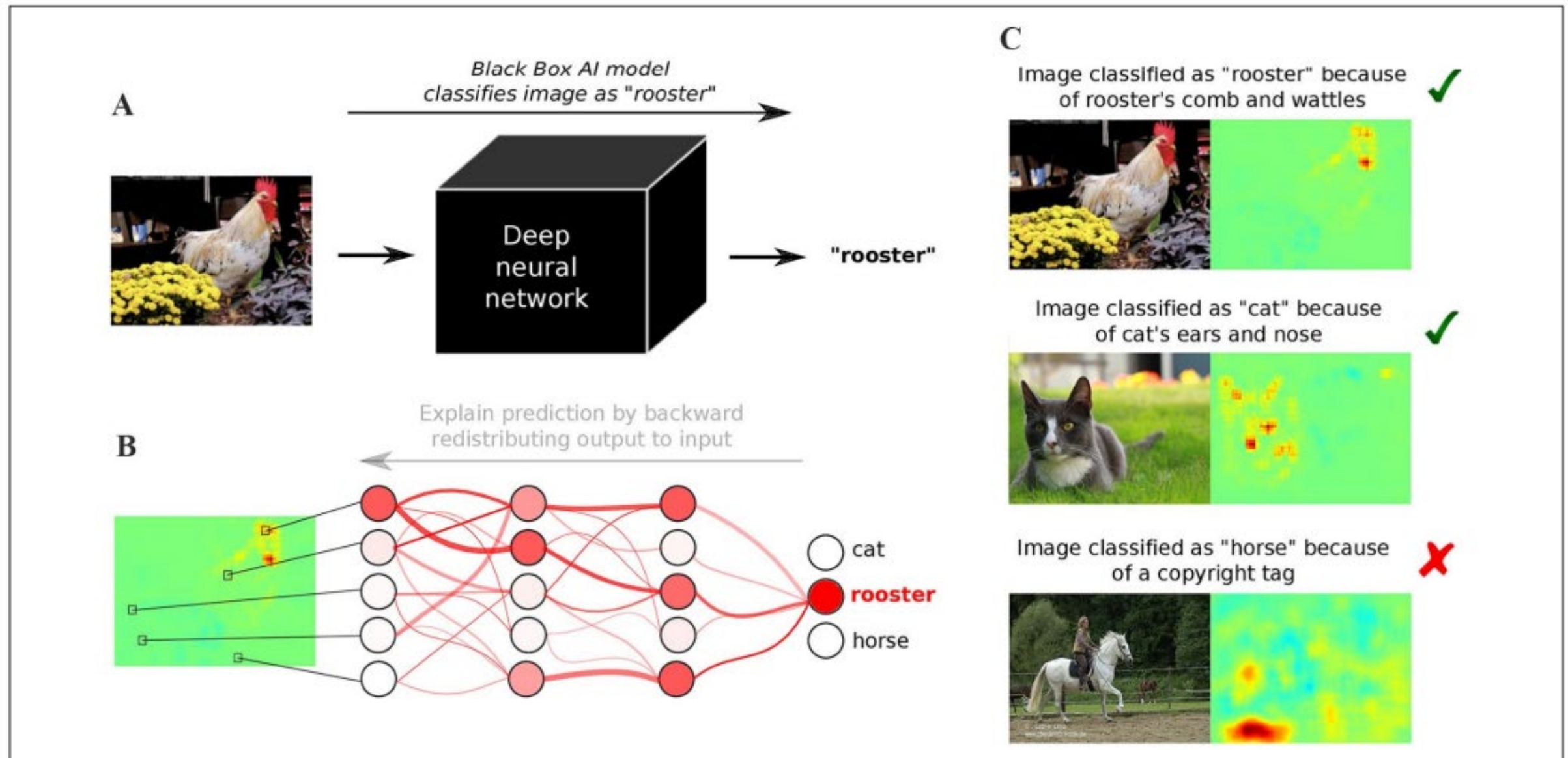


# XAI

---

TA. Bogyong Suh

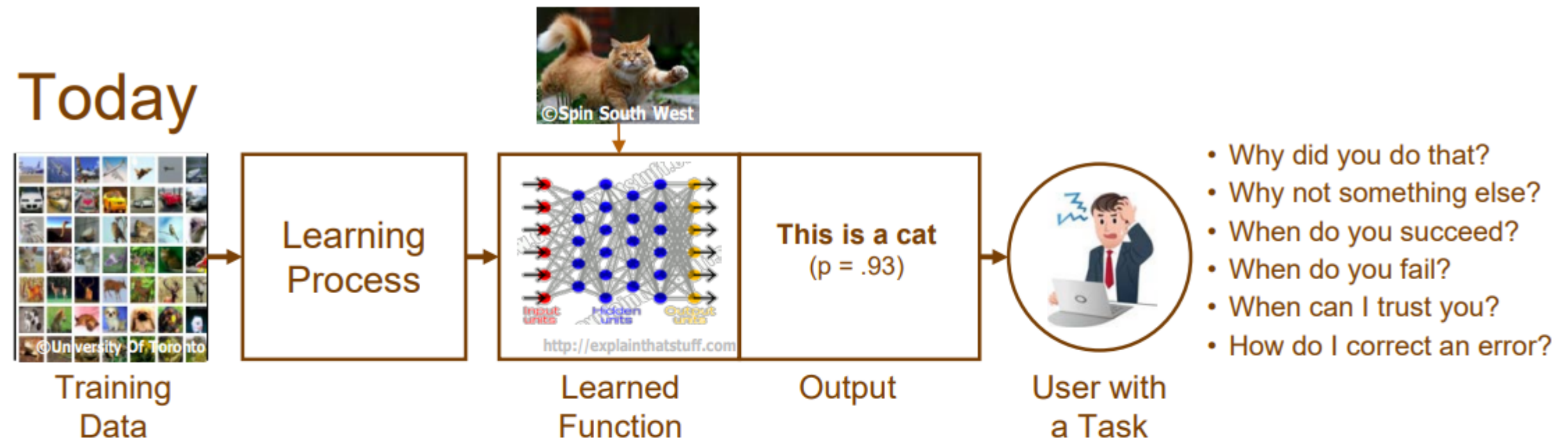
# Black-box model



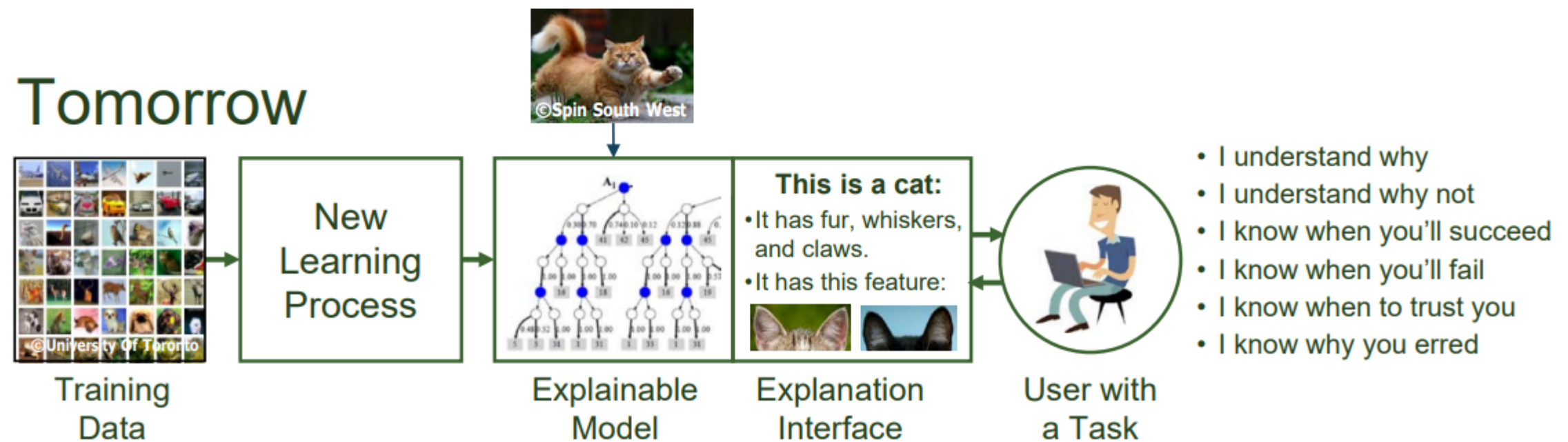
- Internal workings and decision-making processes are not easily understood or transparent in black-box models
- This makes it difficult to understand the reasoning behind the model's decisions and to identify any potential biases or errors in its predictions

# Black-box model

## Today

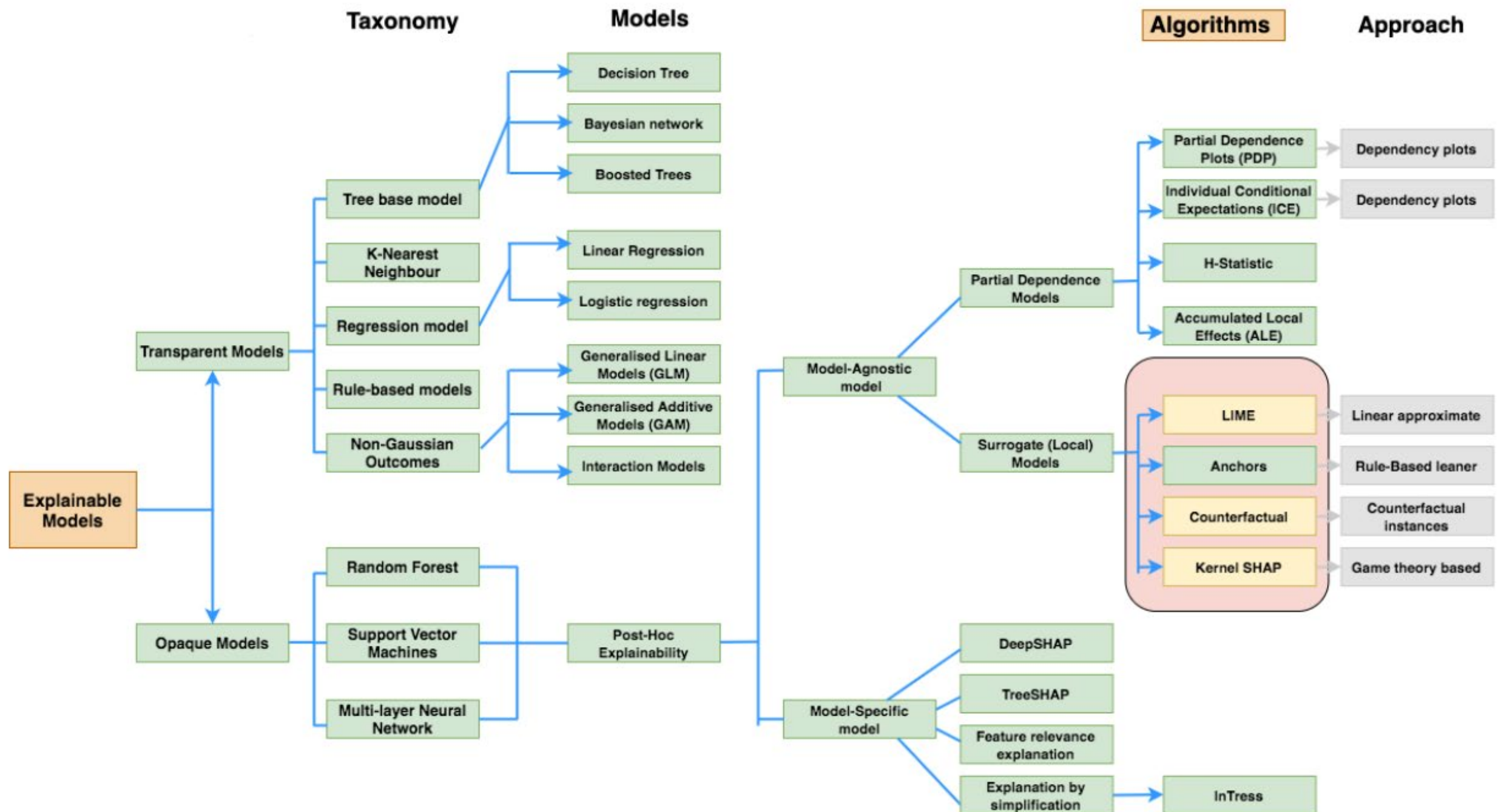


## Tomorrow



- **Explainability** and **interpretability** are important features that reflect the reliability and practicality of AI models

## XAI(eXplainable Artificial Intelligence)



- AI in which humans can understand the decisions or predictions made by the AI
- XAI provides clear and interpretable reasons of the AI models' decision-making processes



## Types of XAI

### Intrinsic vs Post-hoc

- **Intrinsic**: model with an interpretable structure. It may have low accuracy due to less complexity in the structure.
- **Post-hoc**: applied after a model has been trained and deployed.

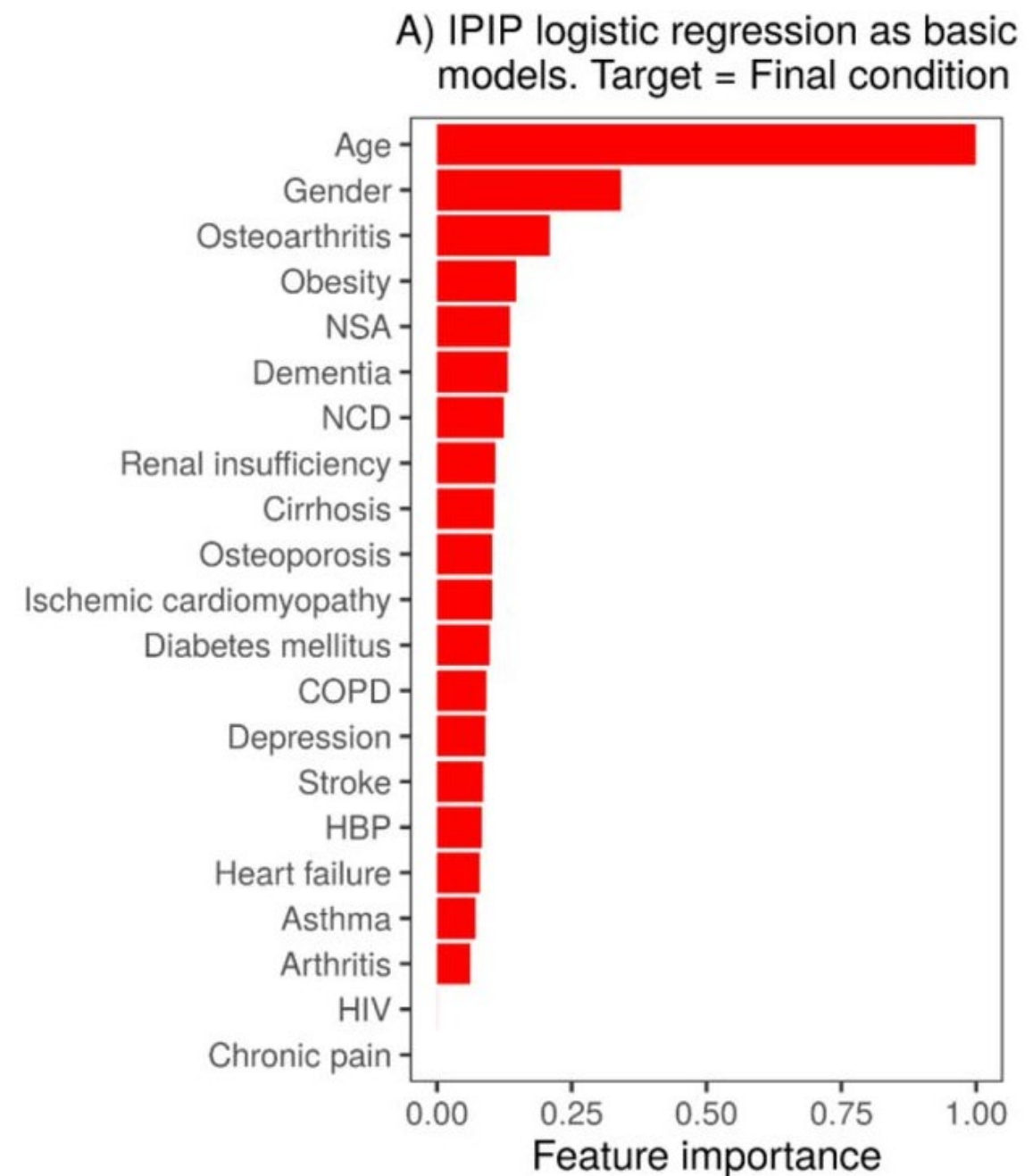
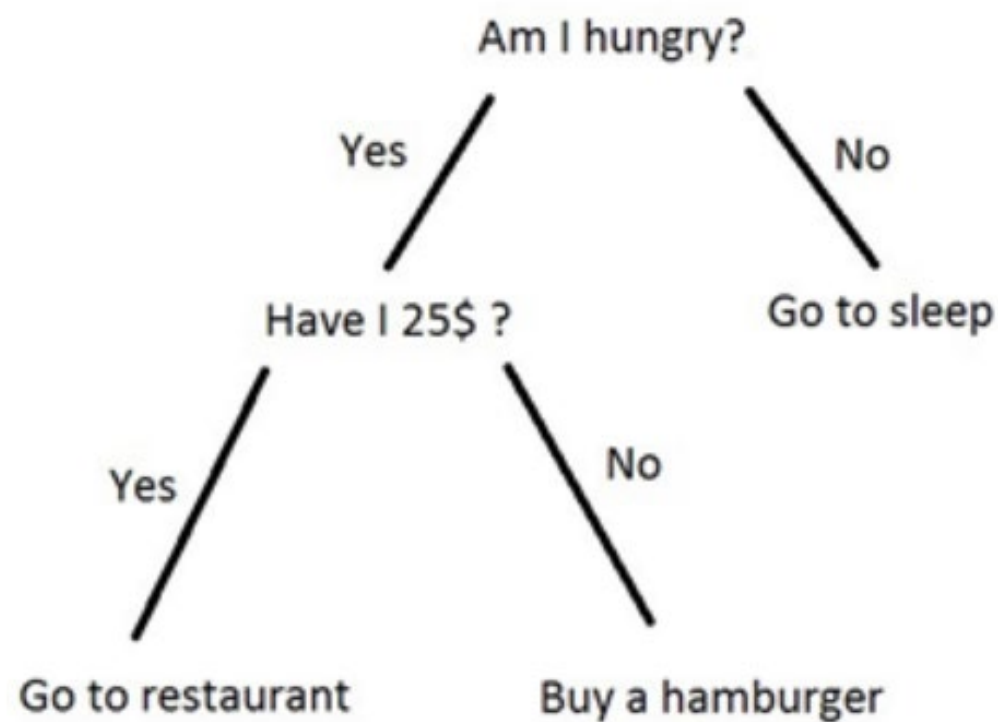
### Global vs Local

- **Global**: explains every predictions that a model makes.
- **Local**: explains certain decision mechanism or one prediction.

### Model-specific vs Model-agnostic

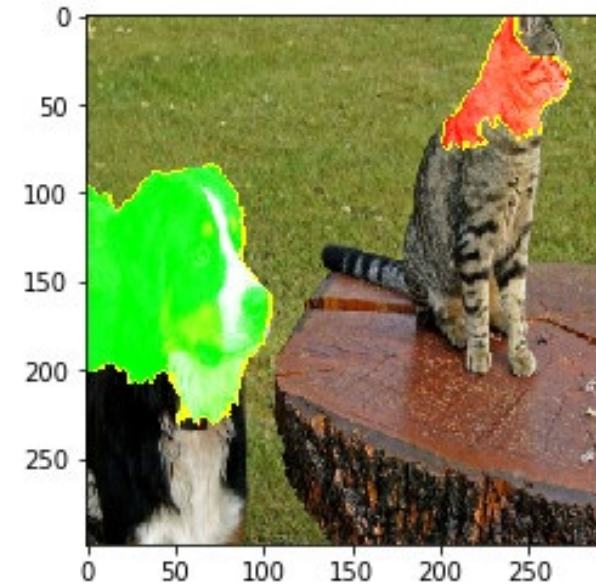
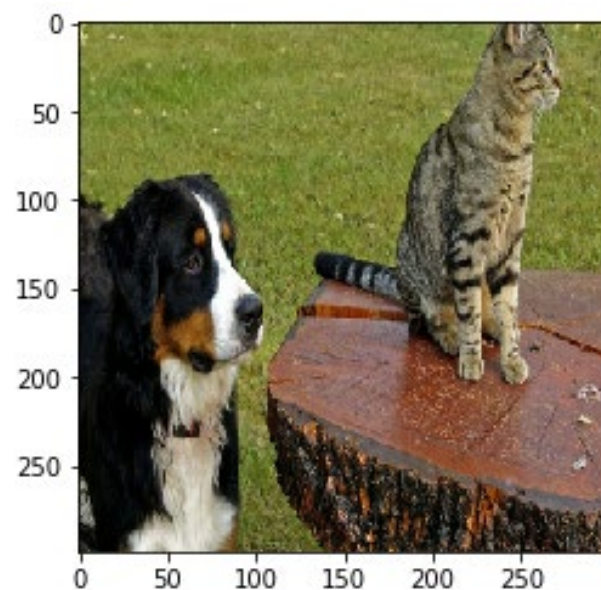
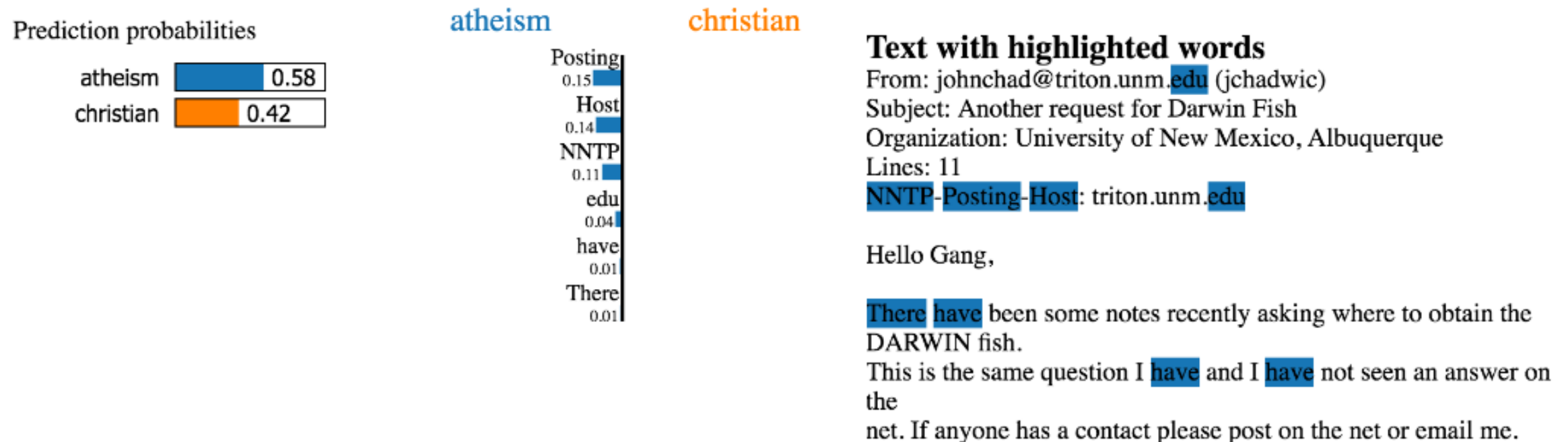
- **Model-specific**: can be applied to certain types of models.
- **Model-agnostic**: can be applied to any types of models.

## Model-intrinsic XAI(Interpretable model)

**The model itself is explainable**

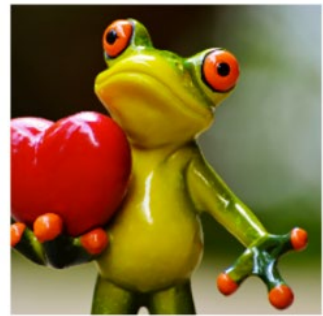
- 1) Decision Tree: Process of deriving the prediction results and its interpretation are intuitive
- 2) Linear model: Coefficients of the model can be used to determine the importance of each features in the prediction (Logistic Regression, Support Vector Machines, LASSO regression, etc)

## LIME(Local Interpretable Model-agnostic Explanation)



- Model-agnostic, and provide local interpretability by explaining individual predictions
- LIME provides explanation by permuting the observation(dataset) and watching the resulting predictions from the model
- Text data, tabular data, images in classification

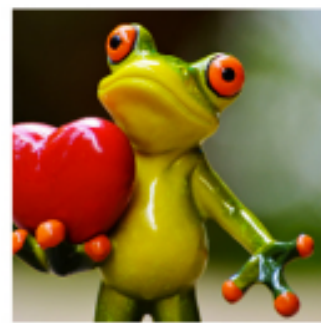
# LIME(Local Interpretable Model-agnostic Explanation) e.g) image classification



Original Image

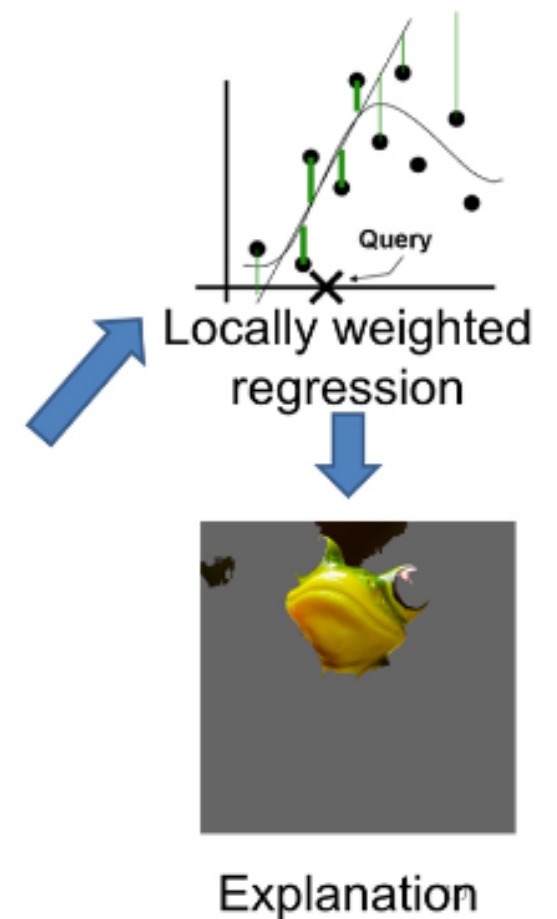


Interpretable Components



Original Image  
 $P(\text{tree frog}) = 0.54$

Perturbed Instances	$P(\text{tree frog})$
	0.85
	0.00001
	0.52

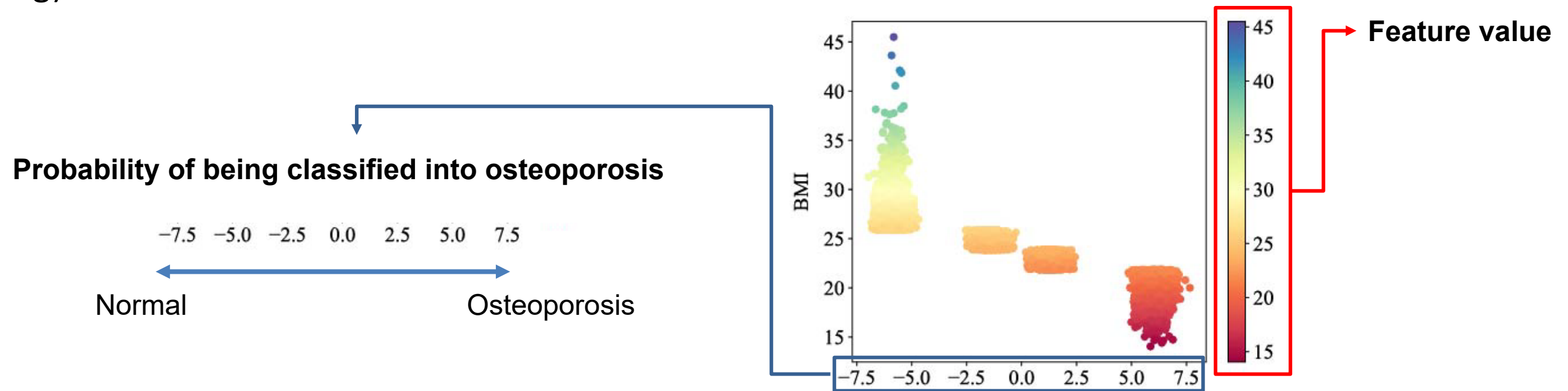


- Perturbation can be made by making superpixels as gray components
- If the prediction results change a lot, then perturbed pixels are the important components when making the prediction



LIME(Local Interpretable Model-agnostic Explanation)

e.g) feature contribution rank



### Individualized risk assessment for osteoporosis

Model prediction for osteoporosis (%)

-20.00 -10.00 0.00 10.00 20.00 30.00 40.00 50.00 60.00 70.00 80.00

-1.36 -4.33 -5.09 21.77 19.05 10.07 8.68 5.40 5.06 3.05

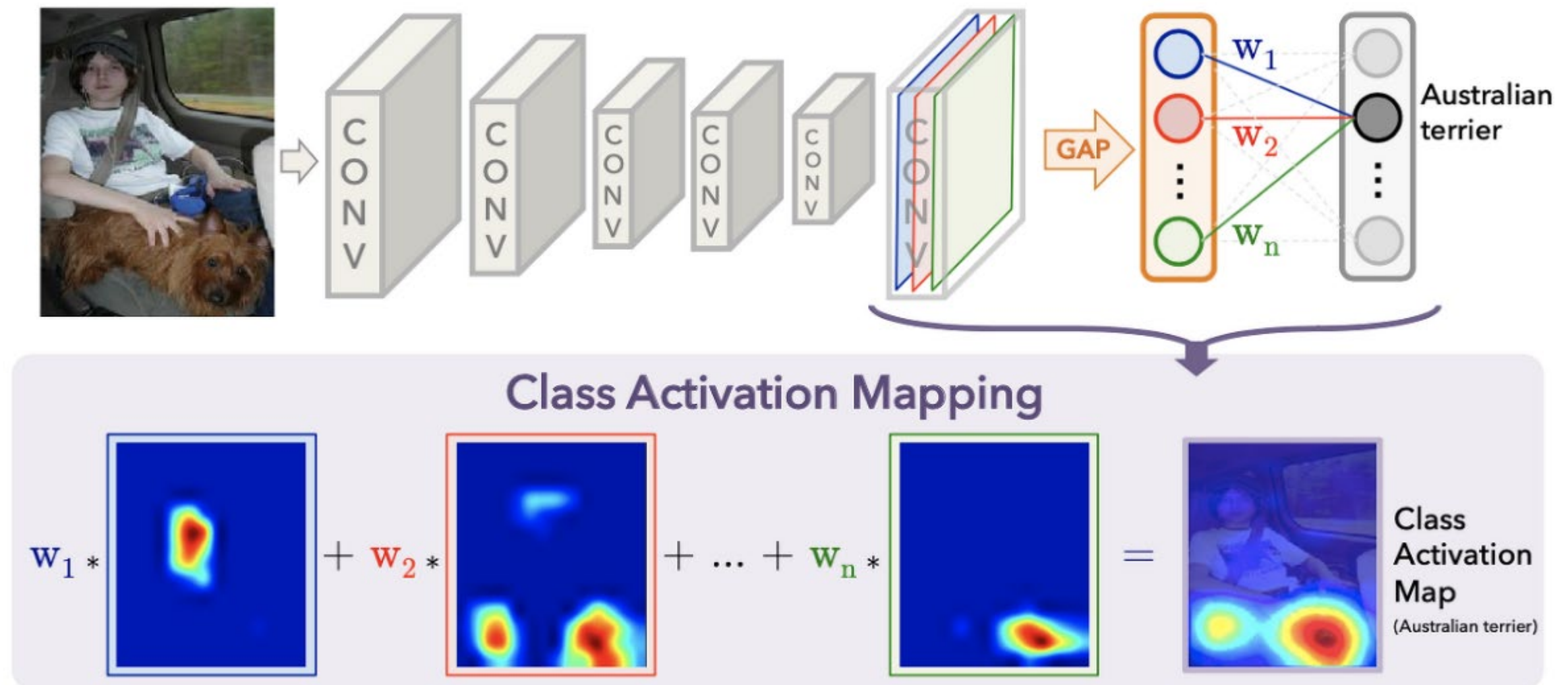
\*Total probability of **osteoporosis** = 62.31%

- Sex=Female
- Prevalence of obesity=Underweight
- Age when diagnosed osteoarthritis=73 (year)
- Education level=2
- High-risk drinking

- Age=78 (year)
- BMI=17.65 (kg/m<sup>2</sup>)
- Diabetes=No
- Age when diagnosed hypertension=Not diagnosed
- Diagnosis of depression=Not diagnosed

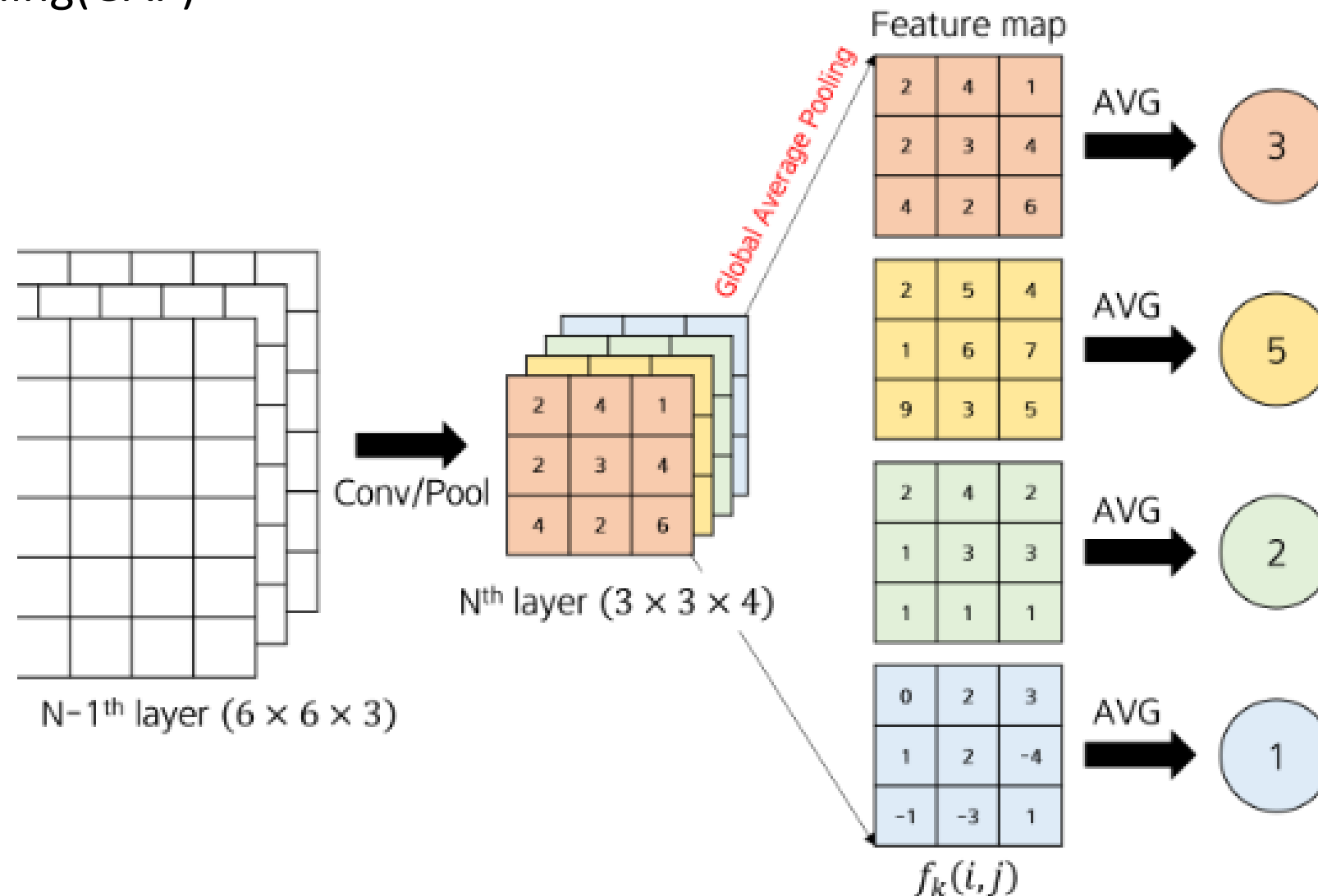
- Feature contribution rank can be made with the analysis of feature importance from LIME

## CAM(Class Activation Map)



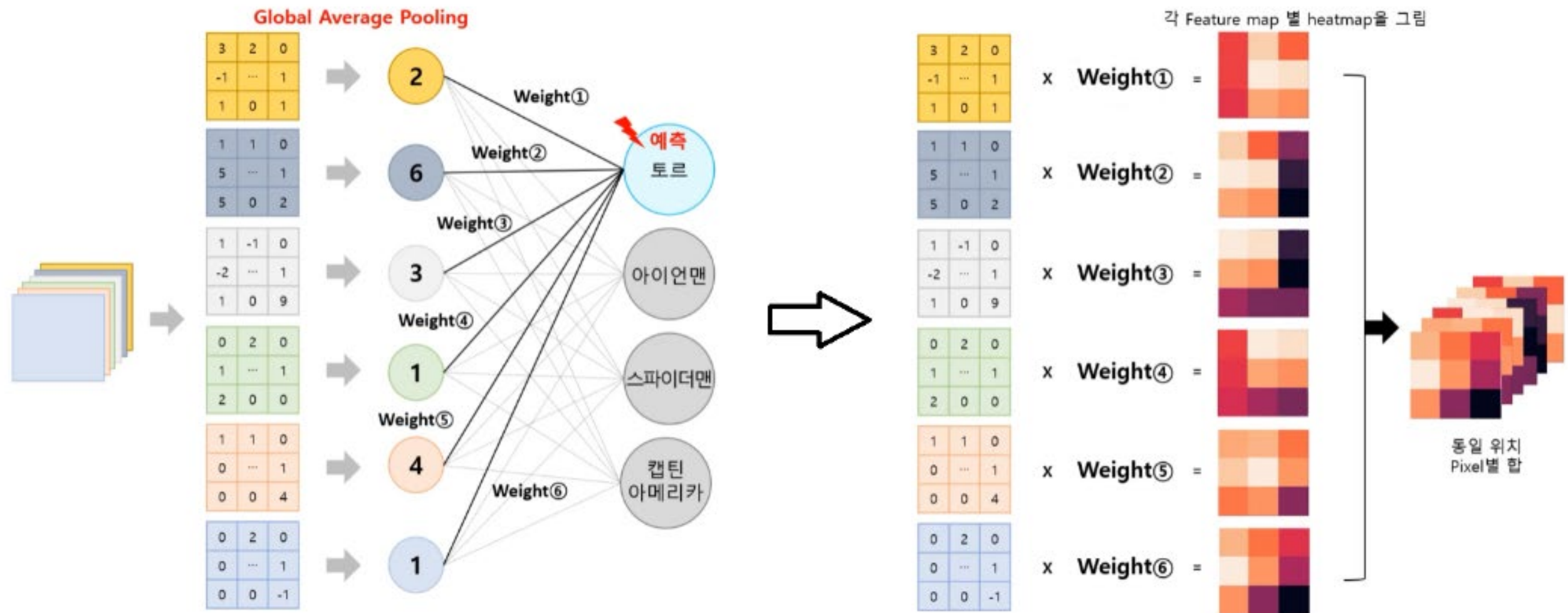
- CAM is a type of visualization in deep learning to understand the regions of an image that are important for a particular classification decision made by a Convolutional Neural Network(CNN)

## CAM(Class Activation Map) -Global Average Pooling(GAP)



- Instead of flattening the data with FC(Fully Connected) layer in the last layer of CNN, using GAP(Global Average Pooling) can generate a heatmap that highlights the regions of the input image that are most relevant to the prediction
- With GAP, feature maps of each channel are converted into a vector with the length equivalent to the number of channels

## CAM(Class Activation Map)

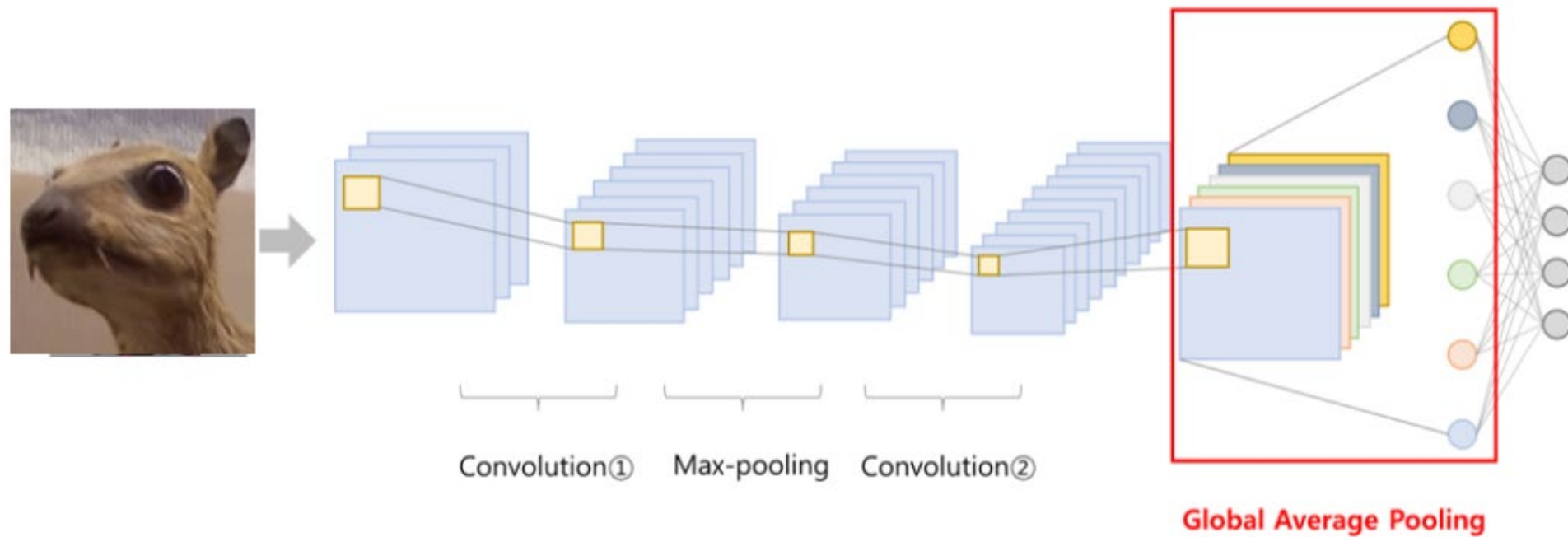


- The output of GAP layer is given as input data to the FC layer which is attached right after the GAP layer
- Heatmap can be made by multiplying each feature map with the weight and then adding to each pixels
- Training is needed to find the weight of FC layer

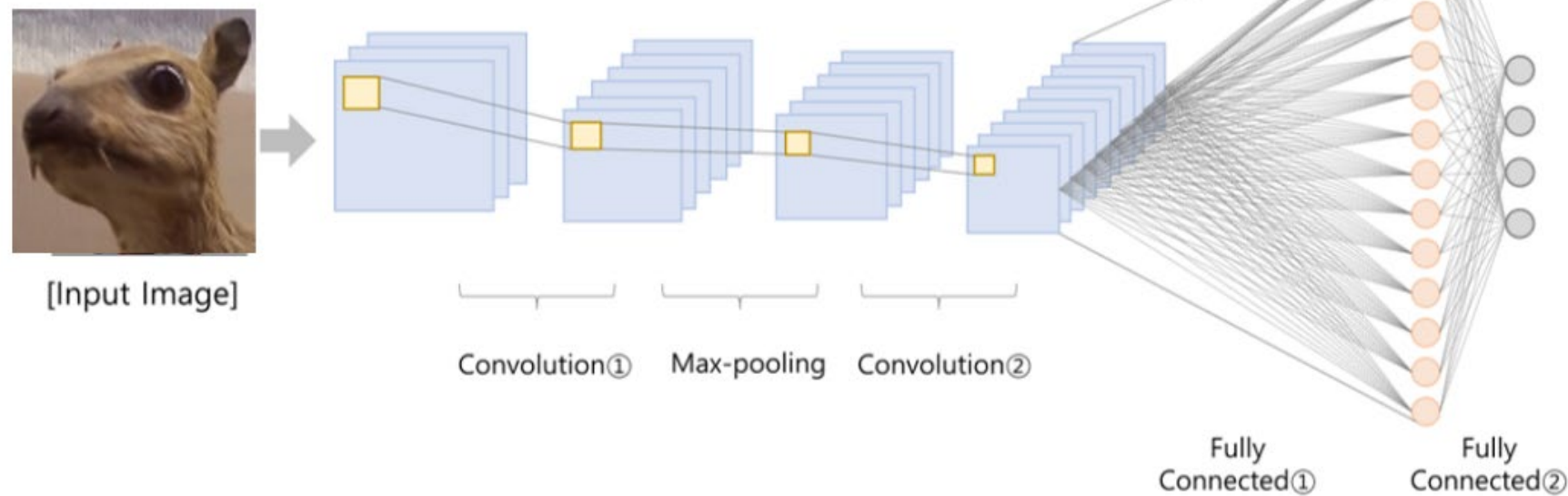


## Grad-CAM

CAM



Grad-CAM



- **CAM** is only applicable with CNN performing GAP over convolutional maps immediately prior to prediction (heatmap can be printed out only from the feature maps of the last convolution layer)
- **Grad-CAM** has similar structure to the original CNN (two FC layers without GAP), there is no need to retrain the CNN model (weight can be derived with gradient information)

## CAM on Cat vs Dogs classification CNN model

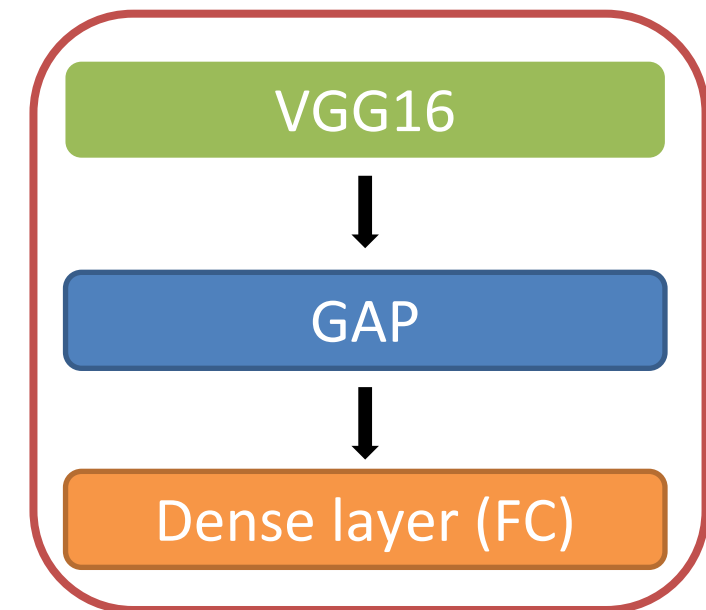
```
#We use VGG16 for pretrained base model
base_model = tf.keras.applications.VGG16(input_shape= (224, 224, 3),
                                         weights='imagenet',
                                         include_top=False)

# add a GAP layer
output = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)

# output has two neurons for the 2 classes(dogs and cats)
output = tf.keras.layers.Dense(2, activation='softmax')(output)

# set the inputs and outputs of the model
model = tf.keras.Model(base_model.input, output)
```

Our model



- We will use pre-trained VGG16 for the classification of Cat vs Dogs with image dataset from tensorflow\_datasets
- Heatmap can be made by using GAP and multiplying the weights of FC layer and feature maps from the convolutional layer

## CAM on Cat vs Dogs classification CNN model



```

cam_model = tf.keras.Model(model.input, outputs=(model.layers[-3].output, model.layers[-1].output))
cam_model.summary()
gap_weights = model.layers[-1].get_weights()[0]
  
```

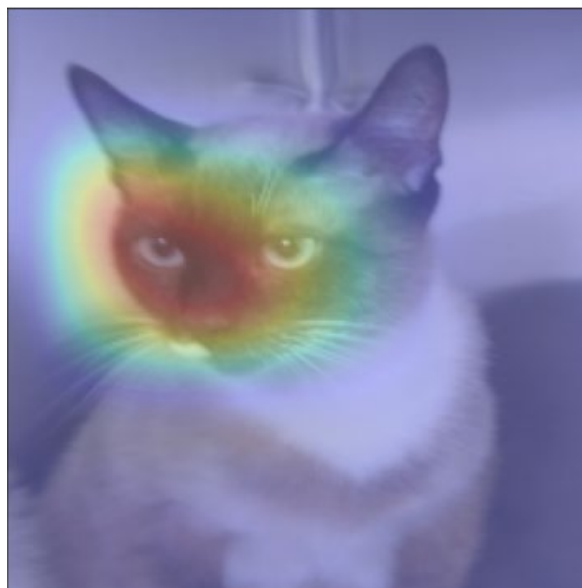
Weight of Dense layer (FC layer)

```

features, results = cam_model.predict(img)
features_for_img = features[0]
prediction = results[0]

class_activation_weights = gap_weights[:,label]
class_activation_features = sp.ndimage.zoom(features_for_img, (224/7, 224/7, 1), order=2)
cam_output = np.dot(class_activation_features, class_activation_weights)
cam_output = tf.reshape(cam_output, (224,224))
  
```

Element-wise product of  
feature map and weight



Ground Truth : Cat  
Prediction : Cat



Ground Truth : Dog  
Prediction : Dog