



# *MLOps 101*

## *Episode 4: ML 생애주기 (3) 모델 해석*

한석진  
마이크로소프트

---

## Episode 4

### ML 생애주기 (3)

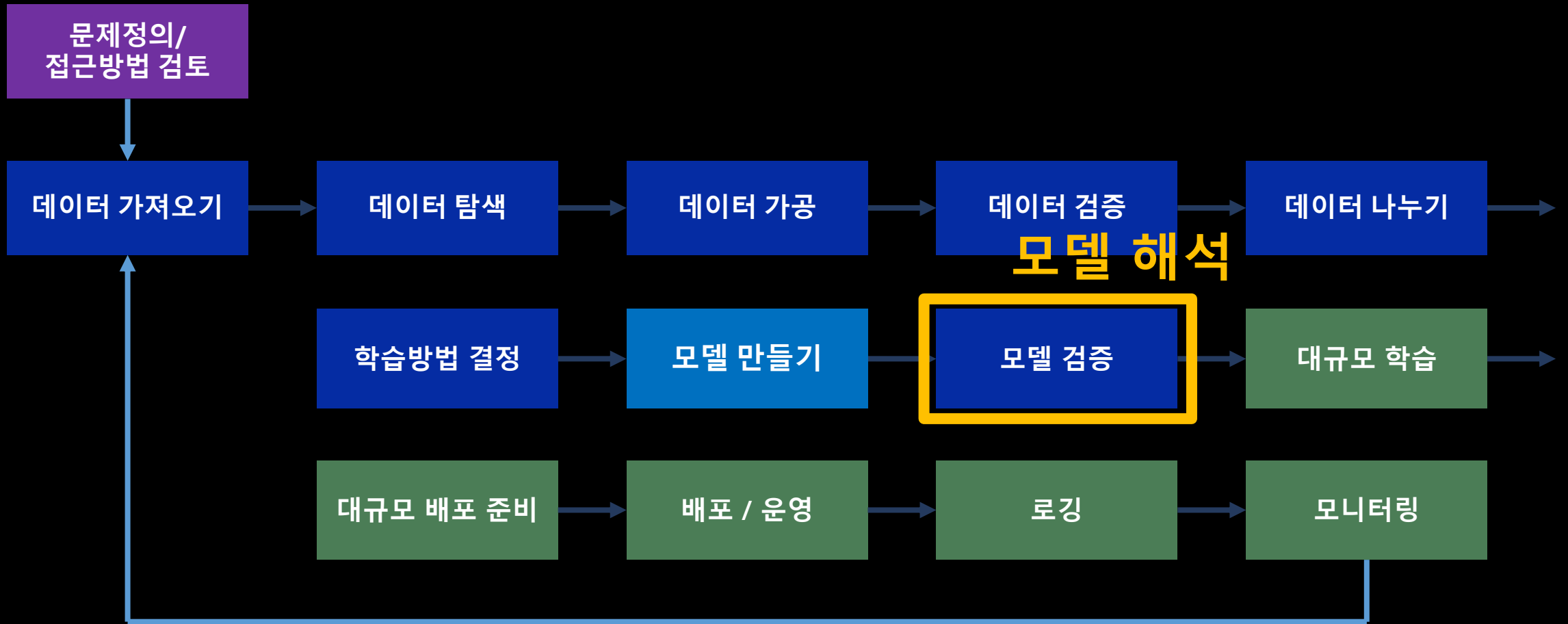
#### 모델 해석

---

#### ML 생애주기 (3) 모델 해석

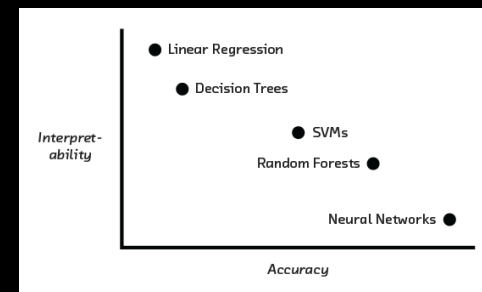
- 모델 해석이 왜 중요한가
- 모형을 해석하려는 시도
- azureml.interpret 들여다보기
- Explainer 관련 시각화 예시
  - 애저머신러닝에서 모델 해석 *DEMO*

# ML 생애주기

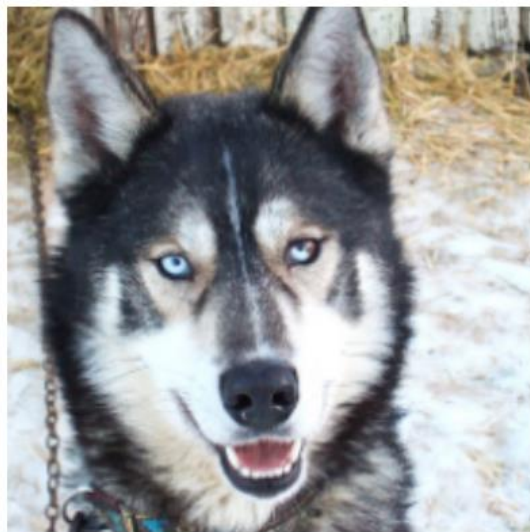


# 모델 해석이 왜 중요한가?

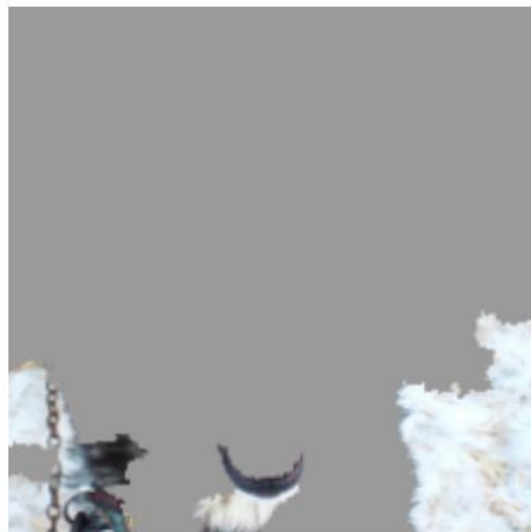
- “단순하게 설명할 수 없다면, 충분히 제대로 이해한 것이 아니다” — Albert Einstein
- 정확도만으로는 더 이상 충분하지 않다
- 내 모델이 예측을 잘 하는 것 같지만 사실은 완전 착각하고 있지는 않는가



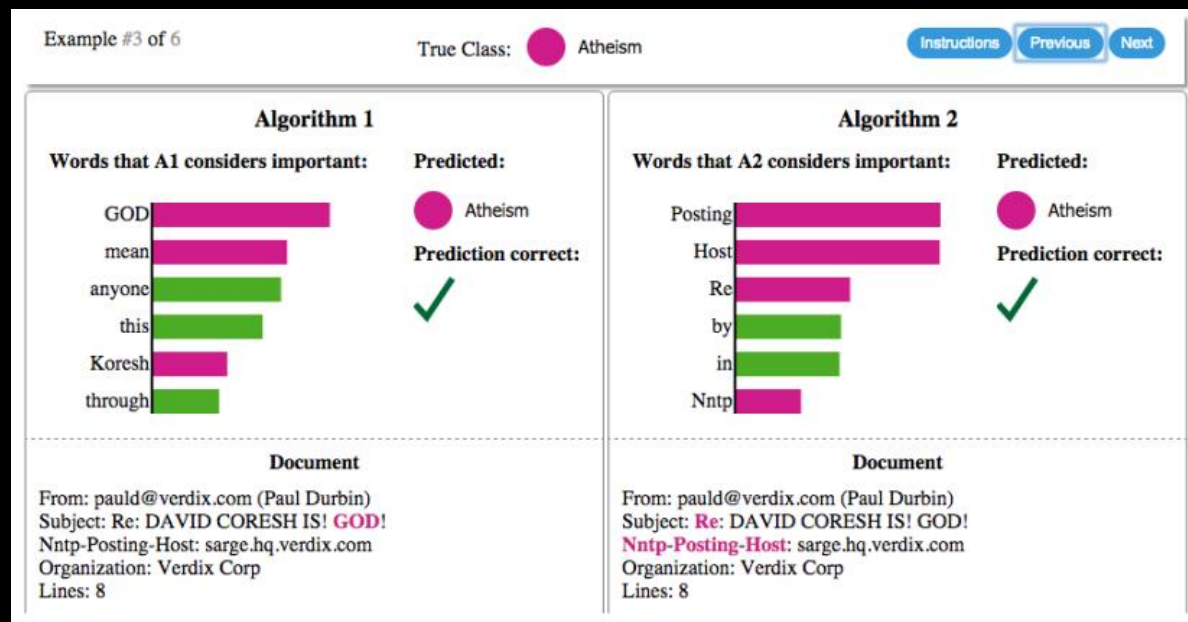
해석이 더 어렵다고 포기?



(a) Husky classified as wolf



(b) Explanation



# 모델 해석이 왜 중요한가?

모델 디버깅

왜 내 모델이 잘못 예측했나?

모델 공정성 판단

내 모델이 알게 모르게 차별을 하고 있지는 않나?

사람과 AI의 협력

나는 어떻게 모델의 예측을 이해하고 믿을 수 있나?

규제 및 컴플라이언스

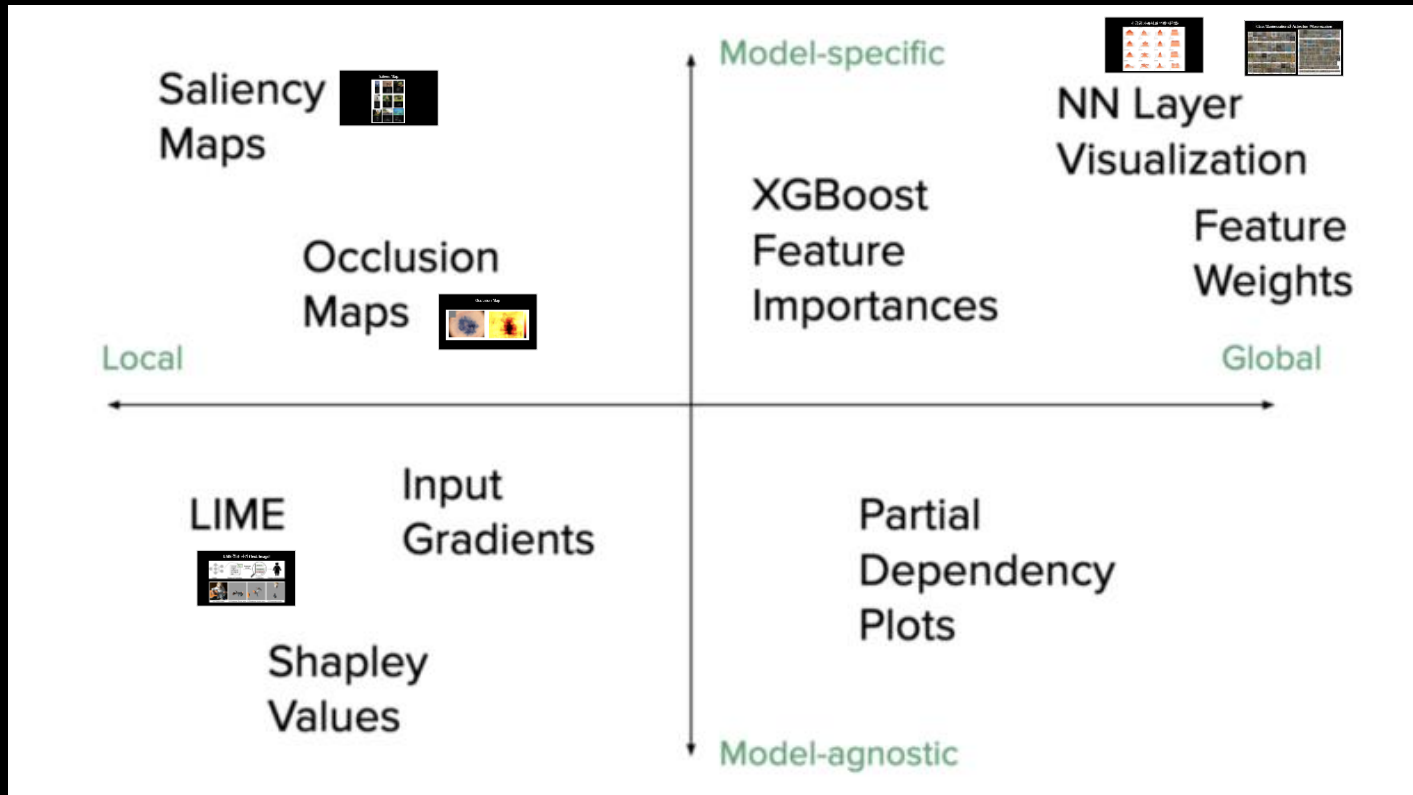
내 모델이 법적인 요건을 만족하는가?

고리스크 영역

의료, 금융, 법률 등

출처: <https://github.com/interpretml/interpret>

# 모델을 해석하려는 시도



X축	Local	일부를 뜯어보기 (신경망의 개별 필터 등)
	Global	전체를 보기 (신경망의 가중치 분포나 전파를 시각화)
Y축	Model-agnostic	다수의 모델 유형에 적용 가능
	Model-specific	특정 유형의 모델에만 적합

## 4 Interpretable Models

### 4.1 Linear Regression

### 4.2 Logistic Regression

### 4.3 GLM, GAM and more

### 4.4 Decision Tree

### 4.5 Decision Rules

### 4.6 RuleFit

### 4.7 Other Interpretable Models

## 5 Model-Agnostic Methods

### 5.1 Partial Dependence Plot (PDP)

### 5.2 Individual Conditional Expecta...

### 5.3 Accumulated Local Effects (AL...

### 5.4 Feature Interaction

### 5.5 Permutation Feature Importance

### 5.6 Global Surrogate

### 5.7 Local Surrogate (LIME)

### 5.8 Scoped Rules (Anchors)

### 5.9 Shapley Values

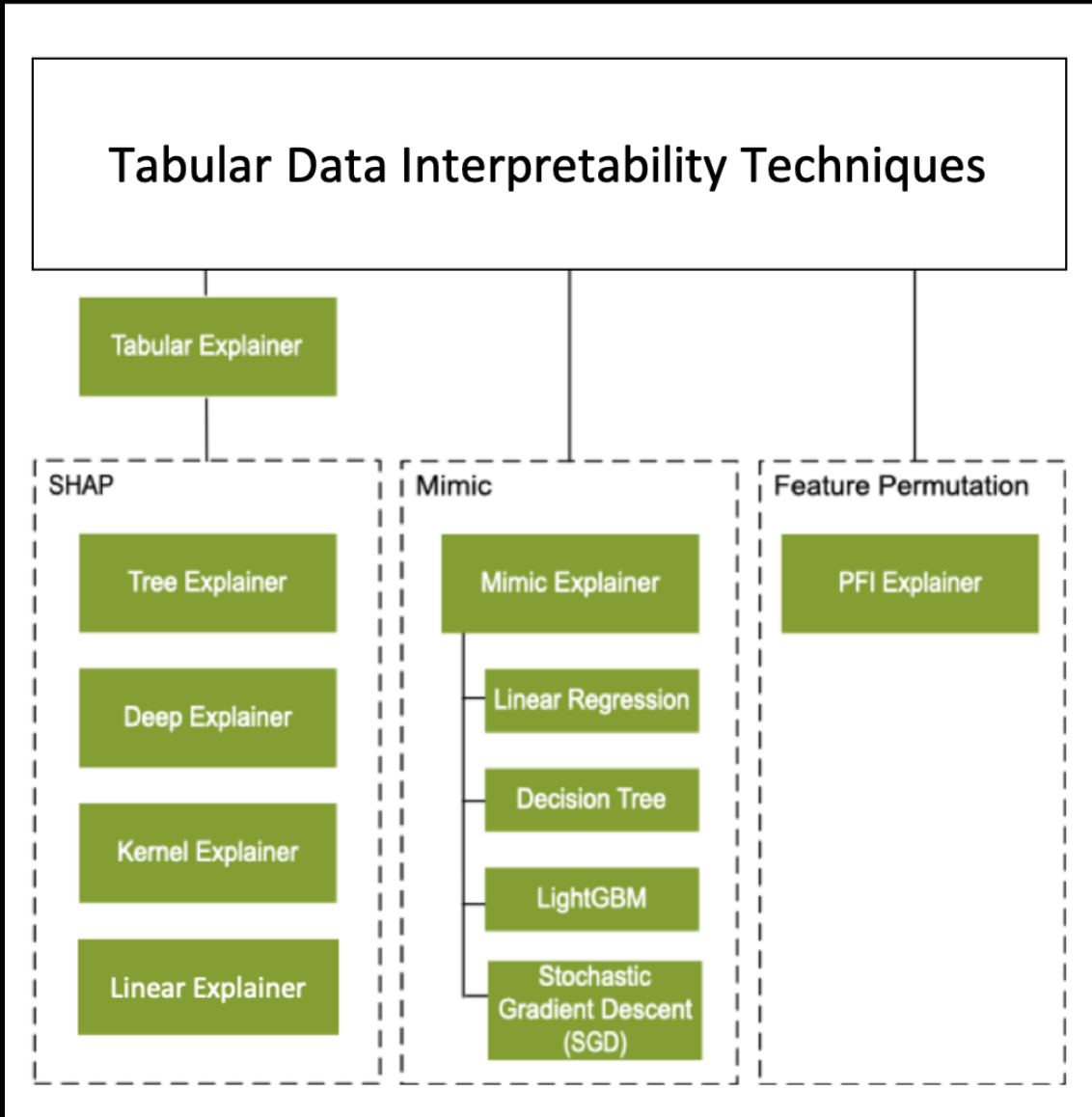
### 5.10 SHAP (SHapley Additive exPI...

## Explaining Black Box Models and Datasets

- Anchor** (Osherson et al., 2017) - An open-source tool for data scientists, machine learning researchers, and practitioners to audit for interpretable models and to make informed and equitable decisions around developing and deploying predictive risk-assessment tools.
- Alibi** (Osherson et al., 2017) - Alibi is an open source Python library aimed at machine learning model inspection and interpretation. The initial focus of the library is on black-box, instance-based model explanations.
- analyzer** (Osherson et al., 2017) - Code for the paper "High precision model agnostic explanation", a model-agnostic system that explains the behavior of complex models with high-precision rules called anchors.
- captum** (Osherson et al., 2017) - model interpretability and understanding library for PyTorch developed by Facebook. It contains general purpose implementations of integrated gradients, saliency maps, smoothgrad, vargrad and others for PyTorch models.
- class** (Osherson et al., 2017) - Example of using classifier-agnostic saliency map extraction on ImageNet presented in the paper "Classifier-agnostic saliency map extraction".
- ContrastiveExplanation** (Fad "Trev" Osherson et al., 2017) - Python script for model agnostic contrastive/counterfactual explanations for machine learning. Accompanying code for the paper "Contrastive Explanations with Local Top Team".
- DeepLIFT** (Osherson et al., 2017) - Codebase that contains the methods in the paper "Learning important features through propagating activation differences". Here is the slide and the video of the 15 minute talk given at ICML.
- Deepviz** (Osherson et al., 2017) - This is the code required to run the Deep Visualization ToolBox, as well as to generate the session-by-session visualizations using integrated optimization. The toolbox and methods are described casually here and more formally in this paper.
- eli5** (Osherson et al., 2017) - "Explain like I'm 5" is a Python package which helps to debug machine learning classifiers and explains their predictions.
- FACETS** - Facets contains two robust visualizations to aid in understanding and analyzing machine learning datasets. Get a sense of the shape of each feature of your dataset using Facets Overview, or explain individual observations using Facets Dive.
- feature** (Osherson et al., 2017) - feature is a python toolkit to assess and mitigate unfairness in machine learning models.
- feature** (Osherson et al., 2017) - feature is a python toolkit assessing the machine learning models for bias.
- feature** (Osherson et al., 2017) - This repository is meant to facilitate the benchmarking of fairness aware machine learning algorithms based on this paper.
- GBI** - Global Explanations for Bias Identification (Osherson et al., 2017) - An attention-based constructed post-hoc explanations for detection and identification of bias in data. We propose a global explanation and introduce a step-by-step framework on how to detect and test bias. Python package for image data.
- IBM AI Explainability 360** (Osherson et al., 2017) - Interpretability and explainability of data and machine learning models including a comprehensive set of algorithms that cover different dimensions of explanations along with proxy explainability metrics.
- IBM AI Fairness 360** (Osherson et al., 2017) - A comprehensive set of fairness metrics for datasets and machine learning models, explanations for these metrics, and algorithms to mitigate bias in datasets and models.
- INterpretable** (Osherson et al., 2017) - An open-source library for analyzing kernel models visually by methods such as DeepFeatureDecomposition, FeatureSaliency, Saliency Maps, and Integrated Gradients.
- Integrated-Gradients** (Osherson et al., 2017) - This repository provides code for implementing integrated gradients for networks with image inputs.
- interpret** (Osherson et al., 2017) - interpret is an open-source package for training interpretable models and explaining black-box systems.
- kernelviz** (Osherson et al., 2017) - kernelviz is a high-level toolkit for visualizing and debugging your trained kernel model. Currently supported visualizations include Activation Maximization, Saliency Maps, Class Activation Maps.
- L3** (Osherson et al., 2017) - Code for replicating the experiments in the paper "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation" at ICML 2016.
- Lightwood** (Osherson et al., 2017) - A Pytorch based framework that breaks down machine learning problems into smaller blocks that can be glued together seamlessly with an objective to build predictive models with low loss of code.
- ling** (Osherson et al., 2017) - Local Interpretable Model-agnostic Explanations for machine learning models.
- LM30 Importance** (Osherson et al., 2017) - LM30 (Local Model Feature Out) importance calculates the importance of a set of features based on a metric of choice, for a model of choice, by iteratively removing each feature from the set, and evaluating the performance of the model, with a validation scheme of choice, based on the chosen metric.
- MindDB** (Osherson et al., 2017) - MindDB is an Explainable Machine Learning (XML) framework for developers. With MindDB you can build, train and use state of the art ML models in as simple as one line of code.
- ml-explainability** (Osherson et al., 2017) - An Automated Machine Learning (AutoML) python package for tabular data. It can handle Binary Classification, Multi-Class Classification and Regression. It provides feature engineering, explanation and model tuning reports.
- NSTRON** (Osherson et al., 2017) - solver for neural network, deep learning and machine learning models.
- pyGlobal** (Osherson et al., 2017) - A model agnostic tool for decomposition of predictions from black boxes. Break Down Table shows contributions of every variable to a final prediction.
- robustviz** (Osherson et al., 2017) - Code to implement learning robustness based predictions with code for paper "Robustness: Neural Predictions".
- responsibility** (Osherson et al., 2017) - Toolkit for auditing and mitigating bias and fairness of machine learning systems.
- SHAP** (Osherson et al., 2017) - SHapley Additive exPlanations is a unified approach to explain the output of any machine learning model.
- skater** (Osherson et al., 2017) - Skater is a unified framework to enable Model Interpretation for all forms of model to help one build an interpretable machine learning system often needed for real world use-cases.
- TensorBoard's TensorBoard WhatIf** (Osherson et al., 2017) - TensorBoard's WhatIf is a web-based tool to analyze the interactions between inference results and data inputs.
- TensorFlow's DeepFuzz** (Osherson et al., 2017) - An adversarial example library for constructing attacks, building defenses, and benchmarking both. A python library to benchmark systems vulnerability to adversarial examples.
- tensorflow's back** (Osherson et al., 2017) - back is a collection of infrastructure and tools for research in neural network interpretability.
- tensorflow's Model Analysis** (Osherson et al., 2017) - TensorFlow Model Analysis (TFMA) is a library for evaluating TensorFlow models. It allows users to evaluate their models on large amounts of data in a distributed manner, using the same metrics defined in their trainer.
- thorn** (Osherson et al., 2017) - thorn is a Python library built on top of pandas and sklearn that implements feature-wise machine learning algorithms.
- Thorn** (Osherson et al., 2017) - Thorn is a testing-based approach for measuring discrimination in a software system.
- Treeinterpreter** (Osherson et al., 2017) - Package for interpreting scikit-learn's decision tree and random forest predictions. Allows decomposing each prediction into base and feature contribution components as described in <http://blog.dominicam.io/interpreting-random-forests/>.
- uex** (Osherson et al., 2017) - Tools for UCI Transportation mostly used in ScoreCard Model for credit rating.
- uex** (Osherson et al., 2017) - An explainability toolbox for machine learning.

출처: <https://github.com/Harvard-IACS/2020-ComputeFest/>, <https://christophm.github.io/interpretable-ml-book/>, <https://github.com/EthicalML/awesome-production-machine-learning#explaining-black-box-models-and-datasets>

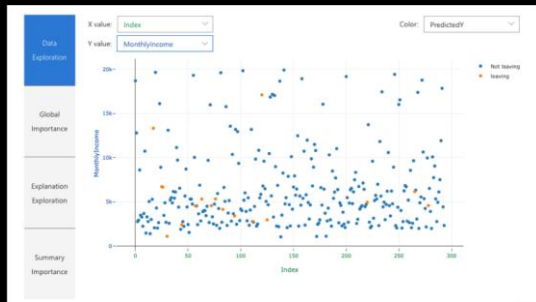
# azureml.interpret 들여다보기



Explainer 종류	적용 모델	접근 방법
SHAP Tree Explainer	Tree, Forest 계열	Polynomial time fast <a href="#">SHAP</a> value estimation
SHAP Deep Explainer	TensorFlow (+ Keras), PyTorch	딥러닝에서 <a href="#">SHAP</a> value 추정을 <a href="#">DeepLIFT 기법</a> 과 연계하여 빠르게 수행
SHAP Linear Explainer	선형 모델	선형 모델을 위한 <a href="#">SHAP</a> value 계산 (inter-feature correlation을 고려)
SHAP Kernel Explainer	임의의 모델	Local 선형 회귀모델에 특정 가중치를 적용하여 <a href="#">SHAP</a> value 추정
Mimic Explainer	임의의 모델	LightGBM, Linear Regression, SGD, Decision Tree 등 설명하기 쉬운 전역 대체 모델( <a href="#">global surrogate model</a> )을 추정하여 원 모델 대신 해석
Permutation Feature Importance Explainer	임의의 모델	<a href="#">Breiman's Random Forests paper</a> (section 10)에 기반하여, 특정 feature를 변경시 최종 모델 성능에 미치는 영향을 측정하여, 임의 모델에서 성능에 중요한 feature를 추출
Tabular Explainer	임의의 모델	SHAP Explainer들에 대한 wrapper로서, 모델의 형태에 따라서 Tree, Deep, Linear, Kernel Explainer 중 선택하여 수행함 (예를 들어 Tree 모델이면 무조건 Tree Explainer를 수행하고, Tree가 아닌 딥러닝 모델이면 Deep Explainer를 실행)

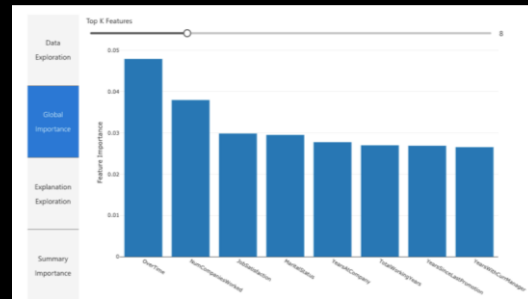
# Explainer 관련 시각화 예시

## 데이터 탐색



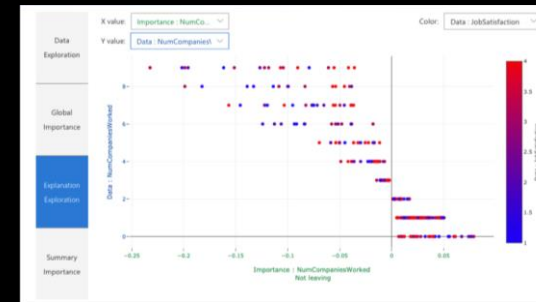
데이터셋을 예측값과 함께 표시

## Global Feature Importance



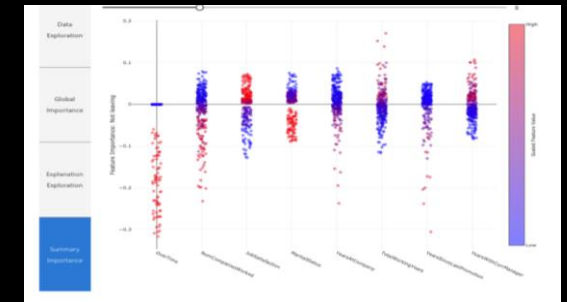
중요한 데이터 속성 N개를 글로벌 관점에서 표시

## Explanation 탐색



개별 feature가 모델 예측에 미치는 영향 분석 (feature interaction 확인가능)

## Summary Importance



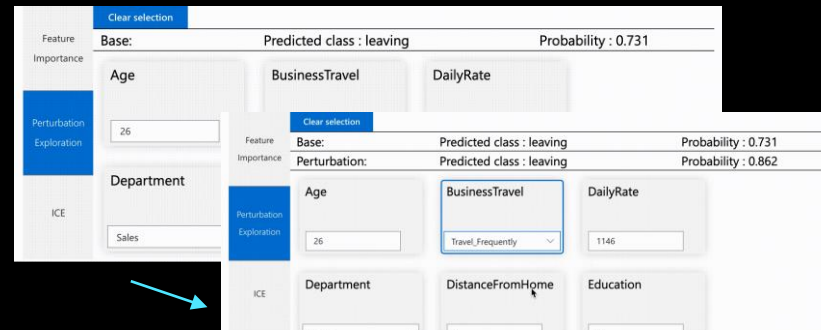
local feature importance를 각 data point에 대해 표시하여 예측값에 미치는 영향의 분포 확인

## Feature Importance



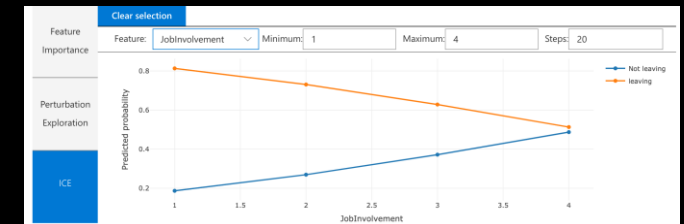
Data 탐색화면에서 특정 데이터 포인트를 선택하면, 해당 데이터 포인트에 대해 해당 모델의 지역적(local) feature importance 시각화

## Perturbation 탐색



feature 값을 특정 값으로 변경했을 때 예측값에 영향을 주는 결과를 시뮬레이션할 수 있는 기능.

## ICE (Individual Conditional Expectation)



feature 값을 특정 값이 아닌 최소에서 최대까지의 값으로 단계적으로 변화시켰을 때 예측값에 미치는 영향 확인



---

## Episode 4

### ML 생애주기 (3)

#### 모델 해석

---

#### ML 생애주기 (3) 모델 해석

- 모델 해석이 왜 중요한가
- 모형을 해석하려는 시도
- azureml.interpret 들여다보기
- Explainer 관련 시각화 예시
  - 애저머신러닝에서 모델 해석 *DEMO*

# {다음 시간에는}

---

## Episode 5 ML 생애주기 (4) 배포/서빙

---

### ML 생애주기 (4) 배포/서빙

- 패키징, 배포 (서빙)
- 모델의 모니터링: Data Drift

### 데모

- 애저머신러닝에서 배포
- 애저머신러닝에서 Data Drift 모니터링