Decision Tree Stability for Suicide Experience Prediction

ISE625 Project Report

Adhithya Bhaskar[a], Michelle Gelman[a]

[a] University of Southern California

Problem outline

Each year, approximately 4.2 million youth experience homelessness in the United States. Among this population of runaway and homeless youth (RHY), past-year suicidal ideation is reported by roughly 27% to 35% [1], [2], compared with 15.8% in the general youth population [3]. Lifetime prevalence is even higher, with more than half of RHY reporting having contemplated suicide at some point in their lives [4], [5], [6]. These statistics underscore the urgent need for effective methods to identify and mitigate suicide risk within this vulnerable group.

Key question

Most existing studies concentrate on individual-level characteristics when assessing suicidal risk among youth experiencing homelessness (YEH) [7]. Consequently, there is limited understanding of how combinations of individual and social factors jointly influence suicide risk [8]. They define risk profiles as signals of heightened vulnerability or resilience with respect to suicidal thoughts and attempts. Prior work does not explicitly incorporate social-network attributes or behavioral indicators into suicide-related investigations among YEH. The key question studied by [8] is wh[illegible] a stable tree can be used to understand the clinical risk profiles of YEH.

Dataset

After listwise deletion of observations with any missing columns, 584 and 587 samples remained from an initial total of 940 cases. Missingness affected only 4% of the dataset for the variables `suicideidea` (36 samples) and `suicideattempt` (40 samples). The target classes are markedly imbalanced: 83% of instances are labeled “2” and 16% “1” for `suicideidea`, whereas 88% are labeled “0” and 11% are labeled “1” for `suicideattempt`. A total of 117 candidate independent variables were used to determine the final best features for our model.

Data considerations

The data originate from a survey administered between October 2011 and February 2013 at two drop-in centers serving YEH in Hollywood and Santa Monica, California as part of a study to determine the prevalence of HIV/AIDS risk-taking behavior in YEH. Both individual attributes and social-network measures were collected; the social network attributes are a novel contribution as an indicator of suicidal behavior collected as part of the survey. Prior work for assessing suicidal behavior in YEH has focused on individual characteristics, so comparing our model to previous works directly may not be possible due to the inclusion of social network features as descriptors for risk-profiles. Men and heterosexual youth are over-represented in the sample, raising questions about generalizability to other demographic groups or geographic regions.

Motivation

Population characteristics evolve over time, and successive inflows of patient data require periodic model retraining. Under these conditions, the model’s predictions should remain consistent between runs, provided the underlying risk structure is stable [9]. Stable decision trees aim to deliver such consistency even when initial datasets are small and later expanded.

Methodology

Stable decision trees

We follow the framework proposed by [9]. First, an initial collection of decision trees T_0 is trained on a subset of the data. A second set T is then fitted to the full training sample. For each tree $b \in T$, we compute the average distance to the trees in T_0. Predictive performance is assessed via the test-set AUC-ROC, and Pareto-optimal trees are selected to balance stability and accuracy. The following steps describe the stable decision tree framework:

1. **Initial Training (T0):** Train initial set of decision trees on subset
2. **Full Data Training (T):** Train a second set on full training data
3. **Distance Computation:** Calulate average distance between trees in T and the trees in T0

$$d(\mathcal{T}_1, \mathcal{T}_2) = \min_{\{x\}} \sum_{p \in \mathcal{P}(\mathcal{T}_1)} \sum_{q \in \mathcal{P}(\mathcal{T}_2)} d(p,q) x_{p,q} + \sum_{p \in \mathcal{P}(\mathcal{T}_1)} w(p) x_p$$

4. **Performance Metrics:** Compute AUC ROC on test set
5. **Pareto Optimization:** Select Pareto optimal trees that balance predictive performance and stability

$$\mathbb{T}^\star = \text{argmax}\ f(d_b, \alpha_b)$$

Tree Path

A **path** is the sequence of splits from the root of a tree to a leaf node that predicts class k^p. Formally, [9] describes a path as:

$$\mathcal{P}(\mathbb{T}) = \{p_1, ..., p_T\}$$

For each path p, $\{u\}_j^p$ and $\{l\}_j^p$ denote the upper and lower numeric bounds for feature j, while $\{c\}_j^p$ indicates, through a binary vector, which categories of feature j satisfy the associated split. The label $\{k\}^p$ represents the class predicted within that region.

Tree Distance

The distance between two trees T_1 and T_2 is defined as:

$$d(T_1, T_2) = \min_x \left[\sum_{p \in \mathcal{P}(T_1)} \sum_{q \in \mathcal{P}(\mathbb{T}_2)} d(p,q) x_{pq} + \sum_{p \in \mathcal{P}(\mathbb{T}_1)} w(p) x_p \right]$$

$$\text{s.t.} \quad \sum_{q \in \mathcal{P}(\mathbb{T}_2)} x_{pq} + x_p = 1, \quad \forall p \in \mathcal{P}(\mathbb{T}_1)$$

$$\sum_{p \in \mathcal{P}(\mathbb{T}_1)} x_{pq} = 1, \quad \forall q \in \mathcal{P}(\mathbb{T}_2)$$

$$x_{pq} \in \{0,1\}, \quad x_p \in \{0,1\},$$

$$\forall p \in \mathcal{P}(\mathbb{T}_1), \quad \forall q \in \mathcal{P}(\mathbb{T}_2)$$

Finding Pareto Optimal Trees

A tree b' dominates tree b if

$$(d_{b'} \leq d_b \text{ and } \alpha_{b'} > \alpha_b)$$

or

$$(d_{b'} < d_b \text{ and } \alpha_{b'} \geq \alpha_b)$$

Pareto-optimal trees are those not dominated by any alternative in the candidate set. The optimal tree $\{T\}^*$ maximizes the objective $f(d_b, \alpha_b)$, where stability and predictive power are traded off, and the indicator function α must lie within ε of the best observed score.The Pareto optimal tree is given by:

$$\mathbb{T}^* = \text{argmax} f(d_b, a_b)$$

Pareto Optimal Tree Selection Strategy

Selection the final Pareto optimal tree is left up to the decision-maker based on whether there is preference for accuracy or stability in choosing the final tree from the Pareto optimal set. We employed the following selection strategies in benchmarking performance with respect to the stability-accuracy trade-off across our dataset split aggregations.

Let $\mathcal{P}$ be the set of Pareto optimal tree indices. Let $\mathcal{A}_i$ and $\mathcal{D}_i$ be the respective out-of-sample AUC and distance scores for each pareto optimal tree.

For the **Stability-AUC balancing strategy**, we define a “high-AUC” candidate set, $\mathcal{C}$:

$$\mathcal{C} = \left\{ i \in P \mid \mathcal{A}_i \geq (1 - \epsilon) \max_j \, (\mathcal{A})_j \right\}$$

and we choose the final best tree via:

$$\text{“} i_* = \text{argmax}_{i \in \mathcal{C}} (\mathcal{A}_i - \mathcal{D}_i) \text{.”}$$

AUC Maximizing Strategy

$$i_{\text{AUC}} = \text{argmax}_{i \in \mathcal{P}} (\mathcal{A}_i)$$

Distance Minimizing Strategy

$$i_{\text{AUC}} = \text{argmin}_{i \in \mathcal{P}} (\mathcal{D}_i)$$

Implementation

bertsimas_suicide_notebook

May 8, 2025

1 Implementing a stable decision tree (Bertsimas et al. 2013) for suicide prediction

1.1 Imports

[1]:
```python
import sys
import itertools
from pathlib import Path
from typing import Dict, List
from itables import show
from IPython.display import Markdown
src_path = Path("../src/dt-distance").resolve()
data_path = Path("../data").resolve()
sys.path.append(str(data_path))
sys.path.append(str(src_path))

import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import roc_auc_score
from sklearn.utils import resample
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from dt_distance.distance_calculator import DistanceCalculator
import matplotlib.pyplot as plt
from alive_progress import alive_bar
from imblearn.over_sampling import SMOTE, SVMSMOTE
np.random.seed(42)
DATA_PATH = "../data/DataSet_Combined_SI_SNI_Baseline_FE.csv"
```

1.2 Configuration parameters

[2]:
```python
DEPTHS = list(range(3, 13))
MIN_SAMPLES = [3, 5, 10, 30, 50]
NUM_BOOTSTRAPS = 25

FEATURE_SETS: Dict[str, List[str]] = {
    "suicidea": [
        "age", "gender", "sexori", "raceall", "trauma_sum", "cesd_score",␣
↪"harddrug_life", "school", "degree", "job", "sex", "concurrent", "exchange",␣
↪"children", "weapon", "fight", "fighthurt", "ipv", "ptsd_score", "alcfirst",␣
↪"potfirst", "staycurrent", "homelage", "time_homeless_month", "jail",␣
↪"jailkid", "gettherapy", "sum_alter", "sum_family", "sum_home_friends",␣
↪"sum_street_friends", "sum_unknown_alter", "sum_talk_once_week",␣
↪"sum_alter3close", "prop_family_harddrug", "prop_friends_harddrug",␣
↪"prop_friends_home_harddrug", "prop_friends_street_harddrug",␣
↪"prop_alter_all_harddrug", "prop_enc_badbehave", "prop_alter_homeless",␣
↪"prop_family_emosup", "prop_friends_emosup", "prop_friends_home_emosup",␣
↪"prop_friends_street_emosup", "prop_alter_all_emosup",␣
↪"prop_family_othersupport", "prop_friends_othersupport",␣
↪"prop_friends_home_othersupport", "prop_friends_street_othersupport",␣
↪"prop_alter_all_othersupport", "sum_alter_staff", "prop_object_badbehave",␣
↪"prop_enc_goodbehave", "prop_alter_school_job", "sum_alter_borrow"],
    "suicattempt": [
        "age", "gender", "sexori", "raceall", "trauma_sum", "cesd_score",␣
↪"harddrug_life", "school", "degree", "job", "sex", "concurrent", "exchange",␣
↪"children", "weapon", "fight", "fighthurt", "ipv", "ptsd_score", "alcfirst",␣
↪"potfirst", "staycurrent", "homelage", "time_homeless_month", "jail",␣
↪"jailkid", "gettherapy", "sum_alter", "prop_family", "prop_home_friends",␣
↪"prop_street_friends", "prop_unknown_alter", "sum_talk_once_week",␣
↪"sum_alter3close", "prop_family_harddrug", "prop_friends_harddrug",␣
↪"prop_friends_home_harddrug", "prop_friends_street_harddrug",␣
↪"prop_alter_all_harddrug", "prop_enc_badbehave", "prop_alter_homeless",␣
↪"prop_family_emosup", "prop_friends_emosup", "prop_friends_home_emosup",␣
↪"prop_friends_street_emosup", "prop_alter_all_emosup",␣
↪"prop_family_othersupport", "prop_friends_othersupport",␣
↪"prop_friends_home_othersupport", "prop_friends_street_othersupport",␣
↪"prop_alter_all_othersupport", "sum_alter_staff", "prop_object_badbehave",␣
↪"prop_enc_goodbehave", "prop_alter_school_job", "sum_alter_borrow"],
}

MODEL_PARAMS = {
    "suicidea": dict(min_samples_leaf=10, min_samples_split=20, max_depth=4),
    "suicattempt": dict(min_samples_leaf=10, min_samples_split=30, max_depth=4),
}

LABELS = ["suicidea", "suicattempt"]
```

1.3 Data Import and Preparation

[3]:
```
def prepare_data(df: pd.DataFrame, features: List[str], label: str, rng,␣
↪imbalance=None):
    df = df.replace('NaN', pd.NA)  # replace the string 'NaN' with actual NaN␣
↪values
```

```
    df_full_cleaned = df[features + [label]].dropna().copy()
    X = df_full_cleaned[features]
    y = df_full_cleaned[label]

    if imbalance == "SMOTE":
        sm = SVMSMOTE(random_state=42)
        X, y = sm.fit_resample(X, y)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=rng.integers(0, 2**32 - 1), stratify=y)
    X_full = df_full_cleaned[features]
    y_full = df_full_cleaned[label]
    return X_full, y_full, X_train, X_test, y_train, y_test

df = pd.read_csv(DATA_PATH)
X_full, y_full, X_train, X_test, y_train, y_test = prepare_data(df,␣
 ↪FEATURE_SETS["suicidea"], "suicidea", np.random.default_rng(1234))

print(f"Number of samples in the full dataset: {len(X_full)}")
print(f"Number of samples in the training set: {len(X_train)}")
print(f"Number of samples in the test set: {len(X_test)}")
print(f"Shape of training set: {X_train.shape}")
```

```
Number of samples in the full dataset: 586
Number of samples in the training set: 439
Number of samples in the test set: 147
Shape of training set: (439, 56)
```

[4]:
```
X_train.head()
```

[4]:
```
      age  gender  sexori  raceall  trauma_sum  cesd_score  harddrug_life  \
105  21.0     0.0     1.0      0.0         0.0         9.0            4.0
830  22.0     0.0     0.0      0.0         3.0        13.0           15.0
361  23.0     0.0     1.0      1.0         4.0        10.0           10.0
278  20.0     0.0     0.0      1.0         4.0        18.0            3.0
149  22.0     0.0     0.0      0.0         3.0        16.0            9.0

     school  degree  job  …  prop_family_othersupport  \
105     2.0     2.0  2.0  …                       0.0
830     2.0     2.0  2.0  …                       0.0
361     1.0     2.0  2.0  …                       0.0
278     2.0     2.0  2.0  …                       0.0
149     2.0     2.0  2.0  …                       0.0

     prop_friends_othersupport  prop_friends_home_othersupport  \
105                       0.00                             0.0
830                       0.00                             0.0
```

```
361                         0.00                                 0.0
278                         0.00                                 0.0
149                         0.11                                 0.0

     prop_friends_street_othersupport  prop_alter_all_othersupport  \
105                              0.00                         0.00
830                              0.00                         0.40
361                              0.11                         0.16
278                              0.00                         0.00
149                              0.11                         0.11

     sum_alter_staff  prop_object_badbehave  prop_enc_goodbehave  \
105              0.0                   0.46                 0.67
830              0.0                   0.70                 1.00
361              3.0                   0.68                 1.00
278              0.0                   0.62                 1.00
149              0.0                   0.25                 1.00

     prop_alter_school_job  sum_alter_borrow
105                   0.33               0.0
830                   0.50               6.0
361                   0.37               1.0
278                   0.83               6.0
149                   0.67               3.0

[5 rows x 56 columns]
```

Balancing the dataset using SMOTE:

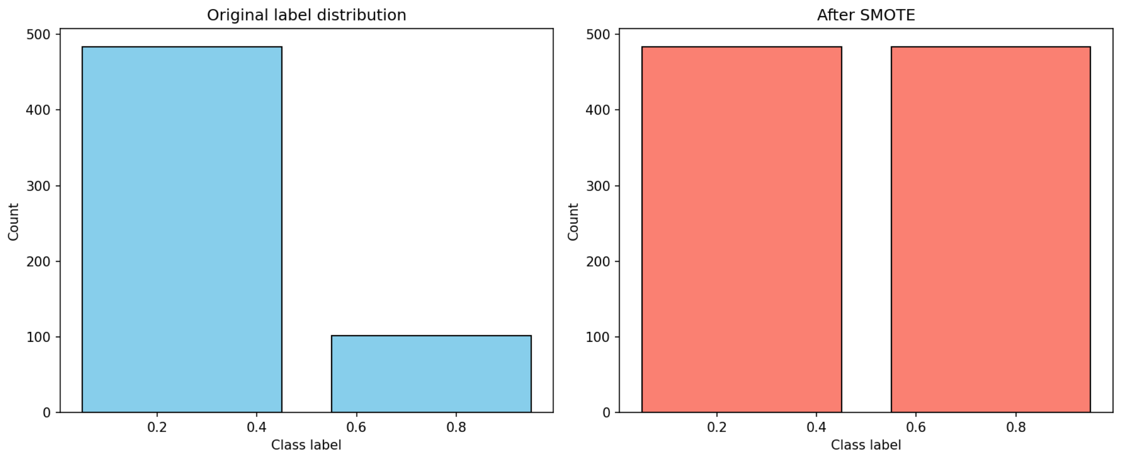
[5]:
```
orig_counts = y_full.value_counts().sort_index()
sm = SVMSMOTE(random_state=42)
X_smote, y_smote = sm.fit_resample(X_full, y_full)
smote_counts = y_smote.value_counts().sort_index()

fig, (ax1, ax2) = plt.subplots(
    ncols=2,
    figsize=(12, 5),
    dpi=150,
)

ax1.hist(
    y_full,
    bins=len(orig_counts),
    rwidth=0.8,
    color='skyblue',
    edgecolor='black'
)
```

```
ax1.set_title("Original label distribution")
ax1.set_xlabel("Class label")
ax1.set_ylabel("Count")

ax2.hist(
    y_smote,
    bins=len(smote_counts),
    rwidth=0.8,
    color='salmon',
    edgecolor='black'
)
ax2.set_title("After SMOTE")
ax2.set_xlabel("Class label")
ax2.set_ylabel("Count")
plt.tight_layout()
plt.show()
```



1.4 Train test split implementation

[6]:
```
def random_train_split(X, y):
    X_values = X.values if hasattr(X, 'values') else X
    y_values = y.values if hasattr(y, 'values') else y

    N = X_values.shape[0]
    indices = np.random.permutation(N)
    X0, y0 = X_values[indices[:N // 2]], y_values[indices[:N // 2]]
    return X0, y0

X0, y0 = random_train_split(X_train.values, y_train.values)
X0.shape, y0.shape
```

```
[6]: ((219, 56), (219,))
```

```
[7]: def train_decision_tree(X, y, depth, min_samples_leaf):
         clf = DecisionTreeClassifier(max_depth=depth,␣
      ↪min_samples_leaf=min_samples_leaf)
         clf.fit(X, y)
         return clf

     def bootstrap_trees(X, y, depths, min_samples, B):
         # Create B bootstrap trees by sampling with replacement from X_0
         trees = []
         for _ in range(B):
             X_sample, y_sample = resample(X, y, replace=True)
             depth = np.random.choice(depths)
             min_leaf = np.random.choice(min_samples)
             tree = train_decision_tree(X_sample, y_sample, depth, min_leaf)
             trees.append(tree)
         return trees
```

1.5 Bootstrap trees

```
[8]: T0 = bootstrap_trees(X0, y0, DEPTHS, MIN_SAMPLES, NUM_BOOTSTRAPS)
     print("Number of trees in T0:", len(T0))

     T = bootstrap_trees(X_train.values, y_train.values, DEPTHS, MIN_SAMPLES,␣
      ↪NUM_BOOTSTRAPS)
     print("Number of trees in T:", len(T))
```

```
Number of trees in T0: 25
Number of trees in T: 25
```

1.6 Computing tree distances

```
[9]: def compute_average_distances(T0, T, X_train, y_train):
         X_train_values = X_train.values if hasattr(X_train, "values") else X_train
         distances: list[float] = []

         with alive_bar(len(T), title="Computing average tree distances",␣
      ↪dual_line=True, spinner="waves", bar="smooth") as bar:
             for tree_b in T:
                 d_b = 0.0
                 for tree_beta in T0:
                     calc = DistanceCalculator(tree_beta, tree_b, X=X_train_values,␣
      ↪y=y_train)
                     d_b += calc.compute_tree_distance()

                 mean_dist = d_b / len(T0)
```

```
            distances.append(mean_dist)
            bar()

    return distances
distances = compute_average_distances(T0, T, X_train, y_train)
```

```
Computing average tree distances |                         |
25/25 [100%] in 16.4s (1.52/s)
```

1.7 Evaluating Predictive Power

Computing the AUC score on the test set:

```
[10]: def evaluate_predictive_power(trees, X_holdout, y_holdout):
          auc_scores = []
          for tree in trees:
              y_prob = tree.predict_proba(X_holdout)[:, 1]
              auc = roc_auc_score(y_holdout, y_prob)
              auc_scores.append(auc)
          return auc_scores

      auc_scores = evaluate_predictive_power(T, X_test.values, y_test.values)
      print("Average AUC score:", np.mean(auc_scores))
```

```
Average AUC score: 0.6213350286077559
```

1.7.1 Getting the Pareto optimal trees

```
[11]: def pareto_optimal_trees(distances, auc_scores):
          pareto_trees = []
          for i, (d_i, a_i) in enumerate(zip(distances, auc_scores)):
              dominated = False
              for j, (d_j, a_j) in enumerate(zip(distances, auc_scores)):
                  if i != j and ((d_j <= d_i and a_j > a_i) or (d_j < d_i and a_j >=␣
       ↪a_i)):
                      dominated = True
                      break
              if not dominated:
                  pareto_trees.append(i)
          return pareto_trees

      pareto_trees = pareto_optimal_trees(distances, auc_scores)
      print("Number of Pareto optimal trees:", len(pareto_trees))
```

```
Number of Pareto optimal trees: 4
```

1.7.2 Selecting the final tree

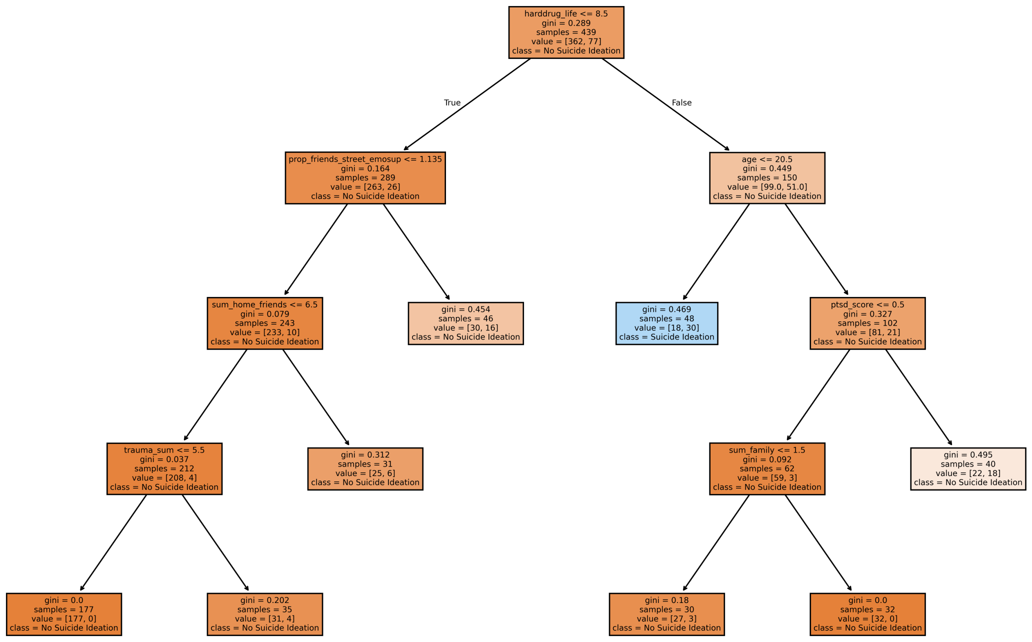
[13]:
```python
def select_final_tree(distances, auc_scores, pareto_indices, epsilon=0.01):
    best_auc = max(auc_scores)
    candidates = [i for i in pareto_indices if auc_scores[i] >= (1 - epsilon) *␣
 ↪best_auc]
    if not candidates:
        candidates = pareto_indices
    best_idx = max(candidates, key=lambda i: auc_scores[i] - distances[i])
    return best_idx

selected_tree_index = select_final_tree(distances, auc_scores, pareto_trees)
print("Selected tree index:", selected_tree_index)
```

```
Selected tree index: 23
```

1.8 Visualizing Selected Tree

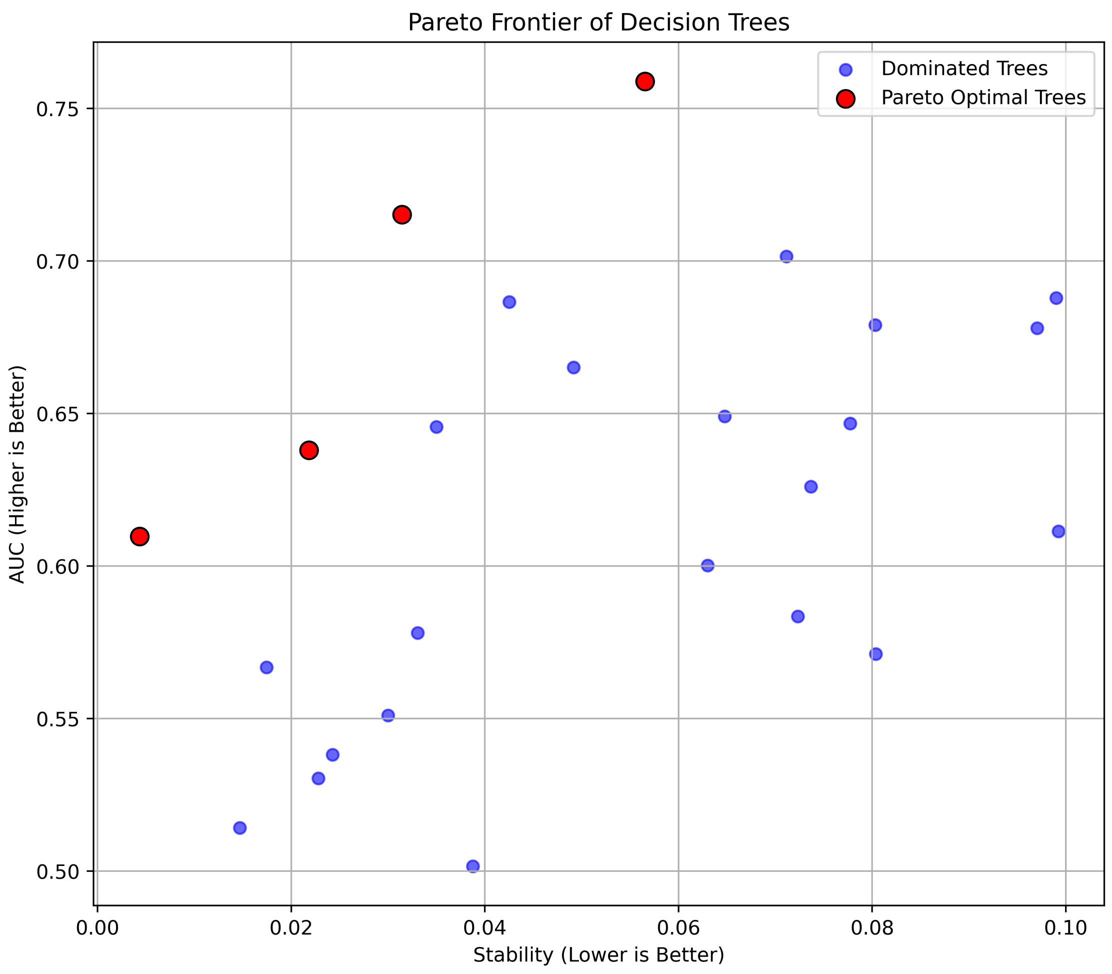
[14]:
```python
selected_tree = T[selected_tree_index]
plt.figure(figsize=(12, 8), dpi=500)
plot_tree(selected_tree,
          feature_names=X_full.columns,
          class_names=["No Suicide Ideation", "Suicide Ideation"],
          filled=True)
plt.tight_layout()
```



1.9 Visualizing the Pareto Frontier

[15]:
```python
def plot_pareto_frontier(distances, auc_scores, pareto_indices):
    distances = np.array(distances)
    auc_scores = np.array(auc_scores)
    pareto_indices = set(pareto_indices)
    is_pareto = np.array([i in pareto_indices for i in range(len(distances))])
    # Plotting
    plt.figure(figsize=(8, 7), dpi=500)
    plt.scatter(distances[~is_pareto], auc_scores[~is_pareto], c='blue',␣
 ↪label='Dominated Trees', alpha=0.6)
    plt.scatter(distances[is_pareto], auc_scores[is_pareto], c='red',␣
 ↪edgecolors='black', s=80, label='Pareto Optimal Trees')
    plt.xlabel("Stability (Lower is Better)")
    plt.ylabel("AUC (Higher is Better)")
    plt.title("Pareto Frontier of Decision Trees")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()

plot_pareto_frontier(distances, auc_scores, pareto_trees)
```


Pareto Frontier of Decision Trees
Dominated Trees
Pareto Optimal Trees
AUC (Higher is Better)
Stability (Lower is Better)
0.50
0.55
0.60
0.65
0.70
0.75
0.00
0.02
0.04
0.06
0.08
0.10

Experiment design

Below is a sample output from the experiment runner for the suicide experience dataset with seed 42, predicting for the label `suicidea`.

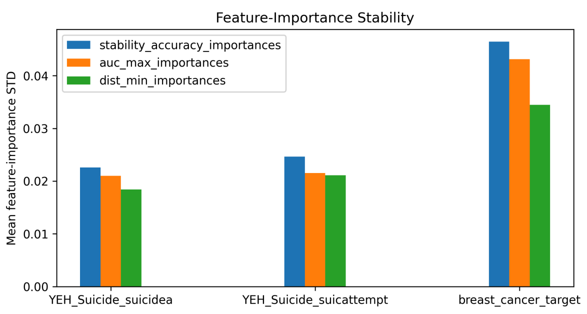
```
================================================
Running for dataset DataSet_Combined_SI_SNI_Baseline_FE with seed 42
================================================
ds_nameDataSet_Combined_SI_SNI_Baseline_FE.csv
Experiment:
experiment_20250501_134848_seed_42_DataSet_Combined_SI_SNI_Baseline_FE_suicidea -
Seed: 42 - Dataset: DataSet_Combined_SI_SNI_Baseline_FE
Number of samples in the full dataset: 586
Number of samples in the training set: 726
Number of samples in the test set: 242
Shape of training set: (726, 56)
Shape of random split: (363, 56), (363,)
Number of trees in T0: 20
Number of trees in T: 20
Computing average tree distances |████████████████████████████████████████| 20/20
[100%] in 20.7s (0.96/s)
Number of distances computed: 20
Average AUC score: 0.821854723038044
Number of Pareto optimal trees: 7
Frequenicies of top 2 common features: [[('trauma_sum', 70.0), ('fight', 20.0)],
[('harddrug_life', 45.0), ('exchange', 15.0)], [('LEAF_NODE', 25.0),
('harddrug_life', 20.0)]]
Selected stability-accuracy trade-off final tree index: 1
Stability-accuracy tree depth: 4, nodes: 23
Selected AUC maximizing tree index: 1
AUC-maximizing tree depth: 4, nodes: 23
Selected distance minimizing tree index: 15
Distance-minimizing tree depth: 11, nodes: 79
Completed experiment:
```

Results

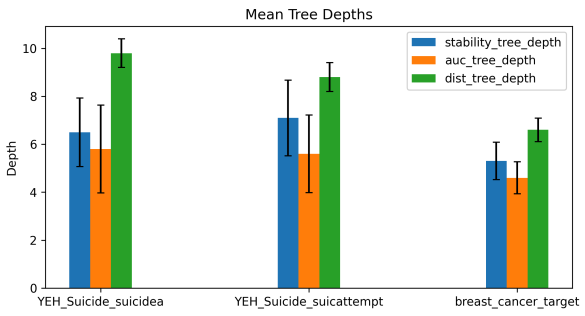
In Figure 1, we present the aggregated results of the experiments. Figure 1a shows the mean feature importance standard deviation, Figure 1b shows the mean tree depth, Figure 1c shows the mean number of nodes, and Figure 1d shows the mean number of distinct top features.

Aggregated metrics

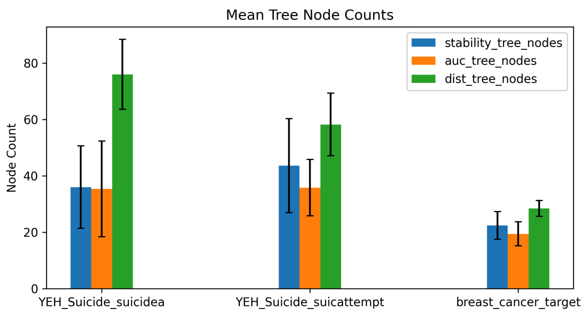
Each of the aggregation metrics plot the mean and standard deviation of the artifact measured. For quantifying stability, we expect to see comparable averages Figure 2 shows the aggregated AUC, distance, feature importance deviation, unique features across top 3 splits, tree depth, and nodes of running variations of our stability algorithm on 10 unique train-test split instances of the initial dataset on an imbalanced class-label dataset and one corrected with SMOTE. The SMOTE-processed data tends to have higher tree nodes across all Pareto selection strategies and much higher AUC as well as slightly lower feature importance deviation. The accuracy (AUC) is much higher (.90) on the SMOTE model variations. Further exploration on the dataset features is needed in order to understand the sharp increase in node complexity on the SMOTE-corrected dataset for the final optimal trees.



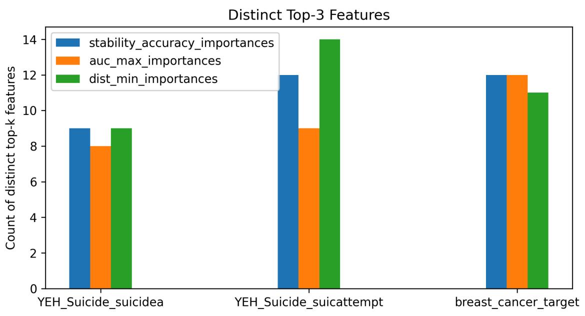
(a) Mean feature importance STD



(b) Mean tree depth



(c) Mean number of nodes



(d) Mean number of distinct top features

Method (CART Pareto)	AUC	Distance	Feature Imp. σ	# Features in top 3	# Nodes	Tree Depth
AUC	0.752 (0.029)	0.043 (0.009)	0.028	11	14.8	4.4
Stability	0.751 (0.029)	0.037 (0.014)	0.03	12	19.2	5.0
Distance	0.596 (0.066)	0.009 (0.005)	0.026	20	49.0	8.4
AUC	0.900 (0.022)	0.054 (0.031)	0.022	9	35.800	5.600
Stability	0.897 (0.023)	0.020 (0.017)	0.025	12	43.600	7.100
Distance	0.852 (0.026)	0.006 (0.001)	0.021	14	58.200	8.800

Figure 2. Aggregated metrics on Unbalanced vs SMOTE-processed data

Discussion

We observe correlations of varying magnitude between tree distance and other aggregate metrics, prompting examination of how distance influences model quality. Deeper trees appear preferable when optimizing for structural distance, but this choice raises classic bias–variance concerns regarding overfitting. Another open question is whether the size of T_0 is sufficient to represent the diversity of T under bootstrapping; future work should test for distributional shifts introduced during sampling.

What does trust mean to a clinician?

For clinicians, trust in a predictive model translates into precise identification of intervention targets, enabling stakeholders to design effective prevention strategies tailored to the clinical risk profiles of YEH. Moreover, trustworthy models reduce algorithmic aversion by demonstrating consistent performance; clinicians are less likely to disregard recommendations from models that reliably flag suicidal risk.

Future work

In the short term, through collaboration with stakeholders, we will refine the operational definition of stability and specify corresponding success metrics. Hyper-parameter tuning will target path-distance class weights, the Pareto selection strategy, and data pre-processing choices. Finally, we will investigate how the stability–performance trade-off affects the ultimate choice of Pareto-optimal tree.

In the long term, a key question to address is whether the proposed implementation reliably identifies suicide risk among YEH. We plan to extend feature selection and incorporate qualitative pre-processing. Robustness will be analyzed through direct perturbations (changes in tree thresholds) and indirect perturbations (modifications of training data) to quantify sensitivity.

Bibliography

[1] M. Kirst, T. Frederick, and P. G. Erickson, "Concurrent Mental Health and Substance Use Problems among Street-Involved Youth," *International Journal of Mental Health and Addiction*, vol. 9, no. 5, pp. 543–553, 2011, doi: 10.1007/s11469-011-9328-3.

[2] L. Rew, M. Taylor-Seehafer, and M. L. Fitzgerald, "Sexual Abuse, Alcohol and Other Drug Use, and Suicidal Behaviors in Homeless Adolescents," *Issues in Comprehensive Pediatric Nursing*, vol. 24, no. 4, pp. 225–240, 2001, doi: 10.1080/014608601753260326.

[3] D. Eaton, "Youth Risk Behavior Surveillance - United States, 2011 - PubMed." Accessed: May 01, 2025. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/22673000/

[4] C. Merscham, J. M. Van Leeuwen, and M. McGuire, "Mental Health and Substance Abuse Indicators among Homeless Youth in Denver, Colorado," *Child Welfare*, vol. 88, no. 2, pp. 93–110, 2009.

[5] B. E. Molnar, S. B. Shade, A. H. Kral, R. E. Booth, and J. K. Watters, "Suicidal Behavior and Sexual/Physical Abuse among Street Youth," *Child Abuse & Neglect*, vol. 22, no. 3, pp. 213–222, Mar. 1998, doi: 10.1016/s0145-2134(97)00137-3.

[6] E. Votta and I. Manion, "Suicide, High-Risk Behaviors, and Coping Style in Homeless Adolescent Males' Adjustment," *Journal of Adolescent Health*, vol. 34, no. 3, pp. 237–243, 2004, doi: 10.1016/j.jadohealth.2003.06.002.

[7] "Preventing Suicide through Connectedness." Accessed: May 01, 2025. [Online]. Available: https://stacks.cdc.gov/view/cdc/11795

[8] A. Fulginiti, E. Rice, H.-T. Hsu, H. Rhoades, and H. Winetrobe, "Risky Integration: A Social Network Analysis of Network Position, Exposure, and Suicidal Ideation among Homeless Youth," *Crisis: The Journal of Crisis Intervention and Suicide Prevention*, vol. 37, no. 3, pp. 184–193, 2016, doi: 10.1027/0227-5910/a000374.

[9] D. Bertsimas and V. D. Jr, "Improving Stability in Decision Tree Models," no. arXiv:2305.17299. arXiv, May 2023. doi: 10.48550/arXiv.2305.17299.