Explainability AI (XAI) – Understanding the black-box of complex machine learning models.

A DATS 6303 Deep Learning Project Individual Report

Aditya Kumar

Abstract

As we solve more complex problems, the complexity of our models also increases. This means that most machine learning models are doing something under the hood that is so complex that we don't have a clue anymore why they behave the way they do. In other words, we don't know the thought process of our model in predicting something. Understanding the behavior of our machine learning model is becoming very important. Judging the model's performance based on its accuracy or other metrics alone is not sufficient anymore as your model can trick you.

This is where Explainable AI or XAI comes to the rescue. XAI aims to make the model's behavior understandable. The explanations produced by this concept should help you to understand why the model behaves the way it does. If the model isn't behaving as expected, there's a good chance you did something wrong in the data preparation phase, or choice of the modeling technique.

Outline of Project Components and Contribution

In this work, I have developed several XAI demonstrations across various kinds of machine learning models for Image Classification. The purpose of these demonstrations is for the user to understand these concepts in detail before utilizing them. Ultimately, I have produced a complete no-code product offering to evaluate the robustness of Image Classification solutions leveraging XAI utilities.

A comprehensive list of all work done by me is –

- Explored LIME XAI Utility in Python in-depth.
- Developed an entire no-code product offering that will work with TensorFlow and PyTorch frameworks. Users can upload any trained model, Custom or Pre-Trained, along with an observation of their choice to analyze the model's inner working.

<u>Description of Contribution</u>

Lime

The acronym LIME stands for Local Interpretable Model-agnostic Explanations. LIME supports explanations for tabular models, text classifiers, and image classifiers. However, for the purpose of this work, the focus is specifically on Image Classifiers.
LIME can provide model agnostic local explanations for solving both regression and classification problems and it can be applied with both structured datasets and even with unstructured datasets like text and images. Lime is a Python Library based on a paper titled “Why Should I Trust You?”, which is referenced towards the end.

The abbreviation of LIME itself gives an intuition about the core idea behind it. LIME is:

- Model agnostic, which means that LIME is model-independent. In other words, LIME is able to explain any black-box classifier.
- Interpretable, which means that LIME provides a solution to understand why your model behaves the way it does, by displaying higher level features of an image such as the super-pixels.
- Local, which means that LIME tries to find the explanation of black-box models by approximating the local linear behavior of the model. Intuitively, an explanation is a local linear approximation of the model's behavior. While the model may be very complex globally, it is easier to approximate it around the vicinity of a particular instance.

How LIME Works:
Internally, LIME tries to interpret a black box model by conducting these four steps:

1. Input data permutation
The first step that LIME does is to create several artificial data points that are close with the instance (image) in the feature space, that is used to explain the model.
LIME will generate several samples that are similar to the input image by turning on and off some of the super-pixels of the image.

2. Predict the class of each artificial data point
Next, LIME will predict the class of each of the artificial data points that has been generated using the trained model. If your input data is an image, then the prediction of each perturbed image will be generated at this stage.

3. Calculate the weight of each artificial data point
The third step is to calculate the weight of each artificial data to measure its importance. To do this, first the cosine distance metric is usually applied to calculate how far the

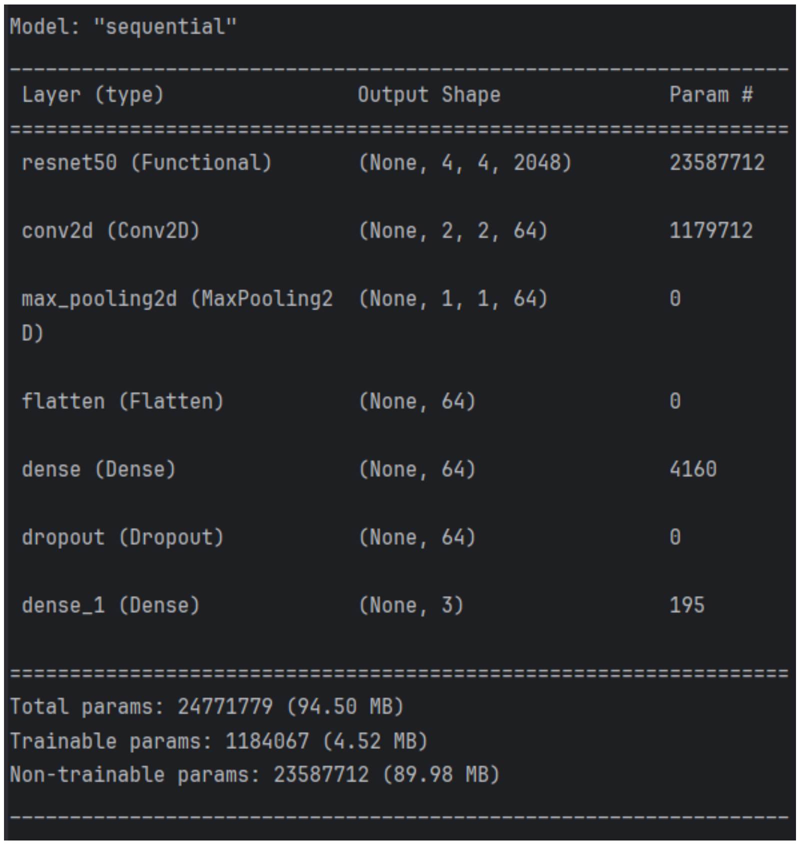
distance of each artificial data point with respect to the original input data. Next, the distance will be mapped into a value between zero to one with a kernel function. The closer the distance, the closer the mapped value to one, and hence, the bigger the weight. The bigger the weight, the bigger the importance of a certain artificial data point.
If the input data is an image, then the cosine distance between each perturbed image and the original image will be computed. The more similarity between a perturbed image to the original image, the bigger its weight and importance.

4. Fit a linear classifier to explain the most important features
The last step is fitting a linear regression model using the weighted artificial data points. After this step, the fitted coefficient of each feature, just like the usual linear regression analysis, is retrieved. Now upon sorting the coefficients, the features that have larger coefficients are the ones that play a big role in determining the prediction of the black-box machine learning model.

Methodology

To analyze the LIME framework, the No-Code Product Offering is utilized as that is integrated with LIME. For the purpose of this analysis, a Kaggle dataset of images containing Dogs, Cats, Pandas is utilized for an image classification task.

1. A custom model architecture is built using ResNet18 as the base followed by Convolutional Layers for image classification. TensorFlow is used as the framework of choice. This model is used to evaluate the working of the no-code product offering for model Explainability. The results of this are shown in the next section. Model architecture is shown below:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 conv2d (Conv2D)             (None, 2, 2, 64)          1179712

 max_pooling2d (MaxPooling2  (None, 1, 1, 64)          0
 D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 24771779 (94.50 MB)
Trainable params: 1184067 (4.52 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
```

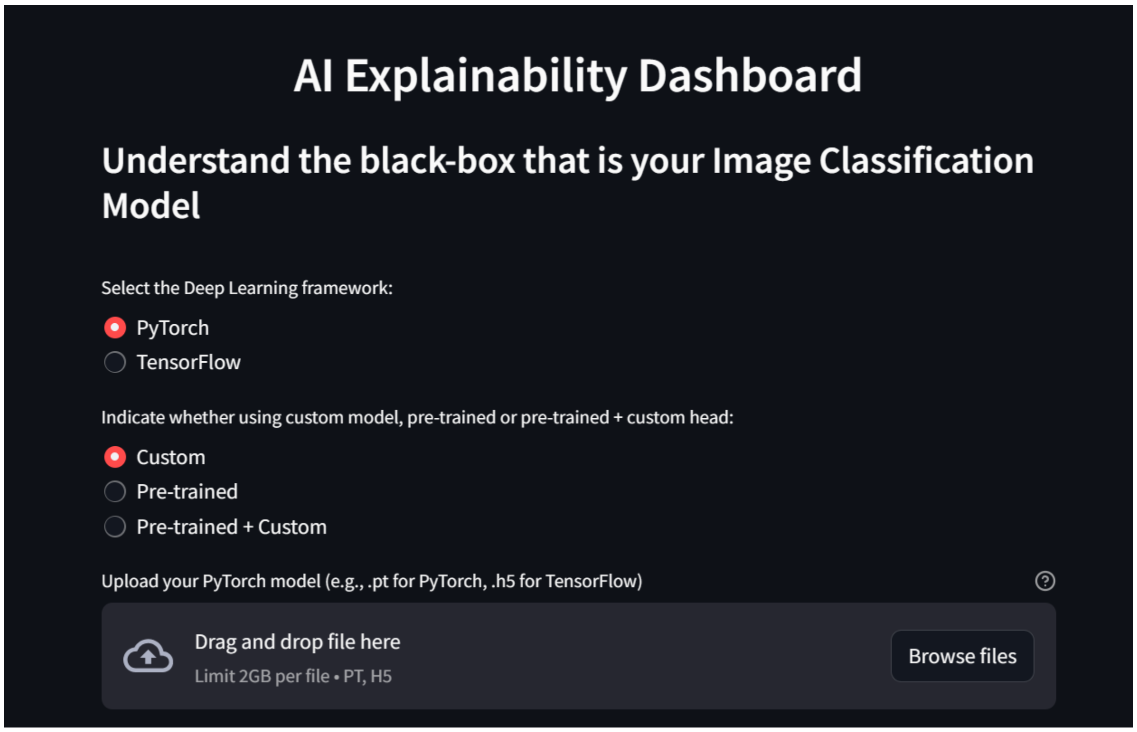
2. A pre-trained model, InceptionNet_V3, from the PyTorch library is utilized as the subsequent model to evaluate and explain its working using the no-code model Explainability offering.

Experimental Setup

Streamlit has been selected as the Web Application Python framework to create an interactive dashboard for both the XAI Demos, as well as the no-code Model Explainability Offering. Streamlit is a free and open-source framework to rapidly build and share machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers.

Results

XAI Product Offering:
The page for the XAI Product Offering looks like this –



AI Explainability Dashboard

Understand the black-box that is your Image Classification Model

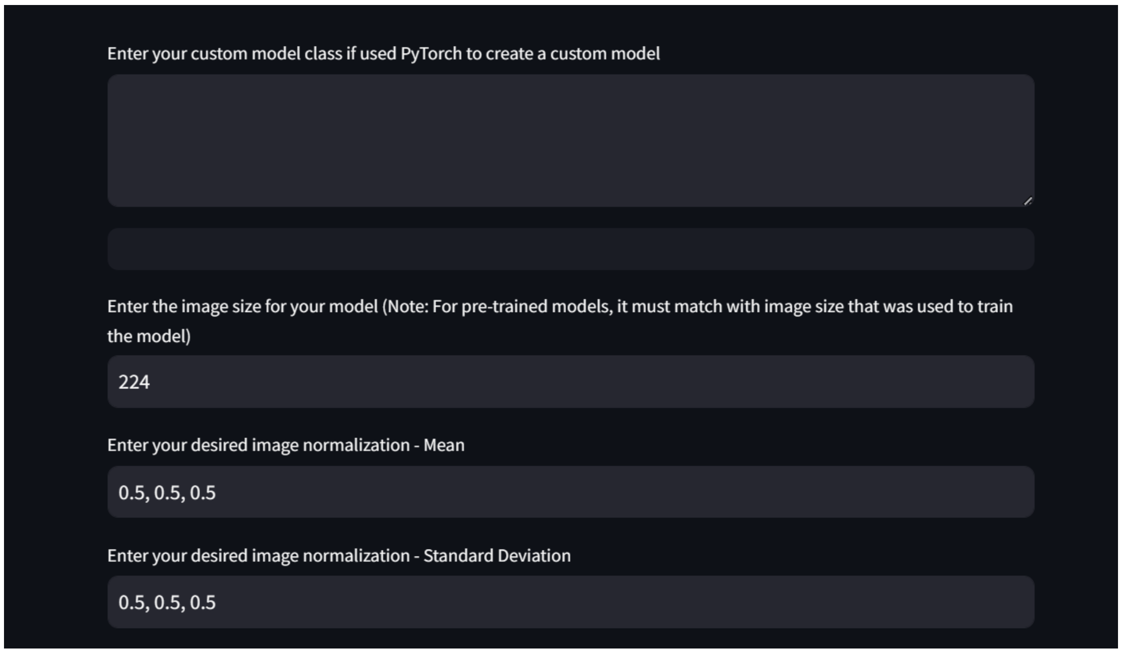
Select the Deep Learning framework:

- [✓ PyTorch]
- TensorFlow

Indicate whether using custom model, pre-trained or pre-trained + custom head:

- [✓ Custom]
- Pre-trained
- Pre-trained + Custom

Upload your PyTorch model (e.g., .pt for PyTorch, .h5 for TensorFlow)



Enter your custom model class if used PyTorch to create a custom model

Enter the image size for your model (Note: For pre-trained models, it must match with image size that was used to train the model)

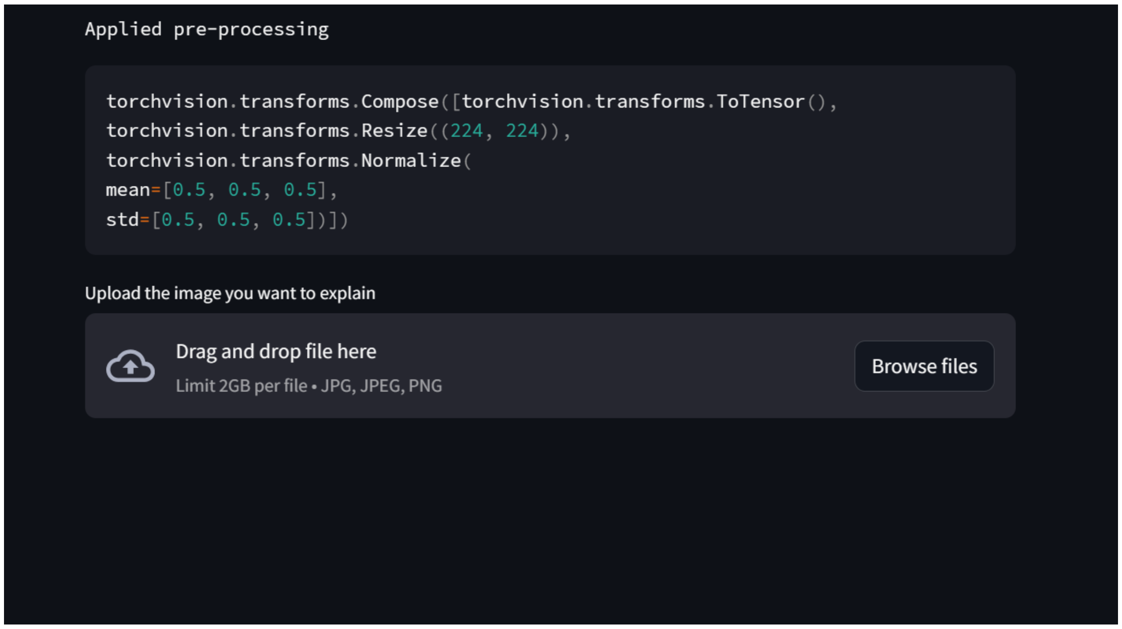
224

Enter your desired image normalization - Mean

0.5, 0.5, 0.5

Enter your desired image normalization - Standard Deviation

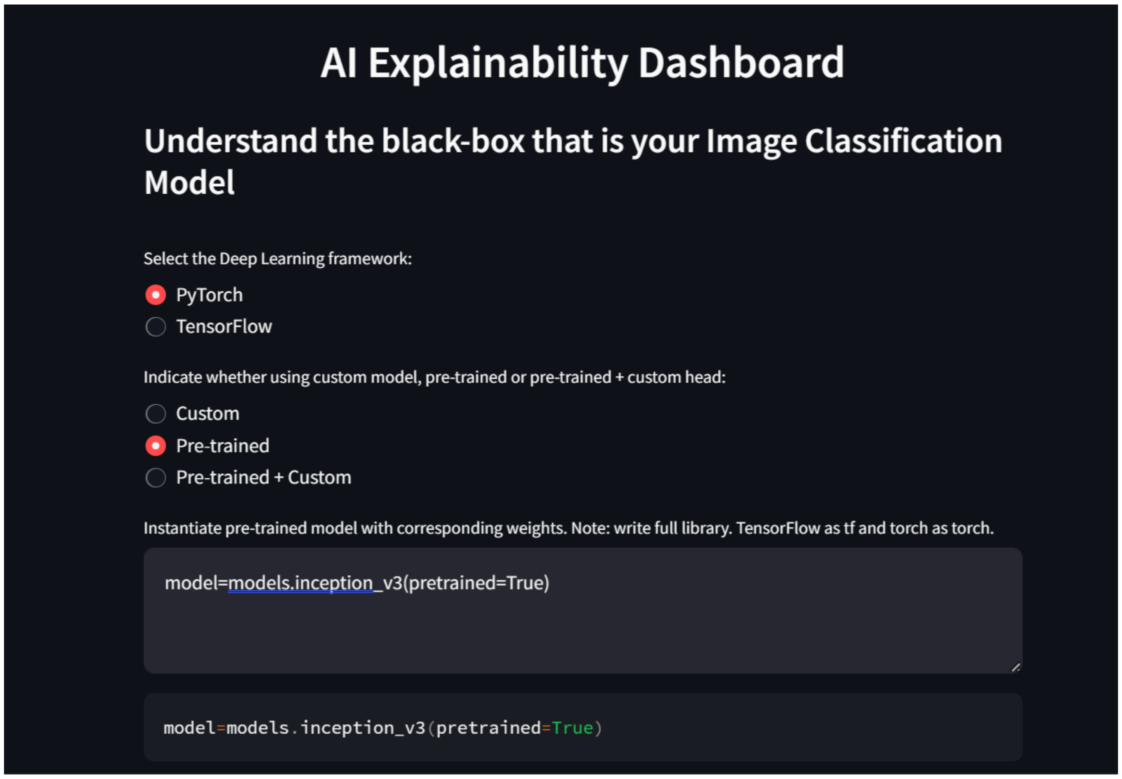
0.5, 0.5, 0.5


Applied pre-processing

```
torchvision.transforms.Compose([torchvision.transforms.ToTensor(),
torchvision.transforms.Resize((224, 224)),
torchvision.transforms.Normalize(
mean=[0.5, 0.5, 0.5],
std=[0.5, 0.5, 0.5])])
```

Upload the image you want to explain

Lime Demonstration: Utilized the problem and methodology defined in the LIME section above.


AI Explainability Dashboard

Understand the black-box that is your Image Classification Model

Select the Deep Learning framework:

[✓ PyTorch] TensorFlow

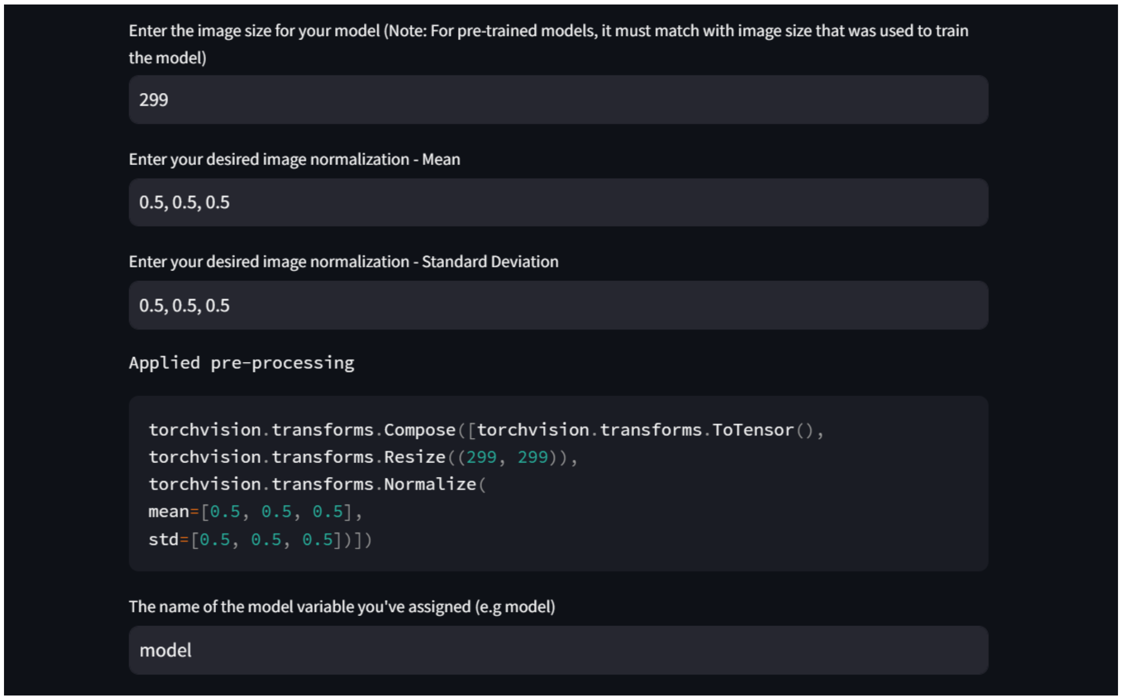
Indicate whether using custom model, pre-trained or pre-trained + custom head:

Custom [✓ Pre-trained] Pre-trained + Custom

Instantiate pre-trained model with corresponding weights. Note: write full library. TensorFlow as tf and torch as torch.

model=models.inception_v3(pretrained=True)

```
model=models.inception_v3(pretrained=True)
```



Enter the image size for your model (Note: For pre-trained models, it must match with image size that was used to train the model)

299

Enter your desired image normalization - Mean

0.5, 0.5, 0.5

Enter your desired image normalization - Standard Deviation

0.5, 0.5, 0.5

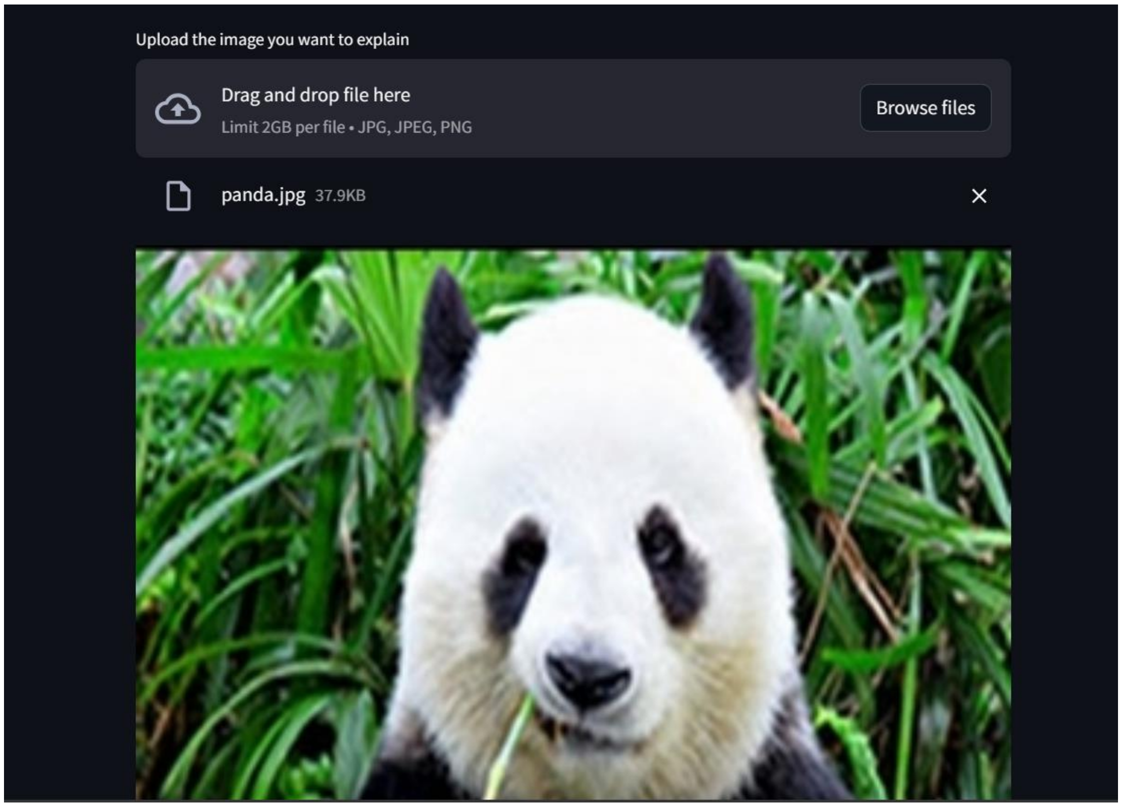
Applied pre-processing

```
torchvision.transforms.Compose([torchvision.transforms.ToTensor(),
torchvision.transforms.Resize((299, 299)),
torchvision.transforms.Normalize(
mean=[0.5, 0.5, 0.5],
std=[0.5, 0.5, 0.5])])
```

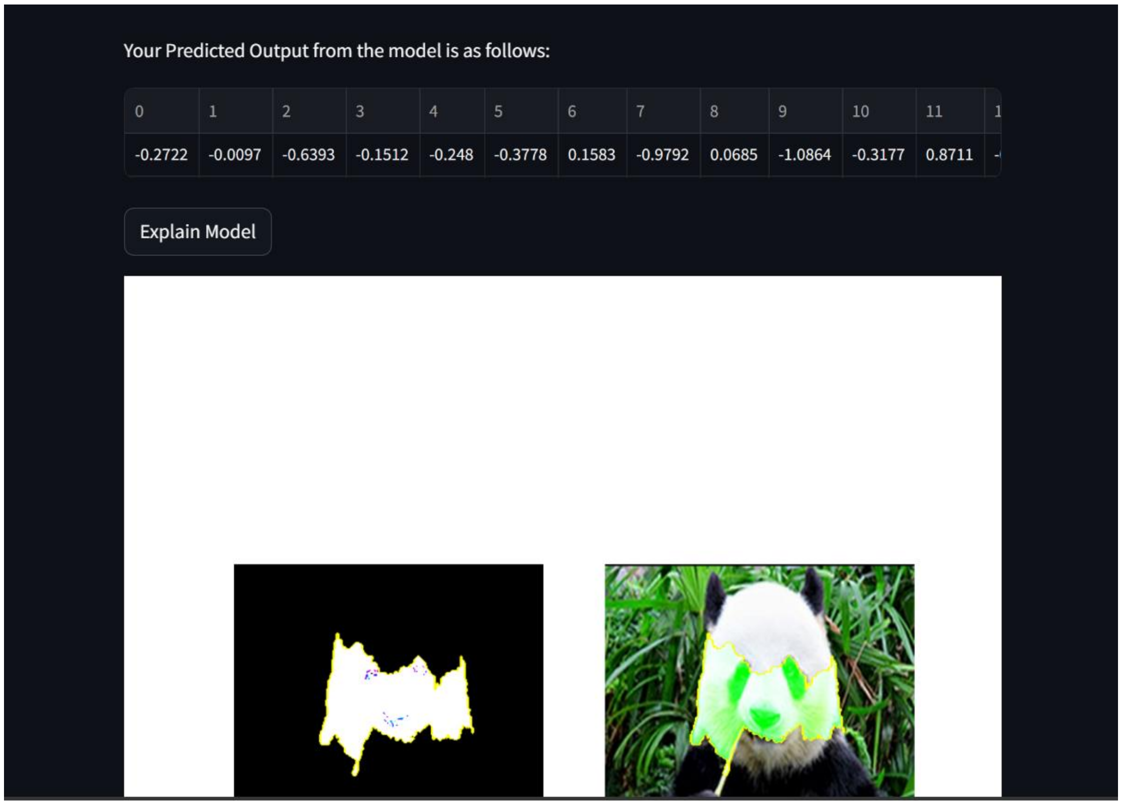
The name of the model variable you've assigned (e.g model)

model

Pre-processing parameters need to match the pre-processing done while training the model.



Upload the image you want to explain



Your Predicted Output from the model is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	1
-0.2722	-0.0097	-0.6393	-0.1512	-0.248	-0.3778	0.1583	-0.9792	0.0685	-1.0864	-0.3177	0.8711	-

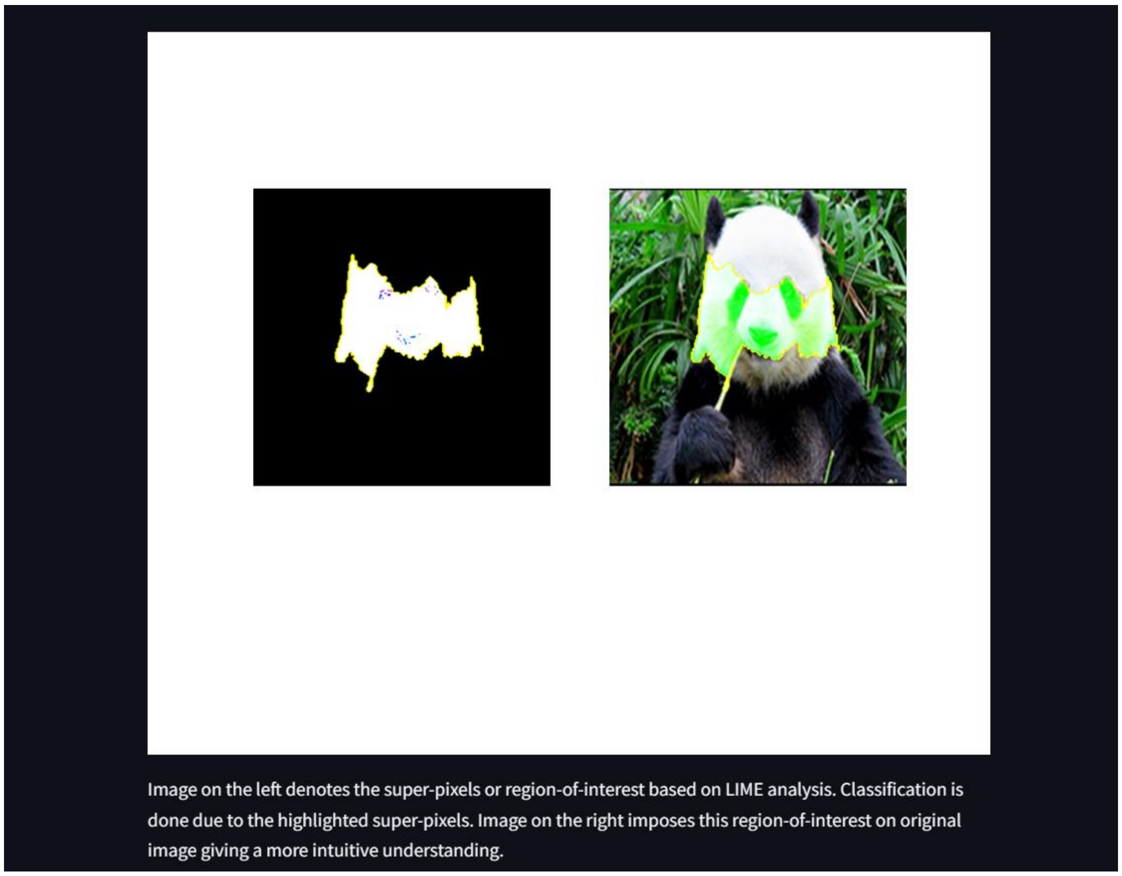
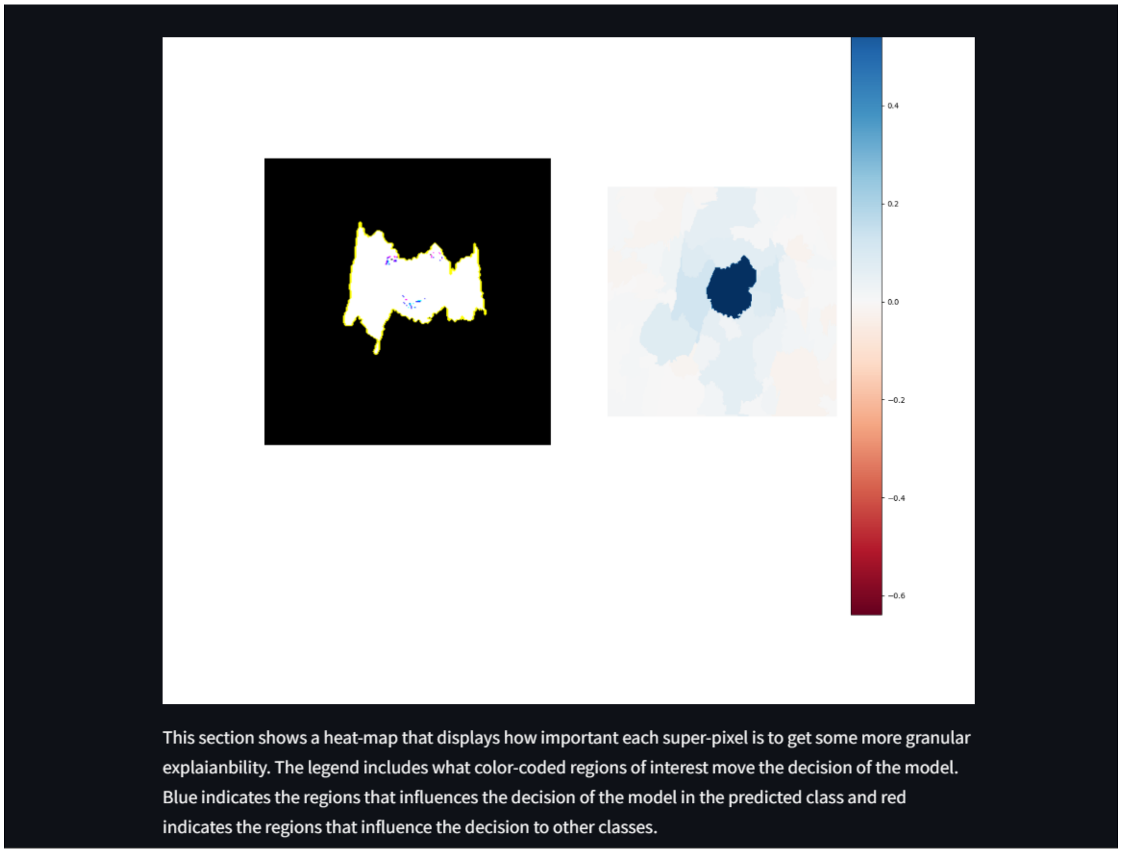


Image on the left denotes the super-pixels or region-of-interest based on LIME analysis. Classification is done due to the highlighted super-pixels. Image on the right imposes this region-of-interest on original image giving a more intuitive understanding.



This section shows a heat-map that displays how important each super-pixel is to get some more granular explaianbility. The legend includes what color-coded regions of interest move the decision of the model. Blue indicates the regions that influences the decision of the model in the predicted class and red indicates the regions that influence the decision to other classes.

The XAI analysis is done when prompted by clicking on the button. A detailed LIME analysis is presented at the end, and appropriate explanations for the plot are provided to help the user understand these results. This analysis helps the user to make the call as to whether their black box model has learnt the right features to be able to make the decision or not. If it hasn't, the modeling approach needs to change.

Conclusion

I have gained an in-depth understanding of XAI utilities, specifically LIME. Every successful Data Scientists needs to understand XAI and the need for this cutting-edge concept. Through this project, I have gained this important understanding and also further developed my web application dashboarding skillset that allows me to successfully display my Data Science projects in an interactive manner.

Future work – To integrate all XAI Python capabilities that are part of the demos into the XAI product offering.

References

- https://towardsdatascience.com/interpreting-image-classification-model-with-lime-1e7064a2f2e5
- https://towardsdatascience.com/how-to-explain-image-classifiers-using-lime-e364097335b4
- https://towardsdatascience.com/interpreting-image-classification-model-with-lime-1e7064a2f2e5
- https://github.com/PacktPublishing/Applied-Machine-Learning-Explainability-Techniques/blob/main/Chapter05/LIME_with_image_data.ipynb
- https://github.com/marcotcr/lime
- https://towardsdatascience.com/lime-how-to-interpret-machine-learning-models-with-python-94b0e7e4432e
- https://towardsdatascience.com/lime-vs-shap-which-is-better-for-explaining-machine-learning-models-d68d8290bb16
- https://www.kaggle.com/code/bygbrains/dog-cat-pandas-image-classifier

Code Originality Index:

- For the Custom-Model-XAI-Streamlit.py → No code has been used directly from the internet.
- For Image-Classification-Model.py → The calculation is as follows: (129 – 20) / (129 + 15) * 100 = 75%