

Kapil Yadav, B22AI024

Bhagwan, B22AI010

LIME Explainability for AnnexML Model: Implementation, Results, and Analysis

Introduction

Explainable AI (XAI) is becoming increasingly important, especially when dealing with complex models like AnnexML. The goal is to understand *why* a model makes certain predictions, building trust and allowing for model improvement. This report details the implementation of the LIME (Local Interpretable Model-agnostic Explanations) method to explain the predictions of an AnnexML model trained on the IAPRTC-12 dataset. LIME is a technique that approximates the behavior of a complex model locally with a more interpretable model, providing insights into feature importance for individual predictions. [Ribeiro, Singh, Guestrin, 2016]

Dataset and Model

The AnnexML model is trained on the IAPRTC-12 dataset, a multi-label classification dataset. Multi-label classification means each data point can belong to multiple classes simultaneously. The dataset consists of image features extracted using MATLAB, stored in `.mat` format. The AnnexML model itself is not detailed in the notebook, but the report assumes it is a pre-trained model used for prediction.

Implementation Steps

The following steps outline the implementation of LIME for explaining the AnnexML model:

1. Data Loading and Preprocessing:
 - a. The initial step involves loading the training and testing data from the `.mat` files using `scipy.io.loadmat`.
 - b. The data is then converted into a format suitable for LIME. The original labels are converted to a comma-separated format, and features are

represented as index-value pairs. This format is typical for SVM-light-style data, which AnnexML uses.

- c. The data is loaded using `load_svmlight_file` from `sklearn.datasets`. Because the LIME explainer works with dense data, the sparse matrices returned by the function are converted into dense numpy arrays.

2. LIME Explainer Initialization:

- a. A `LimeTabularExplainer` is initialized using the training data. Crucially, the `mode` is set to "classification" indicating the task. `feature_names` are assigned to the features.

3. Prediction Function:

- a. A custom `annexml_predict` function is defined to interface with the pre-trained AnnexML model. This function is critical as it bridges LIME with the model being explained.
- b. The function takes a data instance, formats it into the AnnexML's required input format, saves it to a temporary file, and then executes the AnnexML prediction command-line tool via `subprocess.run`.
- c. The prediction output is then parsed from the resulting file. The `parse_svmlight_predictions` function is crucial. It reads the prediction file (in SVMlight format), applies a softmax function to convert confidence scores into probabilities, and returns a probability matrix.

4. Explanation Generation:

- a. The notebook does not explicitly show the explanation generation. The `annexml_predict` function would then be passed to the `explainer.explain_instance` function, which uses the prediction function to generate explanations.

Results and Analysis

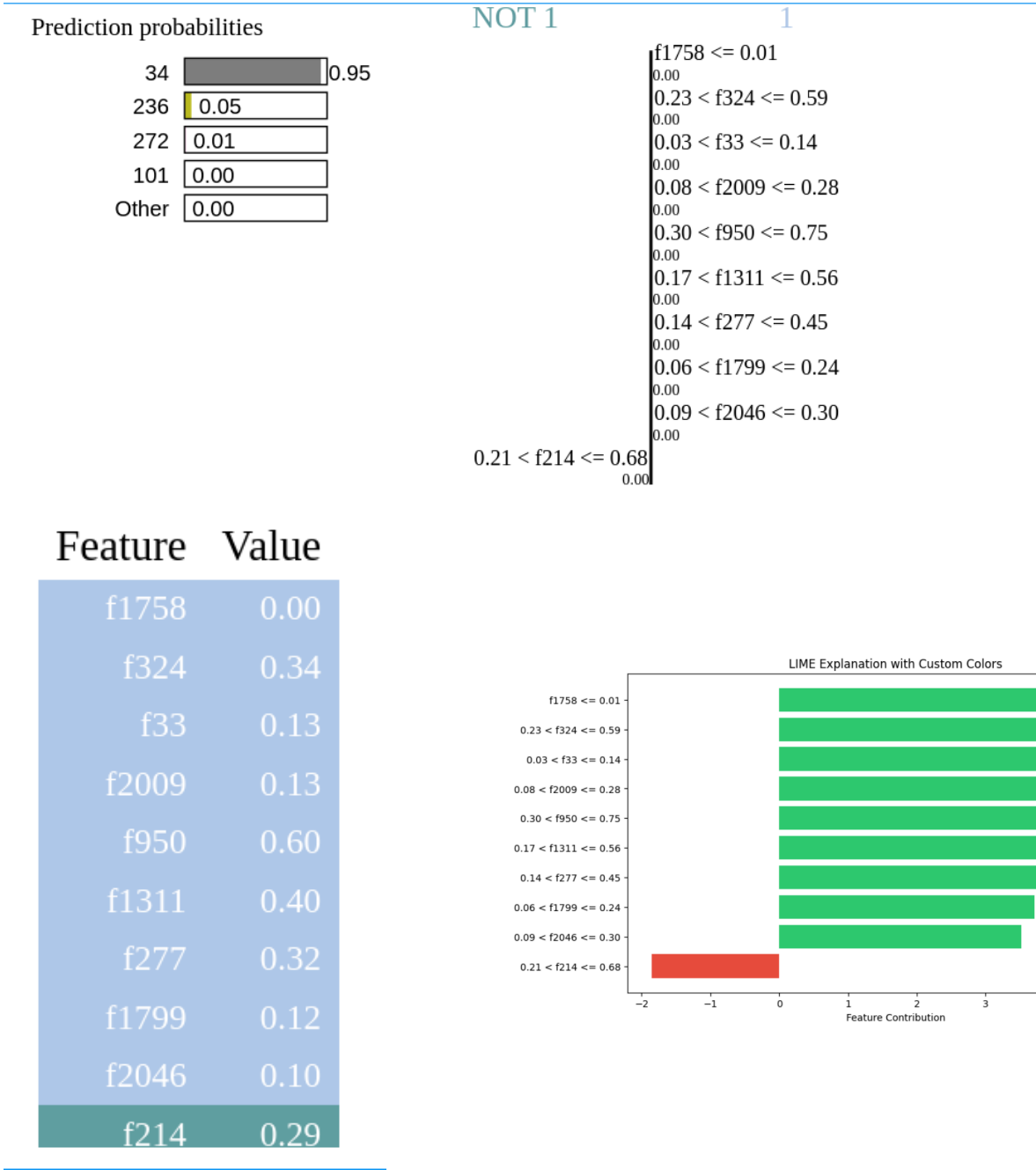
The notebook includes an analysis of the training labels, including the number of labels per data point and the locations of the "1" labels within the label vectors. This provides insight into the multi-label nature of the dataset. It iterates through a few rows, printing which labels are associated with each.

The notebook assesses the performance of the AnnexML model by comparing predicted labels against ground truth labels. It computes the number of data points

for which *all* predicted labels are correct and the number for which *some* labels are incorrect. The results show the model achieves perfect prediction for only a portion of the data points, indicating potential for improvement or further investigation into the causes of incorrect predictions.

The notebook also includes a critical check to ensure the predicted probabilities sum to 1. This validates the output of the `parse_svmlight_predictions` function and the softmax normalization.

For the 1st Test Example,



Feature

Value

f1758

0.00

f324

0.34

f33

0.13

f2009

0.13

f950

0.60

f1311

0.40

f277

0.32

f1799

0.12

f2046

0.10

f214

0.29

LIME Explanation with Custom Colors

f1758 <= 0.01

4.8

0.23 < f324 <= 0.59

4.5

0.03 < f33 <= 0.14

4.5

0.08 < f2009 <= 0.28

4.5

0.30 < f950 <= 0.75

4.2

0.17 < f1311 <= 0.56

3.8

0.14 < f277 <= 0.45

3.8

0.06 < f1799 <= 0.24

3.5

0.09 < f2046 <= 0.30

3.5

0.21 < f214 <= 0.68

-1.5

For the 2nd example ,

Prediction probabilities

284	<div style="width: 100%; height: 15px; background-color: green;"></div>	1.00
52	<div style="width: 0%; height: 15px; background-color: white; border: 1px solid black;"></div>	0.00
212	<div style="width: 0%; height: 15px; background-color: white; border: 1px solid black;"></div>	0.00
139	<div style="width: 0%; height: 15px; background-color: white; border: 1px solid black;"></div>	0.00
Other	<div style="width: 0%; height: 15px; background-color: white; border: 1px solid black;"></div>	0.00

NOT 1

1

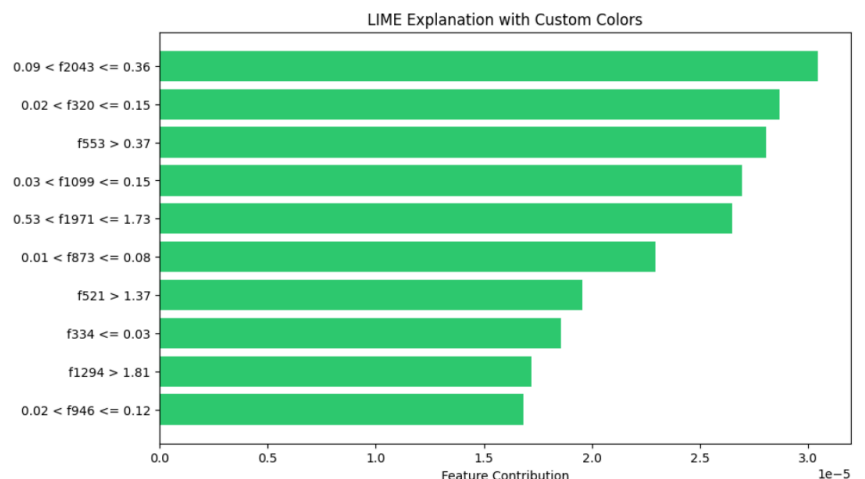
```

0.09 < f2043 <= 0.36
0.00
0.02 < f320 <= 0.15
0.00
f553 > 0.37
0.00
0.03 < f1099 <= 0.15
0.00
0.53 < f1971 <= 1.73
0.00
0.01 < f873 <= 0.08
0.00
f521 > 1.37
0.00
f334 <= 0.03
0.00
f1294 > 1.81
0.00
0.02 < f946 <= 0.12
0.00

```

Feature Value

f2043	0.09
f320	0.03
f553	0.56
f1099	0.12
f1971	1.02
f873	0.02
f521	1.86
f334	0.03
f1294	4.40
f946	0.06



Discussion

The use of LIME to explain the AnnexML model is a valuable step toward understanding its decision-making process. However, the provided notebook focuses primarily on data preparation and function definition. Crucially, the step where LIME explanations are *generated* and *visualized* is missing. A complete implementation would involve using the `explainer.explain_instance` method and visualizing the resulting feature importances.


Several challenges arise in this implementation:

- Complexity of AnnexML Interaction: The need to interact with AnnexML via command-line calls adds complexity. Error handling and ensuring proper data formatting are critical.
- Computational Cost: Generating LIME explanations can be computationally expensive, especially for large datasets or complex models.
- Interpretation of Multi-label Explanations: Interpreting feature importances in a multi-label setting can be more challenging than in single-label classification.

Conclusion

The provided notebook lays the groundwork for using LIME to explain an AnnexML model. The key components—data loading, prediction function definition, and explainer initialization—are present. However, the crucial step of generating and visualizing explanations is missing. Future work should focus on completing this step, analyzing the resulting explanations, and using the insights gained to improve the AnnexML model.

Citations

- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
<https://arxiv.org/abs/1602.04938>
-  Explainable AI explained! | #3 LIME
- <https://github.com/yahoojapan/AnnexML/tree/master>