Python Lime

LIME - Local Interpretable Model-agnostic Explanations

머신러닝 모델은 복잡해질수록 예측의 정확도가 올라가지만 결과의 해석은 어렵다는 단점이 존재하는데 이와 같은 현상을 **블랙박스**라고 한다.

LIME은 이런 블랙박스 현상을 좀 더 자세하게 바라볼 수 있도록 해주는 알고리즘이다.

아이디어

복잡한 데이터의 경우 전역적인 해석이 매우 어렵다.

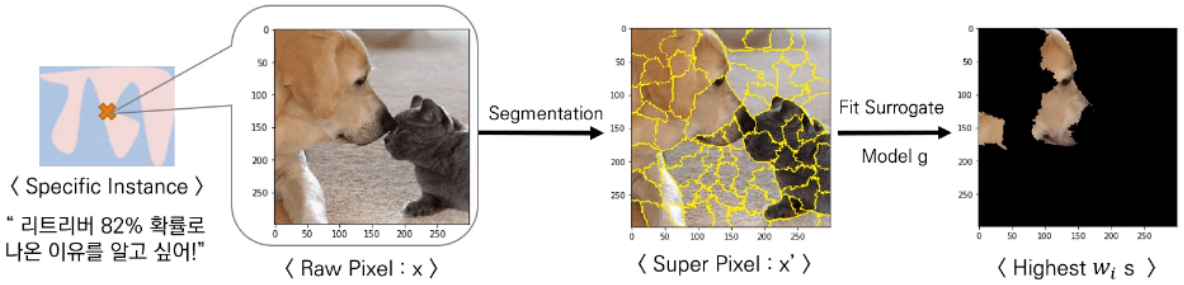
국소적(Local)으로는 비교적 해석이 간단한 Surrogate Model로 근사값 해석이 가능

*Lime은 국소적 접근법이므로 이미지 분석 모델의 경우 여러 Step과정을 거쳐야 한다.

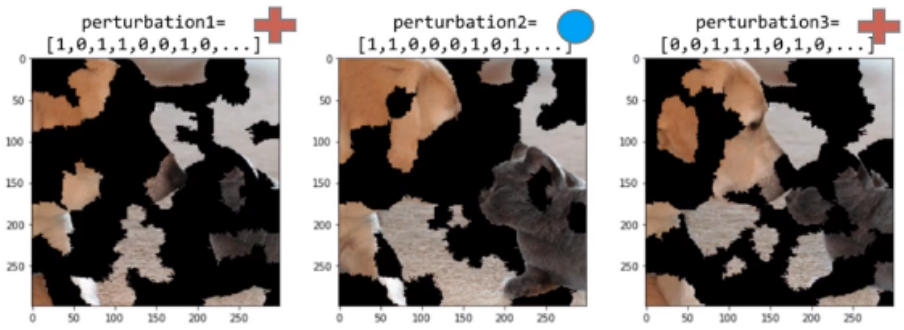
*Surrogate model : 원래 모델 자체로 해석하기 어려울 때 사용하는 대리 모델

Steps 간단 요약 (완벽하게 이해하려 하지말고 그림만 보고 넘어가자)

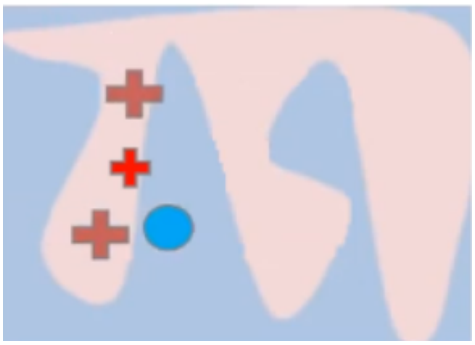
1.해석하고자 하는 관측치(x)를 고르고 x'의 Super Pixel을 구한다.



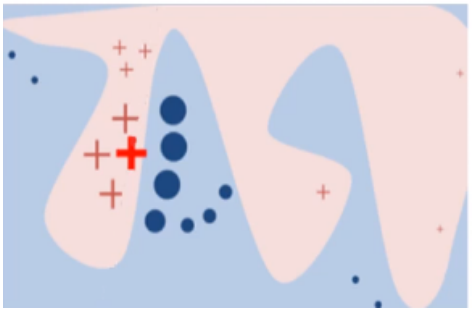
2.x'(하나의 큰 super pixel)을 Perturb(셔플)하여 만든 여러 개의 z(Super Pixel)점들을 만든다.



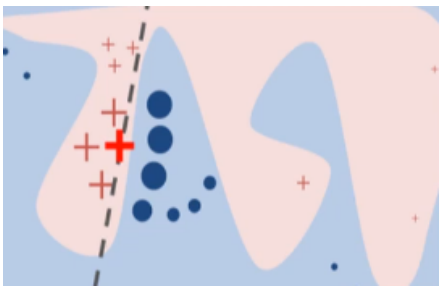
3.h라는 함수를 이용해 각 z들을 다시 mapping 진행



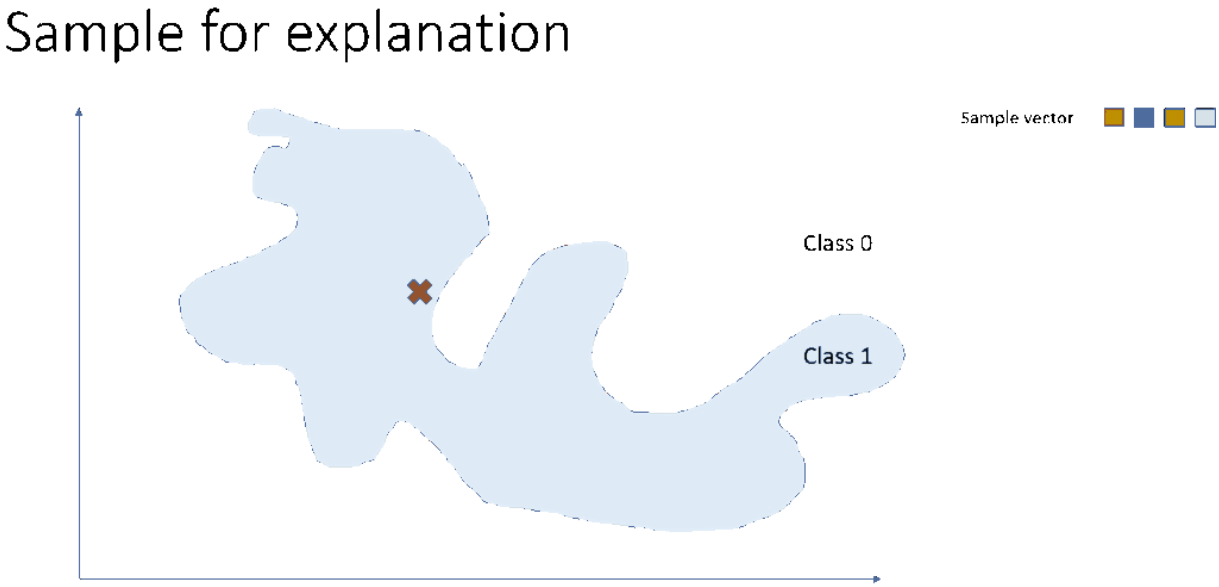
4.구하고자 하는 x라는 값에 가까운 local한 점들에 가중치를 크게 부여(특성 부각)



5.Surrogate model 적용



과정 요약 시각화



장점

1.개별 데이터 인스턴스에 대한 local 해석력을 제공

2.perturb의 방식을 다르게 하면 model-agnostic하게 해석하게 해줌

*model-agnostic : 모델에 구애받지 않고 모두 해석이 가능함을 의미

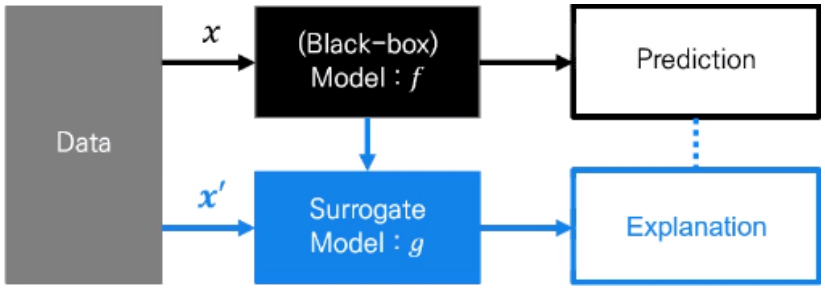
3.Shap에 비해 가벼움

단점

1.국소적으로도 비선형인 경우 선형을 가정으로 하는 Lime의 한계점이 명확함

2.하이퍼 파라미터에 따라서 결과가 천차만별인 불안정성(불공평성)이 존재

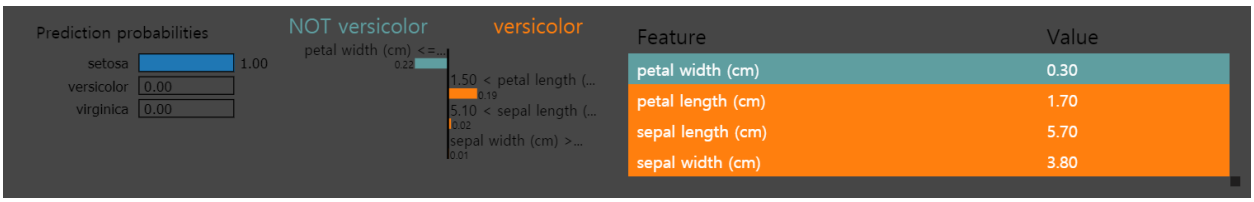
model-agnostic



실습 코드 - 일반 머신러닝(회귀)

```
#라이브러리 설치
import pandas as pd
import numpy as np

import lime
from lime import lime_tabular

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

#load data set
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
df

#x, y split
x = df.iloc[:,:-1]
y = df.iloc[:, -1]

#hold out
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

#Modeling
rf = RandomForestClassifier(n_jobs=-1, random_state=42)
rf.fit(x_train, y_train)
rf.score(x_test, y_test)

#lime explainer
explainer = lime_tabular.LimeTabularExplainer(
    training_data = np.array(x_train),
    feature_names = x_train.columns,
    class_names = ['setosa', 'versicolor', 'virginica'],
    mode = 'classification'
)

#show lime result table
exp = explainer.explain_instance(
    data_row = x_test.iloc[1],
    predict_fn = rf.predict_proba
)

exp.show_in_notebooks(show_table=True)
```

결과



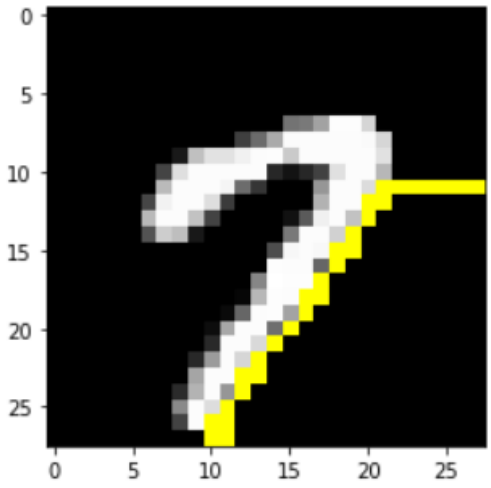
실습 코드 - 딥러닝(이미지)

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten

# 1. 데이터 불러오기
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 2. 이미지 데이터 확인하기 🖼
import matplotlib.pyplot as plt
plt.imshow(x_train[0])

# 3-1. 이미지 데이터 전처리 : 2차원->3차원 🌟🌟🌟
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

# 3-2. 이미지 데이터 전처리 : Normalzation
x_train, x_test = x_train/255.0, x_test/255.0

#흑백 이미지라면 컬러로 변화시켜 주어야 함.
import numpy as np
def to_rgb(x):
    x_rgb = np.zeros((x.shape[0], 28, 28, 3))
    for i in range(3):
        x_rgb[..., i] = x[..., 0]
    return x_rgb
x_train = to_rgb(x_train)
x_test = to_rgb(x_test)

#5. 모델 생성 : CNN 🌟🌟🌟
model = keras.Sequential([Conv2D(512, activation='relu', kernel_size=(5,5), input_shape=(28,28,3), padding='same'),
                          MaxPooling2D(pool_size=(2,2), strides=(2,2)),
                          Conv2D(256, activation='relu', kernel_size=(2,2)),
                          MaxPooling2D(pool_size=(2,2), strides=(2,2)),
                          Dropout(0.3),
                          Flatten(),
                          Dense(512, activation='relu'),
                          Dropout(0.4),
                          Dense(10, activation='softmax')])

#모델 컴파일
model.compile(
  loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  optimizer=keras.optimizers.Adam(),
  metrics=['accuracy']
)

# 7. 모델 학습시키기
# hist = model.fit(x_train, y_train, batch_size=64, epochs=3, validation_data=(x_test,y_test))
hist = model.fit(
        x_train,
        y_train,
        epochs=10,
        batch_size=32,
        validation_data = (x_test, y_test))

# 8. 모델 평가하기
model.evaluate(x_test, y_test)

--------------------------------------------------------------------------------

#LIME 라이브러리 설치
import lime
from lime import lime_image
from skimage.segmentation import mark_boundaries
import random
```

```
explainer = lime_image.LimeImageExplainer(random_state=42)
explanation = explainer.explain_instance(x_train[15], model.predict)
plt.imshow(x_train[15])
image, mask = explanation.get_image_and_mask(
         model.predict(
              x_train[15].reshape((1,28,28,3))
         ).argmax(axis=1)[0],
         positive_only=True,
         hide_rest=True)
plt.imshow(mark_boundaries(image, mask))
```

결과



***위의 딥러닝 사례는 비교적 간단한 분석 데이터이기 때문에 2 Step을 거치지 않음**