Action Influence Graphs for Model Explainability

Challenging Assignments and Mini Projects (CHAMP)

submitted as part of the course
Explainable Artificial Intelligence
BCSE418L

School Of Computer Science and Engineering
VIT Chennai

WINTER 2024-2025

Course Faculty : Dr. B Radhika Selvamani

Submitted By
JOSEPH JITTO (22BAI1428)

AAKANKSHA RAI (22BAI1436)

KEVIN PAUL (22BAI1445)

Abstract

This project focuses on improving model interpretability by generating Action Influence Graphs (AIGs) from trained machine learning models. As AI adoption increases in sensitive domains like healthcare and finance, understanding how models make decisions is essential. Our approach involves converting models like Decision Trees, Random Forests, and Neural Networks into graph-based structures that visually represent how input features influence predictions. These graphs consist of nodes (features or outcomes) and edges (decision flows), enabling intuitive interpretation. The system supports black-box models using SHAP values and includes features like clustering, influence scoring, and shortest path calculation. We demonstrate its effectiveness using the Iris dataset. The resulting graphs offer both visual and quantitative insight into model behavior, supporting transparent and accountable AI development.

Contents

Index

---

1 Introduction

Machine learning models are often criticized for being black boxes. In high-stakes applications, decision-makers need more than just predictions—they need explanations. This project addresses the need for explainability by proposing Action Influence Graphs (AIGs), which visually and structurally represent how features impact outcomes in a model. By integrating traditional interpretable models and black-box models, our tool is aimed at AI practitioners, data scientists, and domain experts who need to trust and verify model decisions.

2 Related Works

Existing tools for explainability include decision trees, feature importance scores, LIME, and SHAP. While decision trees provide transparency through rules, they are limited to simple models. SHAP extends explainability to complex models but can be abstract. Our approach combines the strengths of both by building interpretable graph structures from various model types.

2.1 History

The concept of visualizing decision logic dates back to early expert systems and rule-based classifiers. With the advent of neural networks and ensemble methods, explainability became challenging. SHAP and LIME were introduced to address this. Our work builds on SHAP’s foundation and integrates it into a graph-based visualization pipeline.

3 Design

The design and implementation of the Action Influence Graph framework involve several core modules and classes:

3.1 System Architecture

1. ModelToAIG Class: The primary class responsible for converting a trained model into an AIG. It receives a model, feature names, optional target names, and optional sample data.

2. Graph Construction: A networkx.DiGraph object is used to represent the graph structure, where each node corresponds to a feature condition or an outcome, and edges represent the flow of decision logic.

3. Node Categorization: Nodes are categorized into 'event' nodes (decisions or features) and 'outcome' nodes (final predictions). These are tracked with a node_types dictionary.

4. Explanation Mapping: Each node and edge is annotated with natural-language explanations using a dictionary named explanations.

3.2 Parsing Different Models

1. Tree-Based Models (e.g., Decision Trees, Random Forests): Parsed using the model's internal tree structure (tree_). Nodes are generated for each decision split, and leaves are labeled with the final prediction.

2. Feature Importance Models: Uses the feature_importances_ attribute to determine which features significantly influence the outcome.

3. Black-Box Models (e.g., Neural Networks): SHAP (SHapley Additive exPlanations) is used to calculate feature contributions. SHAP values are aggregated and features with non-zero influence are added to the graph.

3.3 Visualization and Plotting

Visualization is implemented using Plotly, offering:

- Kamada-Kawai or Spring layout options for graph layout.
- Color-coded nodes based on their role or cluster ID.
- Arrows for directed edges.
- Hover tooltips with influence scores and cluster IDs.

3.4 Analytical Features

- Shortest Path: Uses networkx.shortest_path to find the shortest influence route between any two nodes.
- Centrality Measures: Calculates betweenness, closeness, and eigenvector centrality to evaluate node significance.
- Influence Scores: Based on eigenvector centrality.
- Clustering: Spectral Clustering applied on the adjacency matrix of the undirected graph, with cluster IDs stored per node.

The modular design allows extending the tool to support other model types or alternate clustering algorithms. This architecture ensures flexibility, performance, and interpretability.

3.5 Sample

We trained classifiers on the Iris dataset and visualized the resulting AIGs. Each node represents a feature condition or output class. Internal nodes describe decision splits, and leaf nodes represent class outcomes. For neural networks, SHAP values estimate the influence of each feature, which are then connected based on magnitude.

4 Results & Discussion

The visualizations clearly reveal feature importance and decision paths. For example, in a decision tree, paths from nodes like `petal length <= 2.45` lead to class outcomes like `setosa`. Centrality measures highlighted key decision features. We compared our AIG tool's clarity to textual outputs and found it provided superior intuitive insight.

- **Alternative Solutions**: Traditional SHAP plots or feature importance bar graphs.
- **Unimplemented Ideas**: Exporting graphs to interactive web dashboards.
- **Difference from Existing Systems**: Combines symbolic rule extraction with network analytics.

Task Instance	F1 Score	Micro F1 Score
Bengali	0.2920	0.7040
Hindi	0.2894	0.6886
Meitei	0.3215	0.6718
Multilingual	0.2940	0.6650

Table 1: Performance of highest ranked models for various languages (example placeholder)

5 Conclusion & Future Work

The AIG framework provides a powerful, flexible way to make models explainable. Its ability to handle various types of models and support visual and analytical tools makes it suitable for both research and industrial applications.

Future work includes:

Exporting graphs to dashboards

Supporting time-series models

Enhancing scalability for large models

Integrating with cloud-based ML platforms

Acknowledgments

We sincerely thank the organizing committee for recognizing our project. We also extend our gratitude to Dr. B Radhika Selvamani for the guidance and valuable feedback throughout this work.

References

[1] Oluwafemi Oriola and Eduan Kotzé. Evaluating machine learning techniques for detecting offensive and hate speech in South African tweets. *IEEE Access*, 8:21496–21509, 2020.
[2] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[3] Martin Sundermeyer et al. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.