AAAI-25 / IAAI-25 / EAAI-25
FEBRUARY 25 – MARCH 4, 2025 | PHILADELPHIA, USA

BoolXAI: Explainable AI using Expressive Boolean Formulas

Innovative Applications of Artificial Intelligence – Deployed Tools

S. Kadıoğlu, G. Rosenberg, K. Brubaker, M. Schuetz, G. Salton, Z. Zhu, E. Zhu, S. Borujeni, H. Katzgraber

Serdar Kadıoğlu

[1] **Dept. of Computer Science, Brown University**

[2] **AI Center of Excellence, Fidelity Investments**

skadio.github.io

BROWN

BoolXAI Overview

1

Design and Functionality

Showcase of BoolXAI's high-level design and core functionality.

2

Optimization Formulation

Presentation of the underlying optimization problem and solution approach.

3

Results

Demonstration of results from public datasets to showcase BoolXAI's performance.

4

Deployed Service

Illustration of a BoolXAI-powered application deployed as an enterprise service.

The Need for Explainable AI

Increasing Complexity

Machine Learning models are becoming **increasingly complex**, making it difficult to understand and interpret their predictions.

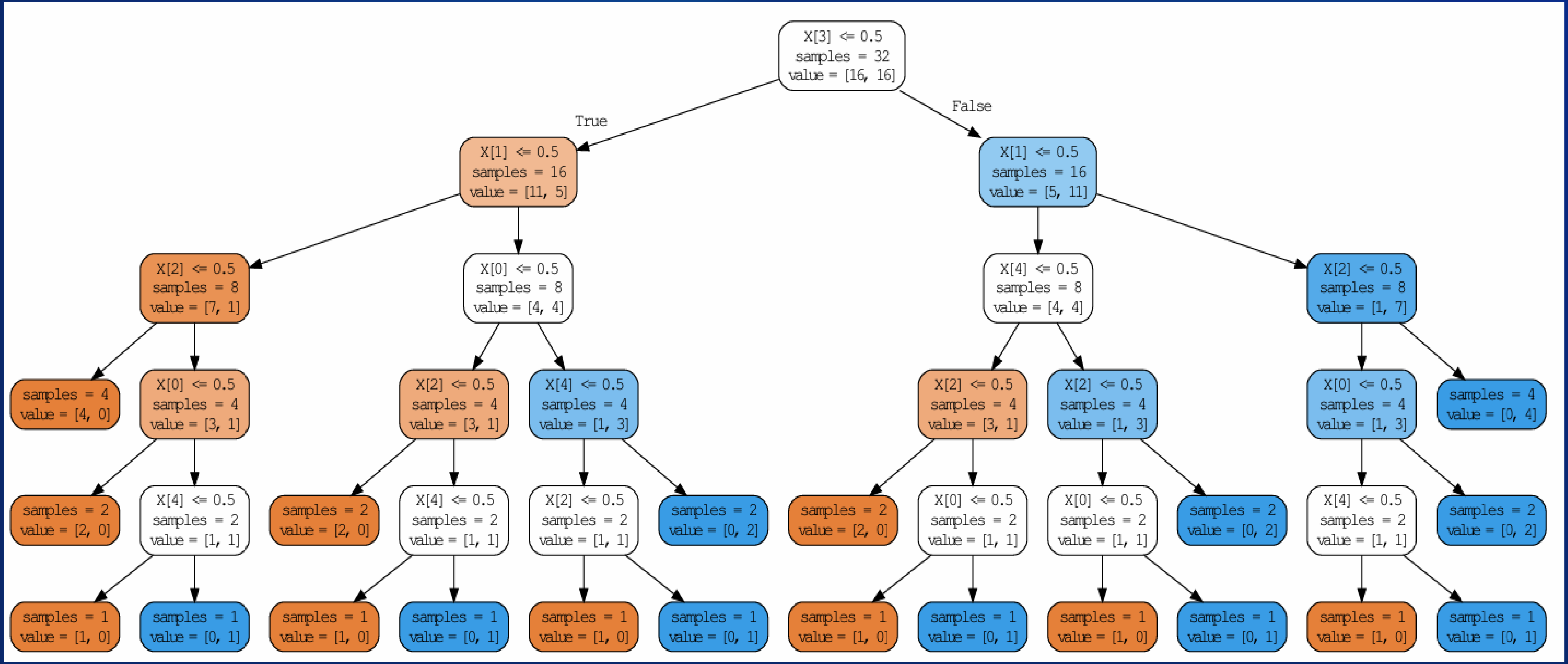
Regulatory Requirements

Explainability is mandatory in several domains such as finance and healthcare due to industry regulations.

Responsible AI

Explainable models help **discover superfluous patterns** and avoid unwanted bias in AI systems.

Limitations of Existing Models



Expressiveness of Boolean Operators

BoolXAI

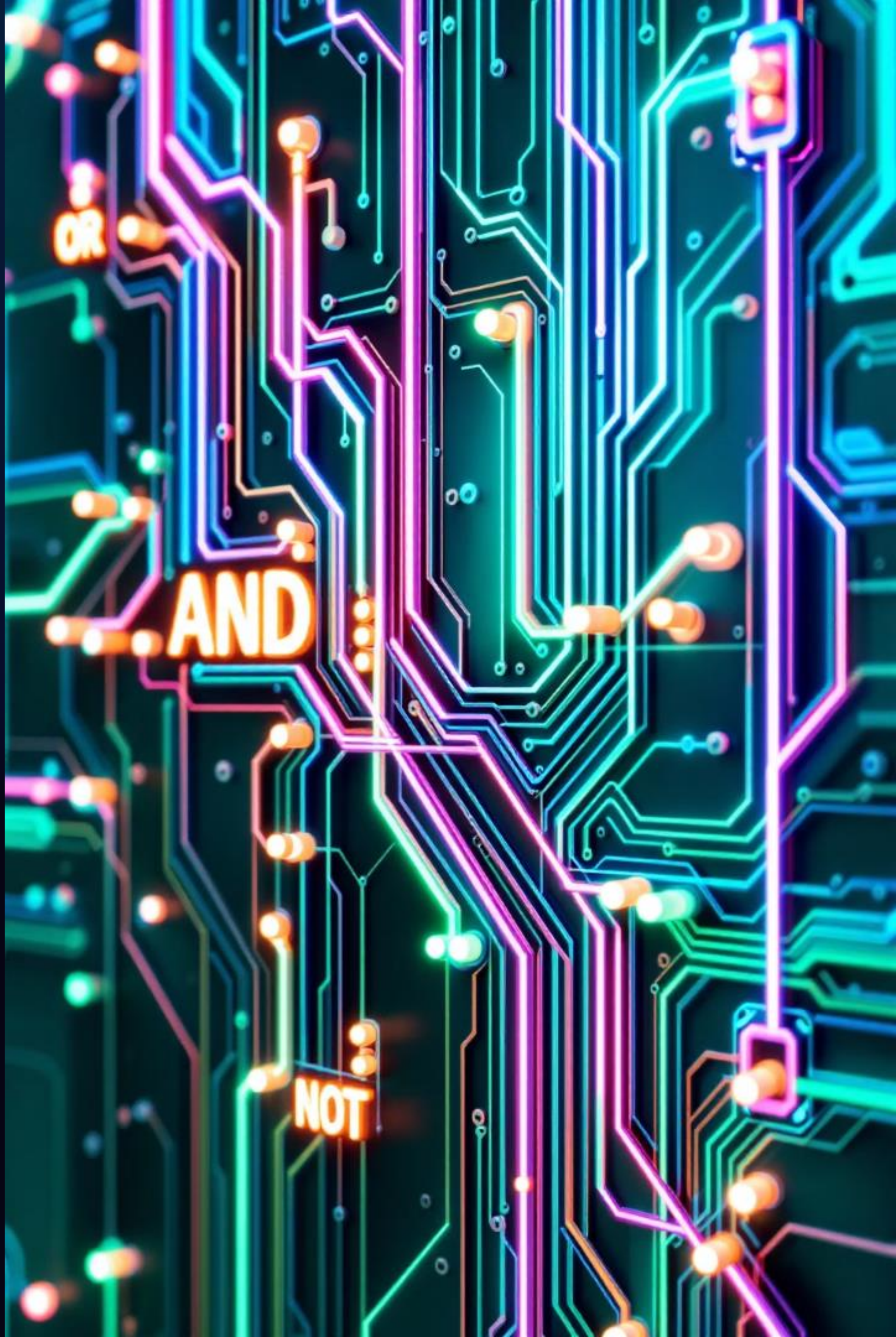
BoolXAI can represent complex conditions concisely **AtLeast3(f1,..,f5)**

Decision Trees

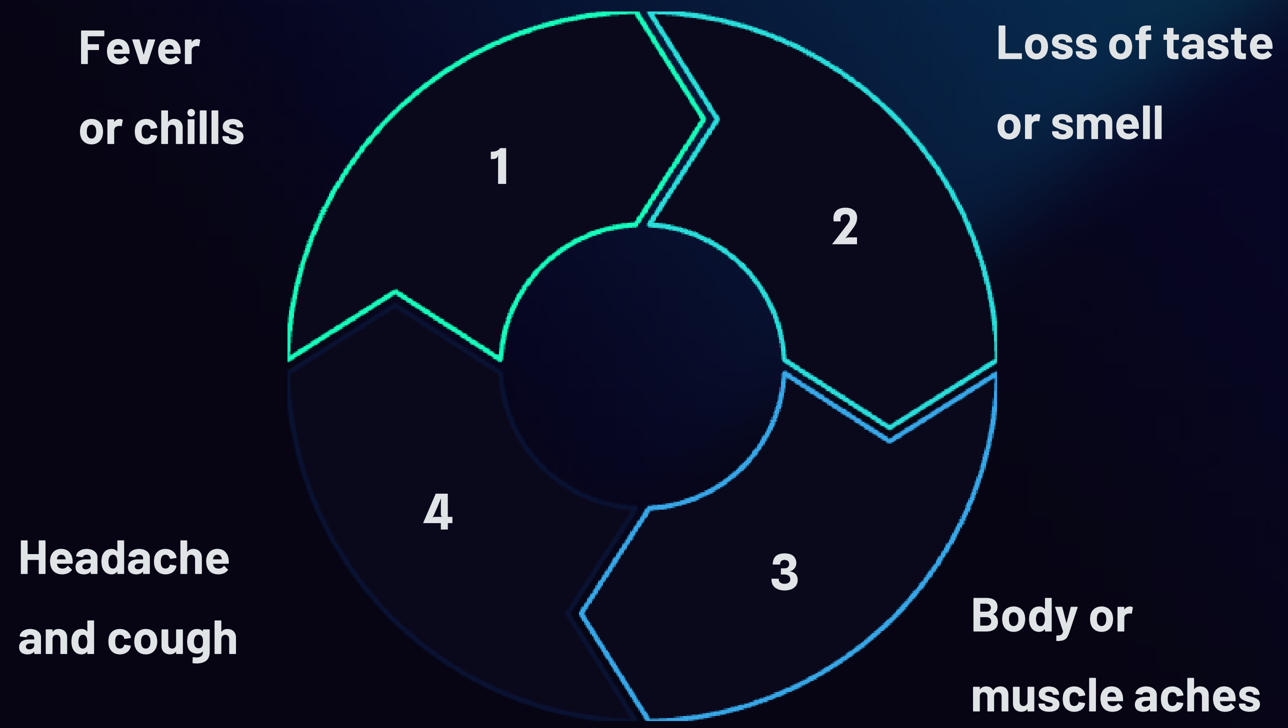
The same requires a decision tree with **19 split nodes** which is less interpretable

Propositional Logic

Using only AND/OR operators would require **13 clauses, 11 variables, 29 literals**



Motivation – Checklists

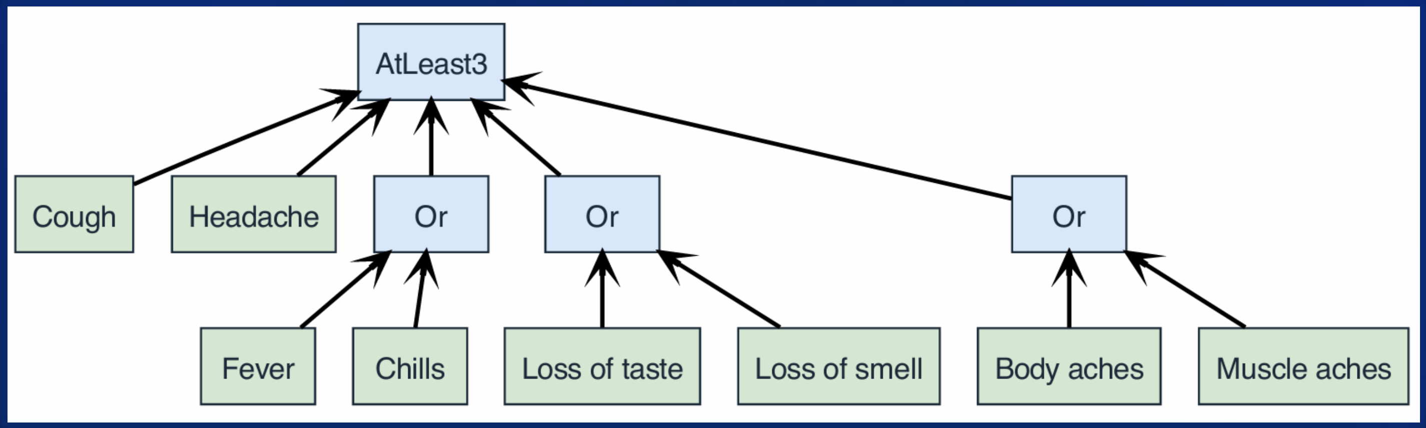


Parameterized operators are motivated by checklists, which are intuitive and widely used in various fields.

Example: a simplified disease checklist
Disease if you have at least three of these symptoms

AtLeast3 (Cough, Headache, Or(Fever, Chills), Or(Loss of taste, Loss of smell), Or(Body aches, Muscle aches))

Complex decision-making processes can be expressed in a clear, interpretable manner using Boolean logic.



Motivation – Logical Formulas in XAI

Comprehensibility

Logical formulas are **highly comprehensible**, making them ideal for explainable AI applications.

Checklist Analogy

Parameterized operators like AtLeast are motivated by checklists as used in medical symptom lists for diagnosis.

Succinct Representations

BoolXAI brings these **succinct representations** in a readily available tool for downstream applications.

Focus on Tabular Data Classification

Supervised Learning

BoolXAI focuses on supervised machine learning, specifically classification from tabular data.

Industrial Applications

This approach is particularly relevant for high-stakes industrial applications where interpretability is crucial.

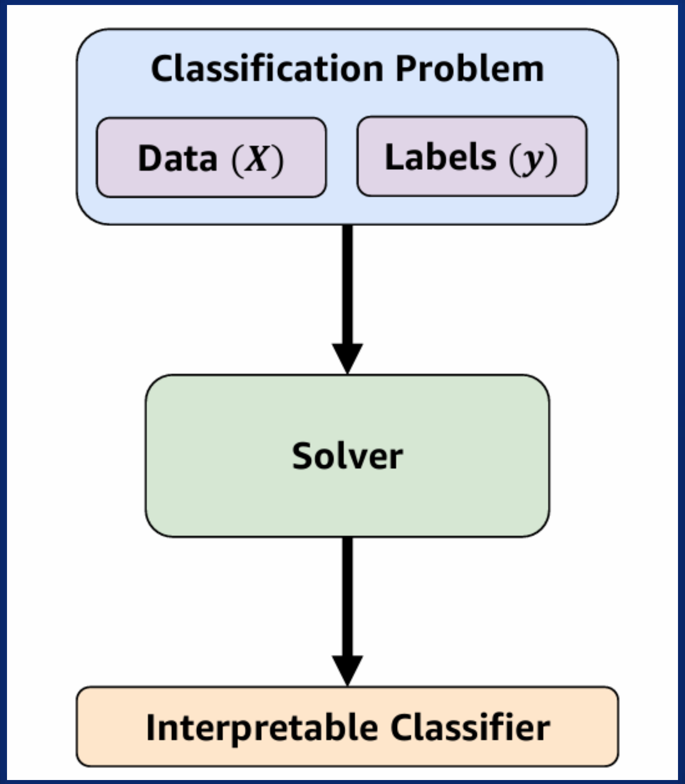
Expressive Boolean Formulas

BoolXAI uses expressive Boolean formulas to create interpretable classification models with tunable complexity.

The Challenge

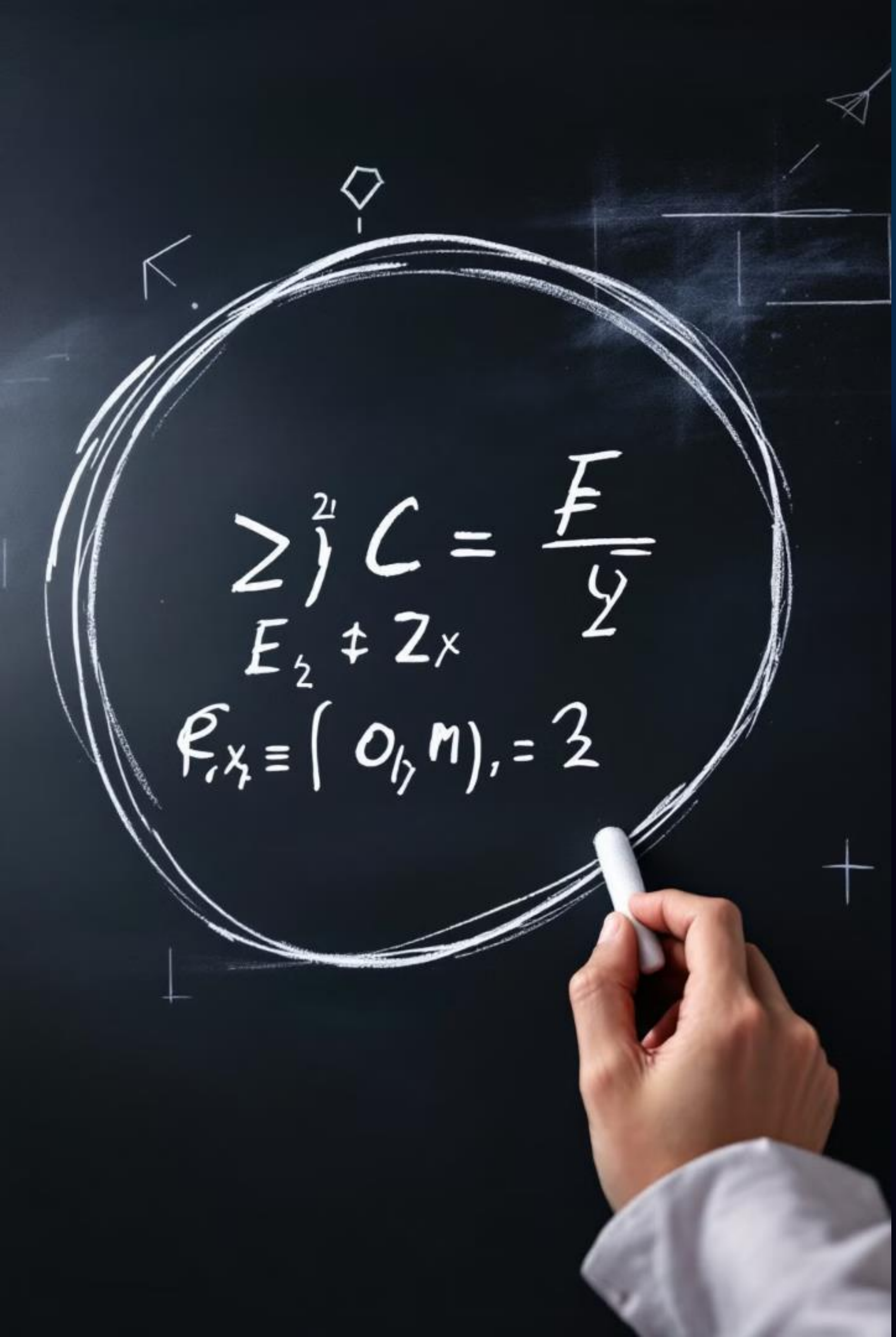
Our research focuses on developing an **interpretable classifier** using expressive Boolean formulas.

These formulas aim to provide a **balance between model complexity and interpretability**, allowing for more nuanced decision-making processes than traditional Decision Trees or CNF expressions.



The challenge lies in creating a Boolean model that can **capture complex relationships** in data while remaining **interpretable to humans**.

This involves **developing a system** that can generate and optimize Boolean formulas based on input data and desired outcomes.



Rule Optimization Problem Definition

Definition 1 (Rule Optimization Problem (ROP)) *Given a binary feature matrix $\boldsymbol{X}$ and a binary label vector $\boldsymbol{y}$, the goal of the Rule Optimization Problem (ROP) is to find the optimum rule R^* that balances the score S of the rule R on classifying the data, and the complexity of the rule C, which is given by the total number of features and operators in R, bounded by a parameter C'.*

Mathematically, our optimization problem can be stated at a high level as:

$$\begin{aligned} R^* &= \arg\max_R [S(R(\boldsymbol{X}), \boldsymbol{y}) - \lambda C(R)] \\ \text{s.t.} \quad & C(R) \leq C', \end{aligned} \tag{1}$$

Rule optimization problem is the foundation for BoolXAI's approach to finding optimal Boolean formulas for classification tasks.

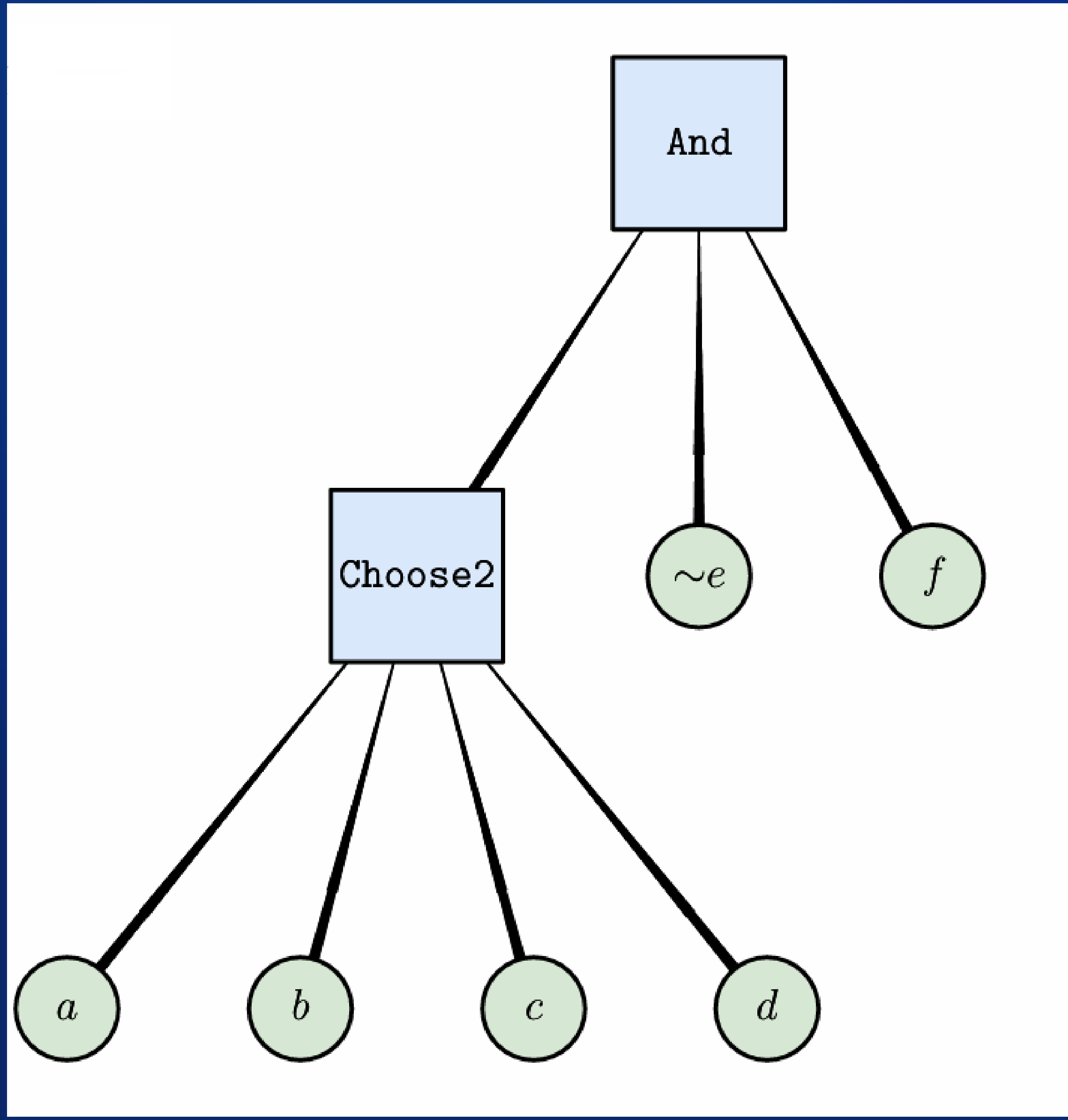
Complexity of Boolean Formula

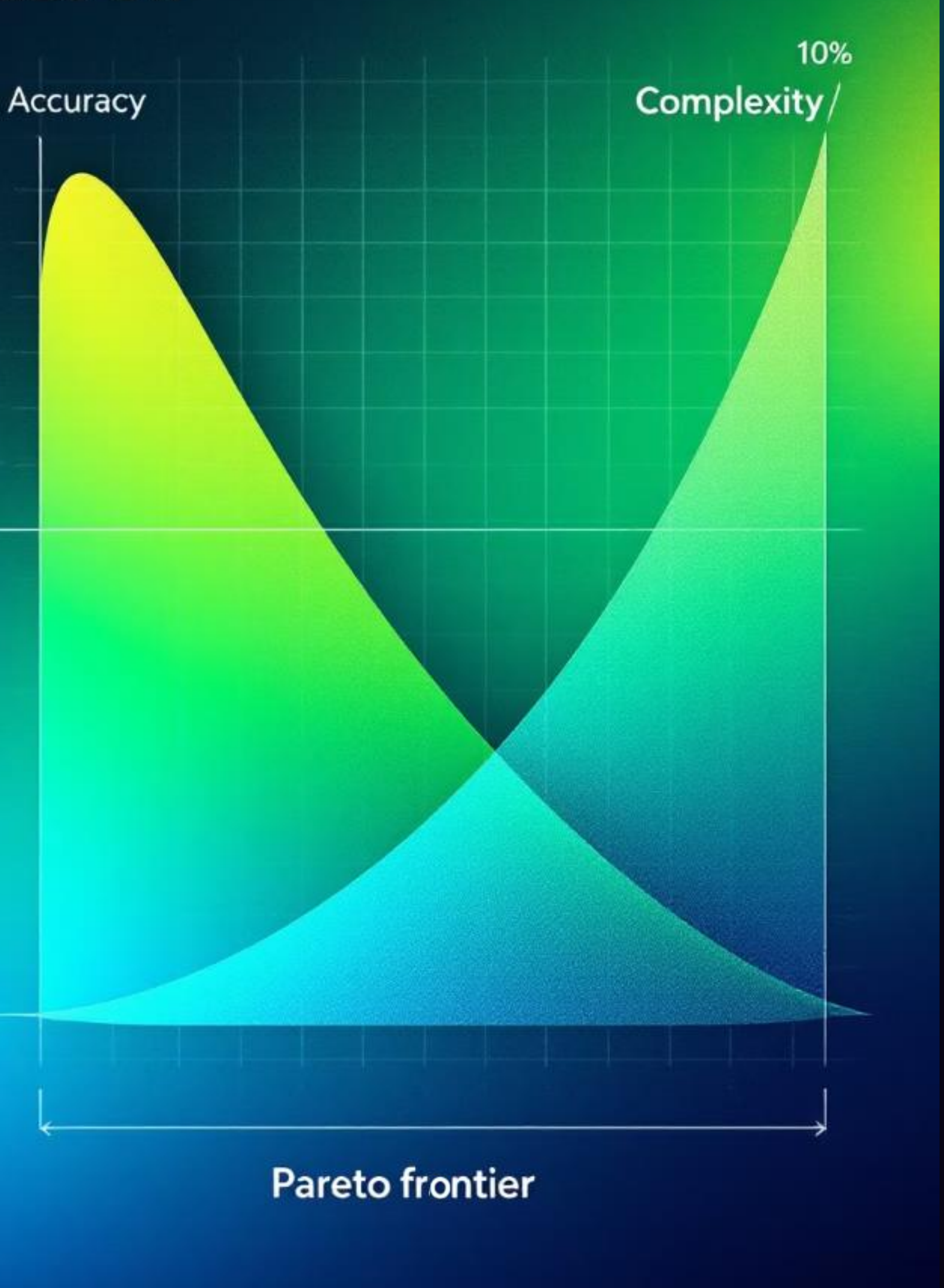
An expressive Boolean formula can be represented as a combination of operators and literals.

We define the **complexity** of a formula as the **total number of operators and literals**.

Syntax tree of **And(Choose2(a, b, c, d), ~e, f):**

- This rule contains six literals (features/columns of X) and two Boolean operators (And and Choose(k = 2)).
- It has a complexity of eight (sum of literals and operators) and a depth of two (longest path from root to leaf).





Score vs. Complexity

1. **Score: performance metric (e.g., balanced accuracy)**
2. **Complexity: total number of operators and literals**
3. **Tradeoff between score and complexity**

In our approach, we consider two key metrics: **score**, a performance metric such as accuracy, and **complexity**, total number of operators and literals in the Boolean formula. There is an **inherent tradeoff** between these two metrics.

Achieving a higher score will require sacrificing some interpretability by increasing the complexity of the formula. BoolXAI aims to find a **balance between these competing objectives**.

BoolXAI Approach

1

Optimization Problem

BoolXAI formulates the classification task as an **optimization problem** to find the smallest logical rule that satisfies the maximum number of samples.

2

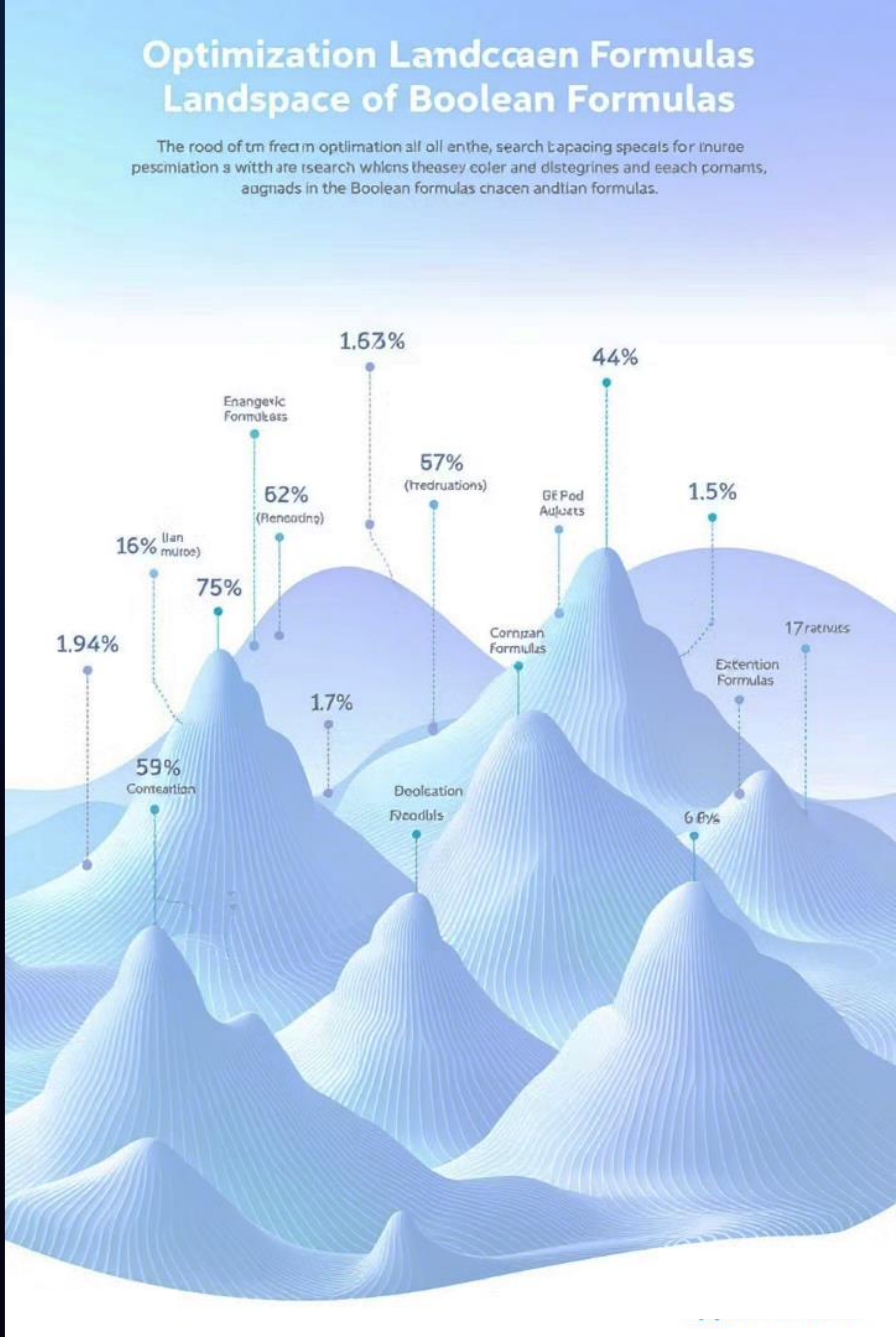
Expressive Operators

It uses expressive Boolean operators **ATLEAST ()**, **ATMOST()**, and **CHOOSE()** in addition to classical **AND/OR**

3

Native Local Optimization

The solution employs **native local optimization** to search the feasible space of all possible formulas efficiently.



Native Local Optimization

1

Native Search Space

BoolXAI optimizes directly in the **space of all valid expressive** Boolean formulas, rather than reformulating the problem in a fixed format like MaxSAT, ILP, or QUBO.

2

Stochastic Local Search

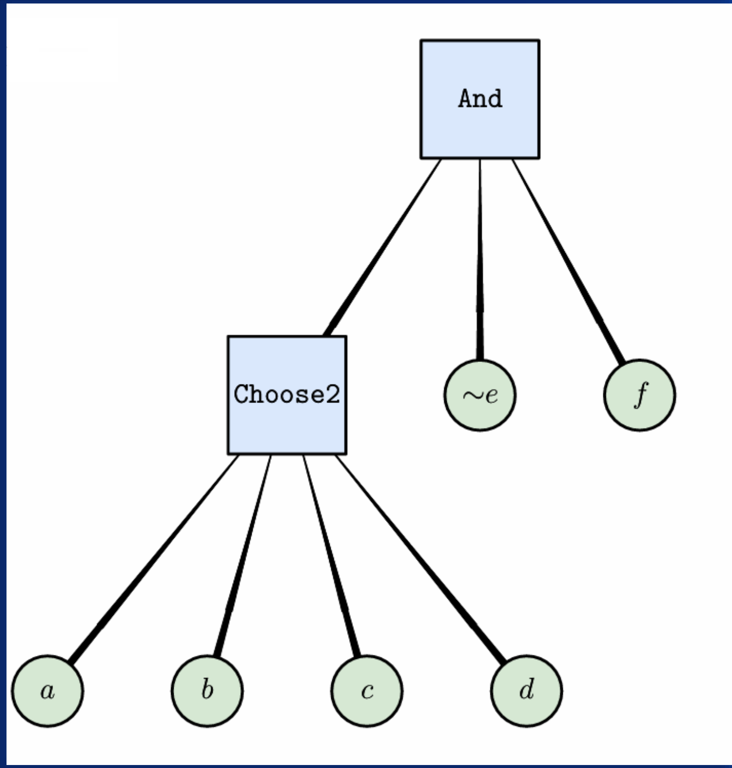
The search space is explored via a series of **(non) local moves** that make changes to the current configuration.

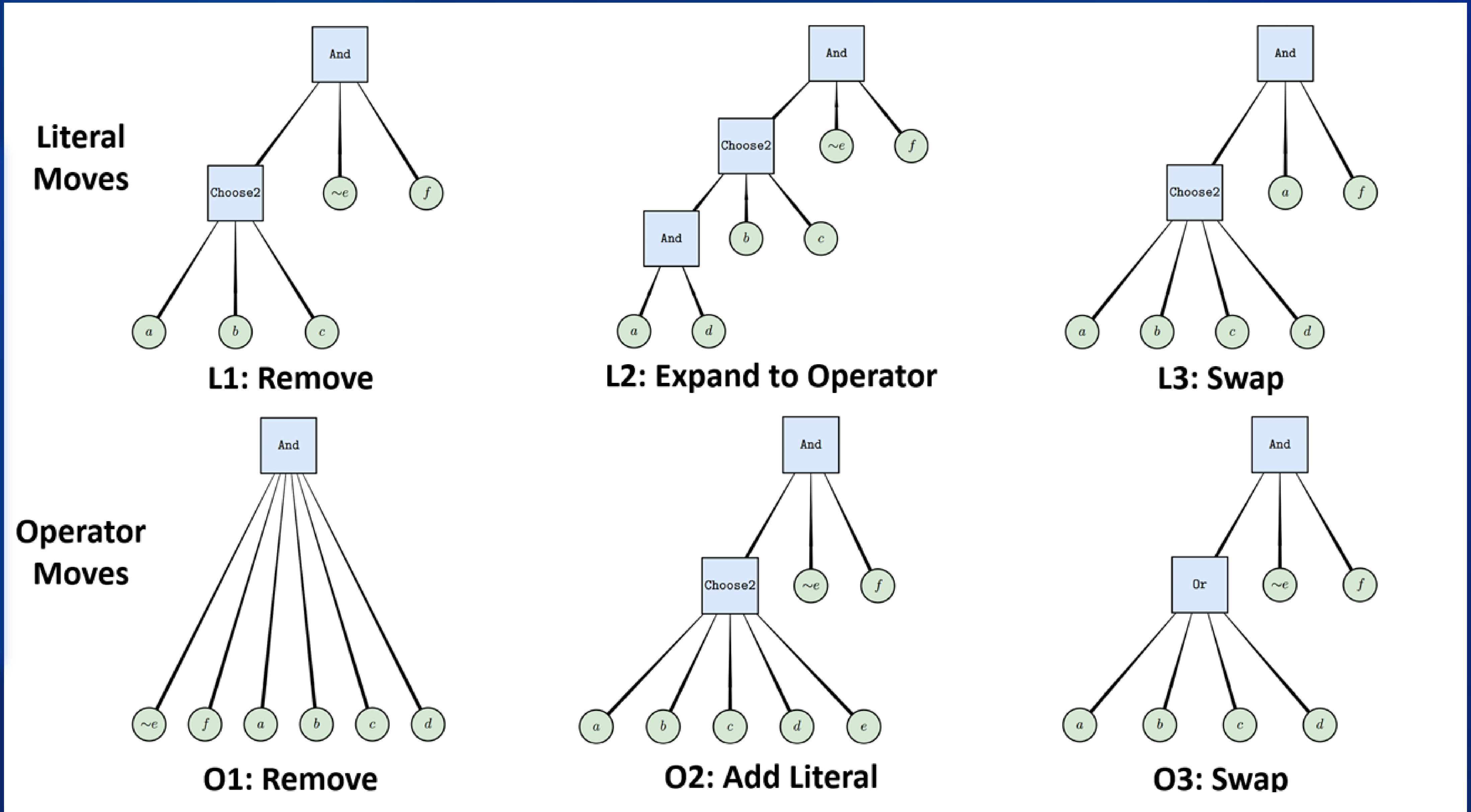
3

Multi-Start Simulated Annealing

The initial rule is constructed by randomly choosing literals and operators within the complexity constraints with **multi-started simulated annealing** process.

Local Moves in BoolXAI





Non-local Optimization

1. **"Non-local" optimization: large neighborhood search**
2. **Explores larger changes to the solution**
3. **Potential for faster improvement than local moves**

"Non-local" optimization refers to exploration of the search space via moves that change a larger part of the solution, a form of **large neighborhood search**. If we can **perform non-local moves faster** than getting the same improvement via local moves, then we'll see an **advantage**.

Hardware Acceleration

Imagine that you have access to a **hardware accelerator**, classical or quantum, that can solve ILP or **QUBO problems** extremely fast. How could you use it to potentially speed up the solver?

Hardware acceleration can be used to perform complex optimization tasks more quickly, potentially allowing for faster **exploration of the search space** or enabling the solver to consider more complex moves that would be too computationally expensive otherwise.

Collaborative Development

Academia

California Institute of Technology and **Brown University** contributed academic expertise.

Financial Technology

AI Center at Fidelity and **Fidelity Center of Applied Technology** provided industry perspective.

High-Tech Industry

Amazon Quantum Solutions and **AWS Center of Quantum Computing** offered cutting-edge technology.

BoolXAI User Base & Quick Start Example

pip install boolxai

100+

Data Scientists

BoolXAI is available to over 100 data scientists.

3000+

Downloads

Launch in Q4 2024, BoolXAI has been downloaded over 3000 times in the broader community.

```
import numpy as np
from sklearn.metrics import balanced_accuracy_score

from boolxai import BoolXAI, Operator

# Create random toy data for binary classification. X and y must be binary!
rng = np.random.default_rng(seed=42)
X = rng.choice([0, 1], size=(100, 10))
y = rng.choice([0, 1], size=100)

# Rule classifier with maximum depth, complexity, possible operators
rule_classifier = BoolXAI.RuleClassifier(max_depth=3,
                                         max_complexity=6,
                                         operators=[Operator.And, Operator.Or, Operator.Choose,
                                         random_state=42)

# Learn the best rule
rule_classifier.fit(X, y)

# Best rule and best score
best_rule = rule_classifier.best_rule_
best_score = rule_classifier.best_score_
print(f"{best_rule=} {best_score=:.2f}")
```

Illustrative Results

Dataset	Expressive BoolXAI Formula	Balanced Accuracy
Airline Customer Satisfaction	And(Inflight entertainment ≠ 5, Inflight entertainment ≠ 4, Seat comfort ≠ 0)	76%
Breast Cancer	AtMost1(worst concave ≤ 0.15, worst radius ≤ 16.43, mean texture ≤ 15.30)	95%
Direct Marketing	Or(dur > 393, employed < 5076, mon=mar)	86%
Online Shopper Intent	AtMost1(PageValues ≤ 5.55, PageValues ≤ 0, BounceRates > 0.025)	87%
Customer Churn	AtMost1(tenure > 5, Contract ≠ Month-to-month, InternetService ≠ Fiber optic)	82%

BoolXAI rules obtained by the native local solver with **max_complexity = 4** across well-known UCI ML datasets. On average, BoolXAI achieves **80% balanced accuracy** with a single Boolean formula of complexity four.

Runtime Performance

500K+

Rows

Dataset size in the case study

100+

Columns

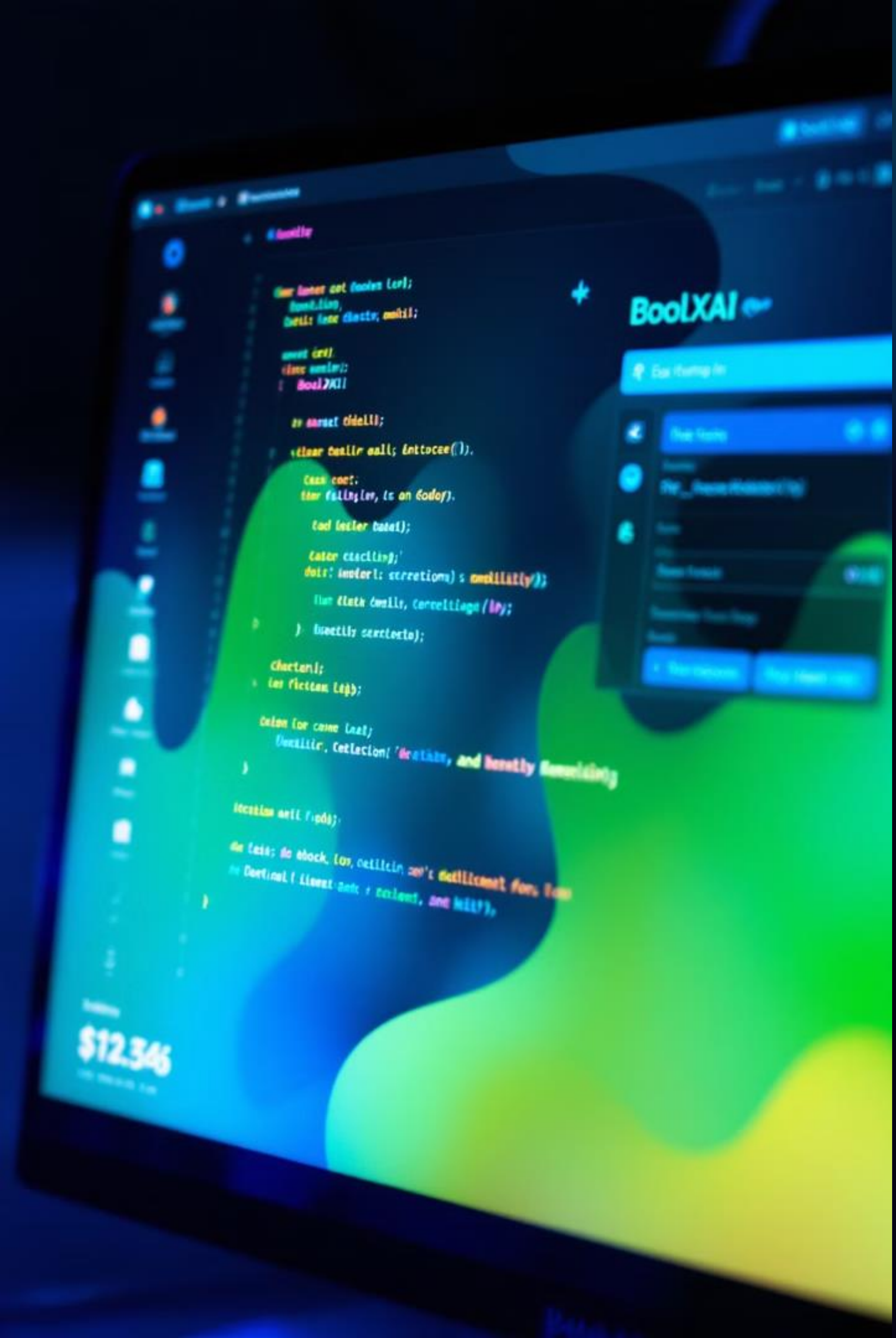
Number of features analyzed

60

Seconds

Runtime on a modern laptop

BoolXAI can process 500,000+ rows and 100+ columns in about 60 seconds on a modern laptop with an Intel i7 2.20 GHz processor using a single core. This translates to **~0.01 sec per rule optimization iteration.**



Practical Considerations & Key Features

1. **Scikit-learn**

   Used in existing pipelines.

2. **Non-Binary Data**

   Allows configuring discretization.

3. **Multi-Class & Label**

   Enables handling of inputs.

Practical Considerations & Key Features

Scikit-learn Interoperability

BoolXAI models can be used in existing pipelines and hyper-parameter tuners as scikit classifiers.

Non-Binary Data Handling

BoolXAIKBinsDiscretizer allows configuring discretization behavior for numerical features.

Multi-Class & Multi-Label

Compatibility with scikit-learn classifiers enables handling of multi-class and multi-label inputs.

Visualization Options

Users can plot the rules or directed networkx graph object for further analysis and interpretation.

Tests & Dependencies & Docs

Only depends on numpy and sklearn Native optimizer is implemented in backed without solver dependency.

Boosting, Bagging, Validation

Meta-classifier to focus on most difficult samples and bagging and cross validation to avoid overfitting.

Custom Operators

Pareto Frontier

Parallelization

User-Driven Enhancements

1

2

3

Predefined User Rules

BoolXAI now allows optimizing only part of a rule, keeping a **user-defined base** rule fixed.

Incremental Rule Generation

Users build formulas **incrementally**, starting with the single best feature and optimizing the remainder.

Controlled Complexity

Users introduce features incrementally and decide when to **stop based on complexity.**

EaSe: Explainability-as-a-Service

Interactive Web Service

EaSe is a web service powered by BoolXAI that does not require programming skills.

Data Upload

Users can **upload their input data** or provide its path on Amazon S3 for analysis.

Visualization

EaSe provides BoolXAI rules and **visualizations** for easy interpretation.

Incremental Optimization

Users can **seed assumptions or extract base rules** from previous runs for incremental re-optimization.

Related Work in XAI

Explaining Black Boxes

Post-hoc explanations for complex models, including local explanations (LIME, SHAP, MUSE) and global explanations that mimic black-box models.

Interpretable Models

Training inherently interpretable models, including decision trees, rule lists, decision sets, and scoring systems.

Combinatorial Methods

Approaches using MaxSAT, ILP, and LP for learning interpretable rules, which are closest to BoolXAI's approach.

BoolXAI Summary

1 Expressive Boolean Formulas

Logical rules with tunable complexity for classification, going beyond classical conjunction and disjunction.

2 Effective Training & Competitive Results

Native local optimization to search the space of feasible formulas efficiently. Competitive with black-box models.

3 Open-Source Library

Provides a high-level user interface for researchers and practitioners, available at https://github.com/fidelity/boolxai

pip install boolxai

skadio.github.io