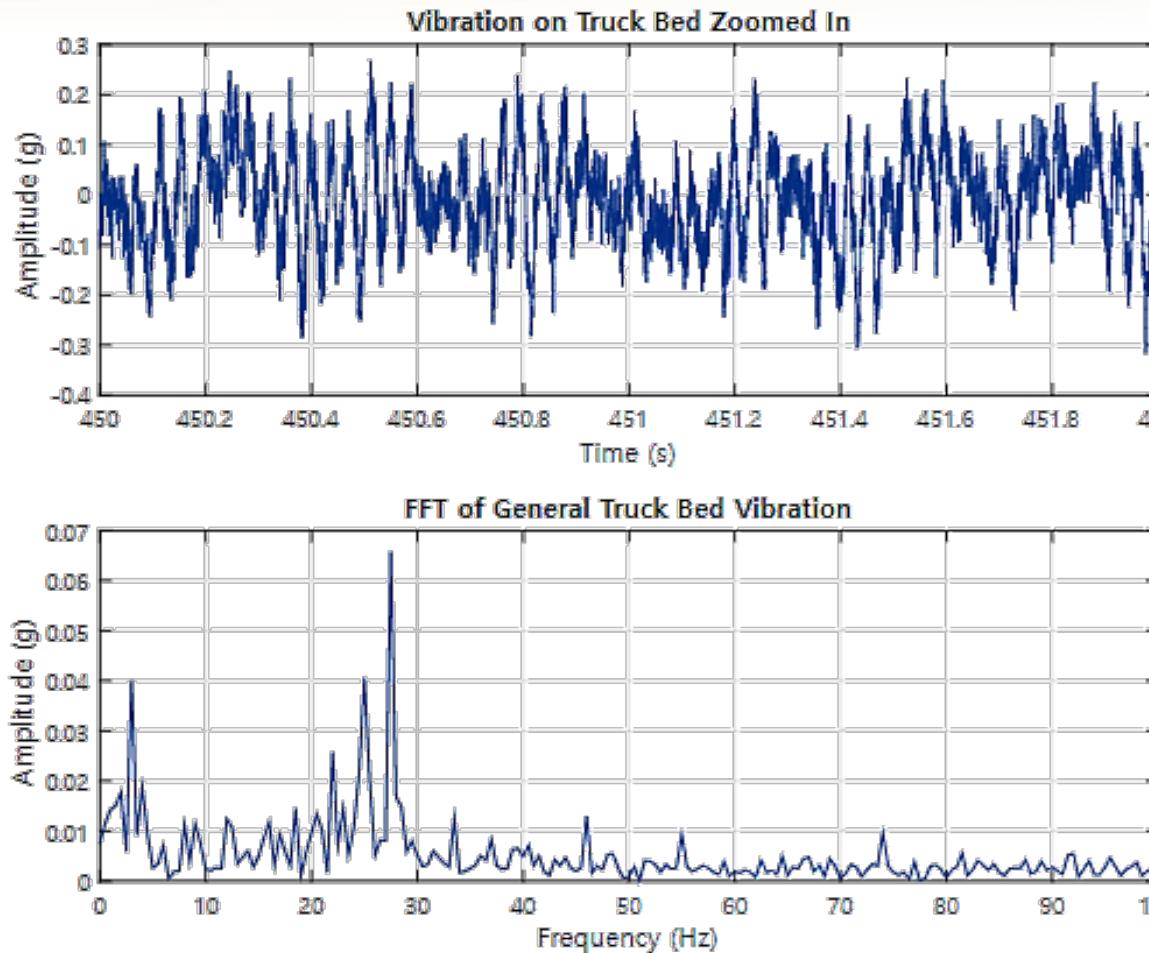
## Adult Income Dataset

※ The circle size reflects the accuracy standard deviation

- Input: basic information about individuals such as: age, gender, education, etc.
- Output: binary (0 or 1) represents high and low income.



# Signal → Frequency Spectrum → Spectrogram



# Load Dataset :: Signal Datasets

```

def load_signal_data(dir_base_path='datasets/sigpeak', split='train', shuffle=True, random_seed=42, verbose=True):
    try:
        T, X, Y = [], [], []
        for dir_name in tqdm(os.listdir(dir_base_path)):
            dir_split_path = os.path.join(dir_base_path, dir_name)

            indices_label_names = sorted([x for x in os.listdir(dir_split_path) if '.' not in x])
            if len(indices_label_names) > 0:
                indice = [int(x.split('_')[0]) for x in indices_label_names]
                label_names = [x.split('_')[1] for x in indices_label_names]

                for index_label_name, index in zip(indices_label_names, indice):
                    dir_signal_path = f'{dir_split_path}/{index_label_name}'
                    path_signal_list = sorted(glob.glob(f'{dir_signal_path}/*'))
                    for path_signal in path_signal_list:
                        T.append(np.loadtxt(path_signal, delimiter=',', skiprows=1)[ :, 0])
                        X.append(np.loadtxt(path_signal, delimiter=',', skiprows=1)[ :, 1])
                        Y.append(tf.keras.utils.to_categorical(index, num_classes=len(label_names), dtype='float32'))
                print('Loaded %d signals from %s' % (len(path_signal_list), dir_signal_path))
            else:
                filename_list = sorted(os.listdir(dir_split_path))
                for filename in tqdm(filename_list):
                    T.append(np.loadtxt(os.path.join(dir_split_path, filename), delimiter=',', skiprows=1)[ :, 0])
                    X.append(np.loadtxt(os.path.join(dir_split_path, filename), delimiter=',', skiprows=1)[ :, 1])
                    Y.append(float(filename.split('_')[0]))
                label_names = ""

        info = {'base_path': dir_base_path, 'split': split, 'label_names': label_names}

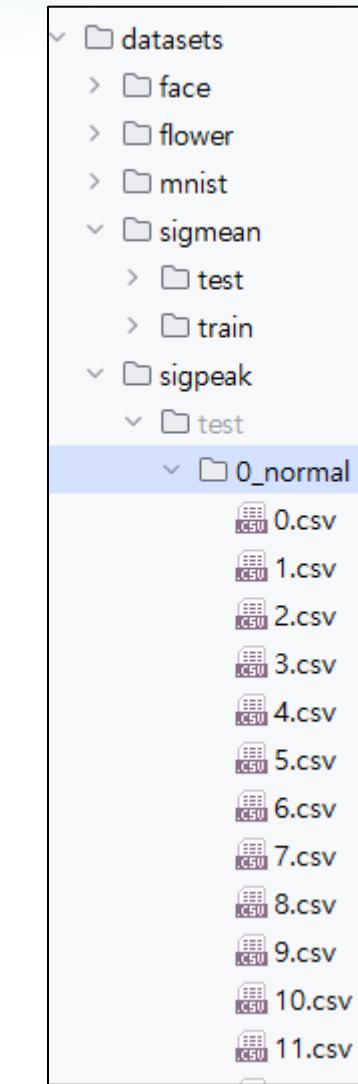
        if shuffle:
            dataset = list(zip(T, X, Y))
            np.random.seed(random_seed)
            np.random.shuffle(dataset)
            T, X, Y = zip(*dataset)

        if verbose:
            if label_names == "":
                print(dir_base_path, f'T length (shape): {len(T)} ({T[0].shape}), X length (shape): {len(X)} ({X[0].shape}), Y length (shape): {len(Y)} ({Y[0].shape})')
            else:
                print(dir_base_path, f'T length (shape): {len(T)} ({T[0].shape}), X length (shape): {len(X)} ({X[0].shape}), Y length (shape): {len(Y)} ({Y[0].shape})')

        return T, X, Y, label_names, info

    except Exception as e:
        print(f'Error: {e}')

```



# Tip :: How to create Preprocessing Code via ChatGPT?

 recommend typical signal preprocessing methods

 When working with signals, there are several typical preprocessing methods that can be applied to enhance the quality of the data or extract relevant features. Here are some commonly used signal preprocessing methods:

1. Filtering: Filtering is used to remove noise or unwanted frequencies from a signal. Common filtering techniques include low-pass, high-pass, band-pass, and notch filters. Filters can be implemented using techniques like moving average, Fourier transform, or digital filter design methods.
2. Normalization: Normalization is used to scale the signal values to a specific range. It helps ensure that different signals or features have comparable scales. Common normalization techniques include min-max scaling, z-score normalization, and unit norm scaling.
3. Smoothing: Smoothing techniques are applied to reduce noise or remove high-frequency variations in the signal. Methods such as moving average, median filtering, or Savitzky-Golay filtering can be used to smooth the signal while preserving important features.
4. Resampling: Resampling is used to change the sampling rate or time resolution of a signal. It can involve upsampling (increasing the sampling rate) or downsampling (decreasing the sampling rate). Techniques like interpolation or decimation are commonly used for resampling.
5. Detrending: Detrending is performed to remove any trend or baseline drift present in the signal. It helps to isolate the variations or fluctuations of interest. Detrending can be achieved using techniques such as polynomial fitting, moving average subtraction, or high-pass filtering.
6. Feature extraction: Feature extraction involves extracting relevant information or characteristics from the signal. This can include methods like Fourier transform, wavelet transform, time-frequency analysis (e.g., spectrograms), or statistical features (e.g., mean, variance, peak values).
7. Artifact removal: If the signal contains unwanted artifacts or interference, specific techniques can be employed to remove or reduce them. This may involve techniques such as baseline correction, artifact rejection based on thresholding, or blind source separation.
8. Segmentation: Signal segmentation involves dividing a continuous signal into smaller segments or windows. Segmentation can be useful for analyzing specific sections of the signal or for feature extraction on localized segments. Techniques like sliding windows or event-based segmentation can be employed.

 python codes

 Certainly! Here are some example Python codes for typical signal preprocessing methods:

## 1. Filtering (Low-pass Filter):

```
python 
import scipy.signal as signal

def lowpass_filter(signal_data, cutoff_freq, fs):
    nyquist_freq = 0.5 * fs
    normalized_cutoff_freq = cutoff_freq / nyquist_freq
    b, a = signal.butter(4, normalized_cutoff_freq, btype='low')
    filtered_signal = signal.lfilter(b, a, signal_data)
    return filtered_signal
```

## 1. Normalization (Min-Max Scaling):

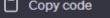
```
python 
def min_max_scaling(signal_data):
    min_val = min(signal_data)
    max_val = max(signal_data)
    scaled_signal = (signal_data - min_val) / (max_val - min_val)
    return scaled_signal
```

## 1. Smoothing (Moving Average):

```
python 
import numpy as np

def moving_average(signal_data, window_size):
    window = np.ones(window_size) / window_size
    smoothed_signal = np.convolve(signal_data, window, mode='same')
    return smoothed_signal
```

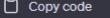
 1. Resampling (Upsampling using Linear Interpolation):

 python

```
from scipy.interpolate import interp1d

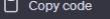
def upsample_signal(signal_data, original_fs, target_fs):
    original_time = np.arange(len(signal_data)) / original_fs
    target_time = np.arange(0, original_time[-1], 1 / target_fs)
    interpolator = interp1d(original_time, signal_data, kind='linear')
    upsampled_signal = interpolator(target_time)
    return upsampled_signal
```

## 1. Detrending (Linear Detrend):

 python

```
def linear_detrend(signal_data):
    detrended_signal = signal_data - np.linspace(signal_data[0], signal_data[-1], len(signal_data))
    return detrended_signal
```

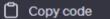
## 1. Feature Extraction (Power Spectral Density):

 python

```
from scipy.signal import Welch

def compute_psd(signal_data, fs):
    f, psd = Welch(signal_data, fs)
    return f, psd
```

## 1. Artifact Removal (Baseline Correction):

 python

```
def baseline_correction(signal_data):
    baseline = signal_data[0] # Assume baseline is the first value
    corrected_signal = signal_data - baseline
    return corrected_signal
```

# Load Dataset :: Plot Signal Data

```

def plot_signals(axis, signals, labels, label_names, i_list=[], axis_name='Time [s]'):
try:
    axis = np.array(axis)
    signals = np.array(signals)
    labels = np.array(labels)
except:
    pass
i_list = range(len(signals)) if i_list == [] else i_list

fig = plt.figure(figsize=(6, 3 * len(i_list)))

ax_list = []
for i_ax, i_signal in enumerate(i_list):
    a = axis[i_signal]
    x = signals[i_signal]
    y = labels[i_signal].astype(np.float16)

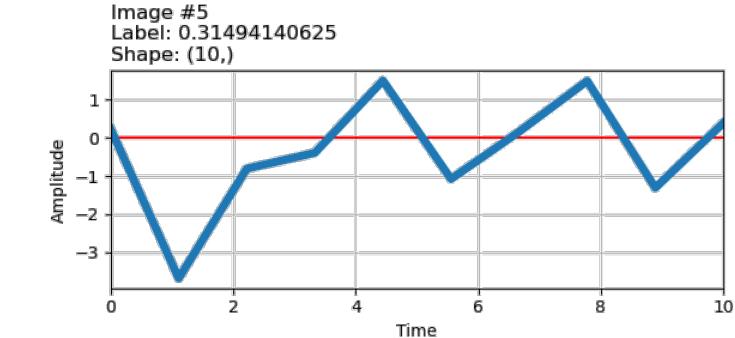
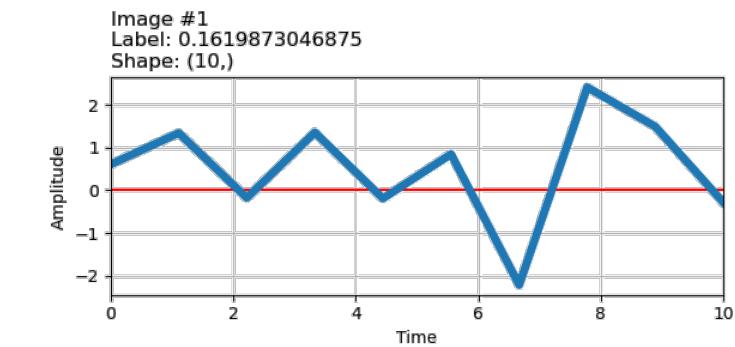
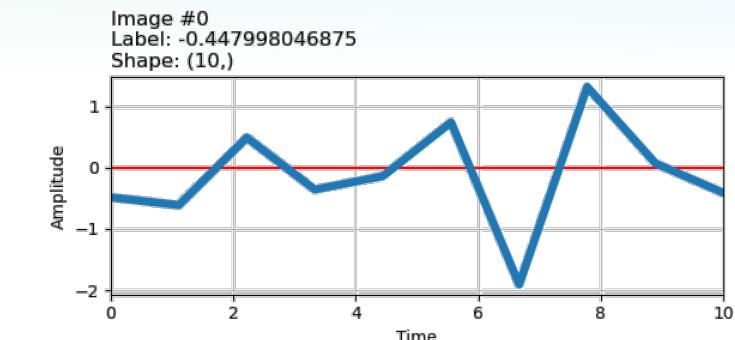
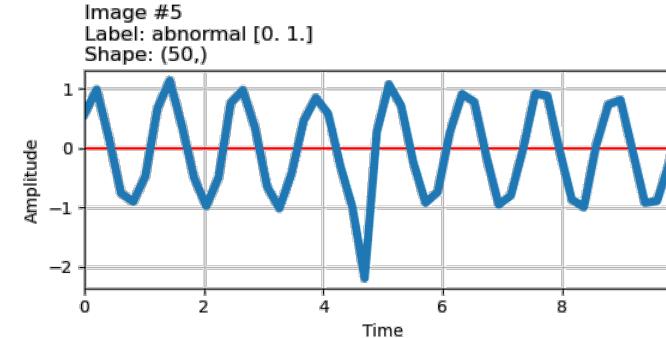
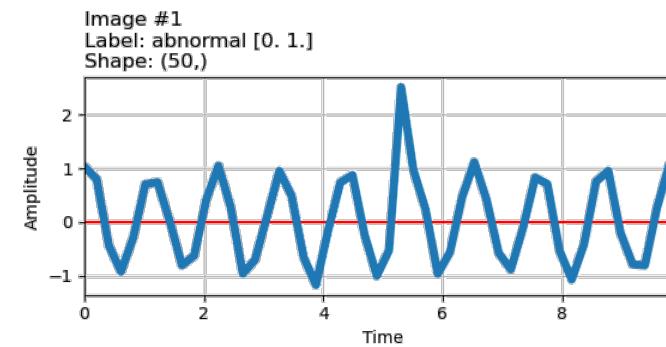
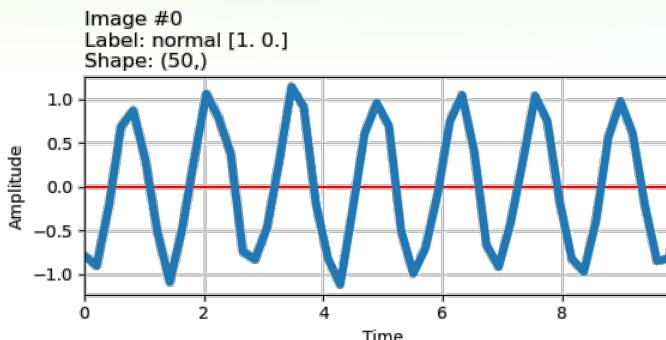
    try:
        y = f'{label_names[np.argmax(y)]} {y}'
    except:
        pass

    ax = fig.add_subplot(len(i_list), 1, i_ax + 1)
    ax.plot(a, x * 0, 'r')
    ax.plot(a, x, linewidth=5)
    ax.set_xlim(a[0], a[-1])
    ax.set_xlabel(axis_name)
    ax.set_ylabel('Value')
    ax.set_title(f'Image #{i_signal}\nLabel: {y}\nShape: {x.shape}', loc='left')
    ax.grid()

    ax_list.append(ax)

fig.tight_layout()
fig.show()

```



# Preprocessing

```
def preprocessing(T, X, Y, i_list=[]):
```

```
    TT, XX, YY = [], [], []
```

```
    for i in tqdm(i_list if len(i_list) > 0 else range(len(X))):
```

```
        t, x, y = T[i], X[i], Y[i]
```

```
        x = absolute_signal(x)
```

```
        x = min_max_scaling_signal(x)
```

```
        x = standardize_signal(x)
```

```
        x = moving_average_signal(x, window_size=10)
```

```
        x = linear_detrend_signal(x)
```

```
        x = remove_outlier(x, threshold=1)
```

```
        x = lowpass_filter(x, t[1] - t[0], cutoff_freq=1)
```

```
        x = highpass_filter(x, t[1] - t[0], cutoff_freq=1)
```

```
        t, x = compute_psd(x, t[1] - t[0])
```

```
        t, x = compute_fft(x, t[1] - t[0])
```

```
        TT.append(t)
```

```
        XX.append(x)
```

```
        YY.append(y)
```

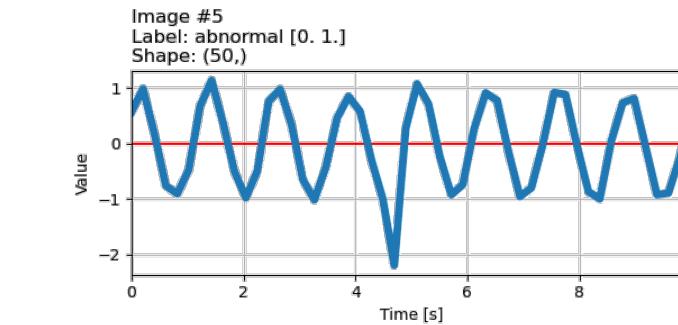
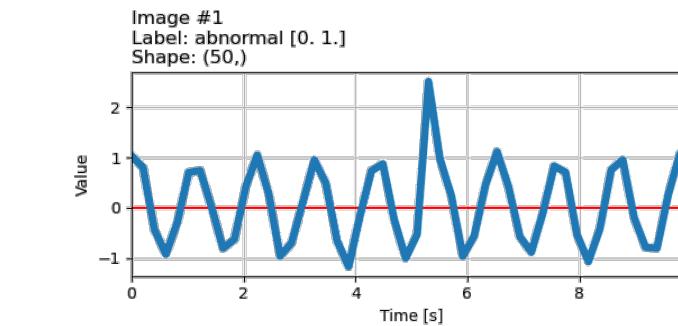
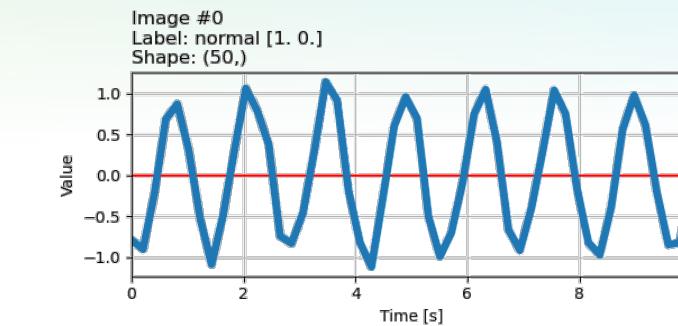
```
    XX, YY = np.array(XX), np.array(YY)
```

```
    return TT, XX, YY
```

```
T, X, Y, label_names, info = load_signal_data(dir_base_path='drive/MyDrive/KIRD/datasets/sigpeak', split='train', shuffle=True, random_seed=42, verbose=True)
plot_signals(axis=T, signals=X, labels=Y, label_names=label_names, i_list=[0, 1, 5])
```

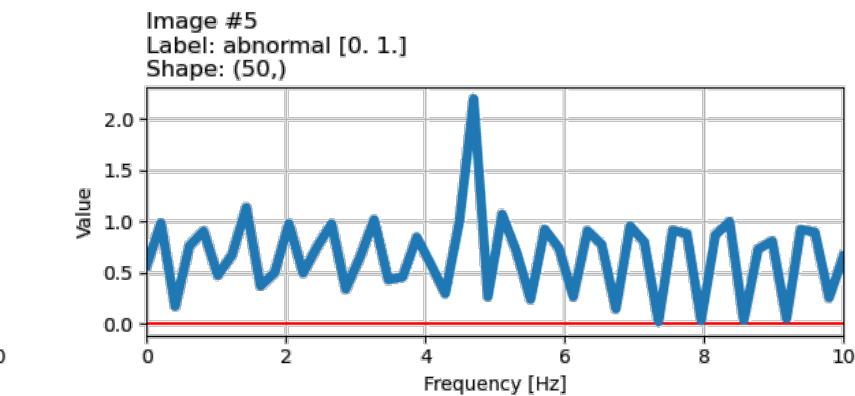
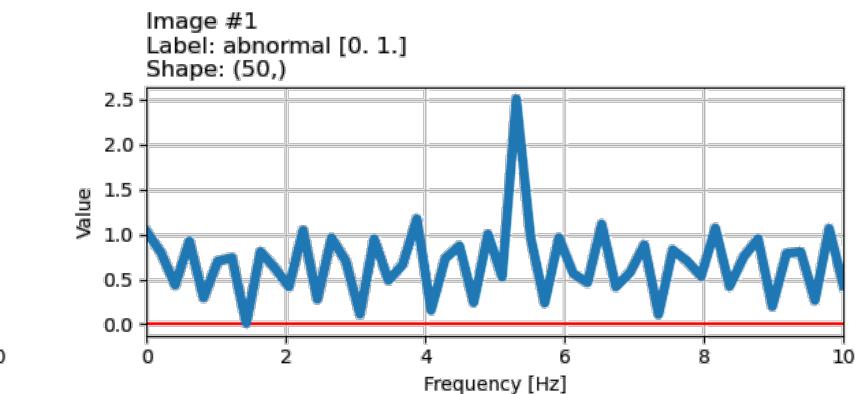
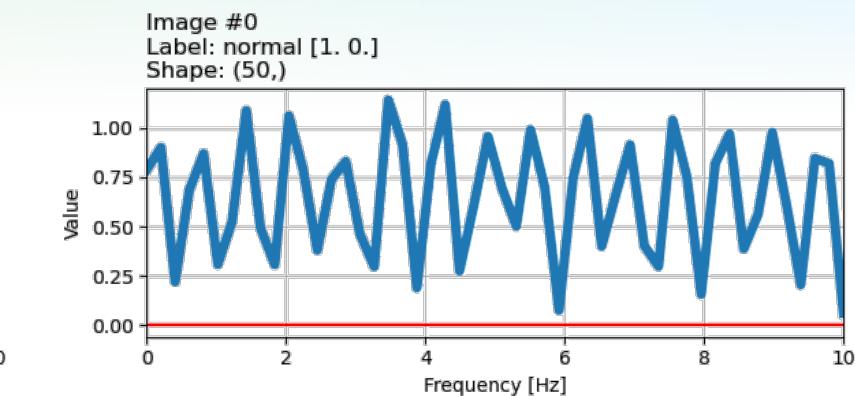
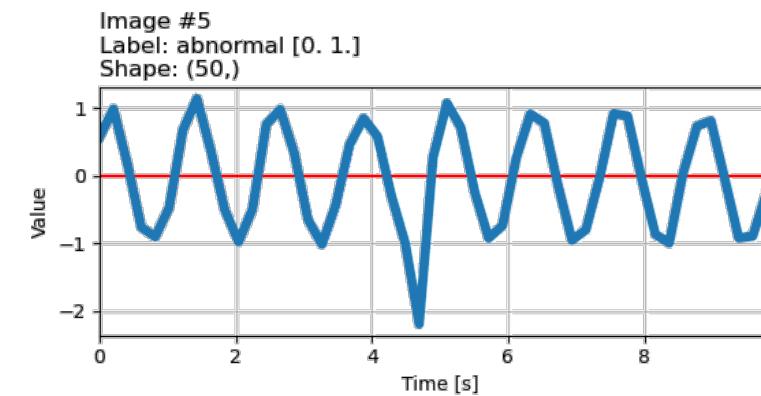
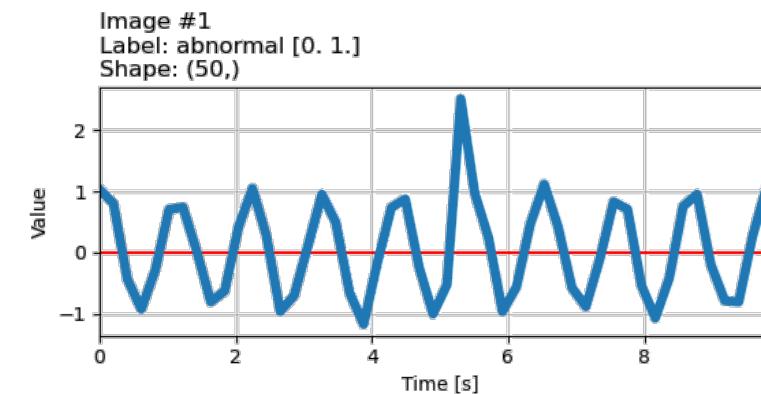
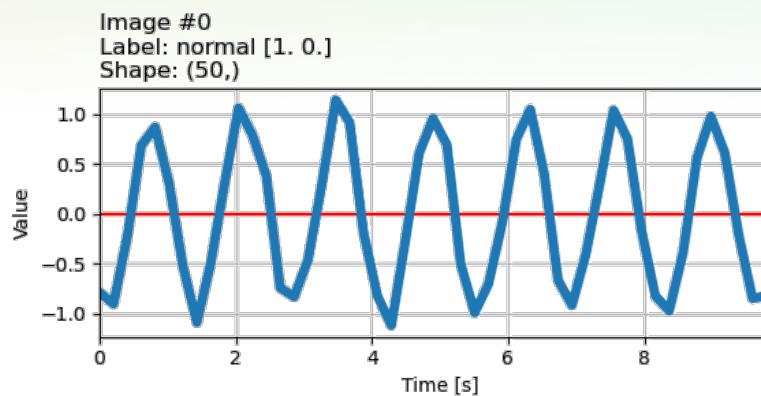
```
TT, XX, YY = preprocessing(T, X, Y)
```

```
plot_signals(axis=TT, signals=XX, labels=YY, label_names=label_names, i_list=[0, 1, 5], axis_name='Frequency [Hz]')
```



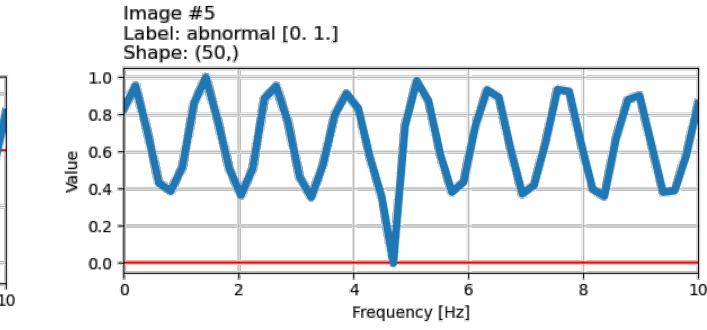
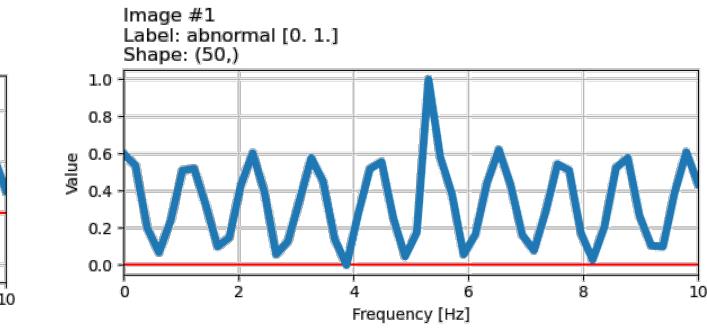
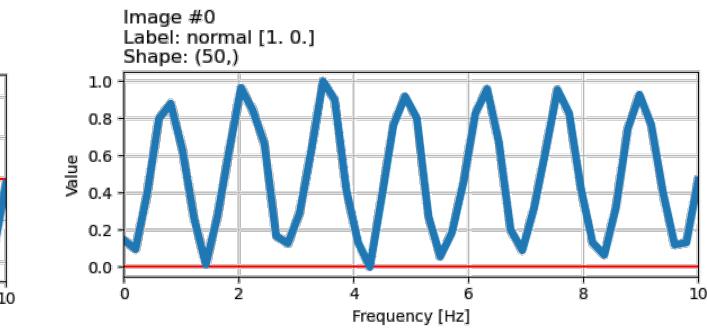
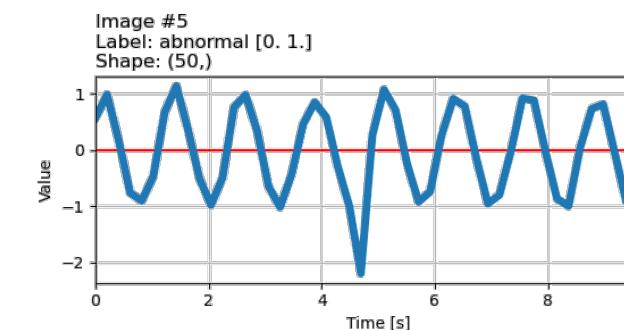
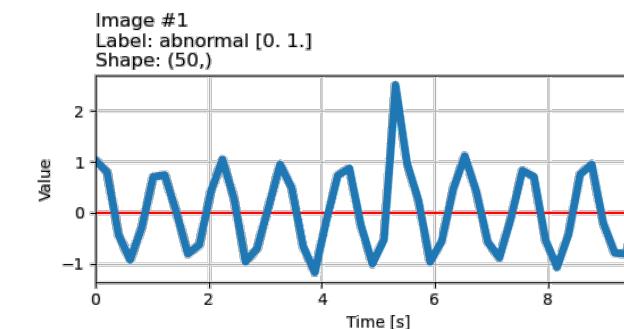
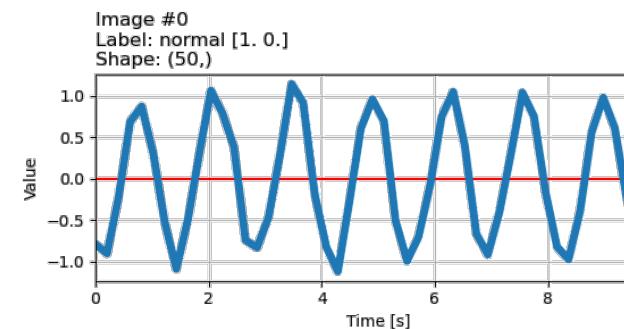
# Preprocessing :: absolute\_signal

```
def absolute_signal(signal_data):
    absolute_signal = np.abs(signal_data)
    return absolute_signal
```



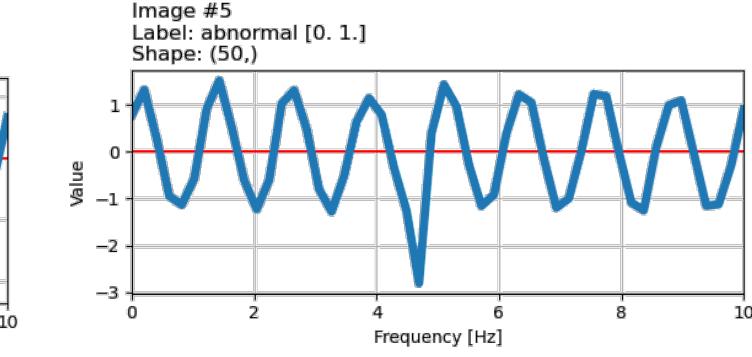
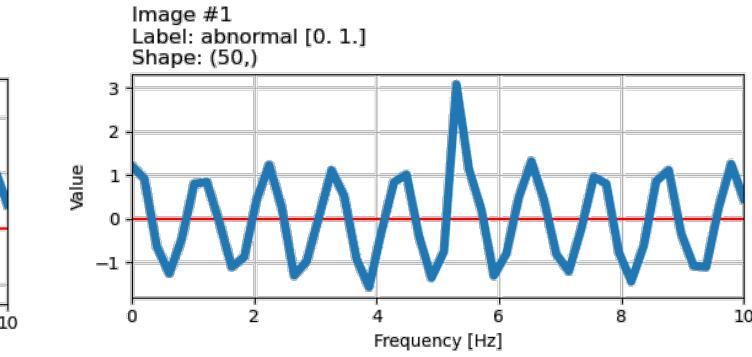
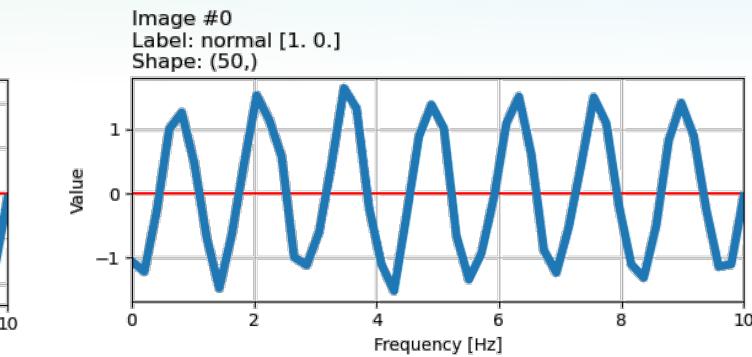
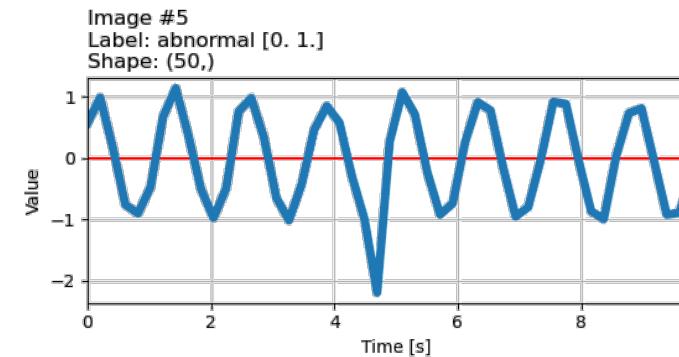
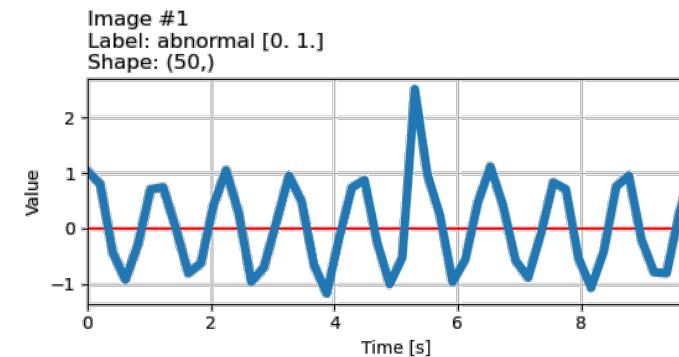
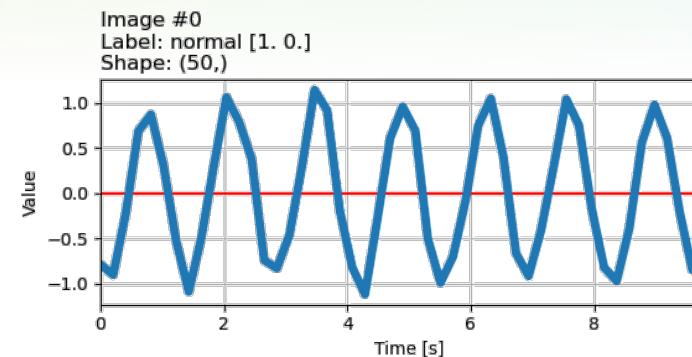
# Preprocessing :: min\_max\_scaling\_signal

```
def min_max_scaling_signal(signal_data):
    min_val = min(signal_data)
    max_val = max(signal_data)
    scaled_signal = (signal_data - min_val) / (max_val - min_val)
    return scaled_signal
```



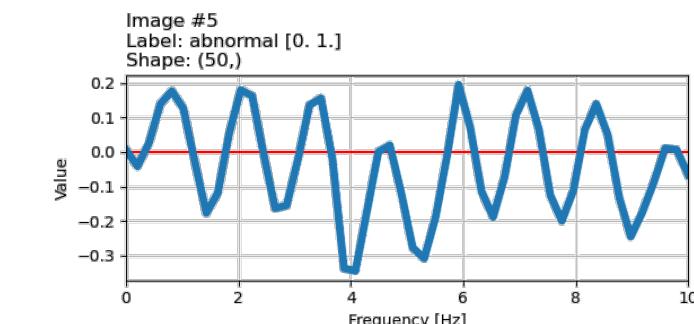
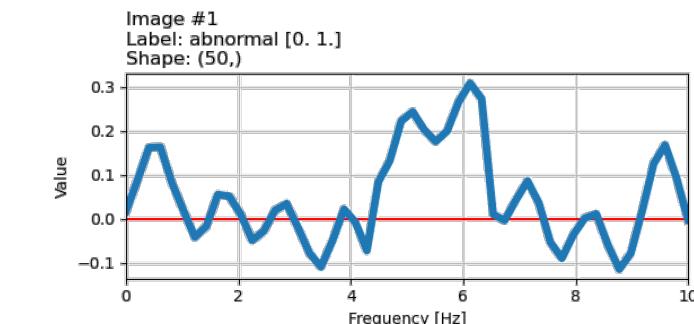
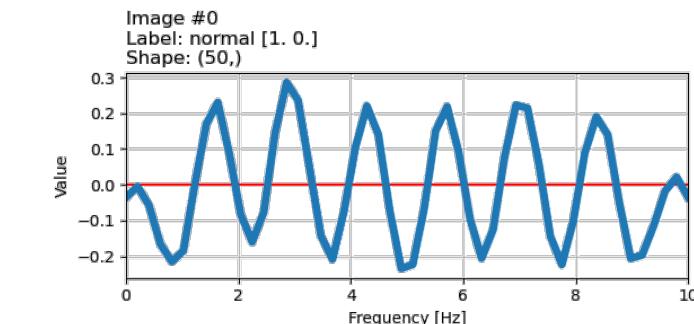
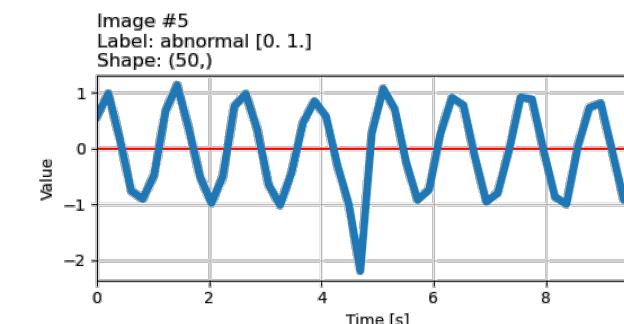
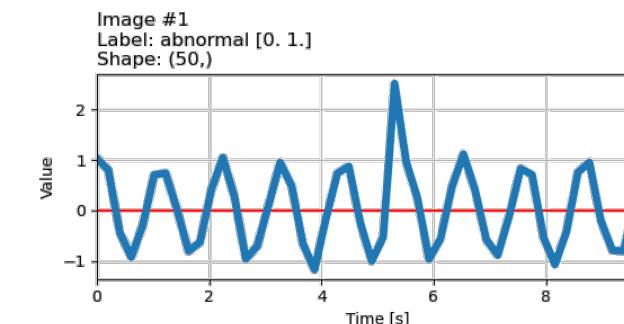
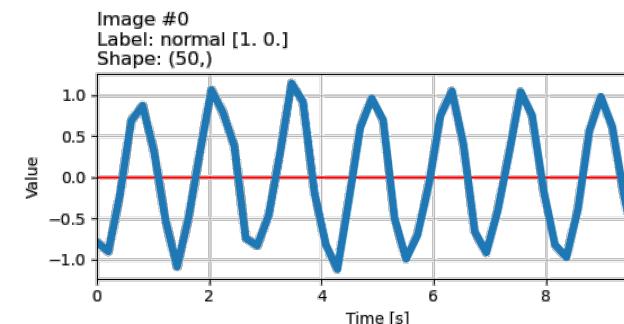
# Preprocessing :: standardize\_signal

```
def standardize_signal(signal_data):
    mean = np.mean(signal_data)
    std = np.std(signal_data)
    standardized_signal = (signal_data - mean) / std
    return standardized_signal
```



# Preprocessing :: moving\_average\_signal

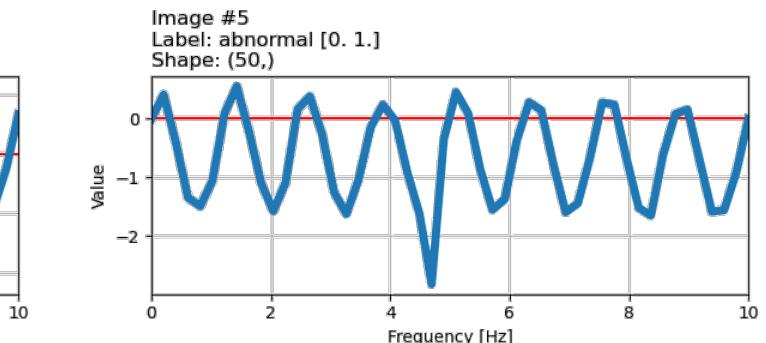
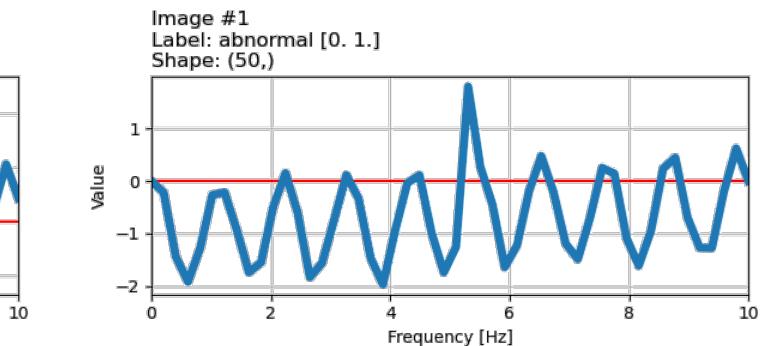
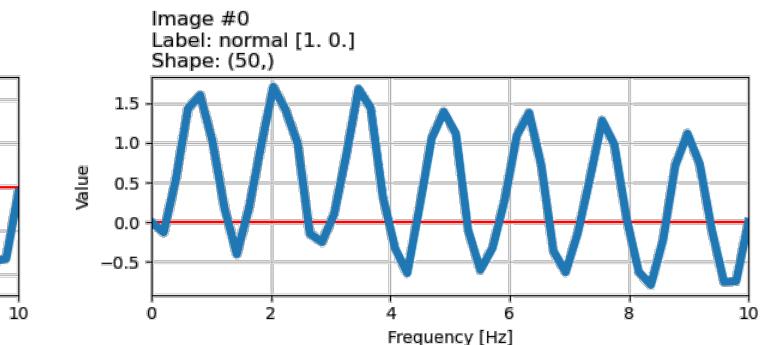
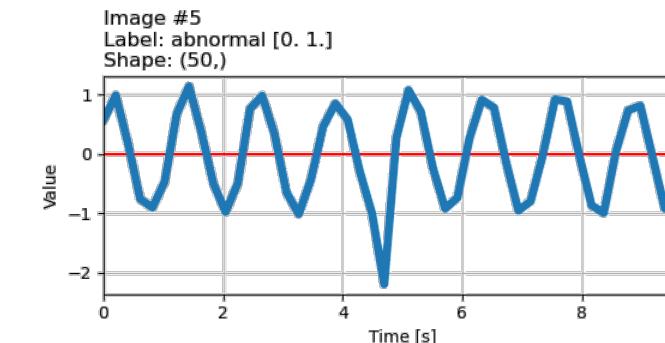
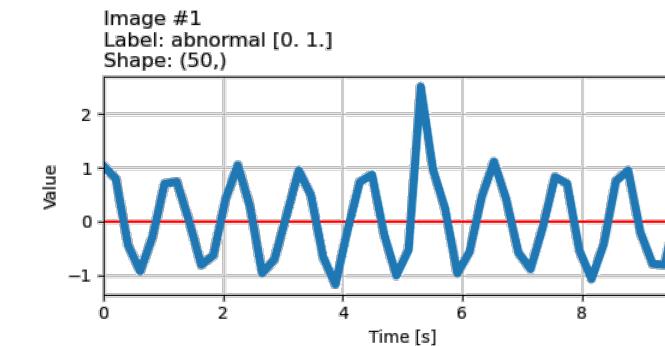
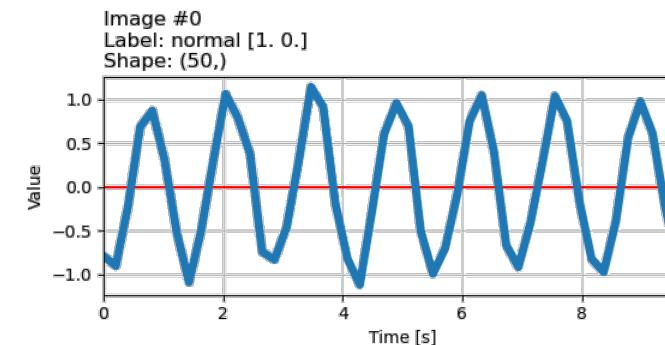
```
def moving_average_signal(signal_data, window_size):
    window = np.ones(window_size) / window_size
    smoothed_signal = np.convolve(signal_data, window, mode='same')
    return smoothed_signal
```



# Preprocessing :: linear\_detrend\_signal

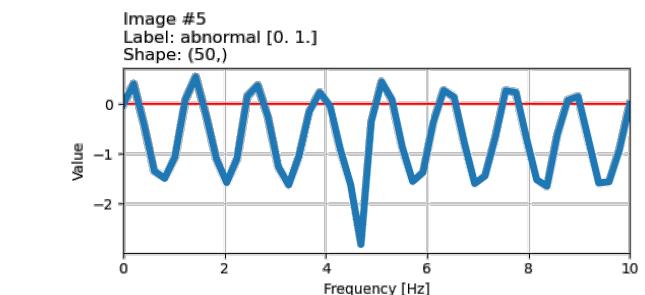
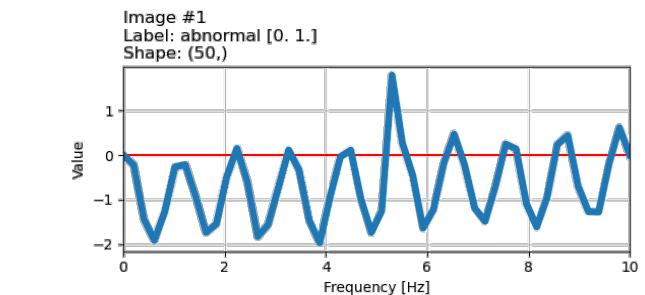
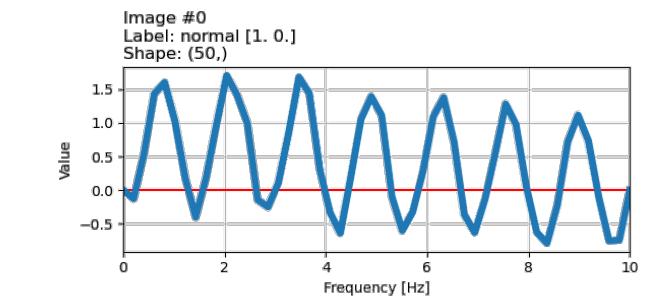
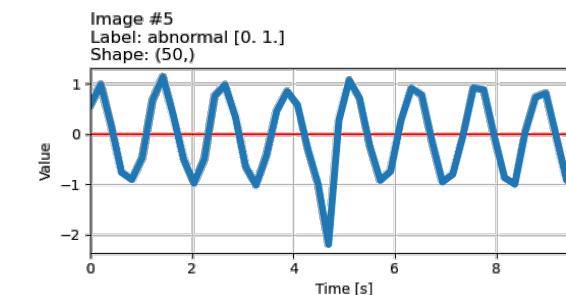
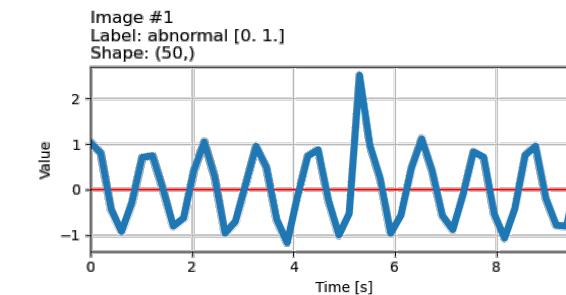
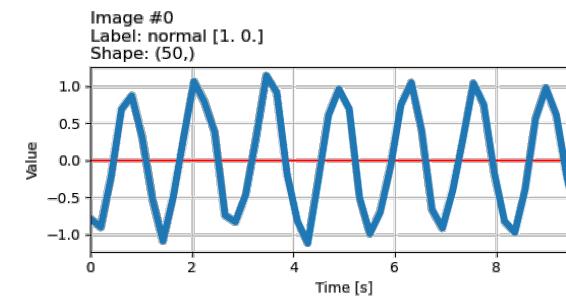
```
def linear_detrend_signal(signal_data):
```

```
    detrended_signal = signal_data - np.linspace(signal_data[0], signal_data[-1], len(signal_data))
    return detrended_signal
```



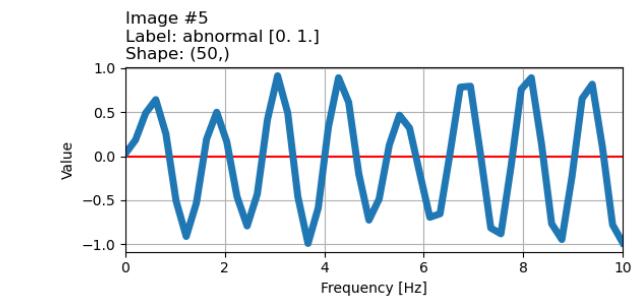
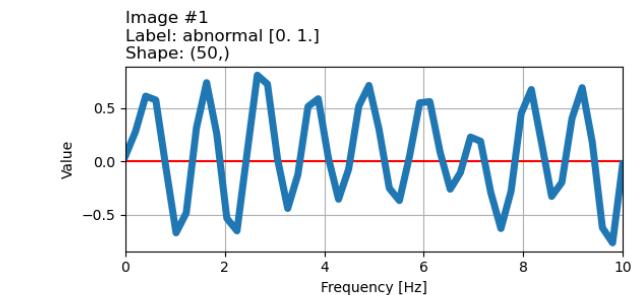
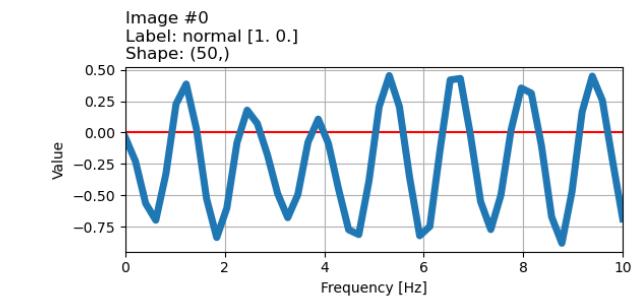
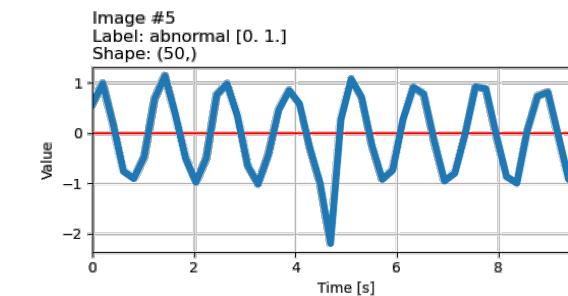
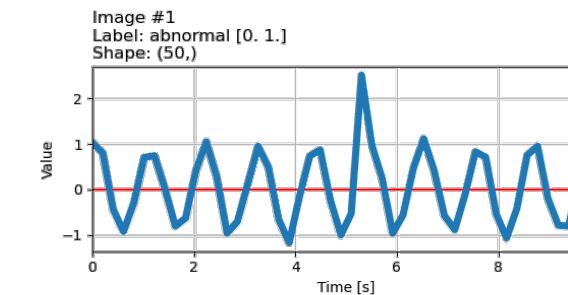
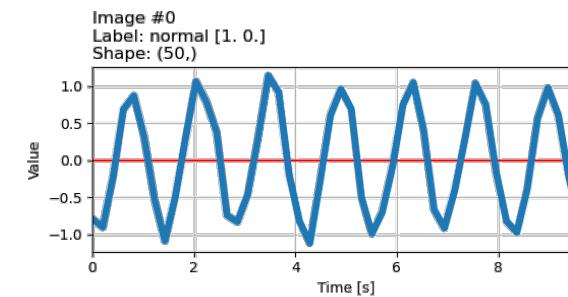
# Preprocessing :: remove\_outlier

```
def remove_outlier(signal_data, threshold=3.5):
    median = np.median(signal_data)
    diff = np.abs(signal_data - median)
    med_abs_deviation = np.median(diff)
    modified_z_score = 0.6745 * diff / med_abs_deviation if med_abs_deviation != 0 else 0
    modified_z_score[modified_z_score == 0] = 1e-18
    indices = np.where(modified_z_score > threshold)
    signal_data[indices] = median
    return signal_data
```



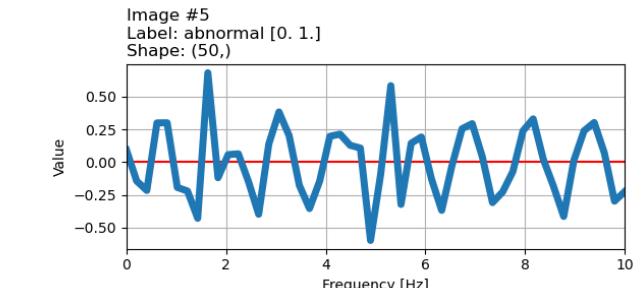
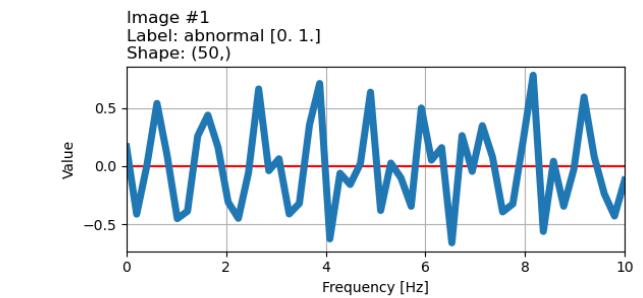
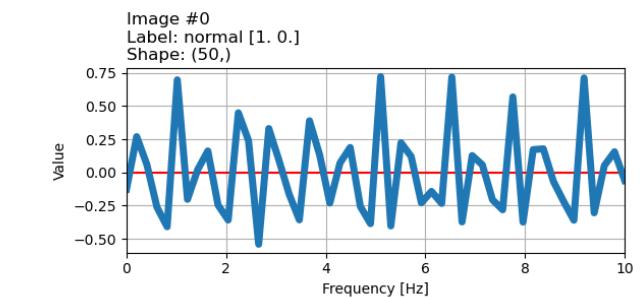
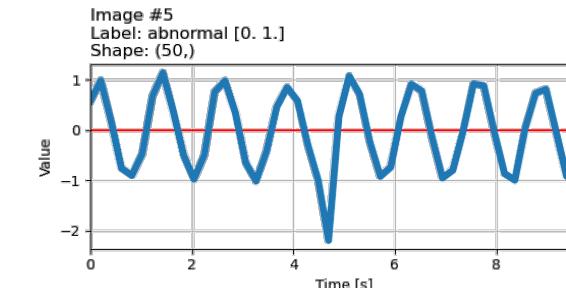
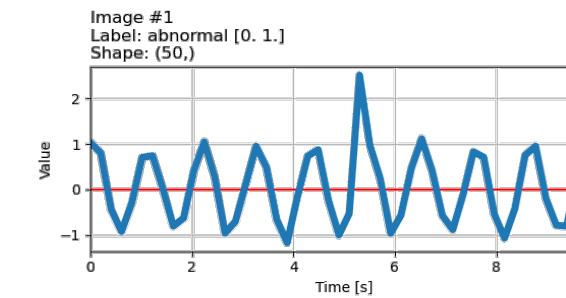
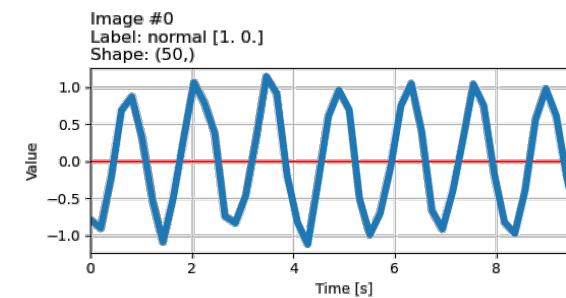
# Preprocessing :: lowpass\_filter

```
def lowpass_filter(signal_data, time_delta, cutoff_freq, order=5):
    sample_rate = 1 / time_delta
    nyquist_freq = 0.5 * sample_rate
    normalized_cutoff_freq = cutoff_freq / nyquist_freq
    b, a = signal.butter(order, normalized_cutoff_freq, btype='low')
    filtered_signal = signal.lfilter(b, a, signal_data)
    return filtered_signal
```



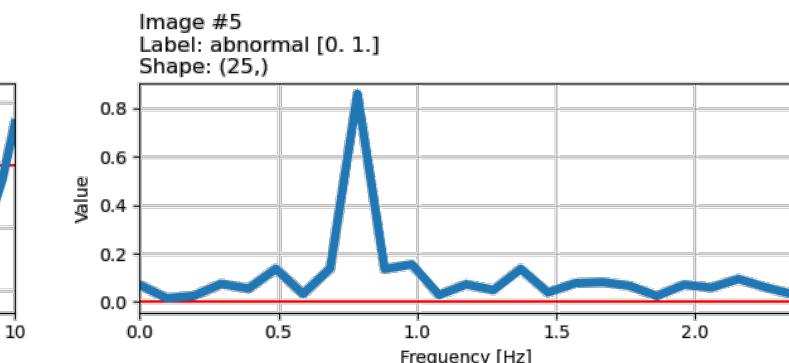
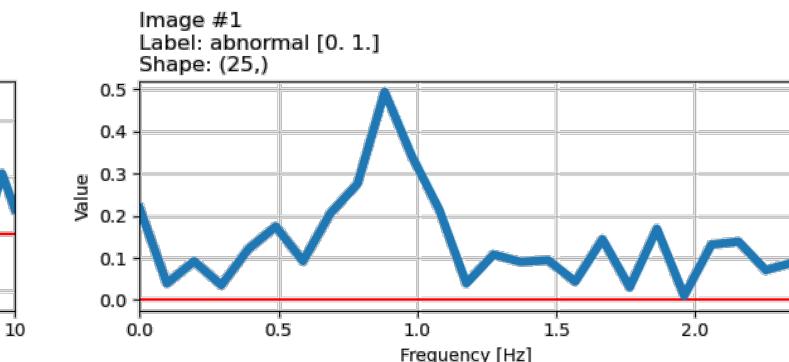
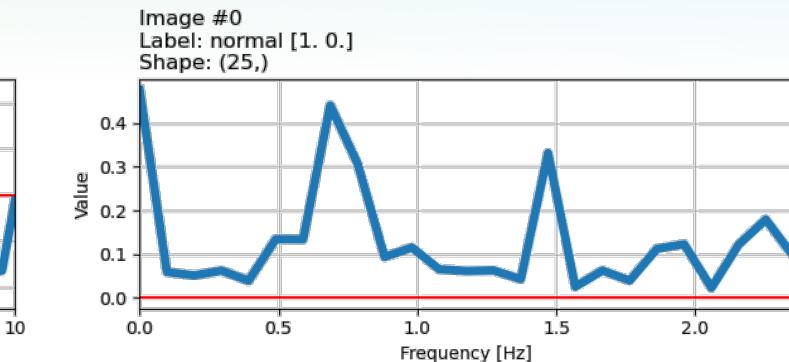
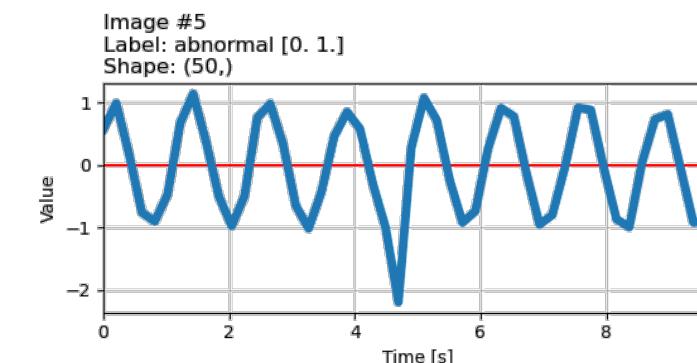
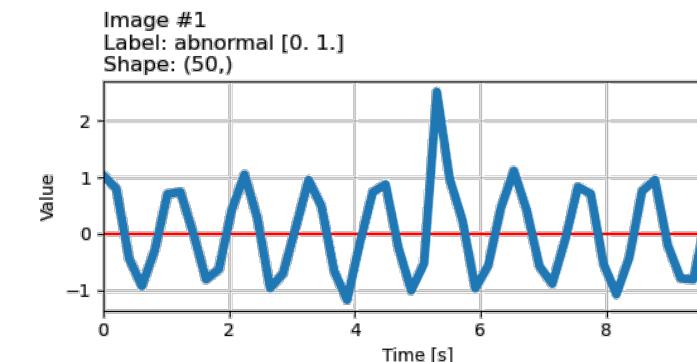
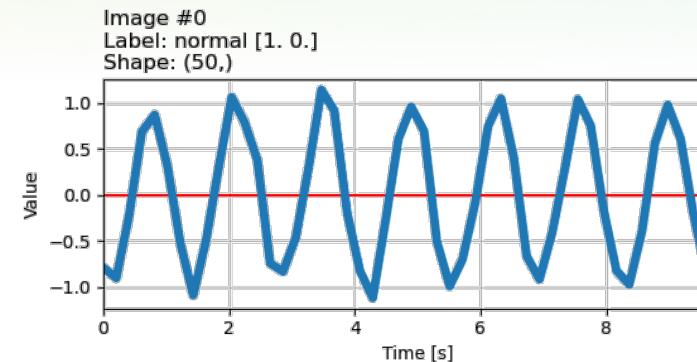
# Preprocessing :: highpass\_filter

```
def highpass_filter(signal_data, time_delta, cutoff_freq, order=5):
    sample_rate = 1 / time_delta
    nyquist_freq = 0.5 * sample_rate
    normalized_cutoff_freq = cutoff_freq / nyquist_freq
    b, a = signal.butter(order, normalized_cutoff_freq, btype='high')
    filtered_signal = signal.lfilter(b, a, signal_data)
    return filtered_signal
```



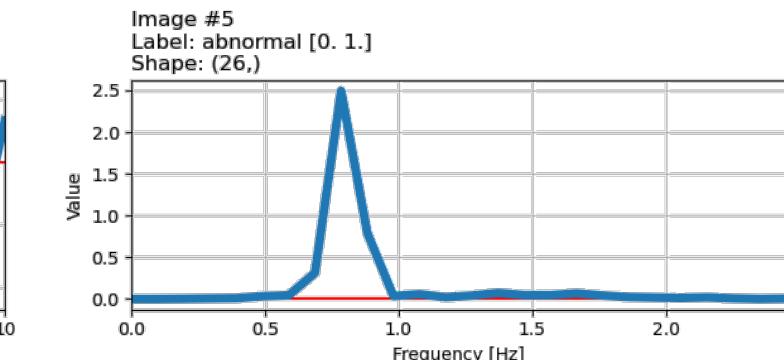
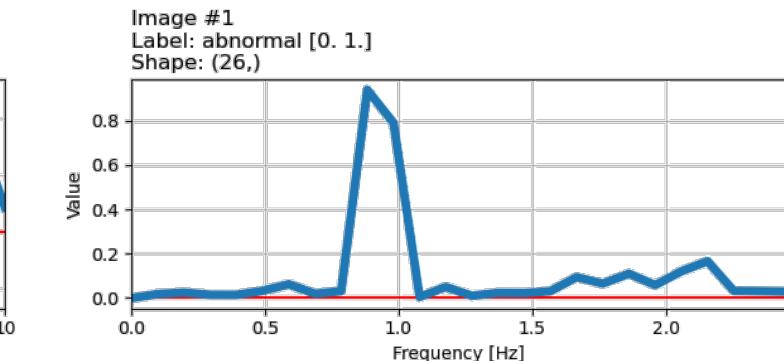
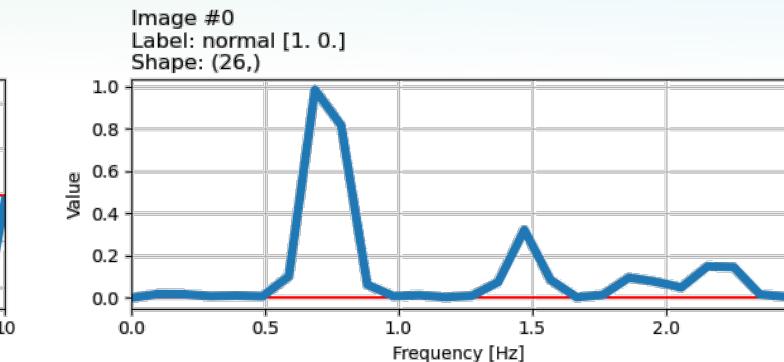
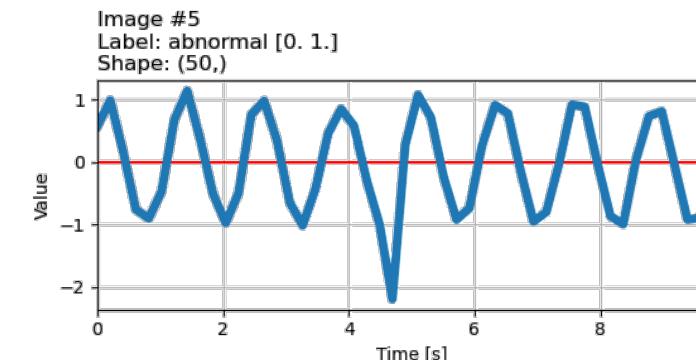
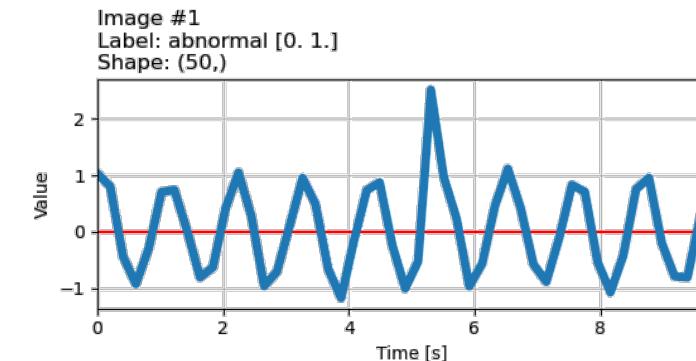
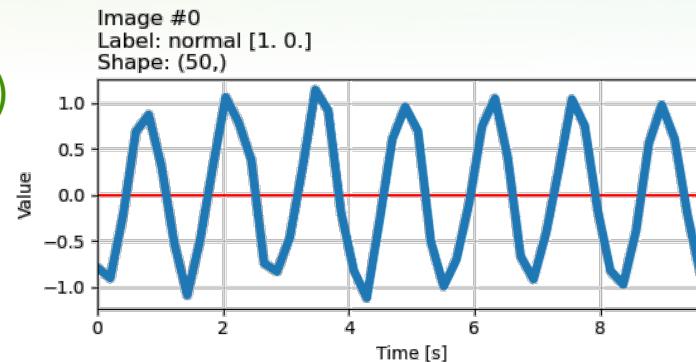
# Preprocessing :: compute\_fft

```
def compute_fft(signal_data, time_delta):
    n = len(signal_data)
    f = np.fft.fftfreq(n, d=time_delta)[:n // 2]
    mag = np.fft.fft(signal_data)
    mag = 2.0 / n * np.abs(mag[0:n // 2])
    return f, mag
```



# Preprocessing :: compute\_psd

```
def compute_psd(signal_data, time_delta):  
    sample_rate = 1 / time_delta  
    f, psd = signal.welch(signal_data, sample_rate)  
    return f, psd
```



# AutoML for Custom Dataset with Preprocessing

```

path_dataset = 'datasets/sigpeak'
objective = 'val_accuracy'
trial_max = 3 # model search trials
epoch_max = 100 # epochs for each model
project_name = 'AutoML_sig_clf'

def preprocessing(T, X, Y, i_list=[]):
    TT, XX, YY = [], [], []
    for i in tqdm(i_list if len(i_list) > 0 else range(len(X))):
        t, x, y = T[i], X[i], Y[i]
        # x = absolute_signal(x)
        # x = min_max_scaling_signal(x)
        # x = standardize_signal(x)
        # x = moving_average_signal(x, window_size=10)
        # x = linear_detrend_signal(x)
        # x = remove_outlier(x, threshold=1)
        # x = lowpass_filter(x, time_delta=t[1] - t[0], cutoff_freq=1)
        # x = highpass_filter(x, time_delta=t[1] - t[0], cutoff_freq=1)

        TT.append(t)
        XX.append(x)
        YY.append(y)

    XX, YY = np.array(XX), np.array(YY)

    return TT, XX, YY

```

Search: Running Trial #3		
Value	Best Value So Far	Hyperparameter
False	False	image_block_1/normalize
resnet	resnet	image_block_1/block_type
True	True	image_block_1/res_net_block_1/pretrained
resnet50	resnet50	image_block_1/res_net_block_1/version
False	False	image_block_1/res_net_block_1/imagenet_size
flatten	flatten	classification_head_1/spatial_reduction_1/reduction_type
0.5	0	classification_head_1/dropout
adam	adam	optimizer
1e-05	1e-05	learning_rate
False	False	image_block_1/res_net_block_1/trainable
Epoch 1/100		

```

T, X, Y, label_names, info = load_signal_data(dir_base_path=path_dataset, split='train')
TT, XX, YY = preprocessing(T, X, Y)
plot_signals(axis=TT, signals=XX, labels=YY, label_names=label_names, i_list=[0, 1, 5], axis_name='Time [s]')

XX_train, XX_val, YY_train, YY_val = train_test_split(np.array(XX)[:, np.newaxis, :], np.argmax(np.array(YY), axis=1), test_size=0.2, random_state=42)

input_node = ak.ImageInput()
output_node = ak.ImageBlock(augment=False)(input_node)
output_node = ak.ClassificationHead()(output_node)
model = ak.AutoModel(inputs=input_node, outputs=output_node, objective=objective, project_name=project_name, overwrite=True, max_trials=trial_max)
model.fit(XX_train, YY_train, validation_data

```

```

model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))
model_best.summary()

```

# Evaluation of Best Model searched by AutoML

```

model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))

T_test, X_test, Y_test, label_names, info = load_signal_data(dir_base_path=path_dataset, split='test')
TT_test, XX_test, YY_test = preprocessing(T_test, X_test, Y_test)
plot_signals(axis=TT_test, signals=XX_test, labels=YY_test, label_names=label_names, i_list=[0, 1, 5], axis_name='Time [s]')

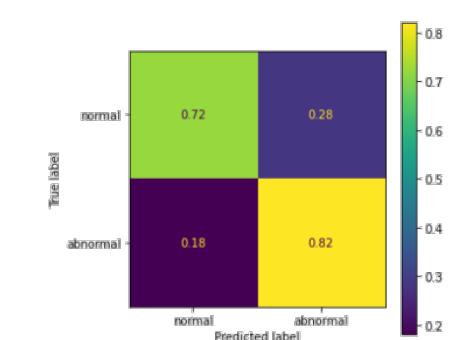
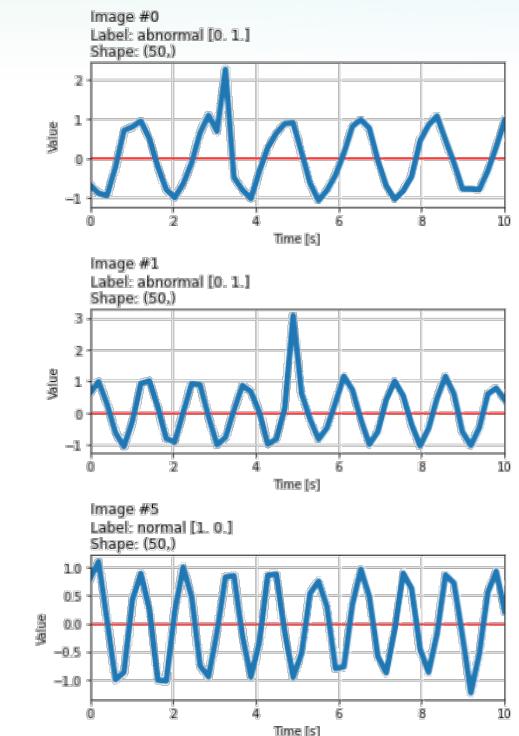
YY_predicted = model_best.predict(np.array(XX_test)[:, np.newaxis, :])
YY_predicted[YY_predicted >= 0.5] = 1
YY_predicted[YY_predicted < 0.5] = 0

accuracy = accuracy_score(np.argmax(np.array(YY_test), axis=1), YY_predicted)
print(f'{path_dataset} :: Test Dataset :: Accuracy = {accuracy:.3f}')

cm = confusion_matrix(np.argmax(np.array(YY_test), axis=1), YY_predicted, normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=label_names)
fig, ax = plt.subplots(figsize=(5, 5)) # Adjust the size (width, height) as needed
disp.plot(ax=ax, values_format='.2f')
plt.show()

```

datasets/sigpeak :: Test Dataset :: Accuracy = 0.770



# XAI :: Local Interpretable Model-agnostic Explanations (LIME)

```

from lime.lime_tabular import LimeTabularExplainer

i_check = 1
y_lim_factor = 1.5

model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))

T, X, Y, label_names, info = load_signal_data(dir_base_path=path_dataset, split='train')
TT_train, XX_train, YY_train = preprocessing(T, X, Y)
T_test, X_test, Y_test, label_names, info = load_signal_data(dir_base_path=path_dataset, split='test')
TT_test, XX_test, YY_test = preprocessing(T_test, X_test, Y_test)
num_features = XX_test.shape[-1]

YY_predicted = model_best.predict(np.array(XX_test)[:, np.newaxis, :])
YY_predicted[YY_predicted >= 0.5] = 1
YY_predicted[YY_predicted < 0.5] = 0
YY_probability = np.c_[1 - YY_predicted, YY_predicted]

def model_func(x):
    xx = x[:, np.newaxis, :]
    yy = model_best.predict(xx)
    return np.c_[1 - yy, yy]

explainer = LimeTabularExplainer(XX_train, feature_names=range(num_features), class_names=label_names, discretize_continuous=False)
explanation = explainer.explain_instance(XX_test[i_check], model_func, num_features=num_features, top_labels=len(label_names))
importance = np.array(explanation.as_list()) # feature_names, weights
importance[:, 1] = importance[:, 1] / max(importance[:, 1])
i_sort = np.argsort(importance[:, 0])
importance = importance[i_sort]

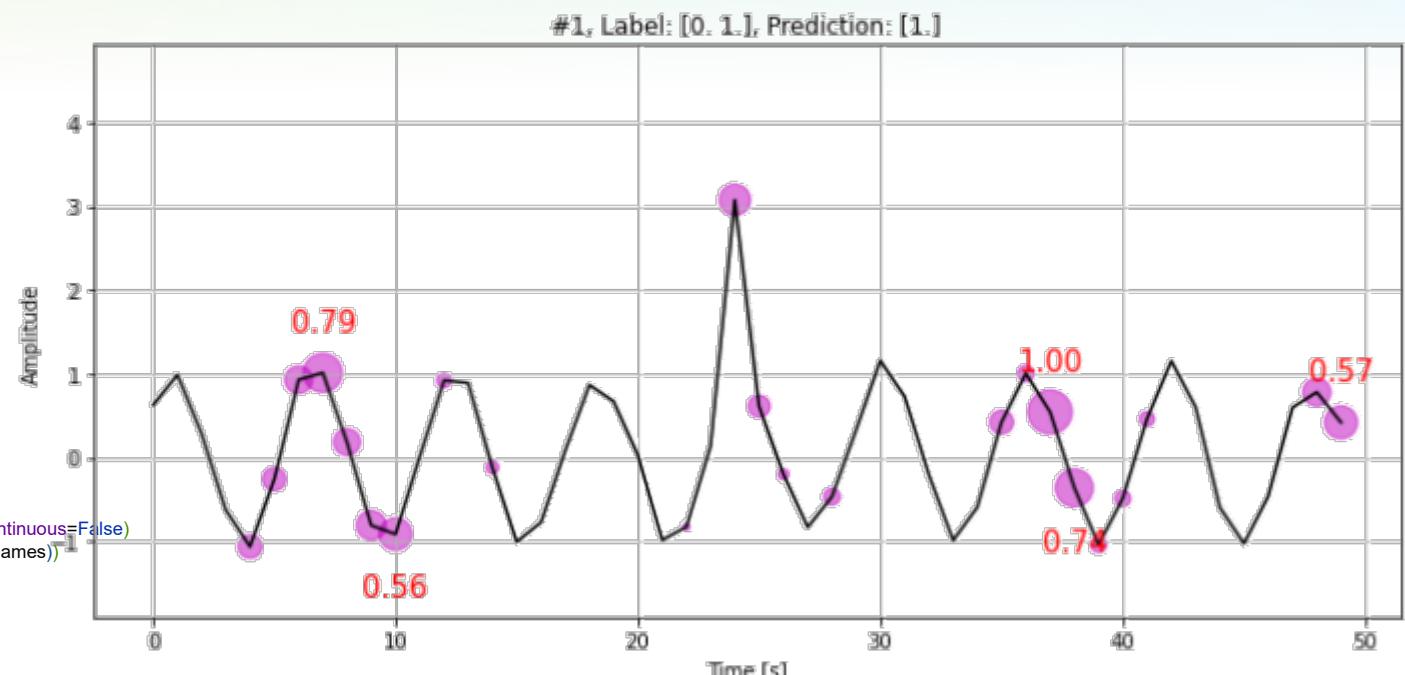
x, y, prediction = XX_test[i_check], YY_test[i_check], YY_predicted[i_check]
print(y, prediction)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

ax.cla()
ax.plot(x, 'k')
ax.scatter(x=range(len(x)), y=x, s=importance[:, 1] * 5e2, c='m', alpha=0.5)
ax.grid()
ymin, ymax = ax.get_ylim()
ax.set_ylim(ymin * y_lim_factor, ymax * y_lim_factor)
ax.set_xlabel('Time [s]')
ax.set_ylabel('Amplitude')
ax.set_title(f'#{i_check}, Label: {y}, Prediction: {prediction}')

i_importance_rank = np.argsort(importance[:, 1])
for i in i_importance_rank[-5:]:
    offset = 0.5 if x[i] >= 0 else -0.75
    ax.annotate(f'{importance[i, 1]:.2f}', (i, x[i] + offset), fontsize=15, color='r', ha='center')
fig.set_size_inches(10, 5)
fig.tight_layout()
fig.show()

```



# XAI :: SHapley Additive exPlanations (SHAP)

```
import shap
```

```
i_check = 1
```

```
model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))

T, X, Y, label_names, info = load_signal_data(dir_base_path=path_dataset, split='train')
TT_train, XX_train, YY_train = preprocessing(T, X, Y)
T, X, Y, label_names, info = load_signal_data(dir_base_path=path_dataset, split='test')
TT_test, XX_test, YY_test = preprocessing(T, X, Y)
num_features = XX_test.shape[-1]

YY_predicted = model_best.predict(np.array(XX_test)[:, np.newaxis, :])
YY_predicted[YY_predicted >= 0.5] = 1
YY_predicted[YY_predicted < 0.5] = 0
YY_probability = np.c_[1 - YY_predicted, YY_predicted]

x = XX_test[i_check]
y = YY_test[i_check]
y_prediction = YY_predicted[i_check]
i_sort = np.argsort(YY_probability[i_check])[:-1]
for i in i_sort:
    print("%d, %s, %.1f%%" % (i, label_names[i], YY_probability[i_check][i] * 100))

def model_func(x):
    xx = x[:, np.newaxis, :]
    yy = model_best.predict(xx)
    return np.c_[1 - yy, yy]

explainer = shap.Explainer(model_func, XX_train, output_names=label_names)
shap_values = np.array(explainer.shap_values[XX_test[i_check:i_check + 1]]) # (n_classes, n_samples, 1, t_length)
print('shap_values:', shap_values.shape) # (n_samples, t_length, n_classes, 1)

shap_values = shap_values[0].transpose([1, 0]) # (n_classes, t_length)
max_values = np.max(np.abs(shap_values), axis=(0, 1), keepdims=True)
shap_values_normalized = shap_values / max_values
print('shap_values_normalized:', shap_values_normalized.shape) # (n_samples, n_classes, t_length)

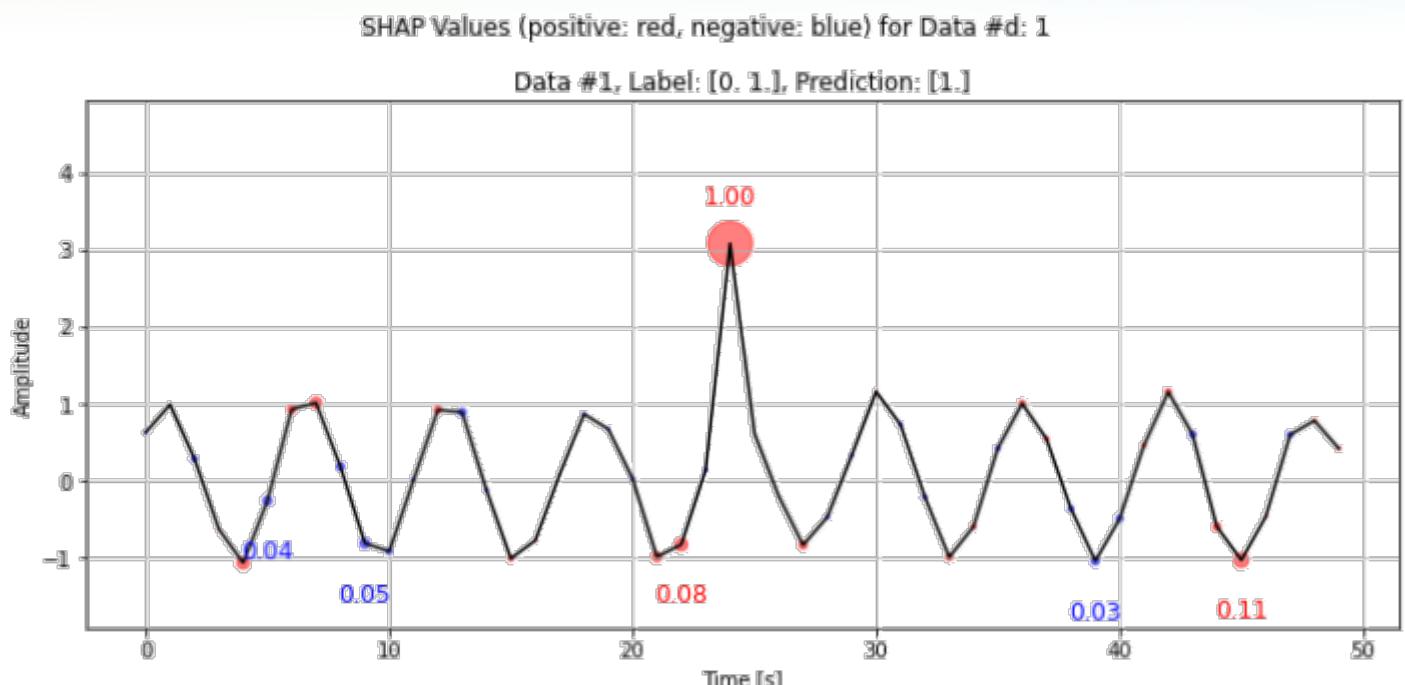
y_lim_factor = 1.5
marker_size_factor = 5e2
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

ax.plot(range(num_features), x, 'k')
ax.scatter(x=range(num_features), y=x, s=shap_values_normalized[0] * marker_size_factor, c='b', alpha=0.5)
ax.scatter(x=range(num_features), y=x, s=shap_values_normalized[1] * marker_size_factor, c='r', alpha=0.5)
ax.grid()
ax.set_xlabel('Time [s]')
ax.set_ylabel('Amplitude')
ymin, ymax = ax.get_ylim()
ax.set_ylim(ymin * y_lim_factor, ymax * y_lim_factor)
ax.set_title(f'Data #{i_check}, Label: {y}, Prediction: {y_prediction}')

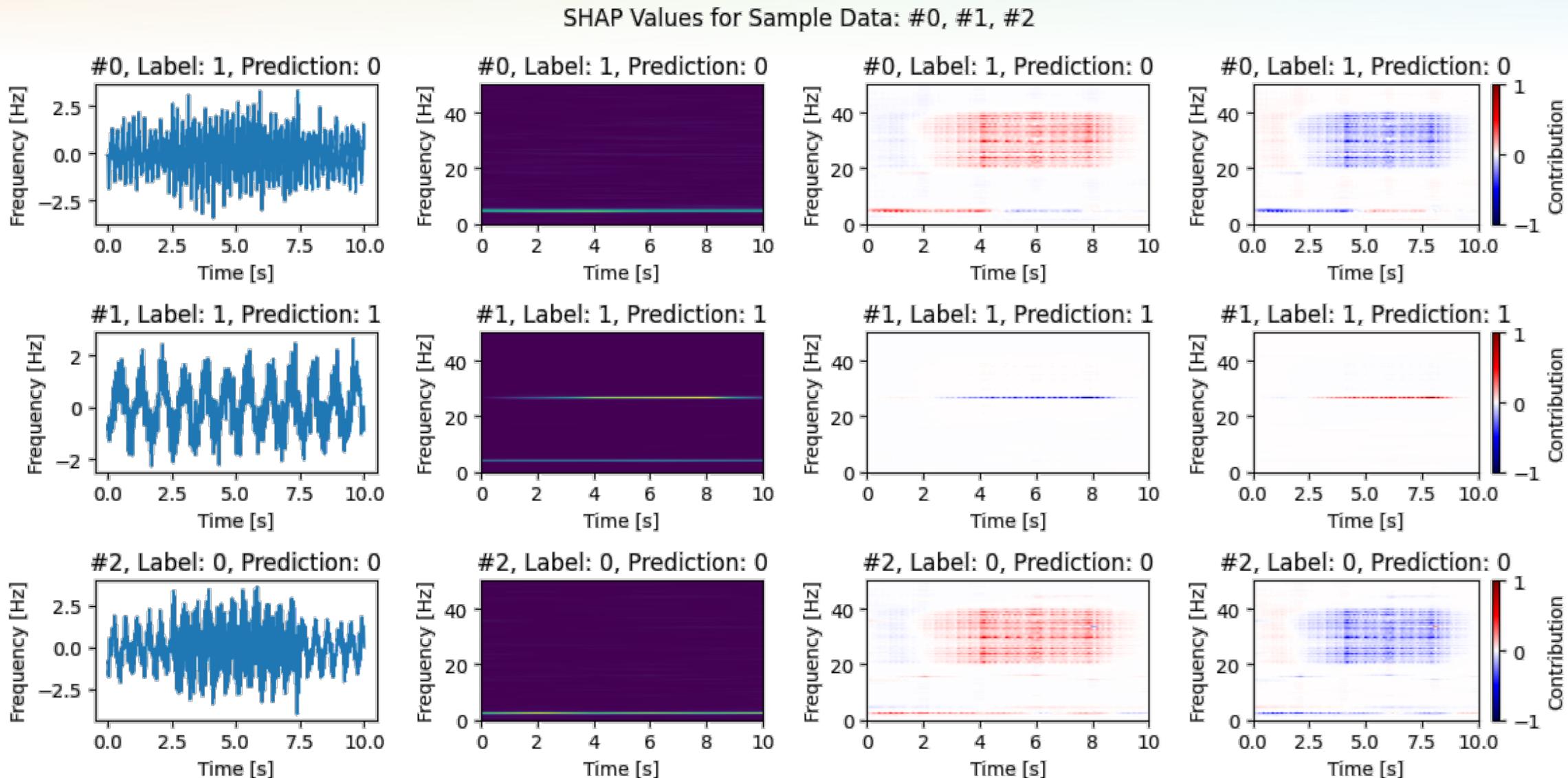
i_v_rank = np.argsort(shap_values_normalized[0])
for i in i_v_rank[-3:]:
    ax.annotate(f'{shap_values_normalized[0][i]:.2f}', (i, x[i] + 0.5 if x[i] >= 0 else x[i] - 0.75), fontsize=12, color='b', ha='center')

i_v_rank = np.argsort(shap_values_normalized[1])
for i in i_v_rank[-3:]:
    ax.annotate(f'{shap_values_normalized[1][i]:.2f}', (i, x[i] + 0.5 if x[i] >= 0 else x[i] - 0.75), fontsize=12, color='r', ha='center')

fig.suptitle('SHAP Values (positive: red, negative: blue) for Data #: %s' % i_check)
fig.set_size_inches(10, 5)
fig.tight_layout()
fig.show()
```



# XAI :: SHapley Additive exPlanations (SHAP)



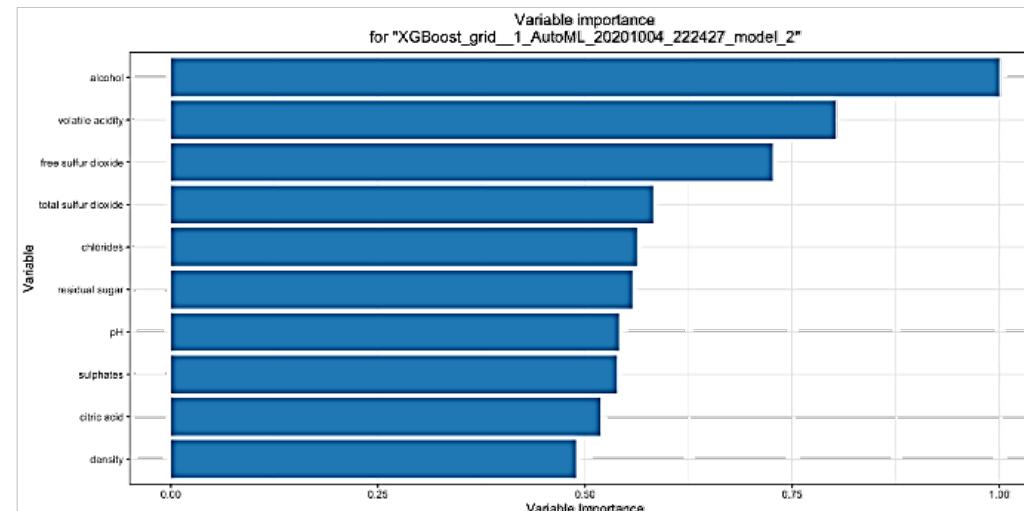
# AutoML Framework :: H2O – Variable Importance & SHAP XAI

## Variable Importance

The variable importance plot shows the relative importance of the most important variables in the model. Variable importance is currently available in for all H2O models except Stacked Ensembles (planned for a future release), so if you happen to use `h2o.explain()` on an AutoML object with a Stacked Ensemble at the top of the leaderboard, it will instead show the variable importance for the top "base model" (non-stacked model). Note for R users: The variable importance plot shown in the `h2o.explain()` output in R is rendered in ggplot2 instead of base R (the `h2o.varimp_plot()` utility function currently only uses base R). We will add a switch to generate the ggplot2 version in a future release.

```
R Python

ra_plot = model.varimp_plot()
```

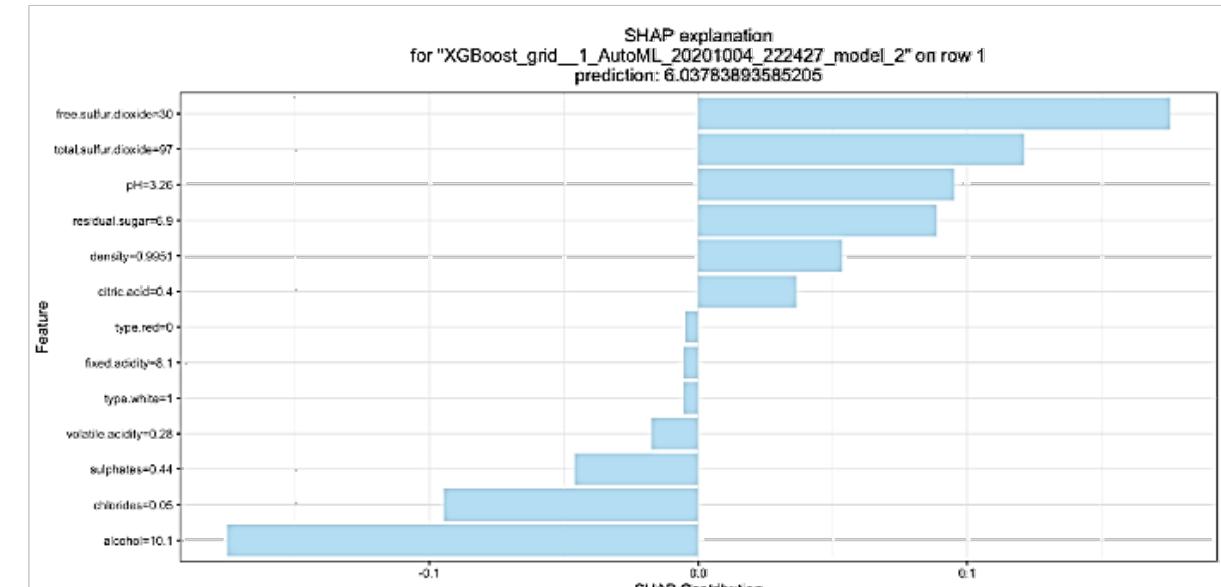


## SHAP Local Explanation

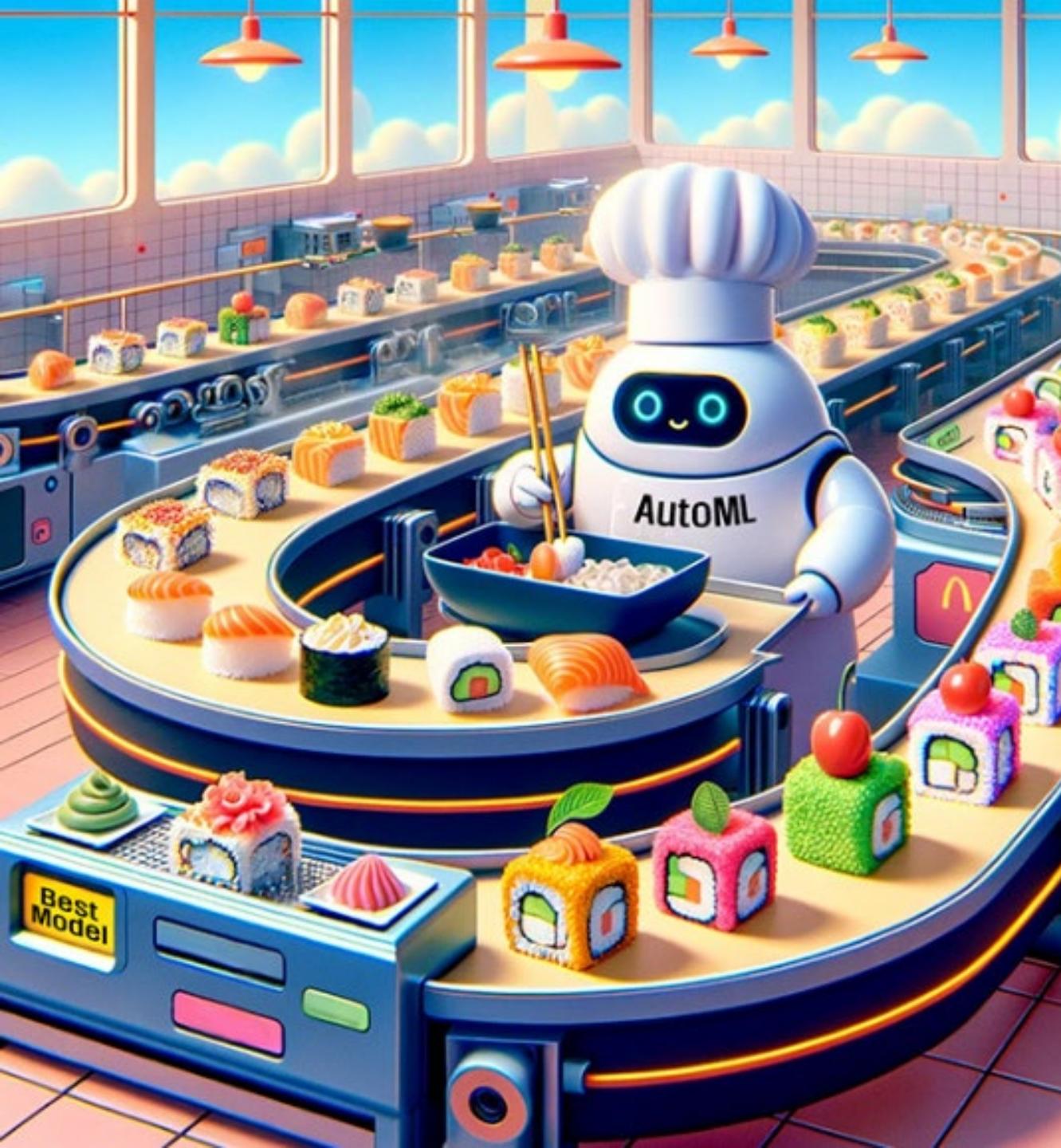
SHAP explanation shows contribution of features for a given instance. The sum of the feature contributions and the bias term is equal to the raw prediction of the model, i.e., prediction before applying inverse link function. H2O implements TreeSHAP which when the features are correlated, can increase contribution of a feature that had no influence on the prediction.

```
R Python

shapr_plot = model.shap_explain_row_plot(test, row_index=0)
```

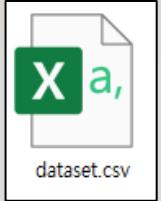


# Contents



0. What is AutoML?
1. 이미지 데이터에 대한 AutoML 활용
2. 신호 데이터에 대한 AutoML 활용
3. AutoML 모델을 활용한  
최적의 입력 데이터 추정

# Tabular Dataset

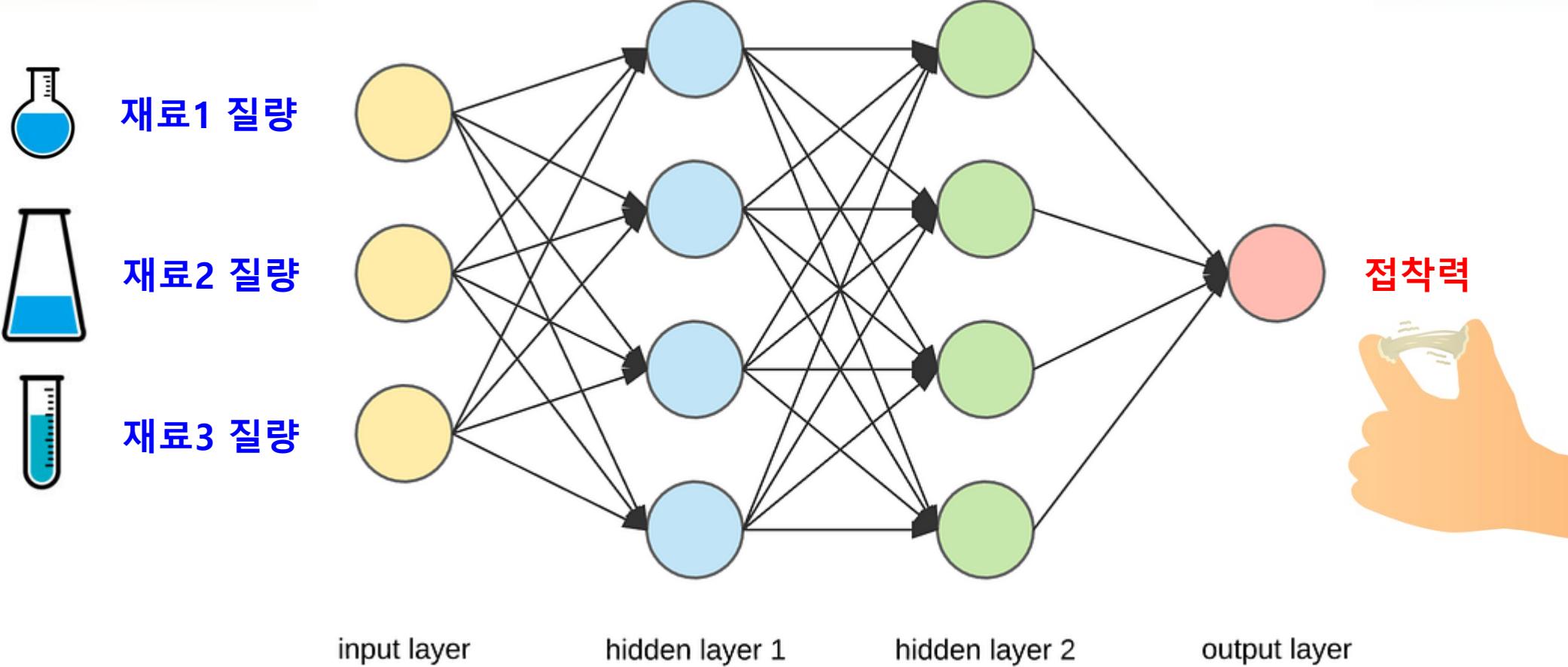


ID	X1	X2	X3	X4	X5	X6	X7	X8	X9
1	-0.59239	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	-33.7619	-6.40671	-48.8371
2	-0.59239	-4.24748	-0.00403		0	-62.7741	0.008934	-33.7619	-28.4571
3	-0.19746	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	0	-28.4571	-81.3951
4	-0.19746	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	-33.7619	-6.40671	-81.3951
5	-0.59239	-4.24748	-0.00081		0	-62.7741	0.008934	-33.7619	-28.4571
6	-0.59239	-4.24748	-0.00403		0	-62.7741	0.008934	0	-28.4571
7	-0.59239	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	0	-6.40671	-81.3951
8	-0.59239	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	-33.7619	-6.40671	-48.8371
9	-0.19746	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	-33.7619	-28.4571	-48.8371
10	-0.59239	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	-33.7619	-28.4571	-81.3951
11	-0.19746	-4.24748	-0.00403		0	-62.7741	0.008934	-33.7619	-28.4571
12	-0.59239	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	0	-28.4571	-81.3951
13	-0.19746	-4.24748	-0.00403		0	-62.7741	0.008934	-33.7619	-6.40671
14	-0.19746	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	0	-28.4571	-48.8371
15	-0.19746	-4.24748	-0.00081		0	-62.7741	0.008934	-33.7619	-6.40671
16	-0.19746	-4.24748	-0.00081		0	-62.7741	0.008934	0	-28.4571
17	-0.59239	-4.24748	-0.00081		0	-62.7741	0.008934	0	-6.40671
18	-0.59239	-4.24748	-0.00403		0	-62.7741	0.008934	0	-6.40671
19	-0.19746	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	0	-6.40671	-48.8371
20	-0.19746	-4.24748	-0.00081		0	-62.7741	0.008934	0	-6.40671
21	-0.59239	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	-33.7619	-6.40671	-48.8371
22	-0.59239	-4.24748	-0.00403		0	-62.7741	0.008934	-33.7619	-28.4571
23	-0.19746	-4.24748	-0.00403	-1.76669	-62.7741	0.008934	0	-28.4571	-81.3951
24	-0.19746	-4.24748	-0.00081	-1.76669	-62.7741	0.008934	-33.7619	-6.40671	-81.3951

X37	X38	X39	X40	Y
-0.34516	21.06096	48.82286	0	-1.76194
-0.34516	21.06096	48.82286	-2.73516	-2.14573
-0.15689	21.06096	48.82286	-2.73516	-1.17579
-0.34516	9.573166	48.82286	-2.73516	-1.50376
-0.15689	21.06096	9.764572	-2.73516	-1.40083
-0.34516	9.573166	48.82286	0	-1.51422
-0.15689	21.06096	9.764572	-2.73516	-1.23859
-0.15689	9.573166	48.82286	0	-2.21551
-0.15689	9.573166	9.764572	-2.73516	-1.45317
-0.15689	9.573166	9.764572	0	-1.49329
-0.34516	9.573166	9.764572	0	-1.81428
-0.34516	21.06096	9.764572	0	-0.82864
-0.15689	21.06096	48.82286	0	-1.51422
-0.34516	9.573166	48.82286	-2.73516	-1.32931
-0.34516	21.06096	9.764572	-2.73516	-1.65727
-0.15689	21.06096	48.82286	0	-1.55435
-0.15689	9.573166	48.82286	-2.73516	-1.19673
-0.34516	9.573166	9.764572	-2.73516	-1.07461
-0.34516	21.06096	9.764572	0	-1.43223
-0.15689	9.573166	9.764572	0	-1.04321
-0.15689	9.573166	9.764572	-2.73516	-2.05851
-0.15689	9.573166	9.764572	0	-1.52469
-0.34516	9.573166	9.764572	0	-1.1566
-0.15689	21.06096	9.764572	0	-2.23296
-0.34516	9.573166	48.82286	0	-1.91895
-0.15689	21.06096	9.764572	-2.73516	-1.46363
-0.34516	9.573166	48.82286	0	-1.09554
-0.34516	21.06096	9.764572	-2.73516	-1.67821
-0.34516	21.06096	48.82286	0	-1.27872
-0.34516	21.06096	48.82286	-2.73516	-1.93639
-0.15689	21.06096	48.82286	-2.73516	-1.67821
-0.15689	9.573166	48.82286	-2.73516	-1.25778
-0.34516	9.573166	9.764572	-2.73516	-1.52469

•••

# 최적화 문제의 개념

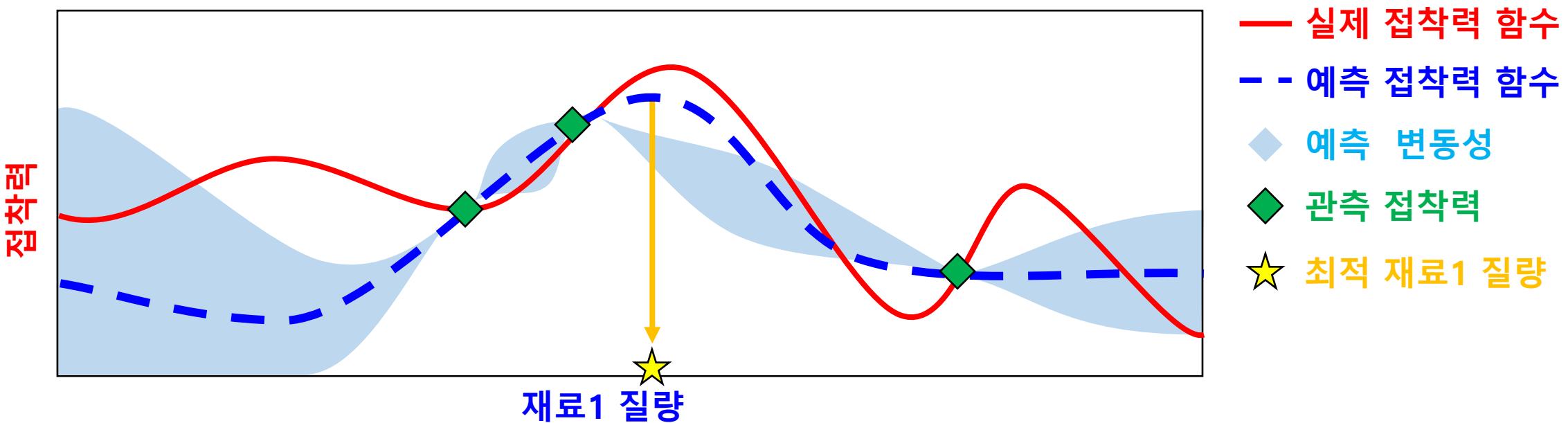
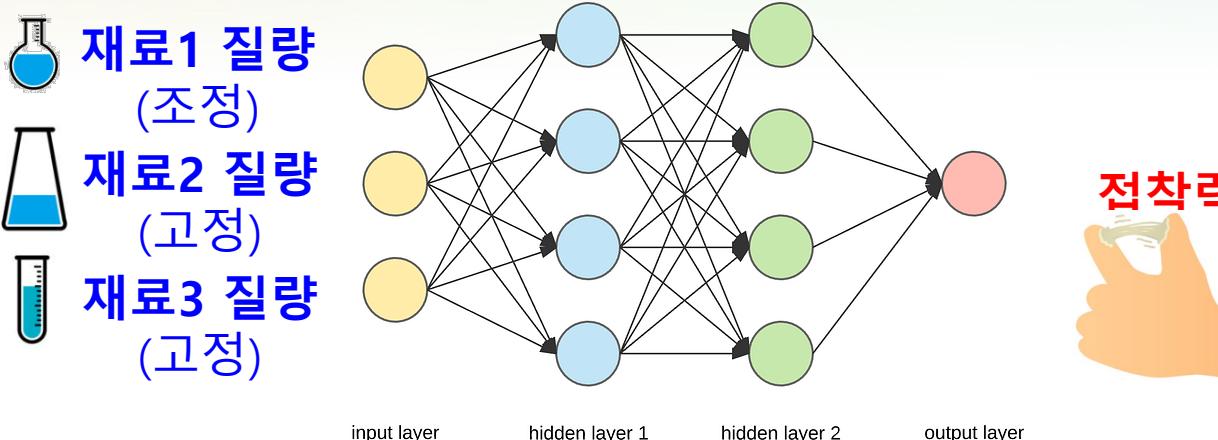


<https://www.istockphoto.com/kr/%EC%9D%BC%EB%9F%AC%EC%8A%A4%ED%8A%B8/%EB%B9%84%EC%BB%A4>

<https://medium.com/@giodegeronimo04/what-the-f-ck-is-a-neural-network-9f7d43d96d0e>

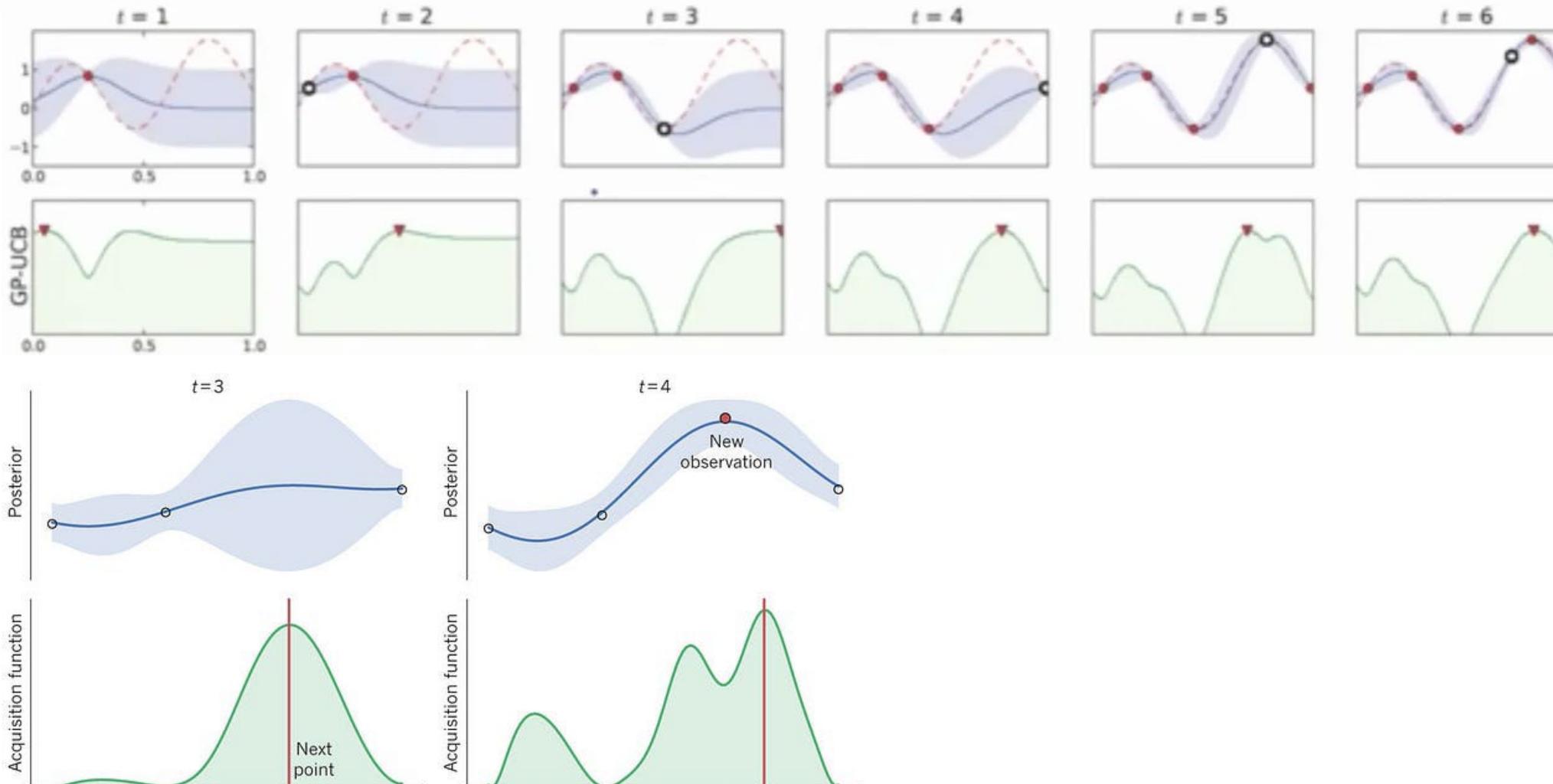
[https://thumb.ac-illust.com/6b/6b811d2f0c996949c200981c6ceffb64\\_t.jpeg](https://thumb.ac-illust.com/6b/6b811d2f0c996949c200981c6ceffb64_t.jpeg)

# 최적화 문제의 개념



# Bayesian Optimization의 개념

재료1 질량에 따라 관측지점이 늘어날수록 예측 변동성이 좁아지면서 실제 접착력 함수에 가까운 예측 가능



# AutoML for Tabular Dataset

```

import autokeras as ak
import numpy as np
import pandas as pd
import tensorflow as tf
from bayes_opt import BayesianOptimization
from sklearn.preprocessing import MinMaxScaler, StandardScaler

path_dataset = 'datasets/dataset.csv'
objective = 'val_loss'
trial_max = 3 #model search trials
epoch_max = 100 #epochs for each model
project_name = 'AutoML_tabular'

### data load
df = pd.read_csv(path_dataset, index_col=0)
df = df.dropna(how='all')

### data shuffling
df = df.sample(frac=1).reset_index(drop=True)

### model training
input_X = df[df.columns[:-1]].to_numpy()
input_Y = df[df.columns[-1]].to_numpy()

scaler_data = MinMaxScaler()
input_X = scaler_data.fit_transform(input_X)
input_Y = scaler_data.fit_transform(input_Y.reshape(-1, 1)).reshape(-1)

model = ak.StructuredDataRegressor(project_name=project_name, overwrite=True, max_trials=trial_max)
model.fit(input_X, input_Y, epochs=epoch_max)

### load best model and predict
model_best = tf.keras.models.load_model(os.path.join(project_name, 'best_model'))
output_Y = model_best.predict(input_X)
for y in output_Y:
    print(y[0])

### Evaluate
squared_errors = (output_Y[:, 0] - input_Y) ** 2
mse = np.mean(squared_errors)
print("RMSE:", np.sqrt(mse))

```

0.19691563
0.80424786
0.24889627
0.79100215
0.14238162
0.14635831
0.7008265
0.7326172
0.18562704
0.8460501
0.29616997
0.2994158
0.43911183
0.20158465
RMSE: 0.04831798642086087

# Bayesian Optimization을 활용한 최적 입력값 도출

```
### optimization
```

```
def model(x01, x02, x03, x04, x05, x06, x07, x08, x09, x10,
          x11, x12, x13, x14, x15, x16, x17, x18, x19, x20,
          x21, x22, x23, x24, x25, x26, x27, x28, x29, x30,
          x31, x32, x33, x34, x35, x36, x37, x38, x39, x40):
    input_data = np.array([[x01, x02, x03, x04, x05, x06, x07, x08, x09, x10,
                           x11, x12, x13, x14, x15, x16, x17, x18, x19, x20,
                           x21, x22, x23, x24, x25, x26, x27, x28, x29, x30,
                           x31, x32, x33, x34, x35, x36, x37, x38, x39, x40]], dtype=np.float32)
    output_data = model_best.predict(input_data)
    return output_data[0, 0]
```

```
pbounds = {}
```

```
min_values = input_X.min(axis=0)
max_values = input_X.max(axis=0)
```

```
for i in range(40):
    pbounds['x%02d' % (i+1)] = (min_values[i], max_values[i] * 10)
```

```
optimizer = BayesianOptimization(
```

```
f=model,
pbounds=pbounds,
random_state=1,
```

```
)
```

```
optimizer.maximize(
    init_points=2,
    n_iter=100,
```

```
)
```

	iter	target	x01	x02	x03	x04	x05	x06	x07	x08	x09	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36	x37	x38	x39	x40		
1	-2.635	4.17	7.203	0.001144	3.023	1.468	0.9234	1.863	3.456	3.968	5.388	4.192	6.852	2.045	8.781	0.2739	6.705	4.173	5.587	1.404	1.981	8.007	9.683	3.134	6.923	8.764	8.946	0.8504	0.3905	1.698	8.781	0.9835	4.211	9.579	5.332	6.919	3.155							
2	-1.418	9.889	7.482	2.804	7.893	1.032	4.479	9.086	2.936	2.878	1.3	0.1937	6.788	2.116	2.655	4.916	0.5336	5.741	6.865	8.346	0.1829	7.501	5.893	1.023	4.141	6.944	4.142	0.4995	5.359	6.638	5.149	9.446	5.866	9.034	1.375	1.393	8.074	3.977	1.654					
3	-1.947	7.259	7.698	0.9315	2.602	0.1399	4.535	0.916	2.919	3.576	1.65	0.8305	6.274	5.097	1.27	2.7	2.507	1.991	2.792	7.581	3.987	5.769	4.637	5.159	7.224	2.402	6.776	7.995	8.488	8.306	5.447	8.159	2.423	0.5366	3.444	9.429	4.481	8.671	5.766	7.809	5.308			