Machine Learning - Principles and Techniques **Summer Project**

Project : Counterfactual Explanation Trees for loan approved systems

Team Name: Sagar, Sonam *Team Members: 23N0071, 23N0061*

Abstract

Counterfactual Explanation (CE) is a method used after a classifier makes a prediction to suggest changes that could alter that prediction. Traditional CE methods focus on altering the prediction for a single instance. However, these methods do not address scenarios where actions need to be assigned to multiple instances simultaneously. To address this, we introduce a new framework called the Counterfactual Explanation Tree (CET), which employs decision trees to suggest changes for multiple instances at once in a clear and consistent manner. We first compute an effective action for the instances and then partition them to balance both effectiveness and interpretability. A stochastic local search algorithm is used to optimize the decision tree's effectiveness and interpretability.

1 Introduction

Counterfactual explanations help us understand how small changes in input data can affect the outcomes of decision-making models. For example, in a decision tree used for loan approval, a counterfactual explanation might show how changing a person's income or credit score could have led to a different decision. This approach makes decision trees more transparent and easier to understand by clearly illustrating what factors influence the final decision. By providing these insights, counterfactual explanations help ensure that decision-making models are fair and their outcomes are more understandable to users.
To offer meaningful insights, post-hoc local explanations must not only explain why a certain prediction is made but also guide users on how to achieve their desired outcomes.Counterfactual Explanations (CE) are designed to fulfill these needs.
For a classifier $f : X \to Y$, a target class $y^* \in Y$, and an instance $x \in X$ where $f(x)! = y^*$, CE aims to find a perturbation vector a that changes the prediction to the desired outcome, i.e., $f(x + a) = y^*$. This perturbation a represents actions users can take to alter the prediction result of the classifier f To determine this perturbation, the following optimization problem is often considered

$$\text{minimize}_{a \in A}\, c(a, x) \quad \text{subject to} \quad f(x + a) = y^*$$

Here, A is a set of feasible actions, or perturbation vectors, and c is a cost function that quantifies the effort required to implement the action a.

In Section 2, we present a comprehensive survey of the existing literature, highlighting relevant studies and approaches that inform our research. Our project proposal, including the core objectives and methodologies, is detailed in Section 4. Section 5 outlines the experimental setup and provides an in-depth analysis of our findings. We discuss potential directions for future research in Section 7. Finally, in Section 8, we conclude with a concise summary of our work and suggest avenues for forthcoming research.

2 Literature Survey

Several methods for extracting local explanations from complex models have been proposed [Ribeiro 2016, Koh 2017, Lundberg 2017]. Counterfactual Explanation (CE), also known as Actionable Recourse, is one such method that has gained increasing attention in recent years [Karimi 2020]. Most existing CE methods focus on providing a single action [Wachter 2018, Karimi 2020, Kanamori 2020, Schut 2021] or multiple diverse actions [Ustun 2019, Mothilal 2020] for a single instance.
While various local explanation methods have been proposed, recent studies have highlighted some issues, such as a lack of robustness [Ghorbani 2019, Dombrowski 2019] and incorrect assumptions [Barocas 2020, Venkatasubramanian 2020, Karimi 2021]. A major problem is that the region where these explanations apply is often unclear [Ribeiro 2018, Rudin 2019]. This issue can lead to local explanation methods failing to explain the global behavior of complex models [Rudin 2019] and may pose a risk of malicious manipulation [Aivodji 2019].
Our Counterfactual Explanation Tree (CET) is most closely related to AReS [Rawal 2020], which provides a global summary of actions using a two-level rule set. AReS helps users understand the recourse actions extracted from complex classifiers. However, AReS has limitations in ensuring transparency and consistency in the assignment of actions due to the nature of rule sets. Specifically, (1) it may not cover the entire input space, and (2) it may assign multiple actions to a single input instance [Freitas 2014].

3 Data set Details

This is the loan dataset, which includes various attributes related to loan applications. The dataset is instrumental in analyzing and predicting loan approval outcomes based on several applicant characteristics. Below is a detailed description of the dataset attributes.

Dataset Size:

Number of Rows: 614
Number of Columns: 13

Dataset Attribute Description:

- Loan_ID: Unique Loan ID
- Gender: Male/Female
- Married(Applicant married): (Y/N)
- Dependents : Number of dependents
- Education Applicant Education: (Graduate/Under Graduate)
- Self_Employed Self employed: (Y/N)

- ApplicantIncome : Applicant income
- CoapplicantIncome : Coapplicant income
- LoanAmount : Loan amount in thousands
- Loan_Amount_Term : Term of loan in months
- Credit_History : Credit history meets guidelines
- Property_Area : Urban/Semi Urban/Rural
- Loan_Status(Target, Loan approved): (Y/N)

Data Preprocessing Steps:

Handle Missing Values:

1. Imputed `Loan_Amount_Term` and `Credit_History` with the mode (most frequent value).
2. Imputed `LoanAmount` with the median value.
3. Dropped rows with missing values in the `Gender`, `Married`, and `Dependents` columns.
4. Imputed `Self_Employed` using K-Nearest Neighbors (KNN).

One-Hot Encoding:

- Applied one-hot encoding to categorical columns: `Loan_Status`, `Gender`, `Married`, `Dependents`, `Education`, `Self_Employed` and `Property_Area`.

Class Imbalance:

- Used Synthetic Minority Over-sampling Technique (SMOTE) to balance the classes in the target variable.

4 Methods and Approaches

Counterfactual Explanation Trees (CET) is a post-hoc explanation method designed to provide **transparency** and **consistency** for any machine learning model. Unlike traditional counterfactual explanation (CE) frameworks that focus on generating actions for single instances, CET addresses the entire input space, offering a more comprehensive and interpretable solution.

Limitations of Existing CE Frameworks

Existing CE methods often fall short because they focus solely on individual instances, failing to account for the broader input space. This localized approach limits their ability to provide consistent and transparent explanations across the entire dataset.

- We have solved the optimization problem using the **GLPK solver**, which is well-suited for large-scale linear programming.
- Used different **Cost Functions:**

  Max Percentile Shift (MPS): MPS quantifies the maximum change in the percentile rank of the model's prediction due to changes in feature values. By focusing on the most significant shifts, MPS helps identify counterfactuals that have a substantial impact on the model's outcome.

$$c(a \mid x) = \max_{d \in [D]} [Q_d(x_d + a_d) - Q_d(x_d)]$$

  Truncated Logarithmic Percentile Shift (TLPS): TLPS enhances MPS by applying a logarithmic transformation to the percentile shift and truncating extreme values. This approach ensures that counterfactuals are less sensitive to outliers, providing more stable explanations.

$$\text{TLPS} = \frac{1}{N} \sum_{i=1}^{N} \min(\text{truncate\_threshold}, \log(1 + |Q_d(x_d + a_d) - Q_d(x_d)|))$$

  Additional Features

- CET is designed to work seamlessly with data that has not been preprocessed. The framework automatically handles tasks such as one-hot encoding and outlier detection, making it adaptable and efficient in generating counterfactual explanations on the go.

4.1 Work Done

There were two problems regarding CET:

Problem 1: CE for Multiple Instances

The primary problem CET solves is **CE for Multiple Instances**. Given a set of N instances X where each instance x satisfies $f(x) = +1$, and a set of actions $A(X)$, the objective is to find an action a that minimizes the following cost function:

$$\text{minimize}_{a \in A(X)}\, g(a \mid X) := \sum_{x \in X} c(a \mid x)$$

Problem 2: Learning CET

Given a set of N instances X such that for each $x \in X$, $f(x) = +1$, and a parameter $\lambda > 0$, find a CET $h \in \mathcal{H}$ that is an optimal solution to the following optimization problem:

$$\min_{h \in \mathcal{H}} o(h \mid X) := \frac{1}{N} \sum_{x \in X} [i(h(x) \mid x)] + \lambda L(h)$$

The first term $o(h \mid X)$ evaluates the average invalidity $i(h(x) \mid x)$, which is based on the cost $c(h(x) \mid x)$ and the loss $l_{01}(f(x+h(x)), +1)$ of the actions $a = h(x)$ assigned to $x \in X$. The second term $L(h)$ represents the total number of leaves (i.e., actions) in the tree h. By tuning the parameter λ, we can balance the trade-off between the effectiveness of the actions assigned by the CET h and the interpretability of h.

This CET framework can be applied to any classifier f and cost function c used in existing CE methods.

- Problem 1 was solved using **GLPK solver**

- Problem 2 was solved using **Stochastic Local Search Algorithm**

Our algorithm consists of two steps:

1. **Determine branching rules of internal nodes**: We use the stochastic local search strategy to determine the branching rules for the internal nodes of the tree.

2. **Optimize actions for each leaf**: We optimize the action assigned to each leaf by solving Problem 1 independently.

This leaf size bound theorem leads to the generation of branch rule for stochastic local search:

Theorem 1 (Leaf Size Bound): Let h be an optimal solution for Problem 2, i.e., $h = \arg\min_h \mathcal{H}_o(h, X)$. Then, we have:

$$L(h) \leq \frac{\gamma + \lambda}{\lambda}$$

- We have used stochastic local search in batch with size 10
we used many different types of classifiers to see how they behave: Logistic Regression, Random forest, LightGBM, Tabnet over different types of data like: attrition, diabetes, student etc.

Algorithm 1 Stochastic Local Search for CET.

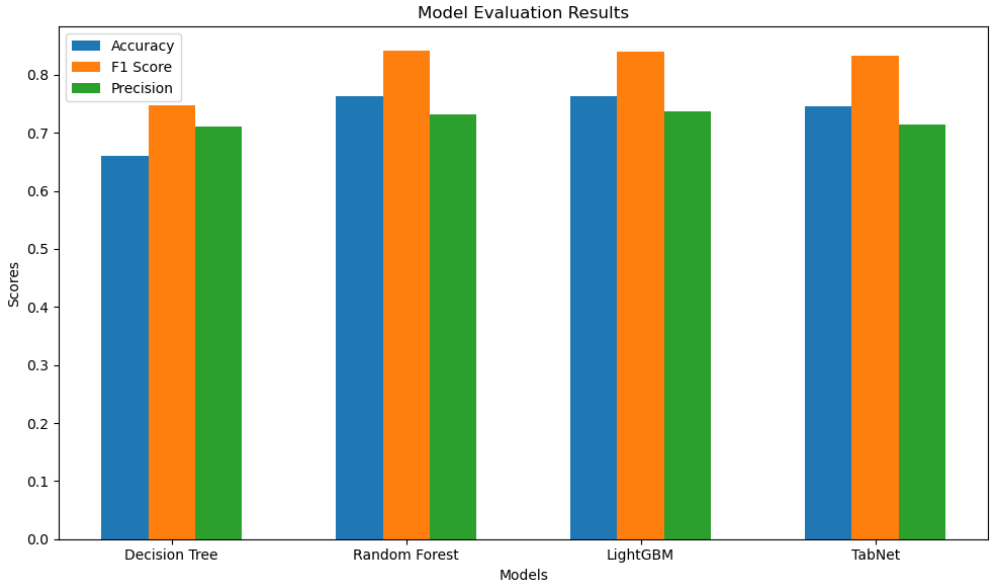
Input: set of instances X, trade-off parameters γ, λ, set of candidate branching rules $\mathcal{R}$, maximum number of iterations T, and accept condition ACCEPT.
Output: CET h^*.

$h^{(0)} \leftarrow$ generate an initial solution randomly;
$h^* \leftarrow h^{(0)}$;
for $t = 1, 2, \ldots, T$ **do**
  $\delta \sim$ random();
  if $\delta \leq 1/3$ and $|\mathcal{L}(h^{(t-1)})| < (\gamma + \lambda)/\lambda$ **then**
    $h^{(t)} \leftarrow$ insert a node with a rule $r \in \mathcal{R}$ to $h^{(t-1)}$ randomly;
  else if $\delta \leq 2/3$ **then**
    $h^{(t)} \leftarrow$ delete a node from $h^{(t-1)}$ randomly;
  else
    $h^{(t)} \leftarrow$ replace the rule of a node in $h^{(t-1)}$ with another rule $r \in \mathcal{R}$ randomly;
  end if
  for $l \in \mathcal{L}(h^{(t)})$ **do**
    $a_l^{(t)} \leftarrow \arg\min_{a \in \mathcal{A}(X_l^{(t)})} g_\gamma(a \mid X_l^{(t)})$;
  end for
  if ACCEPT$(t, h^{(t-1)}, h^{(t)})$ is False **then**
    $h^{(t)} \leftarrow h^{(t-1)}$;
  end if
  $h^* \leftarrow \arg\min_{h \in \{h^*, h^{(t)}\}} o_{\gamma,\lambda}(h \mid X)$;
end for
return h^*;

Figure 1: Stochastic Local Search Algorithm.

5 Experiments

We assessed the efficacy and interpretability of our CET through a range of numerical experiments. The code was developed in Python 3.11.4, and all experiments were performed on a 64-bit system with a 12th Gen Intel(R) Core(TM) i5-12450H 2.00 GHz processor and 16 GB of RAM.



Model	Accuracy	F1 Score	Precision
Decision Tree	0.6610	0.7468	0.7108
Random Forest	0.7627	0.8409	0.7327
LightGBM	0.7627	0.8391	0.7374
TabNet	0.7458	0.8333	0.7143

Table 1: Model Evaluation Results

Training Procedure and algorithm

- **Initialization** Our model begins by initializing with the input data. It discretizes features if required and prepares the tree structure using a dummy node.
- **Tree Generation** A decision tree is generated starting from the root node. The process involves expanding nodes by selecting appropriate features and thresholds. This step forms the basic structure that will be optimized in subsequent steps.
- **Optimization** The tree is optimized by assigning actions (counterfactuals) to the leaf nodes using the `extractAction` method. During this process, the tree structure is adjusted to minimize the total objective function, which considers the following:
  - Cost **(MPS & TLPS)**
  - Action feasibility

  - Model complexity

- **Stochastic Local Search** The `fit` method employs a Stochastic Local Search algorithm to iteratively refine the tree structure. In each iteration, the tree structure undergoes slight modifications. The objective function value of the new structure is evaluated:

  - If the new structure results in a better (lower) objective function value, it is retained.
  - Otherwise, the algorithm continues to explore other possible modifications.

- **Evaluation** Throughout the training process, the algorithm tracks key metrics such as the objective function value and model complexity to ensure the tree's effectiveness. The final optimized tree is the one that achieves the best trade-off between action feasibility and model complexity.

6 Results

6.1 Learning Hierarchical Actionable Recourse Summary

6.2 Data Summary

- **Number of positive cases:** 305
- **Number of negative cases:** 134
- **Total Bins:** 346
- **Number of data points in the train set:** 439
- **Number of used features:** 14

6.3 Parameters

- **lambda:** 0.06
- **gamma:** 1.0
- **max_iteration:** 5
- **leaf size bound:** 17

LightGBM

- **n_estimators:** 100
- **num_leaves:** 16

6.4 Training Details

- **Start training from score:** 0.822472

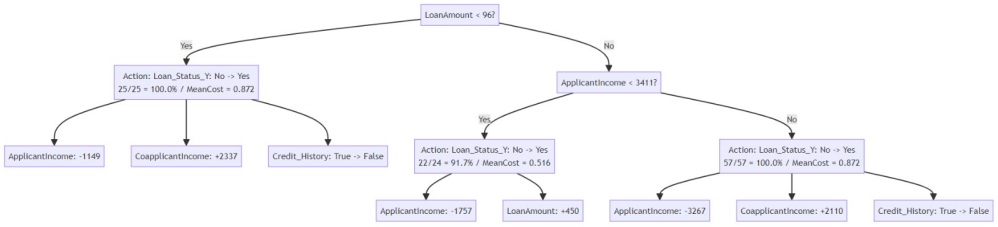


Figure 2: Decision tree using LightGBM.

6.5 Classifier: TabNet

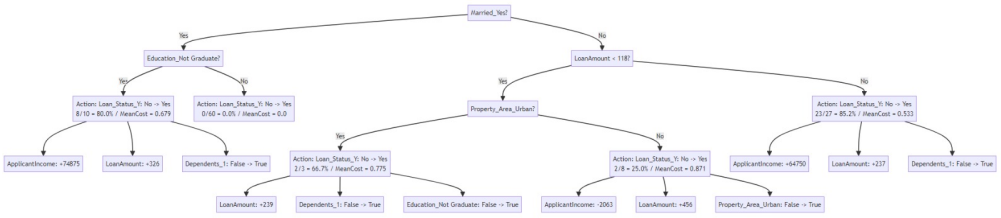


Figure 3: Decision tree using Tabnet classifier.

6.6 Comparison of Classifiers on Training Data

Table 2: Comparison of LightGBM and TabNet Classifiers on Training Data

Classifier	Cost	Loss	g(a—x)	Objective
LightGBM	0.7917	0.0189	0.8106	0.9906
TabNet	0.2822	0.6759	0.9581	1.2581

6.7 Comparison of Classifiers on Test Data

Table 3: Comparison of LightGBM and TabNet Classifiers on Test Data

Classifier	**Cost**	**Loss**	**g(a—x)**
LightGBM	0.7917	0.0189	0.8106
TabNet	0.2822	0.6759	0.9581

7 Future Work

In future work, we aim to develop a more efficient learning algorithm. Since the computational time of Algorithm in our experiments was primarily influenced by solving Problem 1 using MILO at each node, we plan to address this aspect.
Additionally, we will extend our model to handle interactions among multiple instances, as discussed by Karimi et al. (2020b). Another promising direction for future work is to incorporate existing counterfactual explanation (CE) methods that offer better solutions for multiclass problems, and to apply more effective optimization techniques for regression tasks.

8 Conclusion

We introduced the Counterfactual Explanation Tree (CET), a decision tree that suggests appropriate actions for input instances. Our CET makes sure that the process of assigning actions is clear and consistent for all inputs. To create the CET, we first developed a group-wise CE framework that assigns one action to multiple instances. Then, we proposed an algorithm to learn a CET using a stochastic local search and a pruning strategy. Our experiments showed that CET is both effective and easy to understand compared to other methods.

References

Dutta, S., Long, J., Mishra, S., Tilli, C., & Magazzeni, D. (2022). Robust Counterfactual Explanations for Tree-Based Ensembles. *Proceedings of the 39th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research*, 162:5742-5756. Available from `https://proceedings.mlr.press/v162/dutta22a.html`.

Fernández, G., Aledo, J. A., Gamez, J. A., & Puerta, J. M. (2022). Factual and Counterfactual Explanations in Fuzzy Classification Trees. *IEEE Transactions on Fuzzy Systems*, 30(12), 5484-5495. doi: 10.1109/TFUZZ.2022.3179582.

title = Attrition IBM HR Analytics Employee Performance, year = 2022, `https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset`, note = Accessed: Feb. 21, 2022

I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña, "A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence," *IEEE Access*, vol. 9, pp. 11974–12001, 2021. doi: `10.1109/ACCESS.2021.3051315`.