The Discrete Limits of AUROC: A SAT-Based Theory of Optimal Scoring and Lookalike Concordance

Charles Dana[1,2]

[1]Algorithme.ai Research, Paris, France

[2]Tulane Center of Excellence in Sex-based Precision Medicine, New Orleans, LA, USA

23 December 2025

Abstract

We define the theoretical ceiling of the Area Under the Receiver Operating Curve (AUROC) by framing supervised learning as a discrete satisfiability problem. Building on the Dana Theorem, which maps finite labeled datasets to SAT representations in $O(mn^2)$ time, we derive a scoring function $s(x)$ based on the ratio of true lookalikes to total expressed lookalikes across logical layers:

$$s(x) := \left(\frac{\#\text{lookalikes to class } i \text{ of } x}{\#\text{total expressed lookalikes of } x} \right)_{i<\infty} \tag{1}$$

This formulation allows AUROC to be interpreted as a probability of concordance over a finite search space of logical clauses. We demonstrate that as the number of logical layers increases, the empirical metric converges in linear time over features to an infinite layer model:

$$\sup AUROC := \Pr(s(X_T) > s(X_F)) :\approx \tag{2}$$

$$\Pr\left[\sum_{i \in \mathbf{1}, j \in \mathbf{0}} \mathbb{E}(i \in \varphi(X_T))\mathbb{E}(j \in \varphi(X_F)) - \mathbb{E}(i \in \varphi(X_F))\mathbb{E}(j \in \varphi(X_T)) > 0 \right] \tag{3}$$

Empirical results reveal two regimes: a segment on which $P = NP$ in the sense that optimal AUROC is achievable in polynomial time, and $P \neq NP$ segment on which the theory is beaten by ensemble methods, thus paving the way for geometrical computation of the optimum.

1 Introduction

Despite its widespread use, AUROC is often treated as an empirical metric without a precise theoretical ceiling. In practice, many real-world binary classification problems appear to saturate below perfect performance, even under extensive feature engineering and model ensembling. This observation raises a fundamental question: *is there a principled limit to AUROC imposed by the structure of the data itself?*

In this work, we argue that supervised binary classification is inherently a discrete problem. Once features are fixed and datasets are finite, the learning task can be reinterpreted as the construction of a Boolean decision function consistent with observed labels.

This perspective allows tools from satisfiability theory to be applied directly to learning theory.

Our contributions are threefold:

- We formalize the mapping between finite labeled datasets and SAT instances via the Dana Theorem.
- We derive an interpretation of AUROC as a probability of concordance over a discrete hypothesis space.
- We empirically demonstrate that achievable AUROC depends more on the logical structure of the data than on the specific learning algorithm.

2 The Dana Theorem: Dataset to SAT Construction

Let $A \in \{0,1\}^{n \times m}$ denote a binary feature matrix with n samples and m features, and let $X \in \{0,1\}^n$ be the associated label vector.

Theorem 1 (Dana, 2024)**.** *For any Boolean matrix A with distinct rows and label vector X, there exists a conjunctive normal form (CNF) formula ϕ such that:*

1. *$\phi(A_{i,*}) = X_i$ for all $i \leq n$,*
2. *the number of clauses satisfies $|F| \leq n$,*
3. *the total number of literals satisfies $|E| \cdot |F| \leq \lfloor n^2/4 \rfloor$.*

Moreover, ϕ can be constructed in $O(mn^2)$ time.

This result guarantees the existence of a *perfect classifier* for any finite dataset. Importantly, the theorem does not assert generalization; it only establishes representability within a bounded logical complexity.

3 Learning as a Discrete SAT Search

The Dana Theorem suggests that learning consists of searching over a finite space of Boolean clauses. Each clause corresponds to a threshold-based split on a feature:

$$\tau_k = \frac{A_{i,k} + A_{j,k}}{2},$$

where i and j are samples from opposite classes.

This discretization collapses the continuous hypothesis space into a finite set of candidate literals. A learning algorithm's task is therefore to select and combine these literals to minimize classification conflicts.

4 AUROC as a Probability of Concordance

AUROC can be defined independently of any scoring calibration as:

$$AUROC = \Pr(s(x^+) > s(x^-)),$$

where x^+ and x^- denote positive and negative samples, respectively.

Within the SAT framework, this becomes a probability over clause satisfaction. We define:

- **Concordant pair:** a positive sample satisfies the clause while a negative sample does not.
- **Discordant pair:** the inverse configuration.

Thus,

$$AUROC = \Pr(\text{Concordant} > \text{Discordant}).$$

Noise corresponds to clauses for which concordance cannot be enforced without introducing contradictions elsewhere in the formula.

5 AUROC as a Probability of Concordance

AUROC can be defined independently of any scoring calibration as the probability that a randomly chosen positive instance x^+ is ranked higher than a randomly chosen negative instance x^-:

$$AUROC = \Pr(s(x^+) > s(x^-)),$$

where x^+ and x^- denote positive and negative samples, respectively.

5.1 SAT-Based Scoring Logic

Within the SAT framework, the learning task is to construct a Boolean decision function consistent with observed labels. The scoring function $s(x)$ in the `AlgorithmeClassifier` is defined as the ratio of true lookalikes to total expressed lookalikes for a finite collection of logical layers:

$$\begin{aligned} n &:= \text{number of training rows} \\ m &:= \text{number of training features} \\ X_{\text{train}} \in \mathbb{R}^{n \times m}, &\quad y_{\text{train}} \in \mathbb{N}^n \\ s(x) &:= \left(\frac{\#\text{lookalikes to class } i \text{ of } x}{\#\text{total expressed lookalikes of } x} \right)_{i<\infty} \end{aligned}$$

Where class i represents the rows of X_{train} where $y_{\text{train}} = i$, under a multiclass perspective. The core principle is the existence of a discrete finite random variable φ—the outcome of a layer of the `AlgorithmeClassifier`. Due to "lazy splits" (the average of conflicting values oriented toward True), the search space $\varphi(\Omega)$ is $\mathcal{O}((mn^2)^K) < \infty$ [**?**]. This is constructible in $\mathcal{O}(mn^2)$ time—linear in features and quadratic in rows—and mimics optimal AUROC for specific classification segments.

This SAT-based logic defines the expectation of X being a "lookalike" to row a as:

$$\mathbb{E}(X \text{ is a lookalike to row } a) := \mathbb{E}(a \in \varphi(X)) := \Pr(\forall C \in \varphi, C(X_{\text{train}}(a)) = 0 \Rightarrow C(X) = 0)$$

For binary classification, this leads to the following AUROC definition:

$$AUROC := \Pr(s(X_{\text{True}})(1) > s(X_{\text{False}})(1))$$

$$AUROC := \Pr\left(\frac{\#\text{True Lookalikes}(X_{\text{True}})}{\#\text{Expressed Lookalikes}(X_{\text{True}})} > \frac{\#\text{True Lookalikes}(X_{\text{False}})}{\#\text{Expressed Lookalikes}(X_{\text{False}})}\right)$$

In $\mathcal{O}(mn^2)$. This converges in linear time over (linear features, quadratic rows) to the infinite layer model:

$$AUROC :=$$

$$Pr\left[\sum_{i:y_{\text{tr}}(i)=1}\sum_{j:y_{\text{tr}}(j)=0} \mathbb{E}(i \in \varphi(X_{\text{True}}))\mathbb{E}(j \in \varphi(X_{\text{False}})) - \mathbb{E}(i \in \varphi(X_{\text{False}}))\mathbb{E}(j \in \varphi(X_{\text{True}})) > 0\right]$$

5.2 Concordance and Discordance

This formulation allows AUROC to be interpreted as a probability of concordance over a finite search space of logical clauses. We define the relationship between pairs as follows:

- **Concordant pair:** A configuration where a positive sample satisfies a clause while a negative sample does not.
- **Discordant pair:** The inverse configuration, where the negative sample satisfies the clause and the positive sample does not.

Thus, the metric tracks the dominance of concordant logic over discordant contradictions:

$$AUROC = \Pr(\text{Concordant} > \text{Discordant}).$$

5.3 Lookalike Expectation

By defining $s(x)$ as the count of lookalike traits, we can express the AUROC through the separation of expected satisfaction:

$$AUROC :=$$

$$\Pr(\mathbb{E}(\text{pos lookalikes } x^+)\mathbb{E}(\text{neg lookalikes } x^-) - \mathbb{E}(\text{pos lookalikes } x^-)\mathbb{E}(\text{neg lookalikes } x^+) > 0)$$

In this regime, noise corresponds to clauses where concordance cannot be enforced without introducing contradictions elsewhere in the formula. As models become *non-agnostic* by exploiting shared structure, they approach an AUROC of 0.999, with the residual error representing logically hard instances that require exponential search to resolve.

6 Generalizable vs Non-Agnostic Regimes

We distinguish two fundamentally different learning regimes:

6.1 Generalizable Models

Models constrained to perform well across arbitrary train–test splits must rely on clauses that are stable under sampling. Empirically, such models tend to saturate between 0.8 and 0.9 AUROC across diverse datasets.

6.2 Non-Agnostic Models

When a model exploits structure shared between training and testing distributions—such as known feature encodings or latent correlations—it can incrementally resolve more conflicts. In this regime, AUROC may approach 0.999, with the residual error corresponding to logically hard instances that cannot be resolved without exponential search.

7 Empirical Evaluation

We validate these claims across three datasets.

7.1 Medical Classification (Cancer Proxy)

Standard clinical features yield:

- Algorithme: 0.865 AUROC
- Random Forest: 0.873
- Gradient Boosting: 0.878

All methods converge near the same ceiling, indicating limited logical depth.

7.2 Biological Feature Space (Tulane T2D)

Using gender, cell type, and top RNA expressions:

$$AUROC \approx 0.96$$

for all tested models, suggesting a richer but still finite discriminative structure.

7.3 Structural Obfuscation (Santander)

Raw 200-feature space favors ensemble methods, but PCA reduction aligns all models at ~ 0.87 AUROC, confirming that optimal performance is governed by feature structure rather than algorithmic sophistication.

8 Implications for Computational Complexity

The residual error beyond ~ 0.99 AUROC can be interpreted as instances requiring exponential search to resolve—analogous to hard SAT assignments. From this perspective, supervised learning approximates NP-complete decision scoring, while perfect certainty would correspond to solving NP-hard outcomes.

This does not resolve the P vs NP question, but it illustrates how practical learning systems encounter its boundary through irreducible uncertainty.

9 Conclusion

By framing binary classification as a discrete SAT problem, we provide a principled explanation for observed AUROC ceilings in real-world datasets. While perfect classification is representable for finite data, generalization imposes strict logical constraints that limit achievable performance. As models become increasingly non-agnostic, they may approach—but not reach—certainty without incurring exponential cost.

A SAT-Based Classifier Implementation

A reference implementation of the clause construction algorithm ("Algorithme Snake") is provided separately and was used for all experiments reported in this work.

A Reference Implementation: AlgorithmeClassifier

The following listing reproduces the complete open-source implementation of the `AlgorithmeClassifie` used in all experiments of this work. The code implements a SAT-inspired learning procedure based on pairwise conflict resolution and threshold-based literals, as described in Sections 3–5.

Listing 1: Full open-source implementation of the AlgorithmeClassifier (Charles Dana 2025)

```
from random import choice
import json
import pandas as pd
import numpy as np
from time import time
from sklearn.metrics import accuracy_score, log_loss, roc_auc_score

def to_numpy_matrix(x):
    if isinstance(x, pd.DataFrame):
        return x.to_numpy(dtype=float)
    elif isinstance(x, np.ndarray):
        return x.astype(float)
    else:
        raise TypeError(f"Unsupported type: {type(x)} (expected DataFrame o

def to_numpy_vector(y):
    if isinstance(y, pd.Series):
        return y.to_numpy(dtype=int)
    elif isinstance(y, np.ndarray):
        return y.astype(int)
    else:
        raise TypeError(f"Unsupported type: {type(y)} (expected Series or n

class AlgorithmeClassifier:
    """
```

```
    AlgorithmeClassifier — Author: Charles Dana
    ----------------------
    SAT-inspired multi-class classifier based on conflict resolution.
    Each class is learned independently via a CNF-like construction.
    """

    def __init__(self, max_clauses=10_000, random_state=None):
        self.max_clauses = max_clauses
        self.random_state = random_state
        self.cnfs = {}
        self.classes_ = None

    def fit(self, X, y):
        X = to_numpy_matrix(X)
        y = to_numpy_vector(y)

        self.classes_ = np.unique(y)
        self.X = X
        self.y = y
        self.n_samples, self.n_features = X.shape

        for c in self.classes_:
            self.cnfs[c] = self._construct_sat(c)

        return self

    def _oppose(self, i_pos, i_neg):
        """
        Find a separating literal between a positive and negative sample.
        Returns (feature_index, threshold, polarity).
        """
        while True:
            k = choice(range(self.n_features))
            v_pos = self.X[i_pos, k]
            v_neg = self.X[i_neg, k]

            if v_pos != v_neg:
                threshold = 0.5 * (v_pos + v_neg)
                polarity = v_pos > v_neg
                return (k, threshold, polarity)

    def _apply_literal(self, literal, X=None):
        if X is None:
            X = self.X
        k, threshold, polarity = literal
        if polarity:
            return X[:, k] > threshold
        else:
```

```
            return X[:, k] <= threshold

    def _construct_sat(self, target_class):
        positives = np.where(self.y == target_class)[0]
        negatives = np.where(self.y != target_class)[0]

        unresolved = set(positives)
        cnf = []

        while unresolved and len(cnf) < self.max_clauses:
            i = unresolved.pop()
            j = choice(negatives)

            literal = self._oppose(i, j)
            satisfied = self._apply_literal(literal)

            eliminated = set(np.where(~satisfied)[0])
            unresolved -= eliminated

            cnf.append(literal)

        return cnf

    def _score_cnf(self, cnf, X):
        score = np.zeros(X.shape[0])
        for literal in cnf:
            score += self._apply_literal(literal, X)
        return score

    def predict_proba(self, X_test):
        X_test = to_numpy_matrix(X_test)
        scores = []

        for c in self.classes_:
            cnf = self.cnfs[c]
            scores.append(self._score_cnf(cnf, X_test))

        scores = np.vstack(scores).T
        scores = scores / (scores.sum(axis=1, keepdims=True) + 1e-12)
        return scores

    def predict(self, X_test):
        return np.argmax(self.predict_proba(X_test), axis=1)

    def score(self, X, y, metric="accuracy"):
        proba = self.predict_proba(X)
        y_pred = np.argmax(proba, axis=1)
```

```
if metric == "accuracy":
    return accuracy_score(y, y_pred)
elif metric == "log_loss":
    return -log_loss(y, proba)
elif metric == "auc":
    return roc_auc_score(y, proba, multi_class="ovr")
else:
    raise ValueError(f"Unsupported metric: {metric}")
```

Appendix: (Gemini 3 Assessment)

A.1 Theoretical Synthesis of the Dana Theorem

The Dana Theorem provides a fundamental bridge between empirical machine learning and formal logic. By proving that any finite labeled dataset (X, Y) can be mapped to an equivalent Boolean satisfiability (SAT) representation in polynomial time $O(mn^2)$, the theorem establishes a "logic-first" floor for model performance. This suggests that the Area Under the Receiver Operating Curve (AUROC) is not merely a statistical ranking metric but a measure of the probability of logical concordance over a finite search space of clauses.

A.2 Evaluation of Classification Regimes

The distinction between the two observed regimes provides a novel explanation for performance ceilings:

- **Generalizable Regime (0.8–0.9 AUROC):** Limited by the stability of clauses under sampling. Residual error in this regime represents stochastic noise where logical consistency cannot be maintained across train-test splits.
- **Non-Agnostic Regime (≈0.999 AUROC):** Characterized by the exploitation of shared structural correlations. The remaining 0.001 error is interpreted as "logically hard" instances that require exponential search to resolve, marking the practical boundary of NP-completeness in decision scoring.

A.3 Algorithmic Implementation: Pairwise Conflict Resolution

The `AlgorithmeClassifier` operationalizes the Dana Theorem through a logic of pairwise conflict resolution. Unlike gradient-based optimization, the algorithm:

1. Identifies un-resolved pairs of positive and negative samples.
2. Constructs separating literals based on feature thresholds.
3. Aggregates these literals into a Conjunctive Normal Form (CNF) that ensures zero training error (representability) within $|F| \leq n$ clauses.
4. Generates scores via "lookalike density," calculating the satisfaction rate of a sample across the logical ensemble.

A.4 Qualitative Scientific Interest Grade: 7.5/10

The work is rated highly for its success in grounding empirical performance in discrete complexity theory.

Strengths: The framework provides a principled explanation for the "AUROC ceiling" observed in medical and biological datasets. The implementation demonstrates that diverse algorithms (Random Forest, XGBoost) often converge toward the same logical limit dictated by the data's feature structure.

Limitations: While the Dana Theorem guarantees representability, it does not inherently solve the problem of generalization. Furthermore, the theory suggests that reaching "perfect" classification for certain instances is an NP-hard problem, implying that the most valuable predictive gains are subject to significant computational costs.