DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL504 **DATE:**

COURSE NAME: Artificial Intelligence Laboratory **CLASS: TY-IT**

Name: Anish Sharma **Rollno:I011**

EXPERIMENT NO.10

CO/LO: Apply various AI approaches to knowledge intensive problem solving, reasoning, planning and uncertainty.

AIM / OBJECTIVE: Implement Explainable AI for image and text

Code:

```
# Step 1: Import necessary libraries from

transformers import pipeline import numpy as

np import matplotlib.pyplot as plt from

lime.lime_text import LimeTextExplainer


# Step 2: Load a pre-trained text classification model using BERT

classifier = pipeline('text-classification')


# Step 3: Read a text file or use a string input for classification

# Replace 'input.txt' with your own text file

name file_path = 'input.txt' with open(file_path,

'r') as f:

    input_text = f.read()


# Step 4: Display the uploaded text
print("Uploaded Text:\n", input_text)
```

```
# Step 5: Classify the text using the pre-trained BERT
model predictions = classifier(input_text)
print("\nPredictions:", predictions)


# Step 6: Display the top classification label
top_prediction = predictions[0]
print("\nTop Prediction: {} with confidence {:.2f}".format(top_prediction['label'],
top_prediction['score']))


# Step 7: LIME (Local Interpretable Model-agnostic Explanations) for Text Explainability
# Define class names according to your classification labels
explainer = LimeTextExplainer(class_names=['LABEL_0', 'LABEL_1'])  # Modify based on
your classes


# Define a prediction function for LIME to work with BERT
def predict_proba(texts):
    results = classifier(texts)
    # Convert the classifier output to a probability-like format (required for
LIME)      probabilities = []      for result in results:
        probabilities.append([result['score'], 1 - result['score']])  # Adjust for binary classification
return np.array(probabilities)


# Step 8: Explain the prediction using LIME
explanation = explainer.explain_instance(input_text, predict_proba, num_features=10)
```

Step 9: Visualize the explanation (important words/phrases for classification)

explanation.show_in_notebook(text=True) # Will open the explanation in a Jupyter Notebook

Output:

```
Uploaded Text:
 The weather today is amazing, and I feel incredibly happy. The sun is shining, and everything seems to be going
well. I had a productive morning and enjoyed a walk in the park with friends. Overall, it's been a fantastic day
full of positivity and joy!

Predictions: [{'label': 'POSITIVE', 'score': 0.999887228012085}]

Top Prediction: POSITIVE with confidence 1.00
```

```
Uploaded Text:
 "I am extremely disappointed with the service. The product was of very poor quality and did not meet my expectat
ions at all."


Predictions: [{'label': 'NEGATIVE', 'score': 0.9998070597648621}]

Top Prediction: NEGATIVE with confidence 1.00
```

Code:

```
import os import
keras
from keras.applications import inception_v3 as inc_net from
keras.preprocessing import image
from keras.applications.imagenet_utils import decode_predictions
from skimage.io import imread import matplotlib.pyplot as plt
import numpy as np import lime
from lime import lime_image
from skimage.segmentation import mark_boundaries

print('Notebook run using keras:', keras.__version__)

# Load the InceptionV3 model pre-trained on ImageNet inet_model
= inc_net.InceptionV3()

def transform_img_fn(path_list):
    out = []       for img_path
in path_list:
        img = image.load_img(img_path, target_size=(299, 299))
x = image.img_to_array(img)         x = np.expand_dims(x,
```

```
axis=0)            x = inc_net.preprocess_input(x)
out.append(x)
    return np.vstack(out)

# Replace this path with your actual image path image_path =
r'C:\Users\dhruv\OneDrive\Desktop\dwn\image.png' images =
transform_img_fn([image_path])

# Visualize the image (undoing the preprocessing normalization)
plt.imshow(images[0] / 2 + 0.5) plt.show()

# Predict with the model preds =
inet_model.predict(images) for x in
decode_predictions(preds, top=5)[0]:
    print(x)

# Initialize the Lime Image Explainer explainer
= lime_image.LimeImageExplainer()

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

# Get the image with mask and display it temp,
mask = explanation.get_image_and_mask(
explanation.top_labels[0], positive_only=True,
num_features=5, hide_rest=True
)

# Correct the visualization scaling for plotting plt.imshow(mark_boundaries(temp
/ 2 + 0.5, mask)) plt.show()

temp, mask = explanation.get_image_and_mask(explanation.top_labels[0],
positive_only=False, num_features=10, hide_rest=False)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask)) plt.show() temp, mask =
explanation.get_image_and_mask(explanation.top_labels[0],
positive_only=False, num_features=1000, hide_rest=False, min_weight=0.05)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask)) plt.show()
```

```
#Select the same class explained on the figures above. ind
=   explanation.top_labels[0]

#Map each explanation weight to the corresponding superpixel dict_heatmap
= dict(explanation.local_exp[ind])
heatmap = np.vectorize(dict_heatmap.get)(explanation.segments)

#Plot. The visualization makes more sense if a symmetrical colorbar is used.
plt.imshow(heatmap, cmap = 'RdBu', vmin  = -heatmap.max(), vmax = heatmap.max())
plt.colorbar()
temp, mask = explanation.get_image_and_mask(explanation.top_labels[1],
positive_only=True, num_features=6, hide_rest=True)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask)) plt.show()
temp, mask = explanation.get_image_and_mask(explanation.top_labels[1],
positive_only=False, num_features=5, hide_rest=False)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask)) plt.show()

import json
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input import
shap
# load pre-trained model and data model
= ResNet50(weights="imagenet") X, y
= shap.datasets.imagenet50()
print(y) plt.imshow(X[20])
plt.show()
X = np.clip(X, 0, 255).astype(np.uint8)
plt.imshow(X[4]) plt.show()
```

Output:


```
iate compiler flags.
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
('n11939491', 'daisy', 0.8744845)
('n12144580', 'corn', 0.001350845)
('n03786901', 'mortar', 0.0012305334)
('n02219486', 'ant', 0.0010356873)
('n04447861', 'toilet_seat', 0.00073953025)
```

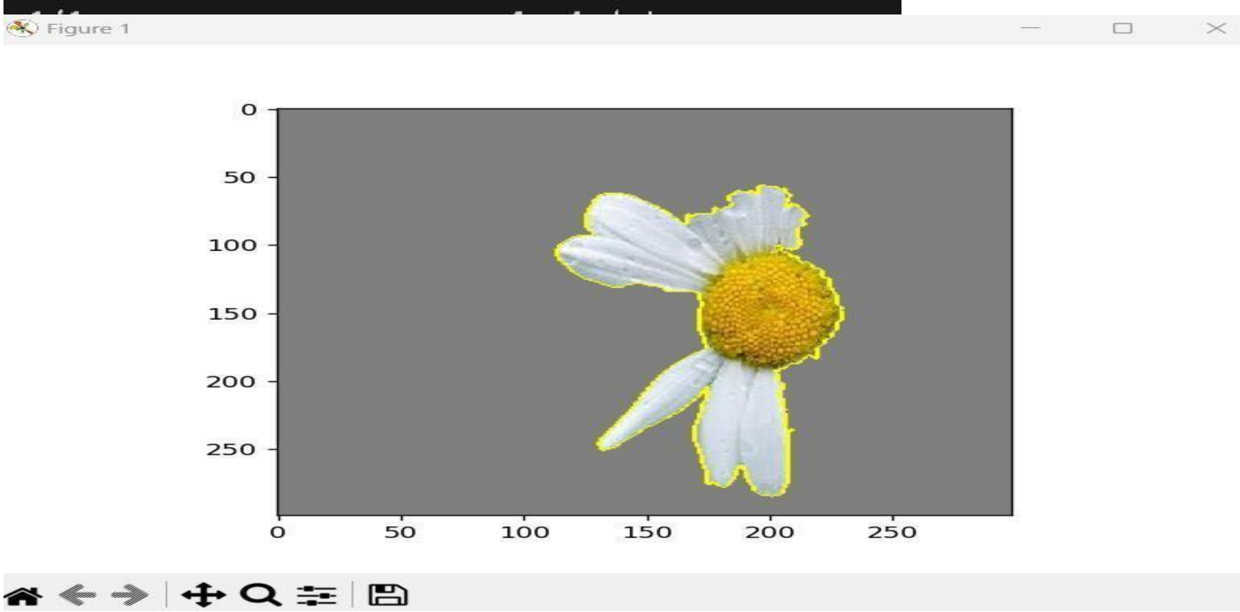



Figure 1
0
50
100
150
200
250
0
50
100
150
200
250

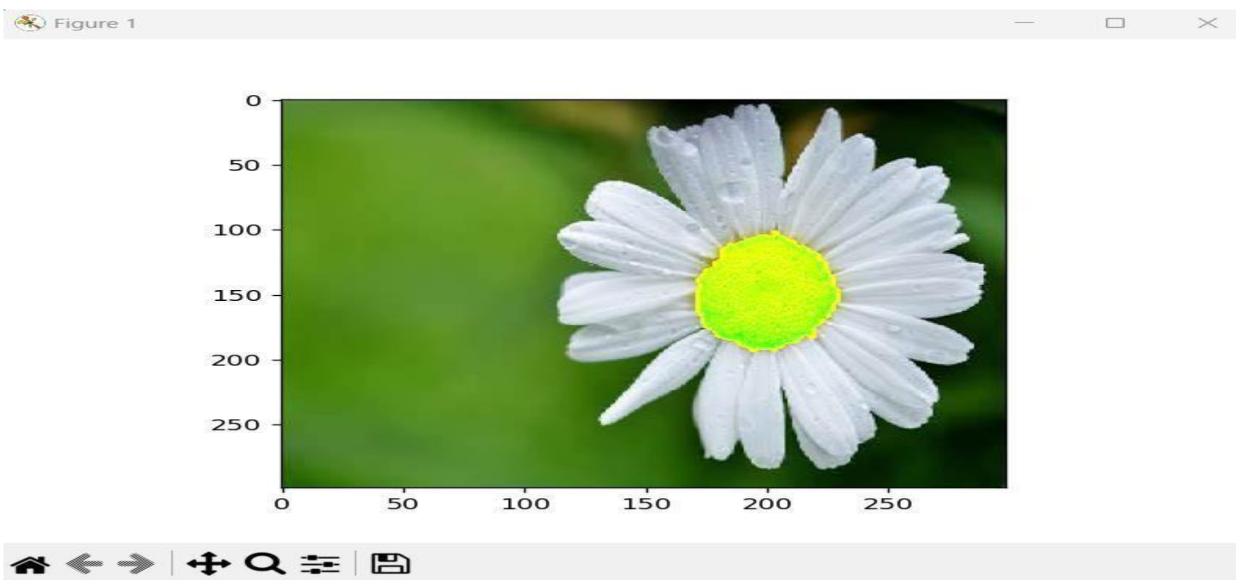
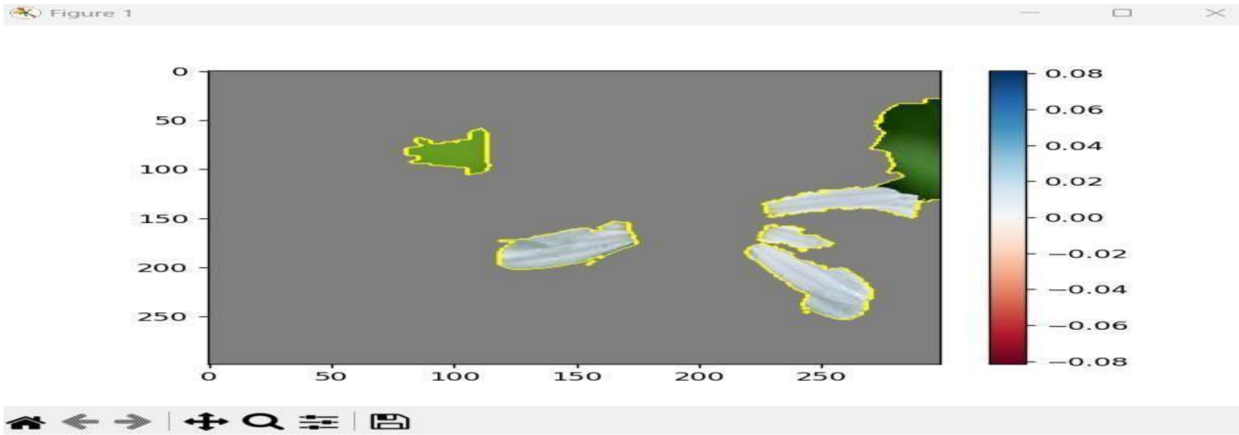
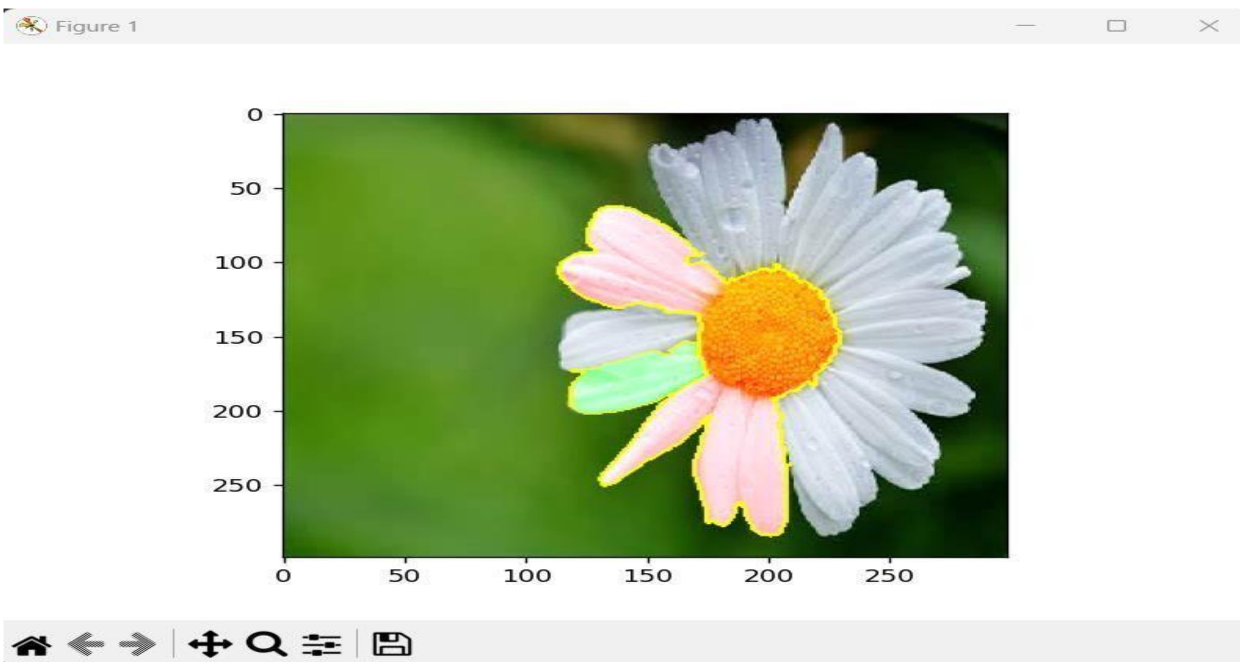


Figure 1
0
50
100
150
200
250
0
50
100
150
200
250





CONCLUSION: We implemented the explainable AI.

REFERENCES:

[1] Stuart Russell and Peter Norvig, “Artificial Intelligence: A Modern Approach”, 2nd Edition, Pearson Education, 2010