1. 개와 고양이 분류 CNN 모델에 LIME 적용

In [1]:
```
import glob
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_
```

In [2]:
```
train_files = glob.glob('training_data/*')
```

In [3]:
```
IMG_DIM = (150, 150)

train_files = glob.glob('training_data/*')
train_imgs = [img_to_array(load_img(img, target_size=IMG_DIM)) for img in train_fil
train_imgs = np.array(train_imgs)
train_labels = [fn.split('₩₩')[1].split('.')[0].strip() for fn in train_files] # fo

validation_files = glob.glob('validation_data/*')
validation_imgs = [img_to_array(load_img(img, target_size=IMG_DIM)) for img in vali
validation_imgs = np.array(validation_imgs)

validation_labels = [fn.split('₩₩')[1].split('.')[0].strip() for fn in validation_f

print('Train dataset shape:', train_imgs.shape,
      '₩tValidation dataset shape:', validation_imgs.shape)
```

```
Train dataset shape: (1000, 150, 150, 3)        Validation dataset shape: (200, 150,
150, 3)
```

In [4]:
```
train_imgs_scaled = train_imgs.astype('float32')
validation_imgs_scaled  = validation_imgs.astype('float32')
train_imgs_scaled /= 255
validation_imgs_scaled /= 255
```

In [5]:
```
print(train_imgs[0].shape)
print(train_imgs[0])
array_to_img(train_imgs[0])
```

```
(150, 150, 3)
[[[101. 103. 100.]
  [123. 125. 122.]
  [116. 118. 115.]
  ...
  [ 45.  41.  30.]
  [ 56.  52.  41.]
  [ 78.  74.  63.]]

 [[ 91.  93.  90.]
  [119. 121. 118.]
  [116. 118. 115.]
  ...
  [ 46.  42.  31.]
  [ 64.  60.  49.]
  [ 84.  80.  69.]]

 [[ 80.  82.  79.]
  [119. 121. 118.]
  [122. 124. 121.]
  ...
  [ 47.  43.  32.]
  [ 72.  68.  57.]
  [ 86.  82.  71.]]

 ...

 [[ 30.  34.  35.]
  [ 32.  36.  37.]
  [ 35.  39.  40.]
  ...
  [ 65.  70.  66.]
  [106. 108. 105.]
  [171. 173. 170.]]

 [[ 31.  35.  36.]
  [ 33.  37.  38.]
  [ 28.  32.  33.]
  ...
  [ 51.  56.  52.]
  [ 93.  95.  92.]
  [152. 154. 151.]]

 [[ 30.  34.  35.]
  [ 33.  37.  38.]
  [ 28.  32.  33.]
  ...
  [ 39.  44.  40.]
  [ 82.  84.  81.]
  [134. 136. 133.]]]
```

Out[5]:

In [6]:

```
batch_size = 30
num_classes = 2
epochs = 30
input_shape = (150, 150, 3)
```

```
# encode text category labels
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(train_labels)
train_labels_enc = le.transform(train_labels)
validation_labels_enc = le.transform(validation_labels)

print(train_labels[0:10], train_labels_enc[0:10])
```

```
['cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat'] [0 0 0 0 0 0
0 0 0 0]
```

In [7]:
```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers

# sequential은 한종류로 쌓겠다는 뜻
# 뒤로갈수록 세밀한 특징을 뽑음
# 커널 개수가 16->64->128로 늘었고, max pooling도 64->128로 늘음
# 즉 커널을 많이 쓰면 같은 feature을 세밀하게 뽑겠다는 뜻

model = Sequential()

model.add(Conv2D(16, kernel_size=(3, 3), activation='relu', # kernel은 3*3짜리 16개
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu')) # kernel은 3*3짜리 64개
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))


model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(),
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 16)      448       
                                                                 
 max_pooling2d (MaxPooling2  (None, 74, 74, 16)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        9280      
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 36992)             0         
                                                                 
 dense (Dense)               (None, 512)               18940416  
                                                                 
 dense_1 (Dense)             (None, 1)                 513       
                                                                 
=================================================================
Total params: 19024513 (72.57 MB)
Trainable params: 19024513 (72.57 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [8]:
```
history = model.fit(x=train_imgs_scaled, y=train_labels_enc,
                    validation_data=(validation_imgs_scaled, validation_labels_enc),
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1)
```

```
Epoch 1/30
34/34 [==============================] - 44s 1s/step - loss: 0.7678 - accuracy: 0.51
70 - val_loss: 0.6895 - val_accuracy: 0.5850
Epoch 2/30
34/34 [==============================] - 43s 1s/step - loss: 0.6841 - accuracy: 0.55
40 - val_loss: 0.6568 - val_accuracy: 0.6050
Epoch 3/30
34/34 [==============================] - 43s 1s/step - loss: 0.6585 - accuracy: 0.59
90 - val_loss: 0.6423 - val_accuracy: 0.6450
Epoch 4/30
34/34 [==============================] - 43s 1s/step - loss: 0.6171 - accuracy: 0.66
70 - val_loss: 0.6148 - val_accuracy: 0.6600
Epoch 5/30
34/34 [==============================] - 43s 1s/step - loss: 0.5805 - accuracy: 0.68
80 - val_loss: 0.6761 - val_accuracy: 0.6000
Epoch 6/30
34/34 [==============================] - 44s 1s/step - loss: 0.5178 - accuracy: 0.73
10 - val_loss: 0.5810 - val_accuracy: 0.7100
Epoch 7/30
34/34 [==============================] - 47s 1s/step - loss: 0.4637 - accuracy: 0.77
20 - val_loss: 0.6658 - val_accuracy: 0.6700
Epoch 8/30
34/34 [==============================] - 41s 1s/step - loss: 0.3969 - accuracy: 0.82
30 - val_loss: 0.9557 - val_accuracy: 0.5800
Epoch 9/30
34/34 [==============================] - 41s 1s/step - loss: 0.3328 - accuracy: 0.85
30 - val_loss: 0.7440 - val_accuracy: 0.7300
Epoch 10/30
34/34 [==============================] - 41s 1s/step - loss: 0.2826 - accuracy: 0.88
90 - val_loss: 0.9775 - val_accuracy: 0.6200
Epoch 11/30
34/34 [==============================] - 40s 1s/step - loss: 0.1888 - accuracy: 0.92
40 - val_loss: 0.8949 - val_accuracy: 0.7100
Epoch 12/30
34/34 [==============================] - 40s 1s/step - loss: 0.1577 - accuracy: 0.94
40 - val_loss: 0.9082 - val_accuracy: 0.6850
Epoch 13/30
34/34 [==============================] - 41s 1s/step - loss: 0.1344 - accuracy: 0.96
40 - val_loss: 1.0171 - val_accuracy: 0.7000
Epoch 14/30
34/34 [==============================] - 43s 1s/step - loss: 0.0726 - accuracy: 0.97
40 - val_loss: 1.3698 - val_accuracy: 0.6900
Epoch 15/30
34/34 [==============================] - 42s 1s/step - loss: 0.0672 - accuracy: 0.97
90 - val_loss: 1.3347 - val_accuracy: 0.7150
Epoch 16/30
34/34 [==============================] - 41s 1s/step - loss: 0.0071 - accuracy: 1.00
00 - val_loss: 1.7554 - val_accuracy: 0.6950
Epoch 17/30
34/34 [==============================] - 42s 1s/step - loss: 0.0826 - accuracy: 0.98
00 - val_loss: 1.7793 - val_accuracy: 0.6250
Epoch 18/30
34/34 [==============================] - 43s 1s/step - loss: 0.0370 - accuracy: 0.99
00 - val_loss: 2.0783 - val_accuracy: 0.6650
Epoch 19/30
34/34 [==============================] - 42s 1s/step - loss: 0.0060 - accuracy: 0.99
80 - val_loss: 1.9362 - val_accuracy: 0.7050
Epoch 20/30
34/34 [==============================] - 41s 1s/step - loss: 5.4585e-04 - accuracy:
1.0000 - val_loss: 2.3239 - val_accuracy: 0.7000
Epoch 21/30
34/34 [==============================] - 41s 1s/step - loss: 1.7216e-04 - accuracy:
1.0000 - val_loss: 2.6105 - val_accuracy: 0.7050
Epoch 22/30
```

```
34/34 [==============================] - 43s 1s/step - loss: 9.7265e-05 - accuracy:
1.0000 - val_loss: 2.6287 - val_accuracy: 0.6850
Epoch 23/30
34/34 [==============================] - 45s 1s/step - loss: 0.2583 - accuracy: 0.96
10 - val_loss: 1.5674 - val_accuracy: 0.6900
Epoch 24/30
34/34 [==============================] - 44s 1s/step - loss: 0.0132 - accuracy: 0.99
60 - val_loss: 1.6115 - val_accuracy: 0.7200
Epoch 25/30
34/34 [==============================] - 41s 1s/step - loss: 8.8247e-04 - accuracy:
1.0000 - val_loss: 1.8995 - val_accuracy: 0.7400
Epoch 26/30
34/34 [==============================] - 41s 1s/step - loss: 3.6507e-04 - accuracy:
1.0000 - val_loss: 2.3728 - val_accuracy: 0.6700
Epoch 27/30
34/34 [==============================] - 42s 1s/step - loss: 0.0210 - accuracy: 0.99
50 - val_loss: 2.4420 - val_accuracy: 0.6850
Epoch 28/30
34/34 [==============================] - 42s 1s/step - loss: 3.2010e-04 - accuracy:
1.0000 - val_loss: 2.6263 - val_accuracy: 0.7000
Epoch 29/30
34/34 [==============================] - 41s 1s/step - loss: 7.4491e-05 - accuracy:
1.0000 - val_loss: 2.8692 - val_accuracy: 0.6900
Epoch 30/30
34/34 [==============================] - 55s 2s/step - loss: 3.6963e-05 - accuracy:
1.0000 - val_loss: 3.0252 - val_accuracy: 0.6850
```

In [9]:

```
!pip install LIME
```

```
Requirement already satisfied: LIME in c:\users\hanjo\anaconda3\lib\site-packages (0.2.0.1)
Requirement already satisfied: matplotlib in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (3.7.1)
Requirement already satisfied: numpy in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (1.24.3)
Requirement already satisfied: scipy in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (1.11.1)
Requirement already satisfied: tqdm in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (4.65.0)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (1.3.0)
Requirement already satisfied: scikit-image>=0.12 in c:\users\hanjo\anaconda3\lib\site-packages (from LIME) (0.20.0)
Requirement already satisfied: networkx>=2.8 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (3.1)
Requirement already satisfied: pillow>=9.0.1 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (9.4.0)
Requirement already satisfied: imageio>=2.4.1 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (2.26.0)
Requirement already satisfied: tifffile>=2019.7.26 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (2021.7.2)
Requirement already satisfied: PyWavelets>=1.1.1 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (1.4.1)
Requirement already satisfied: packaging>=20.0 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (23.0)
Requirement already satisfied: lazy_loader>=0.1 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-image>=0.12->LIME) (0.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-learn>=0.18->LIME) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hanjo\anaconda3\lib\site-packages (from scikit-learn>=0.18->LIME) (2.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hanjo\anaconda3\lib\site-packages (from matplotlib->LIME) (2.8.2)
Requirement already satisfied: colorama in c:\users\hanjo\anaconda3\lib\site-packages (from tqdm->LIME) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\users\hanjo\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->LIME) (1.16.0)
```

In [10]:

```python
from lime import lime_image
from lime.wrappers.scikit_image import SegmentationAlgorithm

explainer = lime_image.LimeImageExplainer()
```

In [11]:

```python
# 이미지를 슈퍼픽셀로 분할하는 알고리즘 설정
# quickshift, slic, felzenswalb 등이 존재
# 과제에 segmenter 설정 조건은(slic...) 그대로 활용
segmenter = SegmentationAlgorithm('slic',
                                  n_segments=100, # 이미지 분할 조각 개수
                                  compactnes=1, # 유사한 파트를 합치는 함수
                                  sigma=1) # 스무딩 역할: 0과 1사이의 float
```

In [12]:
```
X_test = validation_imgs_scaled

olivetti_test_index = 0
exp = explainer.explain_instance(X_test[olivetti_test_index],
                                 classifier_fn=model.predict,
                                 top_labels=1,  # 클래스가 2개인 이진 분류 문제이므로
                                 num_samples=1000,
                                 segmentation_fn=segmenter)
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 321ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 128ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 329ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 176ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 220ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 93ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 97ms/step
```

```
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 104ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 111ms/step
```

In [13]:

```python
X_test = validation_imgs_scaled
olivetti_test_index = 0 # 0번째 장을 집어 넣음 exp=model(우리 딥러닝에서 한것처럼)
exp = explainer.explain_instance(X_test[olivetti_test_index],
                                 classifier_fn=model.predict, # 40개 class 확률 반환
                                 top_labels=5, # 확률 기준 1~5위 (과제: top_labels=2(
                                 num_samples=1000, # sample space
                                 segmentation_fn=segmenter) # 분할 알고리즘
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 206ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 180ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 90ms/step
```

```
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 405ms/step
1/1 [==============================] - 1s 550ms/step
1/1 [==============================] - 0s 227ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 111ms/step
```

In [14]:

```
from skimage.color import label2rgb

def visualize_explanation(image_index, explainer, model, X_data, y_data, num_feature
    # 설명 생성
    exp = explainer.explain_instance(X_data[image_index],
                                     classifier_fn=model.predict,
                                     top_labels=5,
                                     num_samples=1000,
                                     segmentation_fn=segmenter)

    # 시각화
    fig, m_axs = plt.subplots(2, 2, figsize=(8, 8))
    ax = m_axs.ravel()
    for i in ax:
        i.grid(False)

    # Positive Regions
    temp, mask = exp.get_image_and_mask(y_data[image_index],
                                        positive_only=True,
                                        num_features=num_features,
                                        hide_rest=False)
    ax[0].imshow(label2rgb(mask, temp, bg_label=0), interpolation='nearest')
    ax[0].set_title('Positive Regions for {}'.format(y_data[image_index]))

    # Positive/Negative Regions
    temp, mask = exp.get_image_and_mask(y_data[image_index],
                                        positive_only=False,
```

```
                                              num_features=num_features,
                                              hide_rest=False)
    ax[1].imshow(label2rgb(4 - mask, temp, bg_label=0), interpolation='nearest')
    ax[1].set_title('Positive/Negative Regions for {}'.format(y_data[image_index]))

    # Show output image only
    ax[2].imshow(temp, interpolation='nearest')
    ax[2].set_title('Show output image only')

    # Show mask only
    ax[3].imshow(mask, interpolation='nearest')
    ax[3].set_title('Show mask only')
```

In [15]:
```
y_pred = model.predict(validation_imgs_scaled)
y_pred_num = np.argmax(y_pred, axis=1)
# 실제 라벨
y_test_num = validation_labels_enc
```

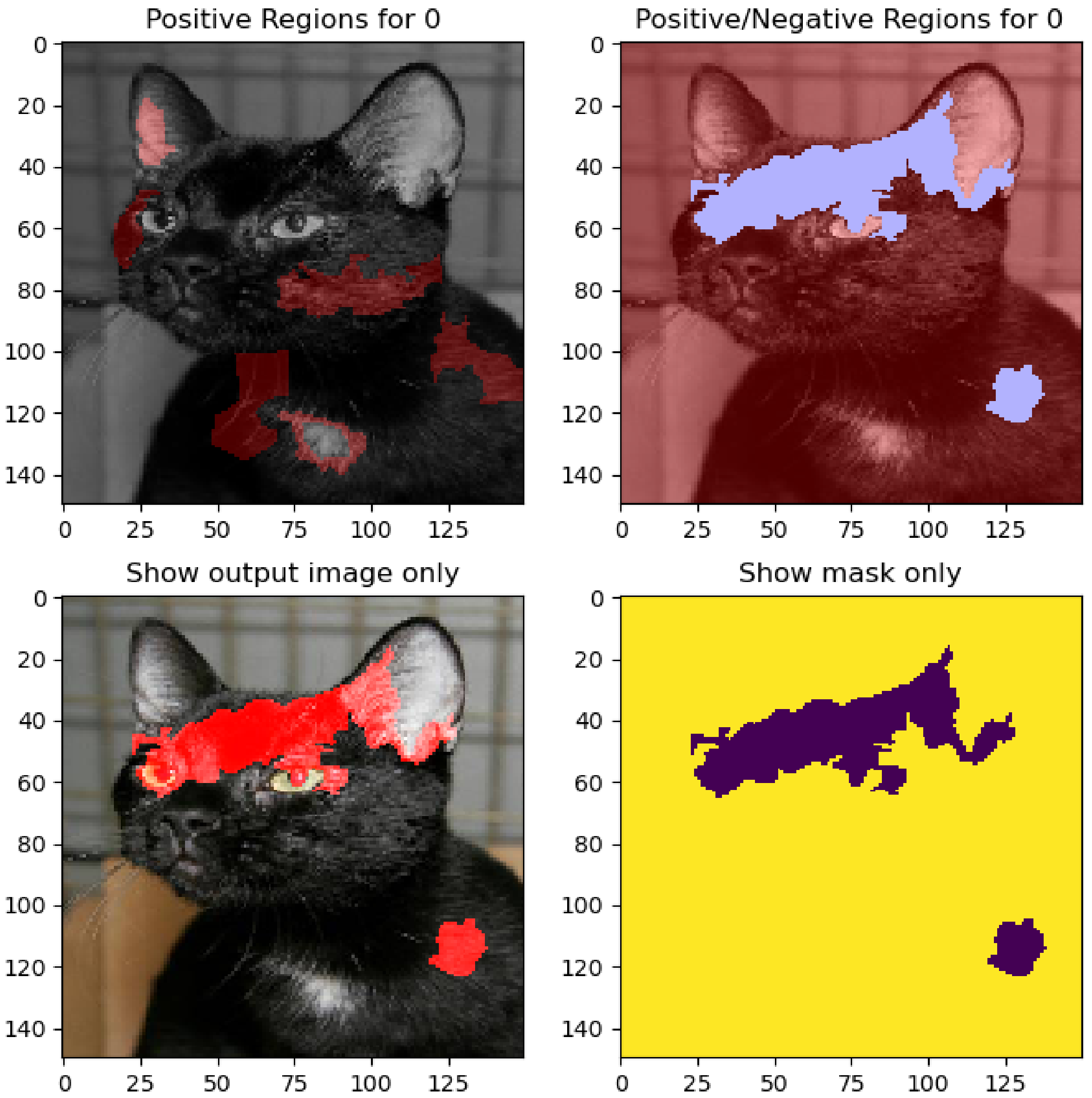
```
7/7 [==============================] - 2s 285ms/step
```

In [16]:
```
visualize_explanation(image_index=0, explainer=explainer, model=model, X_data=X_test
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 219ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 209ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 78ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 68ms/step
```

```
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 261ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 218ms/step
1/1 [==============================] - 0s 209ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 239ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 211ms/step
1/1 [==============================] - 0s 229ms/step
1/1 [==============================] - 0s 220ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 324ms/step
1/1 [==============================] - 0s 218ms/step
1/1 [==============================] - 0s 257ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 238ms/step
1/1 [==============================] - 0s 303ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 437ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 337ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 295ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 223ms/step
1/1 [==============================] - 0s 256ms/step
1/1 [==============================] - 0s 263ms/step
1/1 [==============================] - 0s 192ms/step
```

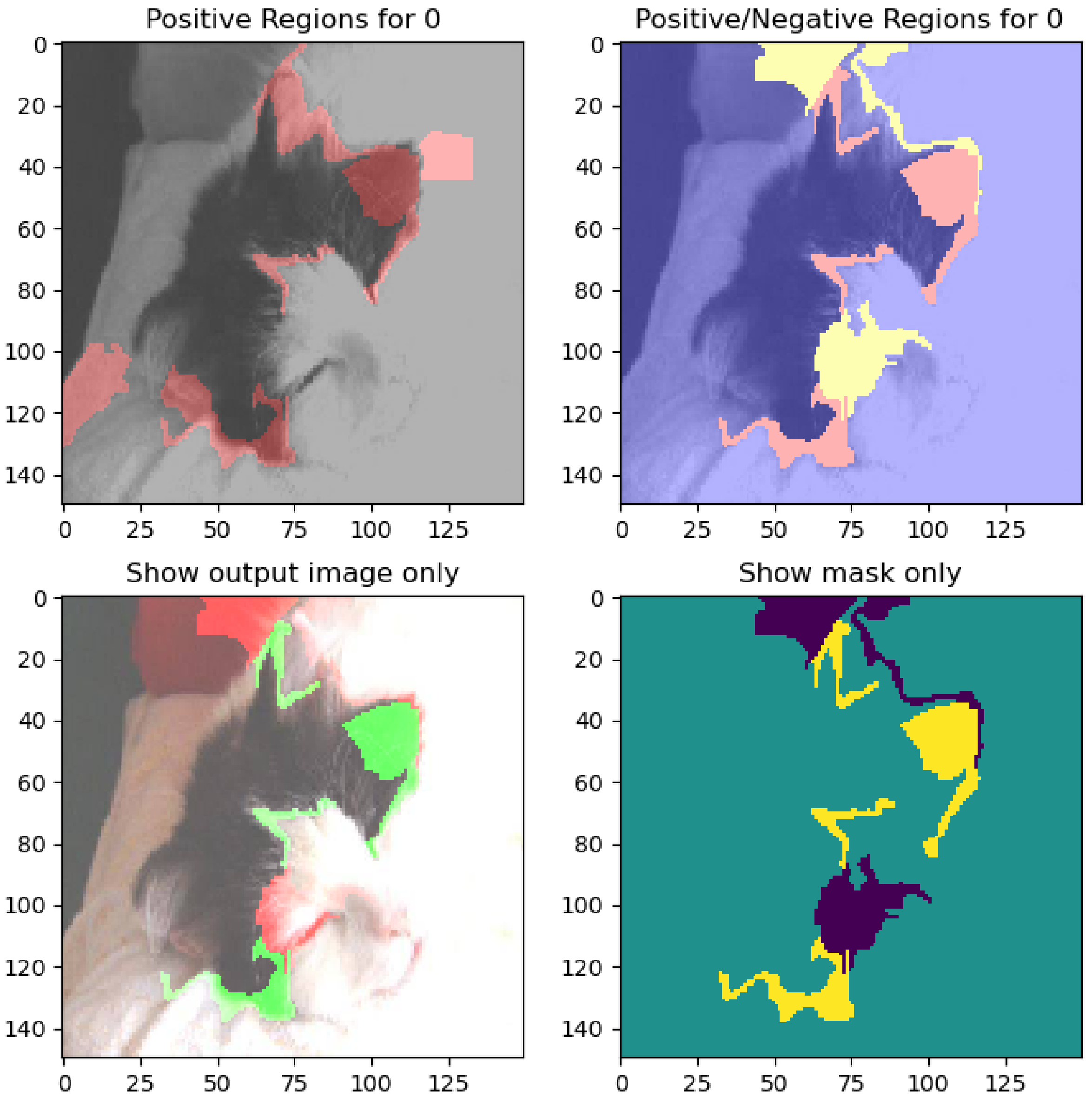


In [17]:
```
visualize_explanation(image_index=3, explainer=explainer, model=model, X_data=X_test
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 90ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 183ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 179ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 81ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 93ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 181ms/step
1/1 [==============================] - 0s 188ms/step
```

```
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 104ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 93ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 132ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 98ms/step
```

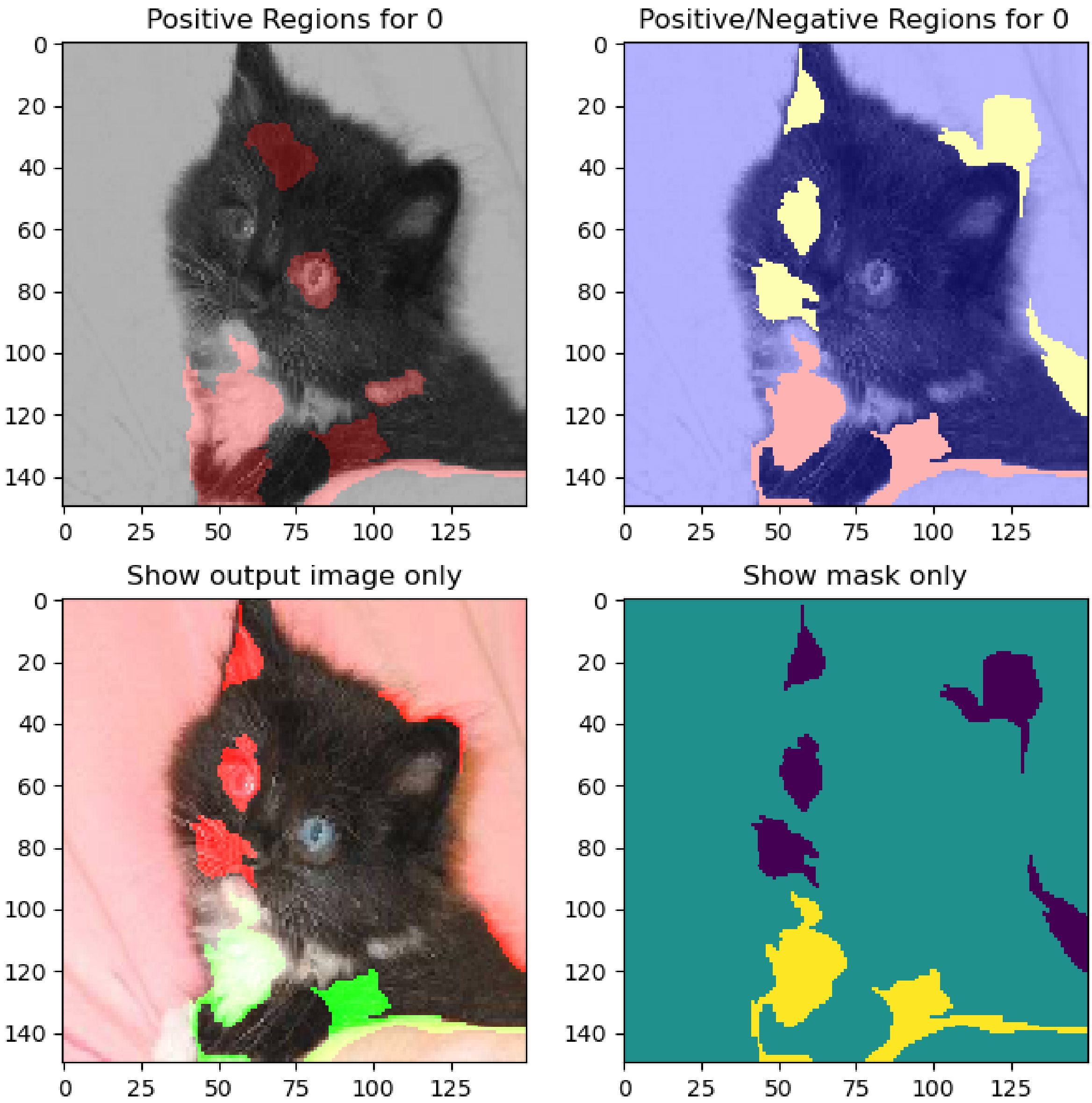


In [18]:
```
visualize_explanation(image_index=6, explainer=explainer, model=model, X_data=X_test
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 86ms/step
1/1 [==============================] - 0s 181ms/step
1/1 [==============================] - 0s 104ms/step
1/1 [==============================] - 0s 93ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 176ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 214ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 87ms/step
```

```
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 92ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 95ms/step
```

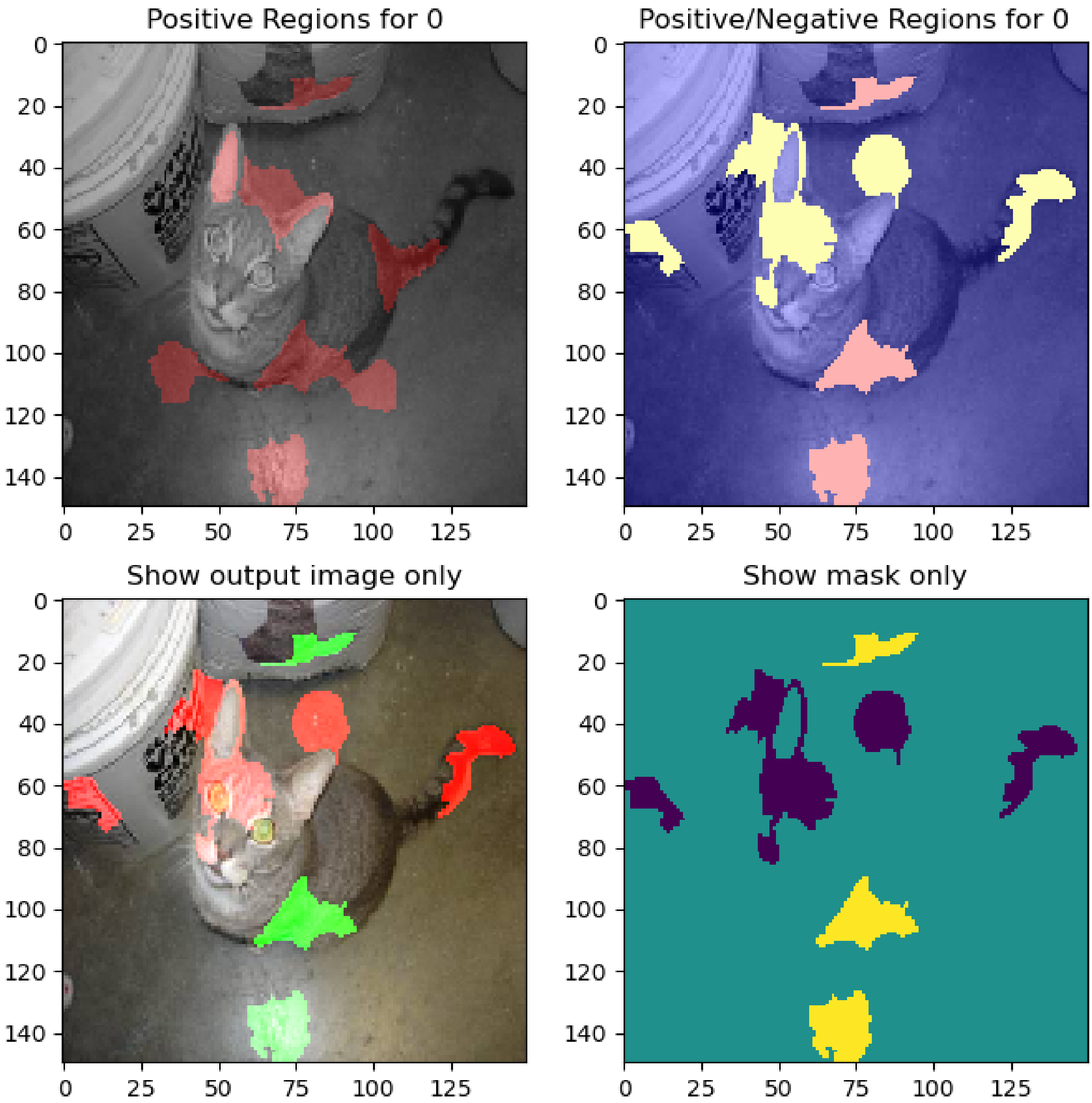



```
In [19]: visualize_explanation(image_index=9, explainer=explainer, model=model, X_data=X_test
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 115ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 88ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 128ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
```

```
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 112ms/step
```



1. 피마 인디언 당뇨 예측 모델에 SHAP 적용

In [20]:
```
!pip install shap
```

```
Requirement already satisfied: shap in c:\users\hanjo\anaconda3\lib\site-packages
(0.44.0)
Requirement already satisfied: numpy in c:\users\hanjo\anaconda3\lib\site-packages
(from shap) (1.24.3)
Requirement already satisfied: scipy in c:\users\hanjo\anaconda3\lib\site-packages
(from shap) (1.11.1)
Requirement already satisfied: scikit-learn in c:\users\hanjo\anaconda3\lib\site-pac
kages (from shap) (1.3.0)
Requirement already satisfied: pandas in c:\users\hanjo\anaconda3\lib\site-packages
(from shap) (2.0.3)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\hanjo\anaconda3\lib\site-pac
kages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in c:\users\hanjo\anaconda3\lib\site-p
ackages (from shap) (23.0)
Requirement already satisfied: slicer==0.0.7 in c:\users\hanjo\anaconda3\lib\site-pa
ckages (from shap) (0.0.7)
Requirement already satisfied: numba in c:\users\hanjo\anaconda3\lib\site-packages
(from shap) (0.57.0)
Requirement already satisfied: cloudpickle in c:\users\hanjo\anaconda3\lib\site-pack
ages (from shap) (2.2.1)
Requirement already satisfied: colorama in c:\users\hanjo\anaconda3\lib\site-package
s (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: llvmlite<0.41,>=0.40.0dev0 in c:\users\hanjo\anaconda
3\lib\site-packages (from numba->shap) (0.40.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hanjo\anaconda3\li
b\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hanjo\anaconda3\lib\site-pac
kages (from pandas->shap) (2022.7)
Requirement already satisfied: tzdata>=2022.1 in c:\users\hanjo\anaconda3\lib\site-p
ackages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\hanjo\anaconda3\lib\site-pa
ckages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hanjo\anaconda3\lib
\site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\hanjo\anaconda3\lib\site-package
s (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

In [21]:
```
import shap
import numpy as np
import pandas as pd
```

In [22]:
```
df = pd.read_csv('diabetes.csv')
```

In [23]:
```
df
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

In [24]:
```
# 데이터를 훈련 데이터셋과 테스트 데이터셋으로 분할
from sklearn.model_selection import train_test_split
X = df.loc[:, df.columns != 'Outcome']
y = df.loc[:, 'Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [38]:
```
# 1. 의사결정트리모델 사용
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

tree = DecisionTreeClassifier(max_depth=4, random_state=42)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('₩nClassification Report:')
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7575757575757576

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81       143
           1       0.72      0.59      0.65        88

    accuracy                           0.76       231
   macro avg       0.75      0.73      0.73       231
weighted avg       0.75      0.76      0.75       231
```

In [26]:
```
# 2. xgboost모델 사용

import xgboost
model = xgboost.XGBRegressor(objective ='reg:linear') # 확률을 숫자로 변환하여 출력ㅎ
model.fit(X_train, y_train)
preds = model.predict(X_test)
```

```
# 전체 피처를 사용해서 학습시킨 모델의 RMSE를 구하는 코드

from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

```
[13:10:52] WARNING: C:₩buildkite-agent₩builds₩buildkite-windows-cpu-autoscaling-grou
p-i-0b3782d1791676daf-1₩xgboost₩xgboost-ci-windows₩src₩objective₩regression_obj.cu:2
09: reg:linear is now deprecated in favor of reg:squarederror.
RMSE: 0.437362
```

In [27]:

```
# 3. 로지스틱회귀모델 사용
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train_scaled, y_train)

y_pred = lr_model.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy:.4f}')
print(f'Classification Report:₩n{class_report}')
```

```
Accuracy: 0.7792
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.92      0.84       143
           1       0.80      0.56      0.66        88

    accuracy                           0.78       231
   macro avg       0.79      0.74      0.75       231
weighted avg       0.78      0.78      0.77       231
```

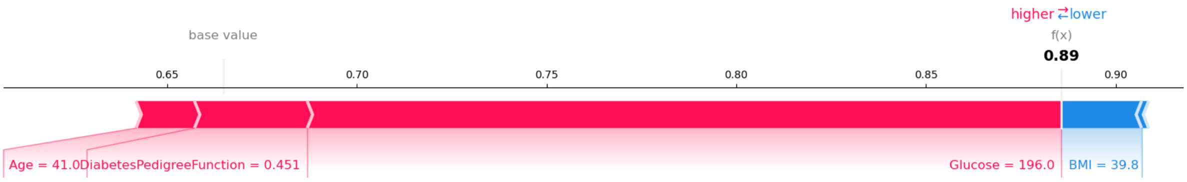
recall이 높은 의사결정모델을 사용하기로 결정함.

In [39]:

```
# SHAP의 설명체를 정의하고 섀플리 값을 계산하는 로직
# SHAP은 자바스크립트를 무조건 사용함

# load JS visualization code to notebook(자바스크립트 쓰겠다고 알려주는 것)
import shap
import matplotlib
shap.initjs()

# explain the model's predictions using SHAP values
# (same syntax works for LightGBM, CatBoost, and scikit-learn models)
# SHAP안에 TreeExplainer 사용
explainer = shap.TreeExplainer(tree)
shap_values = explainer.shap_values(X_train)

# visualize the first prediction's explanation
# (use matplotlib=True to avoid Javascript)
sample_index = int(input("인덱스를 입력하세요."))
shap.force_plot(explainer.expected_value[0], shap_values[0][sample_index, :], X_trai
```

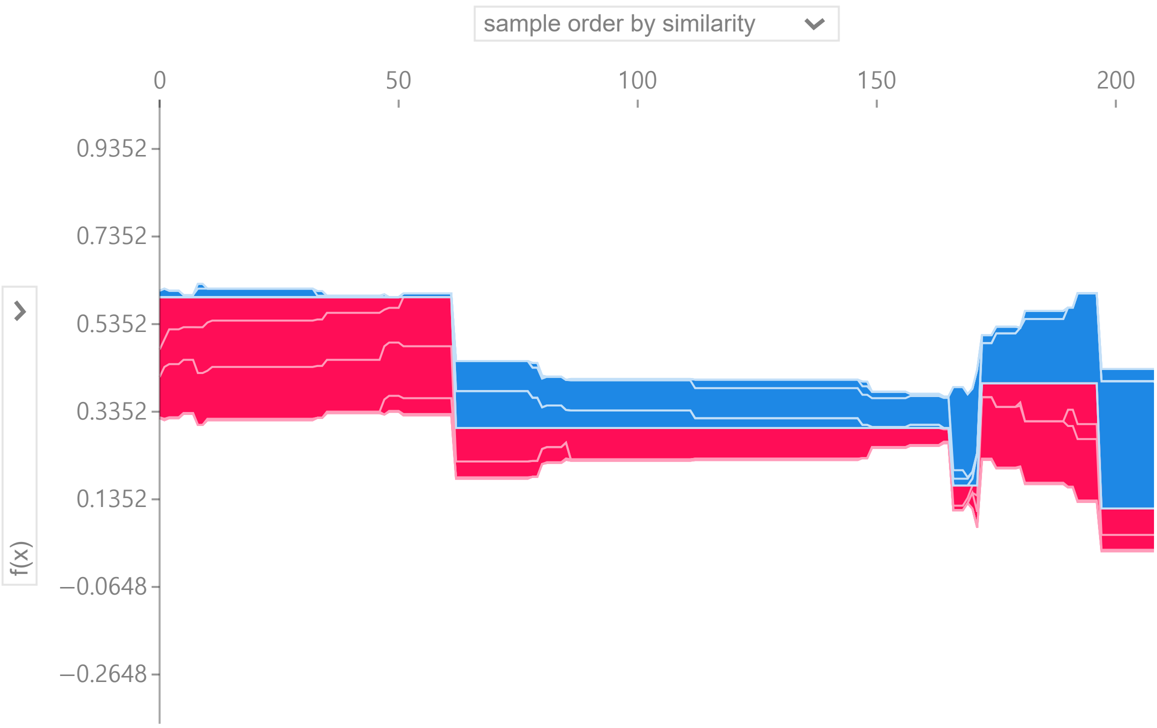
```
# 빨간색: 중요한 변수, 이 값이 높을수록 모델의 예측도 증가
# 파란색: 오히려 방해하는 변수. 중요하지 않은 변수를 보여줌
```

인덱스를 입력하세요.1

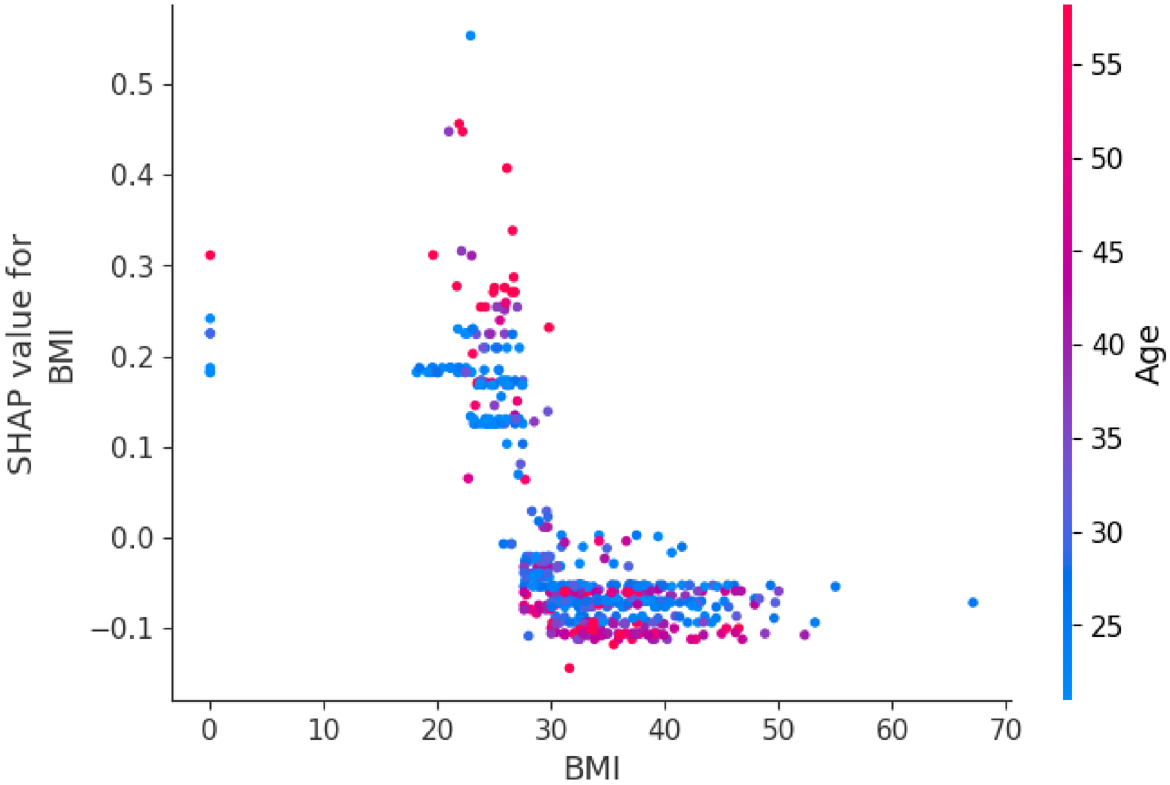


In [40]:
```
# 전체 데이터에 대한 섀플리 값을 플롯으로 그리는 코드

# load JS visualization code to notebook
shap.initjs()

# visualize the training set predictions
shap.force_plot(explainer.expected_value[0], shap_values[0], X_train) # 음성
shap.force_plot(explainer.expected_value[1], shap_values[1], X_train) # 양성
```

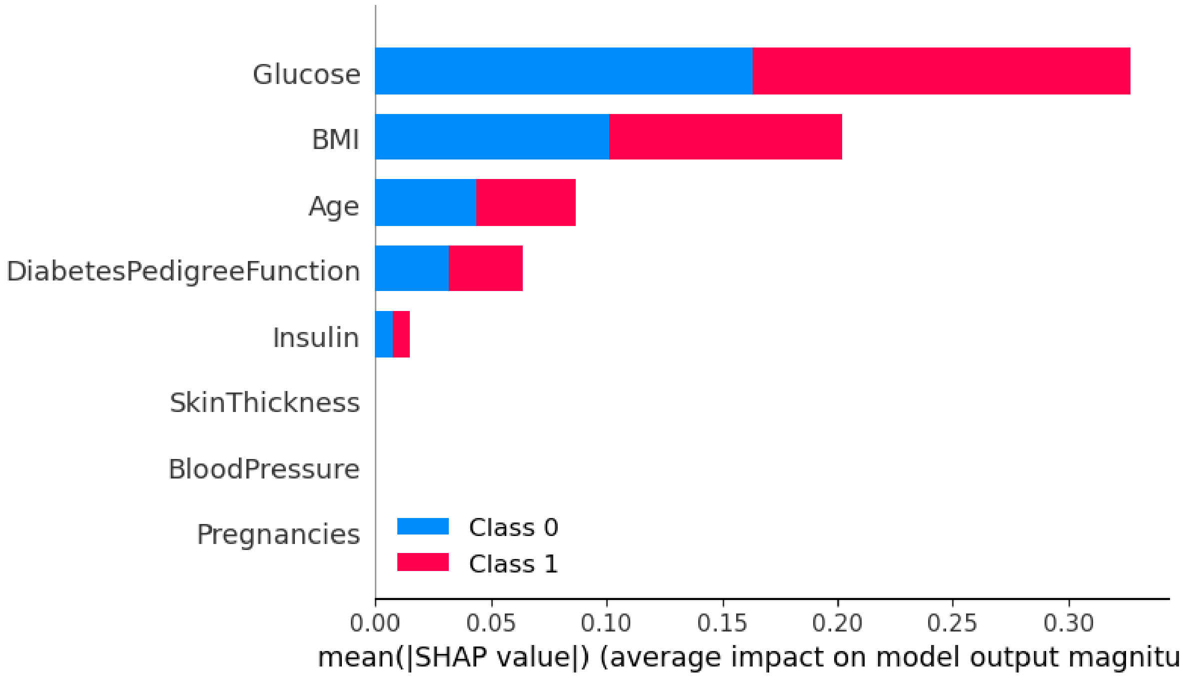
Out[40]:



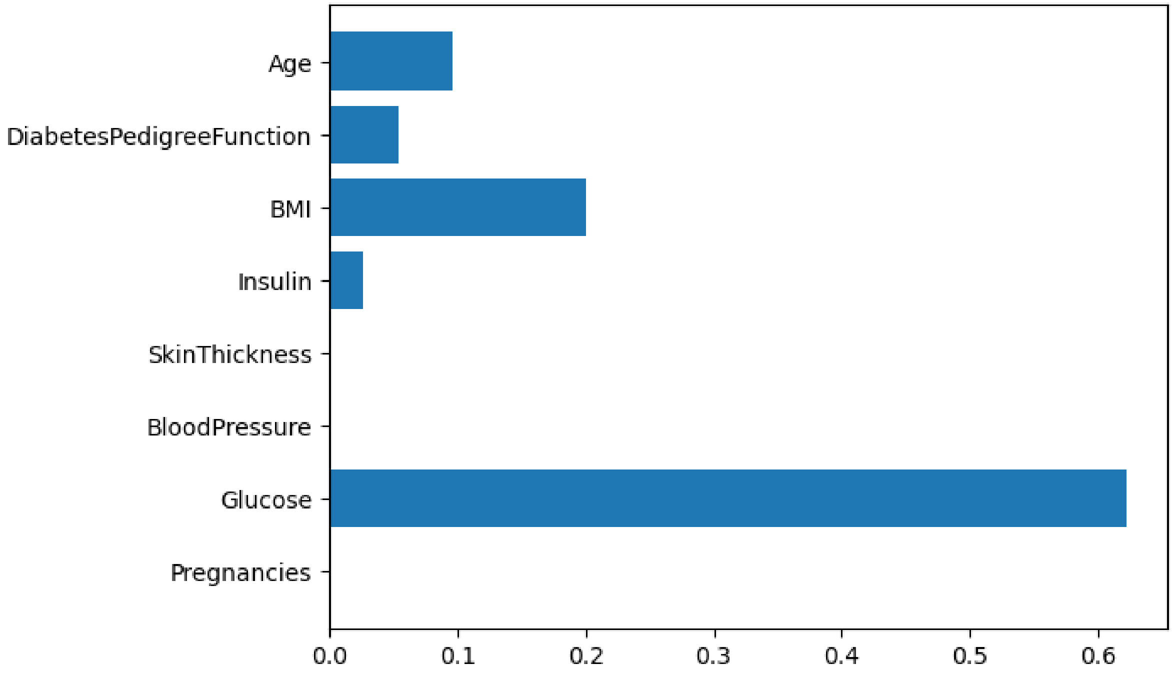
In [41]:
```
shap.dependence_plot("BMI", shap_values[0], X_train)
```



In [42]:
```
# 피처별 섀플리 값을 막대 타입으로 비교하는 코드

shap.summary_plot(shap_values, X_train, plot_type="bar")
```



In [43]:
```
plt.barh(X_train.columns, tree.feature_importances_)
plt.show()
```


Age
DiabetesPedigreeFunction
BMI
Insulin
SkinThickness
BloodPressure
Glucose
Pregnancies
0.0
0.1
0.2
0.3
0.4
0.5
0.6