PyCon Korea 2019

딥러닝 안에서 일어나는 과정을 설명하는 인공지능 기술

모델 불가지론적 방법 모델 (SHAP) 튜토리얼 : SHapley Additive exPlanations (SHAP)

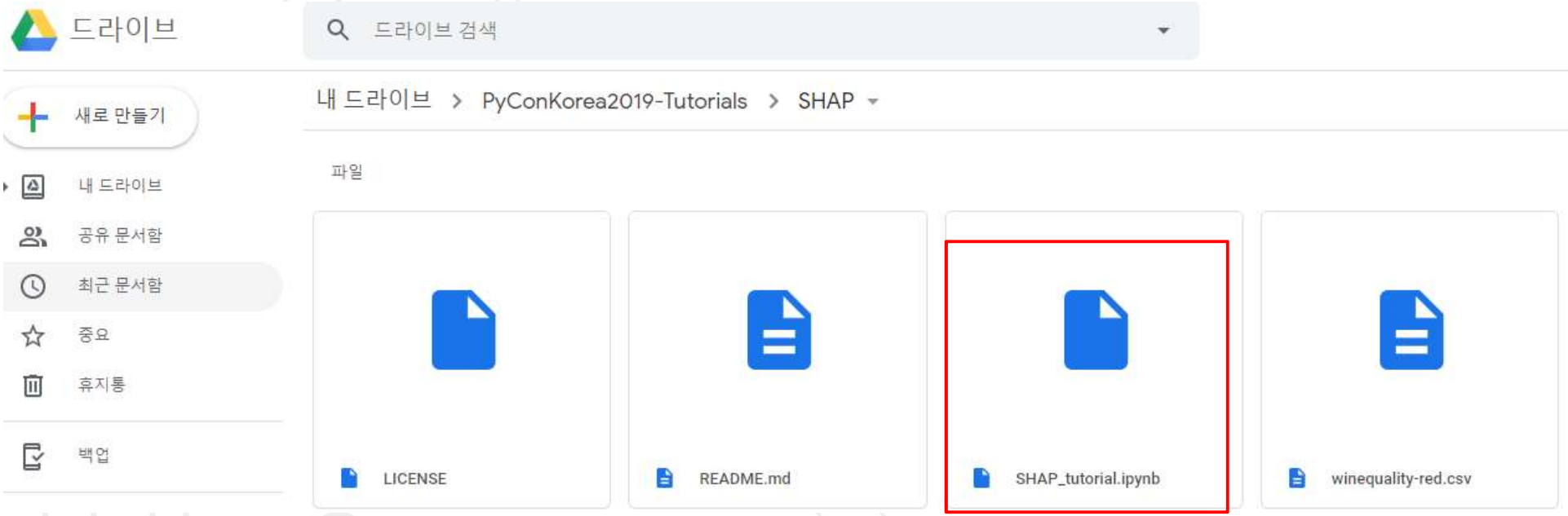
허성만 @ 설명가능인공지능 연구센터
Smheo.cie@gmail.com

Index

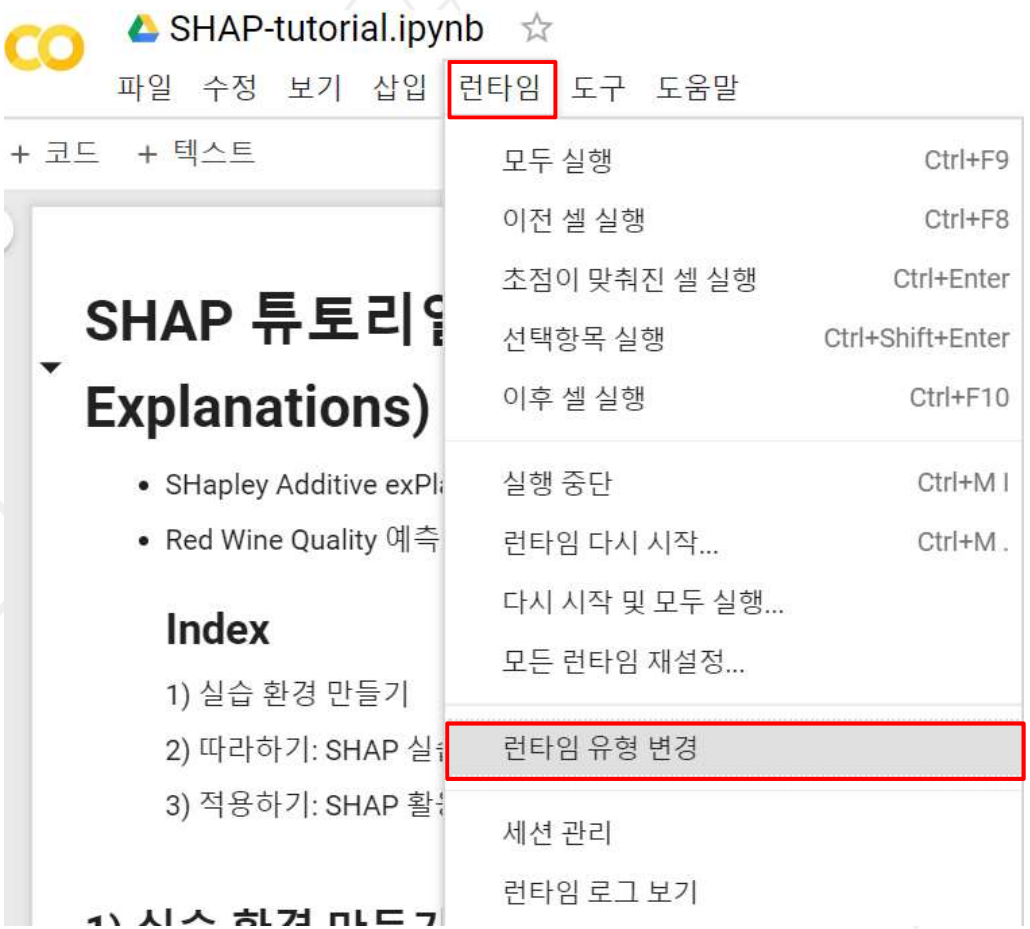
실습 환경 만들기

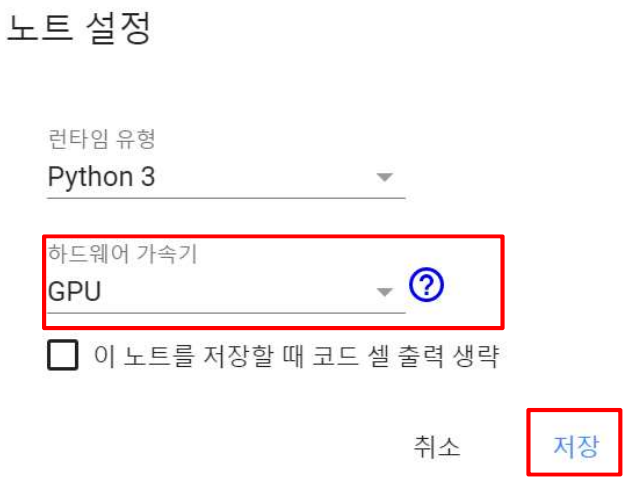
SHAP 모듈 설치

실습 환경 만들기 Google Colab



실습 환경 만들기 Google Colab





실습 환경 만들기 Google Colab

SHAP 모듈 설치 방법

설치 방법 – PyPI로 설치 **pip install**

```
# SHAP 모듈 설치
!pip install shap
```

```
Collecting shap
  Downloading https://files.pythonhosted.org/packages/80/82/bab67238ac27d53214b12f6ed095493dc7b43be07c615b8b0dbb7da33157/shap-0.29.3.tar.gz (230kB)
     |████████████████████████████████| 235kB 4.9MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from shap) (1.16.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from shap) (1.3.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from shap) (0.21.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from shap) (3.0.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from shap) (0.24.2)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.6/dist-packages (from shap) (4.28.1)
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from shap) (5.5.0)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist-packages (from shap) (0.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn->shap) (0.13.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap) (2.4.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap) (2.5.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap) (0.10.0)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->shap) (2018.9)
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (4.7.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (0.7.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (2.1.3)
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (4.4.0)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (41.0.1)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (1.0.16)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (0.8.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from ipython->shap) (4.3.2)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->shap) (4.3.0)
Requirement already satisfied: imageio>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from scikit-image->shap) (2.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->shap) (2.3)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->shap) (1.0.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.1->matplotlib->shap) (1.12.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from pexpect; sys_platform != "win32"->ipython->shap) (0.6.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->shap) (0.1.7)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2->ipython->shap) (0.2.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-packages (from pillow>=4.3.0->scikit-image->shap) (0.46)
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.29.3-cp36-cp36m-linux_x86_64.whl size=344724 sha256=c1781e2c069a392e719d185e20ede7e65cecefb149dfbb5ed29ee158851a6388
  Stored in directory: /root/.cache/pip/wheels/00/20/87/d199e4d7397997f5494e4098104f91313ac8120753bee7b032
Successfully built shap
Installing collected packages: shap
Successfully installed shap-0.29.3
```

실습 환경 만들기 Google Colab

구글 Drive와 Colab 연동

```
# 구글 드라이브와 Colab 연동
from google.colab import drive
drive.mount('/content/drive') # 출력되는 URL에 접속하여 verification code 복사 및 붙여넣기
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318

Enter your authorization code:
· · · · · · · · · · ·
Mounted at /content/drive
```

실습 환경 만들기 Google Colab

구글 Drive와 Colab 연동

```
# 작업할 path로 변경
import os
os.chdir('/content/drive/My Drive/PyConKorea2019-Tutorials/SHAP')
os.listdir(os.getcwd()) # 현재 path에 존재하는 파일 목록 확인
```

```
['LICENSE', 'README.md', 'SHAP_tutorial.ipynb', 'winequality-red.csv']
```

```
# 현재 path 확인
os.getcwd()
```

```
'/content/drive/My Drive/PyConKorea2019-Tutorials/SHAP'
```

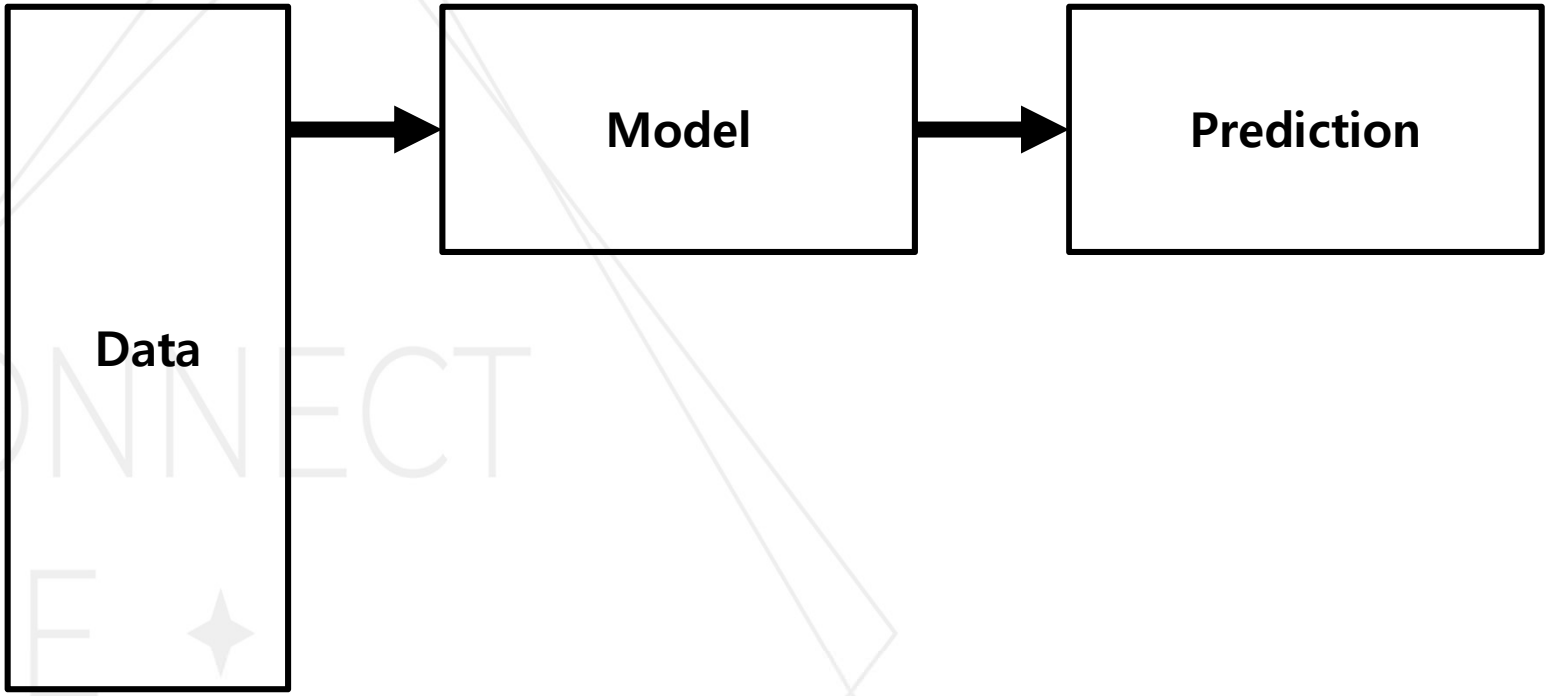
```
# 경고 메시지 무시
import warnings
warnings.filterwarnings(action='ignore')
```

따라하기: SHAP 실습

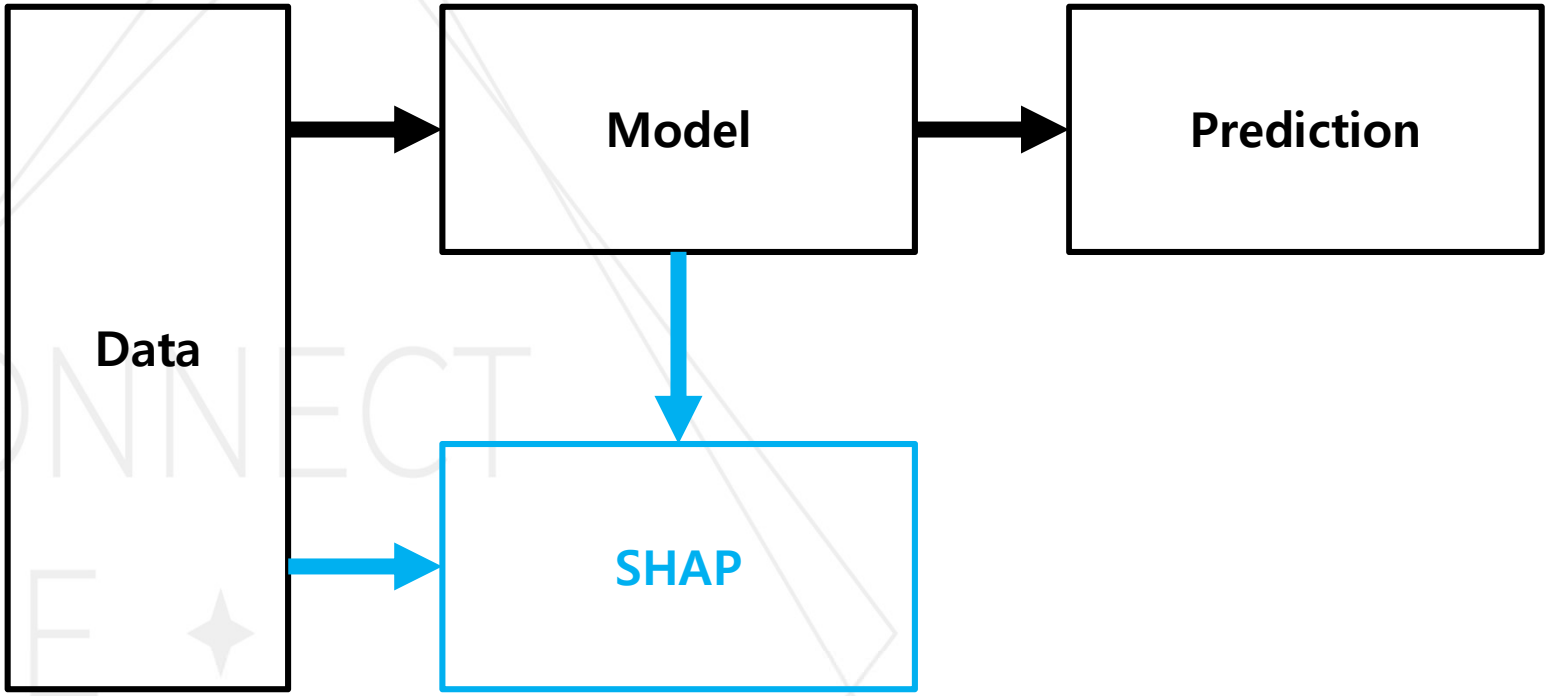
DeepExplainer
KernelExplainer
TreeExplainer

따라하기: **SHAP** 실습

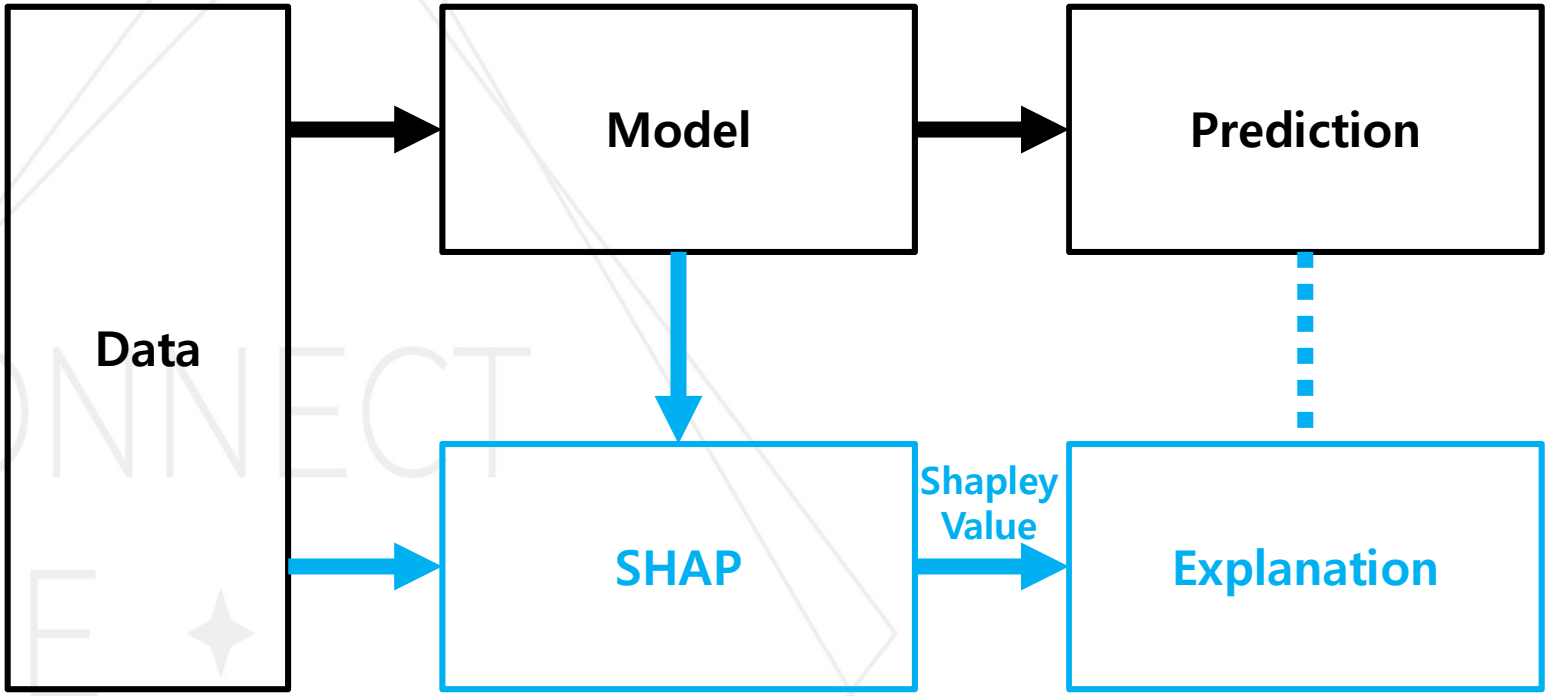
따라하기: **SHAP** 실습



따라하기: **SHAP** 실습



따라하기: **SHAP** 실습



따라하기: SHAP 실습

SHAP이란?

학습 데이터와 **학습된 모델**을 가지고 **설명 모델**을 생성하고 **새로운 입력 데이터**에 대해 **shapley value**계산을 통해 **입력 데이터 features**가 학습된 모델 출력 값에 대해 어떠한 **공헌도**를 가지는지 설명함으로써 기계학습 모델을 설명하는 방법

모델 불가지론적 방법 Model-Agnostic Methods

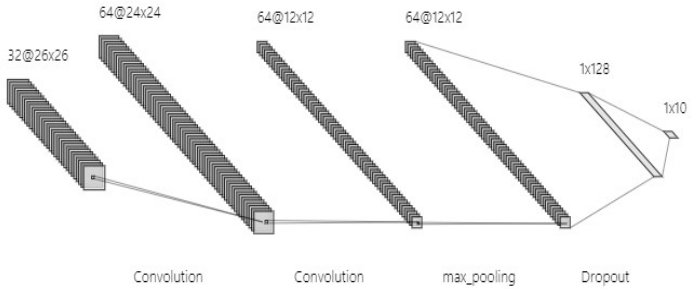
따라하기: SHAP 실습

DeepExplainer
KernelExplainer
TreeExplainer

따라하기: **SHAP** 실습 DeepExplainer

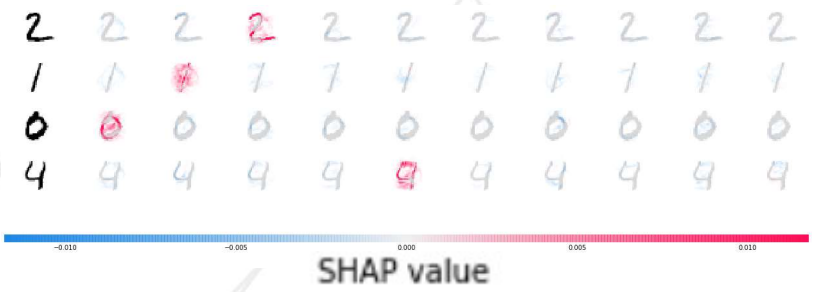
https://snowdeer.github.io/machine-learning/2018/01/09/recognize-mnist-data/

MNIST



CNN

DeepExplainer



따라하기: **SHAP** 실습 DeepExplainer

```
# 사용할 패키지 불러오기
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

keras 모듈에서 제공하는 데이터와 모델 사용

따라하기: **SHAP** 실습 DeepExplainer

```
# 데이터셋 불러와서 훈련셋과 검증셋 분리
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

학습용 데이터 60,000개

검증용 데이터 10,000개

따라하기: **SHAP** 실습 DeepExplainer

```
# 데이터셋 전처리
img_rows, img_cols = 28, 28
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

2차원 배열 이미지 데이터

Shape: (60000, 28, 28) → (60000, 28, 28, 1

Range: (0 ~ 255) → (0 ~ 1)

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

따라하기: **SHAP** 실습 DeepExplainer

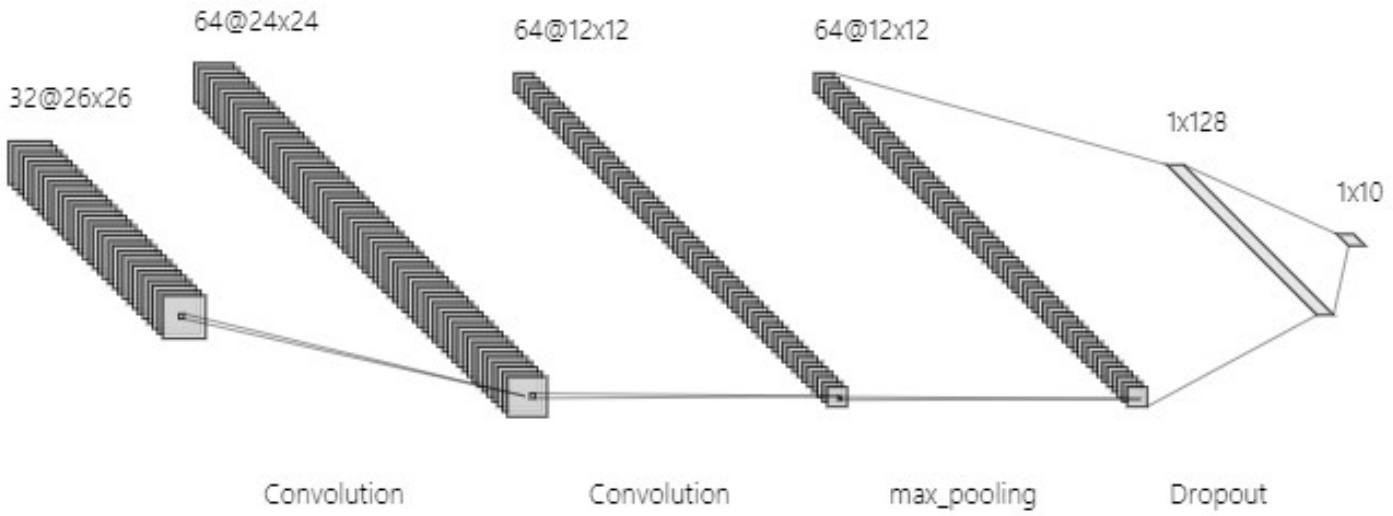
```
# 원핫인코딩 (one-hot encoding) 처리
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

class:
0 1 2 3 4 **5** 6 7 8 9

```
5 → array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

따라하기: **SHAP** 실습 DeepExplainer

```
# 모델 구성
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```



따라하기: **SHAP** 실습 DeepExplainer

```
# 모델 학습과정 설정
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```
# 모델 학습
batch_size = 128
epochs = 2
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 5s 90us/step - loss: 0.2637 - acc: 0.9193 - val_loss: 0.0568 - val_acc: 0.9821
Epoch 2/2
60000/60000 [==============================] - 5s 76us/step - loss: 0.0897 - acc: 0.9731 - val_loss: 0.0443 - val_acc: 0.9851
```

```
# 모델 평가
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.044331139760033694
Test accuracy: 0.9851
```

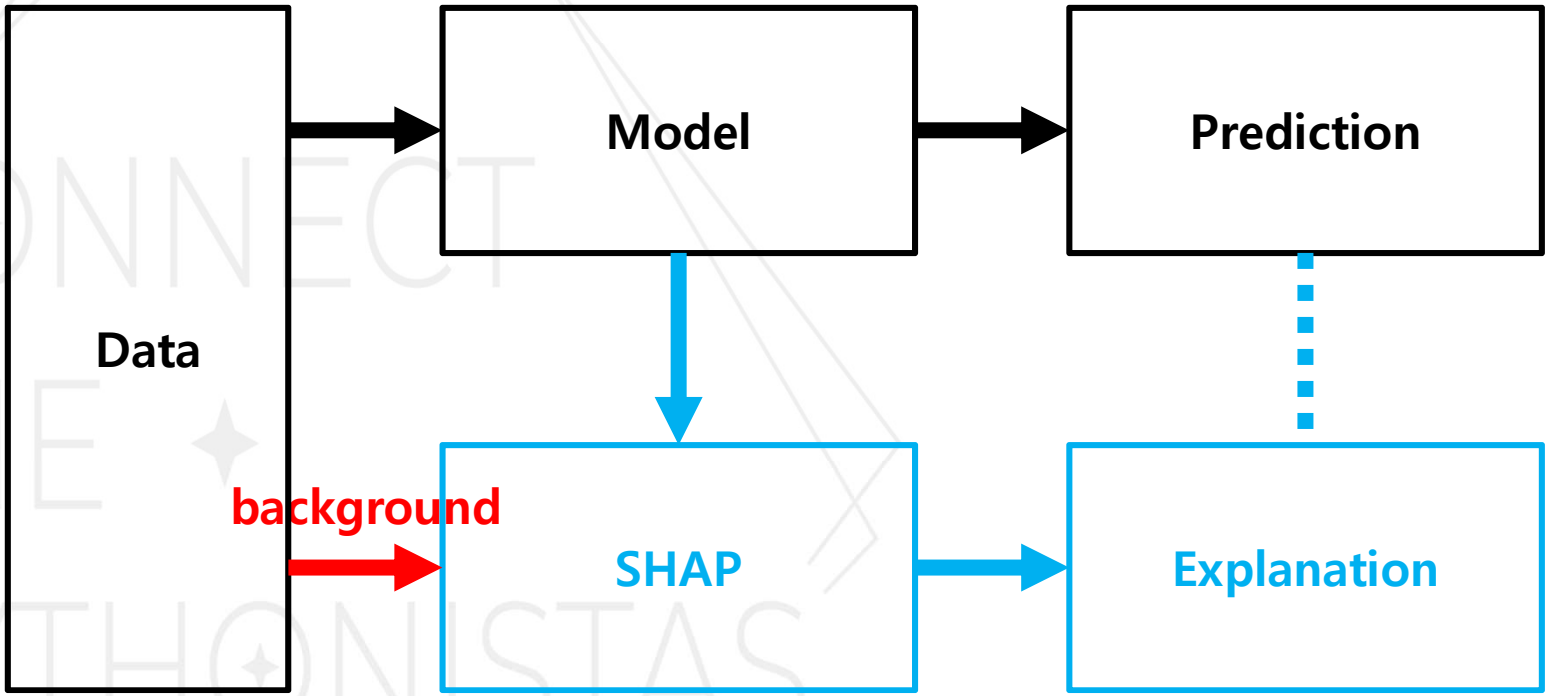
따라하기: **SHAP** 실습 DeepExplainer

```
# 설명 모듈 불러오기
import shap
import numpy as np
```

따라하기: **SHAP** 실습 DeepExplainer

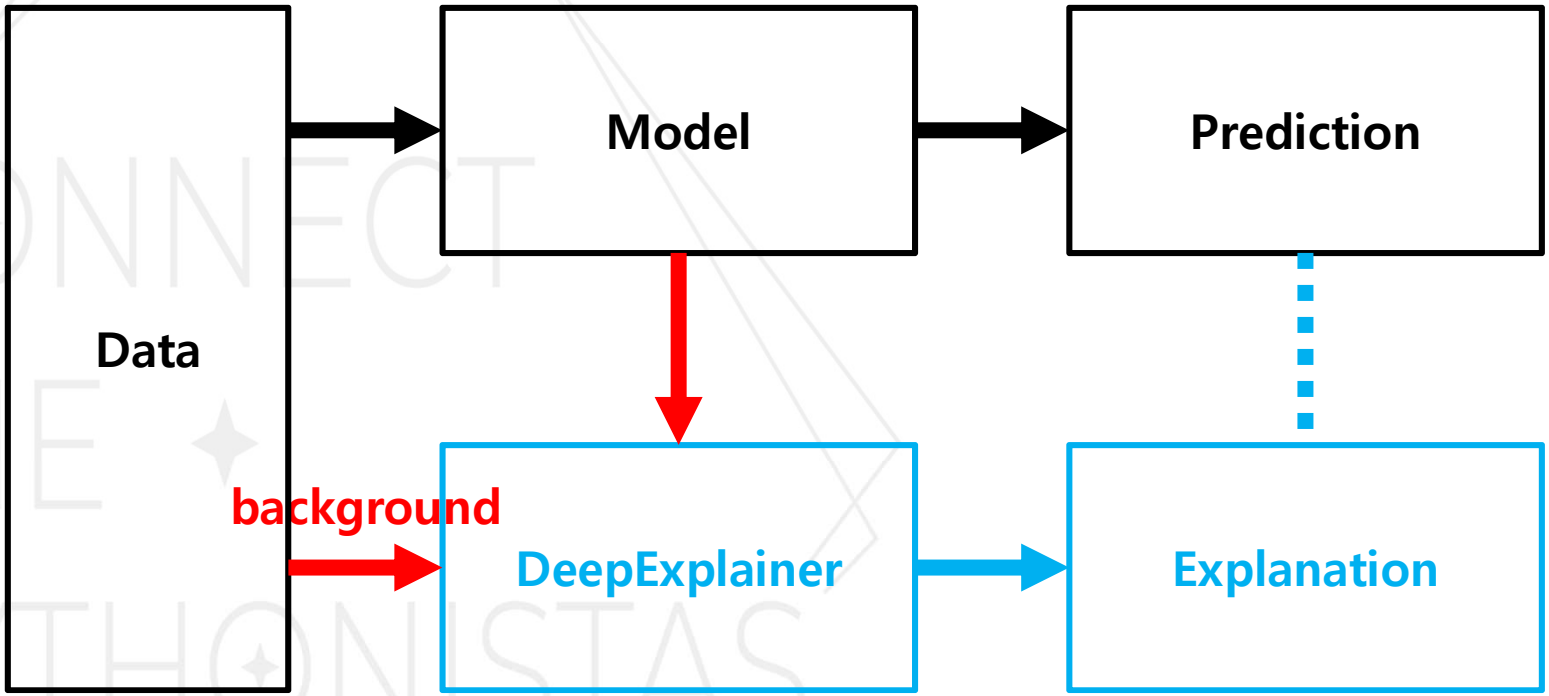
```
# 설명 모듈 불러오기
import shap
import numpy as np
```

```
# shapley value 계산을 위한 백그라운드 데이터 셋 랜덤으로 선택
background = x_train[np.random.choice(x_train.shape[0], 1000, replace=False)]
```



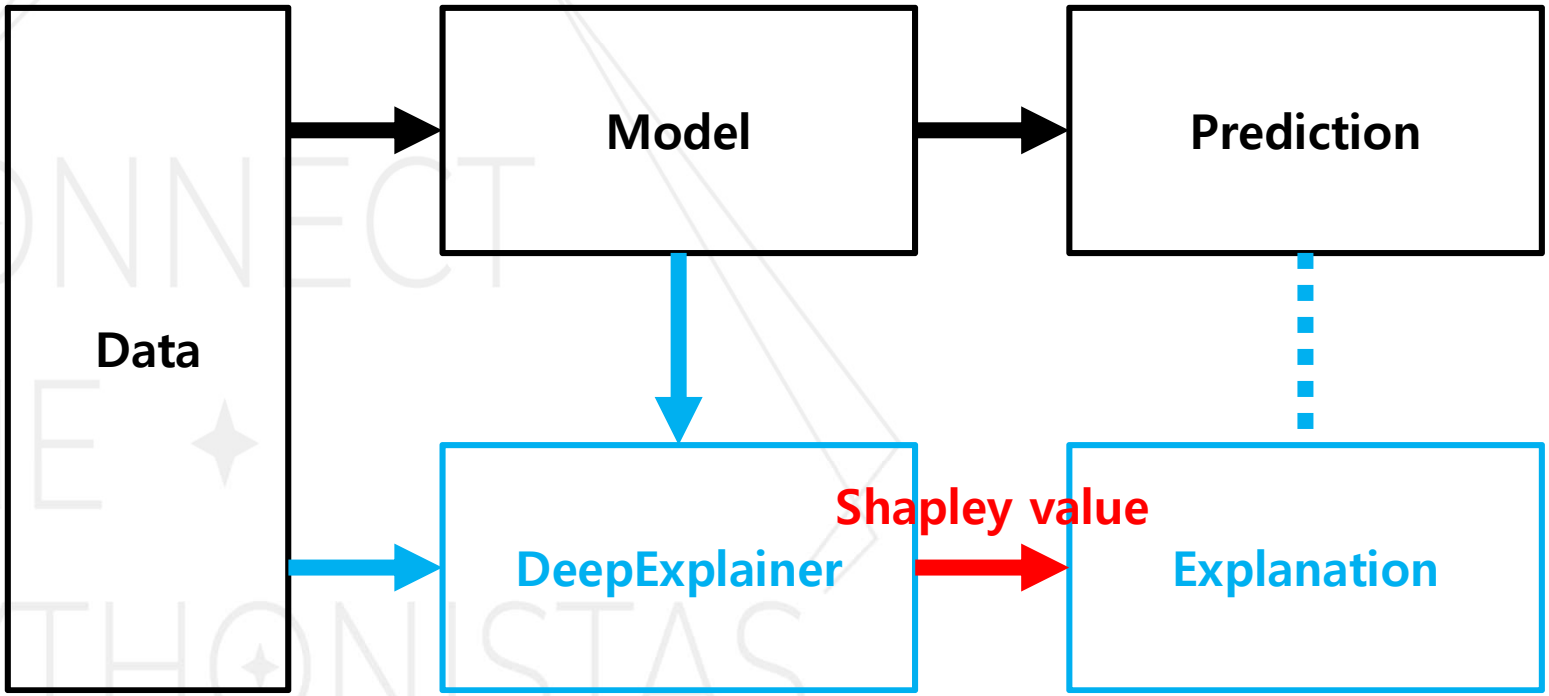
따라하기: **SHAP** 실습 DeepExplainer

```
# 설명 모델 생성
explainer = shap.DeepExplainer(model, background)
```



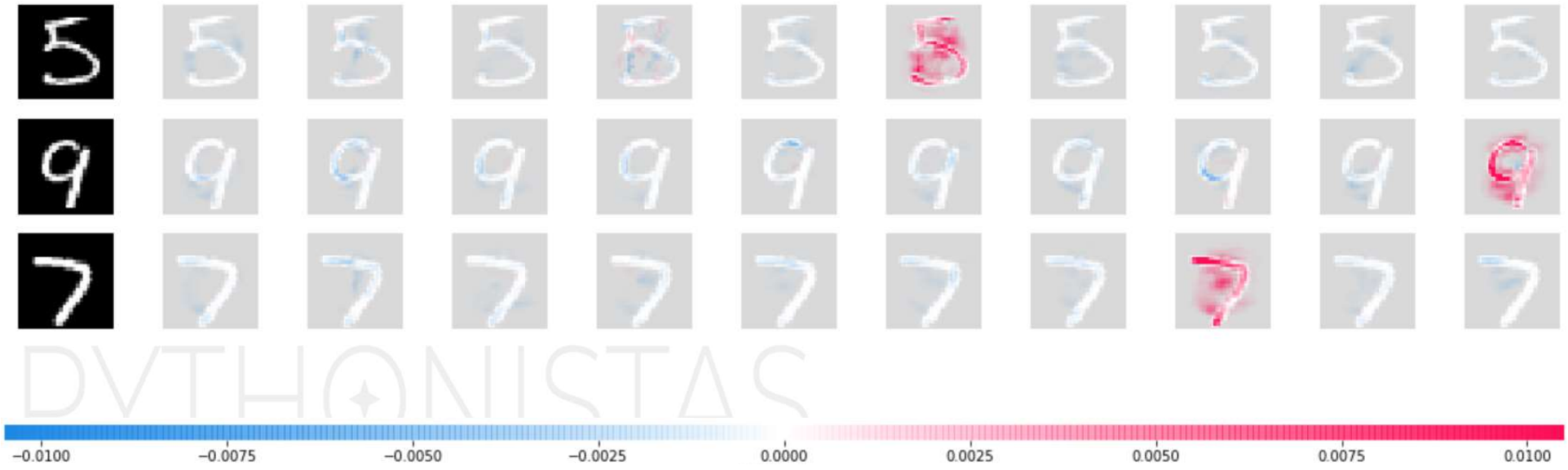
따라하기: **SHAP** 실습 DeepExplainer

```
# 테스트 입력 영상 데이터에 대한 Shap value 계산
test_sample = x_test[15:18]
shap_values = explainer.shap_values(test_sample)

# 입력 데이터 공헌도 시각화
shap.image_plot(shap_values, test_sample)
```



따라하기: **SHAP** 실습 DeepExplainer

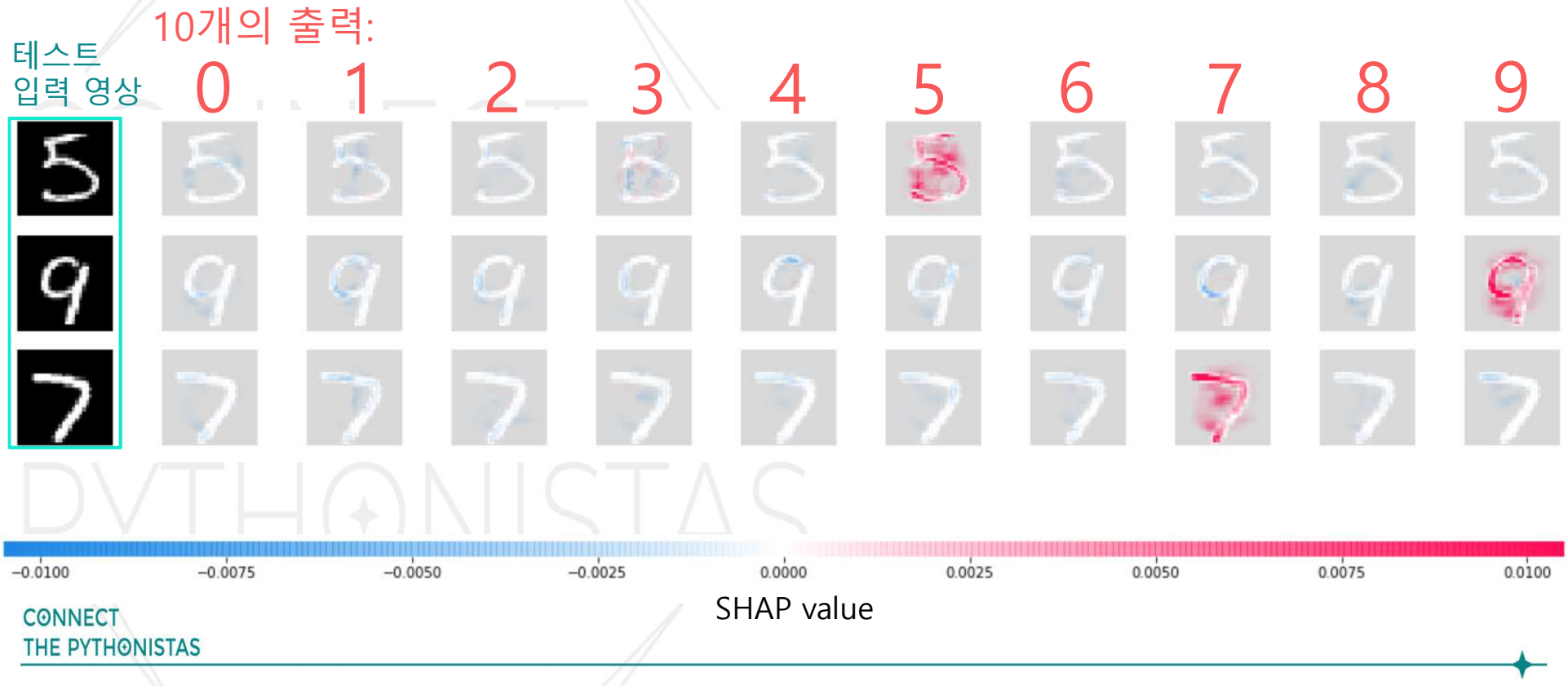
```
# 테스트 입력 영상 데이터에 대한 Shap value 계산
test_sample = x_test[15:18]
shap_values = explainer.shap_values(test_sample)

# 입력 데이터 공헌도 시각화
shap.image_plot(shap_values, test_sample)
```



SHAP value

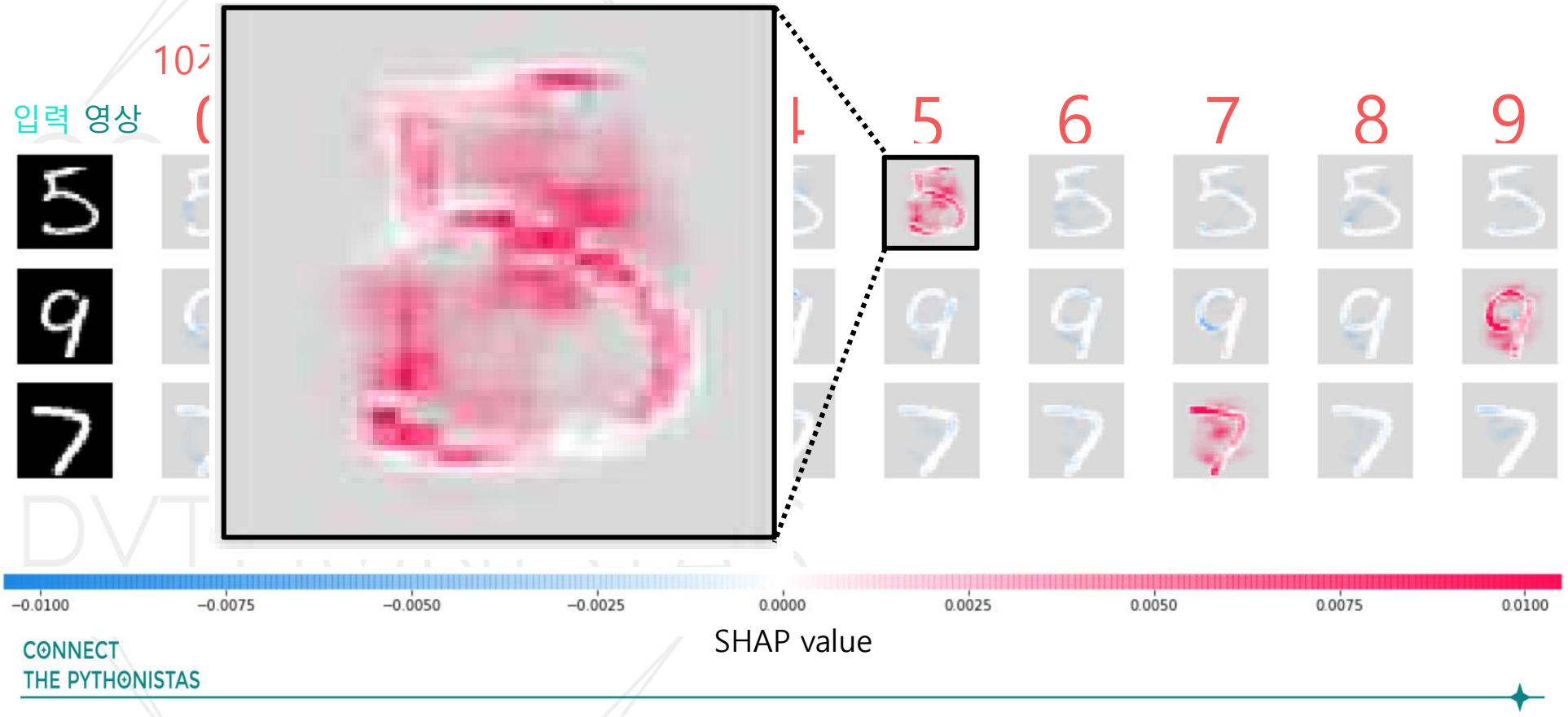
따라하기: **SHAP** 실습 DeepExplainer

```
# 테스트 입력 영상 데이터에 대한 Shap value 계산
test_sample = x_test[15:18]
shap_values = explainer.shap_values(test_sample)

# 입력 데이터 공헌도 시각화
shap.image_plot(shap_values, test_sample)
```



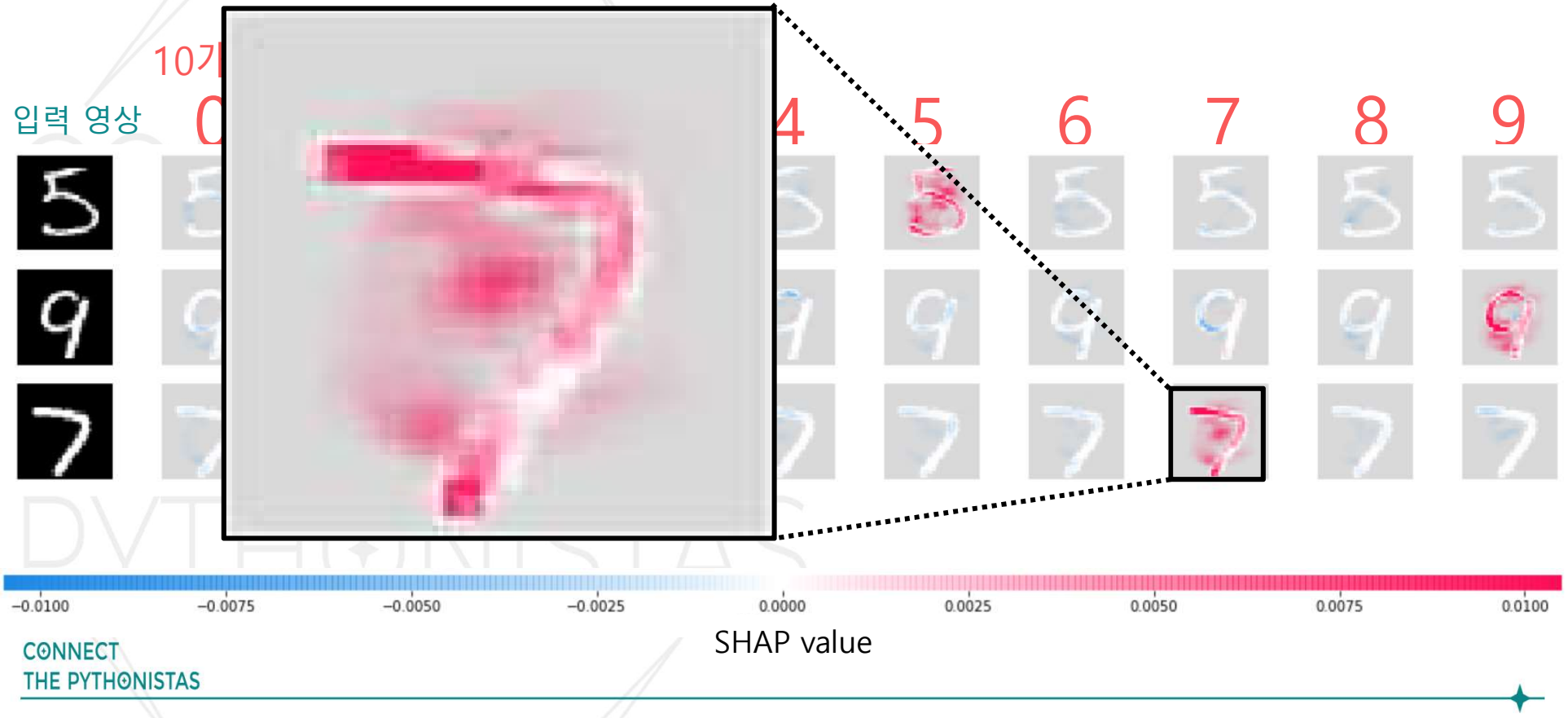
따라하기: **SHAP** 실습 DeepExplainer

```
# 테스트 입력 영상 데이터에 대한 Shap value 계산
test_sample = x_test[15:18]
shap_values = explainer.shap_values(test_sample)

# 입력 데이터 공헌도 시각화
shap.image_plot(shap_values, test_sample)
```



따라하기: **SHAP** 실습 DeepExplainer

```
# 테스트 입력 영상 데이터에 대한 Shap value 계산
test_sample = x_test[15:18]
shap_values = explainer.shap_values(test_sample)

# 입력 데이터 공헌도 시각화
shap.image_plot(shap_values, test_sample)
```



따라하기: SHAP 실습

DeepExplainer

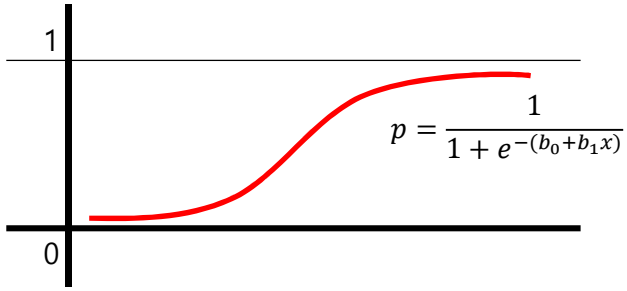
KernelExplainer

TreeExplainer

따라하기: **SHAP** 실습 KernelExplainer

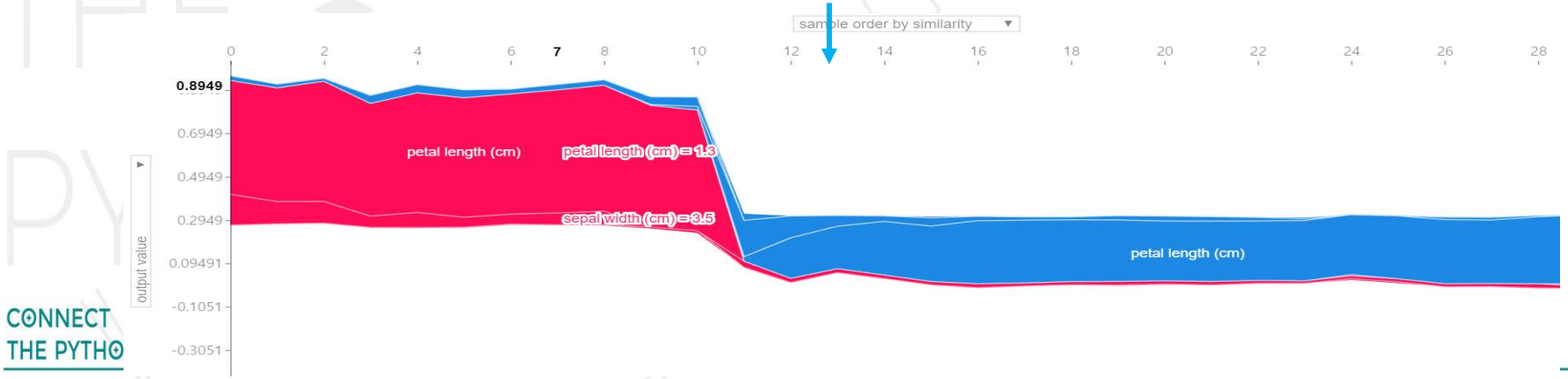
http://www.dicook.org/files/jsm19/slides#11

Iris



Logistic Regression

KernelExplainer



따라하기: **SHAP** 실습 KernelExplainer

```
# 사용할 모듈 불러오기
import sklearn
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import shap
```

```
# Iris 데이터 불러오기
d = sklearn.datasets.load_iris()
df = pd.DataFrame(data=d.data, columns=d.feature_names)
```

따라하기: **SHAP** 실습 KernelExplainer

PYCON KOREA 2019

```
df
```

꽃받침 꽃잎

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
...	...	...	...	...
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

따라하기: SHAP 실습 KernelExplainer

```
print(d.target_names)
print(d.target)
```

```
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

(150,)

Multiclass

- **0: setosa**
- **1: versicolor**
- **2: virginica**

따라하기: **SHAP** 실습 KernelExplainer

PYCON KOREA 2019

```
# 훈련셋과 테스트셋으로 분리
X_train,X_test,Y_train,Y_test = train_test_split(df, d.target,
                                                 test_size=0.2, random_state=0)
```

```
# 분류기 모델 정의
classifier = sklearn.linear_model.LogisticRegression()
```

```
# 분류기 모델 학습
classifier.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# 분류 정확도 출력
def print_accuracy(f):
    print("Accuracy = {0}%".format(100*np.sum(f(X_test) == Y_test)/len(Y_test)))

print_accuracy(classifier.predict)
```

```
Accuracy = 96.66666666666667%
```

따라하기: **SHAP** 실습 KernelExplainer

PYCON KOREA 2019

```
# 설명 모델 생성
explainer = shap.KernelExplainer(linear_lr.predict_proba, X_train)
```

```
# shapley value 계산
test_sample = X_test[0:1]
shap_values = explainer.shap_values(test_sample)
```

```
Y_test
```

2: virginica

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

```
X_test
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
114	5.8	2.8	5.1	2.4
62	6.0	2.2	4.0	1.0
33	5.5	4.2	1.4	0.2

따라하기: **SHAP** 실습 KernelExplainer

```
# 설명 모델 생성
explainer = shap.KernelExplainer(linear_lr.predict_proba, X_train)
```

background

```
# shapley value 계산
test_sample = X_test[0:1]
shap_values = explainer.shap_values(test_sample)
```

```
Y_test
```

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

```
X_test
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
114	5.8	2.8	5.1	2.4
62	6.0	2.2	4.0	1.0
33	5.5	4.2	1.4	0.2

따라하기: **SHAP** 실습 KernelExplainer

```
# 설명 모델 생성
explainer = shap.KernelExplainer(linear_lr.predict_proba, X_train)
```

```
# shapley value 계산
test_sample = X_test[0:1]
shap_values = explainer.shap_values(test_sample)

# shapley value 시각화
shap.initjs()
classNumber = 0 # 0: setosa / 1: versicolor / 2: virginica
shap.force_plot(explainer.expected_value[classNumber],
                shap_values[classNumber], test_sample)
```

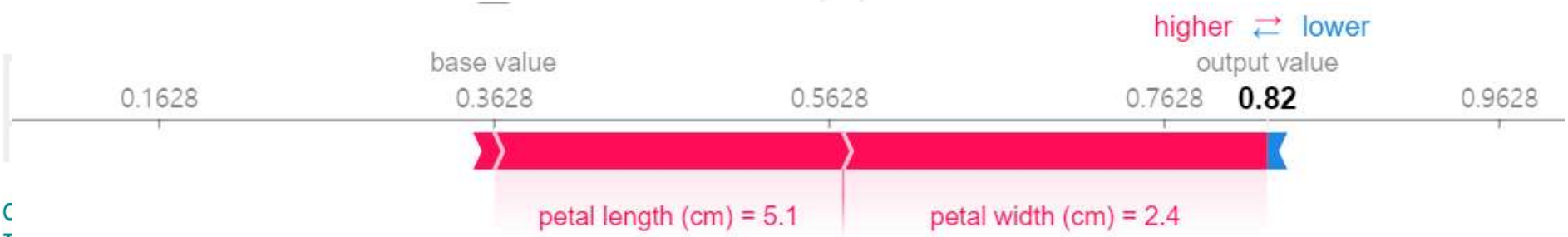
따라하기: **SHAP** 실습 KernelExplainer

PYCON KOREA 2019

classNumber = 0

classNumber = 1

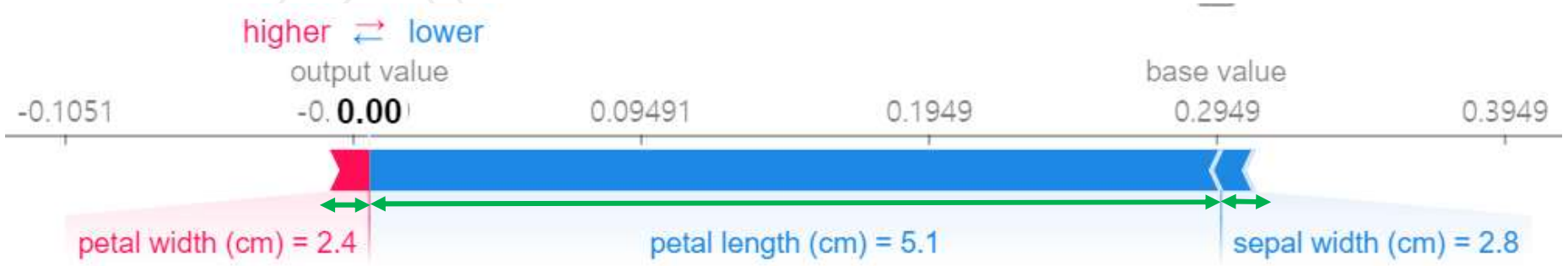
classNumber = 2



따라하기: **SHAP** 실습 KernelExplainer

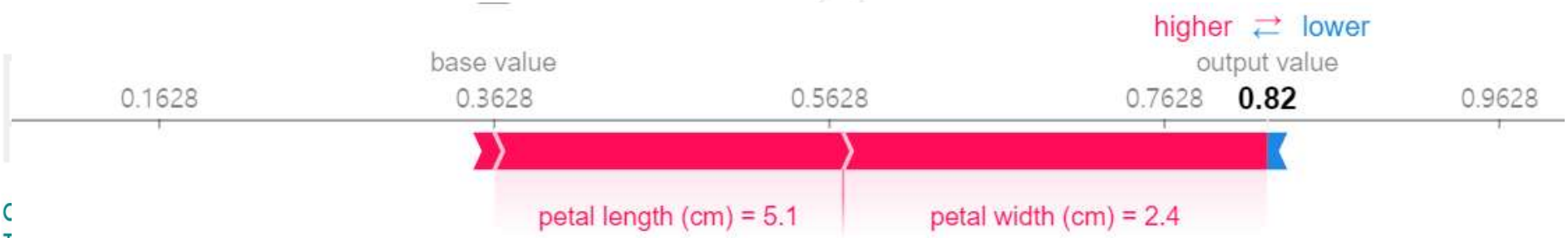
PYCON KOREA 2019

classNumber = 0



classNumber = 1

classNumber = 2



따라하기: SHAP 실습

DeepExplainer
KernelExplainer
TreeExplainer

따라하기: **SHAP** 실습 TreeExplainer

```
# 사용할 모듈 불러오기
import xgboost
import pandas as pd
import shap
```

```
# boston 데이터 불러오기
d = sklearn.datasets.load_boston()
df = pd.DataFrame(data=d.data, columns=d.feature_names)
X = df
y = d.target
```

따라하기: **SHAP** 실습 TreeExplainer

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.09	1.0	296.0	15.3	396.9	4.98

⋮

```
X.shape
```

```
(506, 13)
```

특징 데이터 X:

- CRIM: 범죄율
- INDUS: 비소매상업지역 면적 비율
- NOX: 일산화질소 농도
- RM: 주택당 방 수
- LSTAT: 인구 중 하위 계층 비율
- B: 인구 중 흑인 비율
- PTRATIO: 학생/교사 비율
- ZN: 25,000 평방피트를 초과 거주지역 비율
- CHAS: 찰스강의 경계에 위치한 경우는 1, 아니면 0
- AGE: 1940년 이전에 건축된 주택의 비율
- RAD: 방사형 고속도로까지의 거리
- DIS: 직업센터의 거리
- TAX: 재산세율

따라하기: **SHAP** 실습 TreeExplainer

```
y[0:10]
```

```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9])
```

```
y.shape
```

```
(506,)
```

타겟 데이터 y:

- 1978 보스턴 주택 가격
- 506개 타운의 주택 가격 중앙값 (단위 1,000 달러)

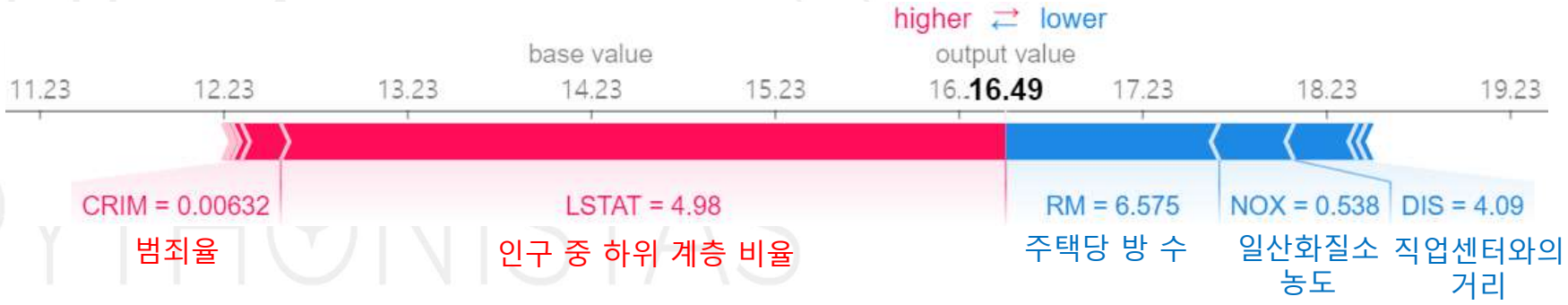
따라하기: **SHAP** 실습 TreeExplainer

PYCON KOREA 2019

```
# 예측 모델 학습(XGBoost)
model = xgboost.train({"learning_rate": 0.01}, xgboost.DMatrix(X, label=y), 100)
```

```
# 설명 모델 생성
explainer = shap.TreeExplainer(model)
# shapley value 계산
shap_values = explainer.shap_values(X)
```

```
# 하나의 데이터에 대해 shapley value 시각화
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```



따라하기: **SHAP** 실습 TreeExplainer

```
# 학습 데이터 전체에 대해 시각화
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values, X)
```

적용하기: SHAP 활용

Red Wine Quality

적용하기: **SHAP** 활용 Red Wine Quality

Data Kernels (201) Discussion (7) Activity Metadata Download (26 KB) **New Notebook**

Usability 8.8 **License** Database: Open Database, Contents: Database Contents **Tags** beginner, food and drink, regression analysis

Description

Context

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).

https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

적용하기: **SHAP** 활용 Red Wine Quality

PYCON KOREA 2019

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
⋮												
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null float64
volatile acidity        1599 non-null float64
citric acid             1599 non-null float64
residual sugar          1599 non-null float64
chlorides               1599 non-null float64
free sulfur dioxide     1599 non-null float64
total sulfur dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

특징 데이터:

- fixed acidity: 고정 산도
- volatile acidity: 휘발성 산도
- citric acid: 구연산
- residual sugar: 자연 발효로 생성된 당분
- chlorides: 염소
- free sulfur dioxide: 유황 이산화황
- total sulfur dioxide: 총 이산화황
- density: 밀도
- pH
- sulphates: 황산염
- alcohol: 알코올
- quality (score between 0 and 10): 품질(0~10점)

PyCon Korea 2019

감사합니다.