"Enhancing the performance of Plant Diseases Detection with Leaf Images by using Explainable Artificial Intelligence Techniques.":

XAI : *LIME*, as an explainable model, ensures that the model is implementable by revealing its inner functions. This technique can explain the complex classification model by providing explanations through the less complex model. It is model agnostic technique that may be used with any machine learning and deep learning model. It allows for some interpretation and explanation latitude without restricting the classification models that might be developed. As we know the CNN is black box, it directly gives the prediction for any image which we pass through the model. For the explanation, we used the LIME which gives the accurate explanation for prediction of our model.

The value of the research is that it explains everything that prior researchers have not explained in depth.The research investigates the identification and explanation of plant disease. Explaining the prediction implies identifying disease that gives a qualitative understanding of their relevance to the model's prediction.

In this research, lime is used for manual filtration of the images. In lime images are passed and Lime divides the picture in the two colours mainly Green and Red in which Green colour indicates positive part from which the model is going to predict the disease and Red colour indicates negative part from which model reads but ignores these parts during the prediction. Also the image shows the colourless part which the model is unable to read.

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Activation, BatchNormalization
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator,DirectoryIterator
from tensorflow.keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from keras import backend as K
import os
import tensorflow as tf
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from zipfile import ZipFile
# file_name = "/content/drive/MyDrive/archive.zip"
file_name = '/content/drive/MyDrive/Mangesh_Project/PROJECT/Exaplainable_AI/val'
# with ZipFile(file_name,'r') as zip:
#   zip.extractall()

print("completed")
```

```
completed
```

```
# Path to train and test directory
dir_ = os.path.join(file_name)


# Generate training and test data with Image Generator
train_datagen = ImageDataGenerator(rescale=1/255,
                                   validation_split = 0.2)


train_generator = train_datagen.flow_from_directory(dir_,target_size=(256, 256),
                                                    batch_size= 20,
                                                    class_mode='categorical',
                                                    shuffle=False,
                                                    subset = 'training')

test_generator = train_datagen.flow_from_directory(dir_,target_size=(256, 256),
                                                   batch_size= 20,
                                                   class_mode = 'categorical',
                                                   shuffle=False,
                                                   subset = 'validation')
```

```
# Fetch the data and the labels
x_train, y_train = next(train_generator)
x_test, y_test  = next(test_generator)

# Fix the filepath
test_filepath = []
for filepath in test_generator.filepaths:
    filepath = filepath.replace('\\', '/')
    test_filepath.append(filepath)
```

```
Found 196 images belonging to 12 classes.
Found 42 images belonging to 12 classes.
```

```
model = Sequential([

    # First convolution
        Conv2D(16, (3,3), activation='relu', input_shape=(256,256, 3)),
        MaxPooling2D(2, 2),

    # Second convolution
        Conv2D(32, (3,3), activation='relu'),
        MaxPooling2D(2,2),

    # Third convolution
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D(2,2),

        Flatten(),

    # Dense hidden layer
        Dense(512, activation='relu'),
        Dropout(0.2),

    # Output neuron.
        Dense(12, activation='softmax')
])
```

Inception model: V3

```
#@title Inception model: V3
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_generator,
                    validation_steps=1,
                    epochs = 10,
                    verbose = 1)
```

```
Epoch 1/10
10/10 [==============================] - 1s 81ms/step - loss: 1.5683 - accuracy: 0.4949
Epoch 2/10
10/10 [==============================] - 1s 75ms/step - loss: 1.1705 - accuracy: 0.5816
Epoch 3/10
10/10 [==============================] - 1s 75ms/step - loss: 0.8672 - accuracy: 0.6939
Epoch 4/10
10/10 [==============================] - 1s 75ms/step - loss: 0.8187 - accuracy: 0.6735
Epoch 5/10
10/10 [==============================] - 1s 78ms/step - loss: 0.6528 - accuracy: 0.7449
Epoch 6/10
10/10 [==============================] - 1s 71ms/step - loss: 0.5469 - accuracy: 0.7959
Epoch 7/10
10/10 [==============================] - 1s 74ms/step - loss: 0.5660 - accuracy: 0.8010
Epoch 8/10
10/10 [==============================] - 1s 74ms/step - loss: 0.5410 - accuracy: 0.7857
Epoch 9/10
10/10 [==============================] - 1s 76ms/step - loss: 0.4358 - accuracy: 0.8061
Epoch 10/10
10/10 [==============================] - 1s 72ms/step - loss: 0.2704 - accuracy: 0.9286
```

Pretrained Inception model: V3

```
from tensorflow.keras.applications import inception_v3 as inc_net
```

```
from skimage.io import imread
from skimage.transform import resize
```

```
from skimage import io
from tensorflow.keras.preprocessing import image
```

```
url = '/content/drive/MyDrive/Mangesh_Project/PROJECT/Exaplainable_AI/val/diplodia_fruits_disease/IMG_20221027_194945 (1).jpg'

def read_and_transform_img(url):

    img = imread(url)
    img = resize(img, (256,256))       #256*256

    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)

    return img

images = read_and_transform_img(url)

preds = model.predict(images)
prediction = np.argmax(preds)
pct = np.max(preds)

# if prediction == 0:
#     print('It\'s a cat!')
# elif prediction == 1:
#     print('It\'s a dog!')
# else:
#     print('It\'s a panda!')

if prediction==0:
      print("Canker_fruits-disease")

elif prediction==1:
      print("Caterpillar worms leaf Disease")

elif prediction==2:
      print("Faint color fruit disease")

elif prediction==3:
      print("InitialBurn leaf Disease")

elif prediction==4:
      print("Myrtle Rust leaf disease")

elif prediction==5:
      print("Nutrition deficiency leaf disease")

elif prediction==6:
      print("SemiBurn leaaf disease")

elif prediction==7:
      print("crack_fruits_disease")
elif prediction==8:
      print("diplodia_fruits_disease")

elif prediction==9:
      print("fungi_fruits_disease")
elif prediction==10:
      print("shrink leaf disease")
elif prediction==11:
      print("uneven size fruit disease")



print("Accuracy of these prediction is",pct)
```

```
1/1 [==============================] - 0s 25ms/step
Myrtle Rust leaf disease
Accuracy of these prediction is 0.72957003
```

```
pip install lime
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lime in /usr/local/lib/python3.8/dist-packages (0.2.0.1)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.8/dist-packages (from lime) (0.18.3)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.8/dist-packages (from lime) (1.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from lime) (3.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from lime) (1.7.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from lime) (4.64.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from lime) (1.21.6)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-image>=0.12->lime) (1.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image>=0.12->lime) (2.8.8)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image>=0.12->lime) (2.9.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.8/dist-packages (from scikit-image>=0.12->lime) (2022.
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image>=0.12->li
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->lime) (1.4.4)
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib->lime) (1.
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=0.18->lime) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=0.18->lime) (3.1.
```

```
from lime import lime_image
explainer = lime_image.LimeImageExplainer()
```

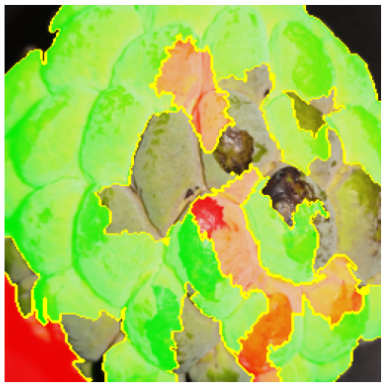
```
explanation = explainer.explain_instance(images[0].astype('double'), model.predict,
                                         top_labels=3, hide_color=0, num_samples=1000)
```

```
100%                                  1000/1000 [00:12<00:00, 50.84it/s]
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
```

```
from skimage.segmentation import mark_boundaries
temp_1, mask_1 = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=15, hide_rest=True)
temp_2, mask_2 = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=False, num_features=30, hide_rest=False)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,15))
ax1.imshow(mark_boundaries(temp_1, mask_1))
ax2.imshow(mark_boundaries(temp_2, mask_2))
ax1.axis('off')
ax2.axis('off')
```

```
(-0.5, 255.5, 255.5, -0.5)
```



```
1/1 [==============================] - 0s 20ms/step
```

```
import skimage
```

```
1/1 [==============================] - 0s 20ms/step
```

```
from tensorflow.keras.applications.imagenet_utils import decode_predictions

def transform_img_fn_ori(url):

    img = skimage.io.imread(url)
    img = skimage.transform.resize(img, (299,299))
    img = (img - 0.5)*2
    img = np.expand_dims(img, axis=0)
    preds = inet_model.predict(img)
    for i in decode_predictions(preds)[0]:
        print(i)
    return img

inet_model = inc_net.InceptionV3()
images_inc_im = transform_img_fn_ori(url)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_orderin
96112376/96112376 [==============================] - 0s 0us/step
1/1 [==============================] - 2s 2s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
('n07760859', 'custard_apple', 0.8643168)
('n07768694', 'pomegranate', 0.0030906138)
('n01664065', 'loggerhead', 0.0026823552)
('n04591157', 'Windsor_tie', 0.0019177982)
('n07754684', 'jackfruit', 0.0015087047)
```

```
explanation = explainer.explain_instance(images[0].astype('double'), inet_model.predict, top_labels=5, hide_color=0, num_samples=1000)

temp_1, mask_1 = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=15, hide_rest=True)
temp_2, mask_2 = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=False, num_features=20, hide_rest=False)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,15))
ax1.imshow(mark_boundaries(temp_1, mask_1))
ax2.imshow(mark_boundaries(temp_2, mask_2))
ax1.axis('off')
ax2.axis('off')
```

```
100%                                                  1000/1000 [00:15<00:00, 66.37it/s]
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
```

Double-click (or enter) to edit

```
1/1 [==============================] - 0s 30ms/step
```

```
1/1 [==============================] - 0s 33ms/step
```

```
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 34ms/step
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
```