Adaptive Decision Trees with Dynamic Depth Pruning for Uneven Data Distributions

By: Aarav Gupta, Rohith Yelisetty

Introduction

- Many machine learning datasets are imbalanced, with dense and sparse regions
- Traditional decision trees tend to overfit in dense areas and underfit in sparse regions
- Our solution: an adaptive decision tree that dynamically adjusts depth based on data density

Problem Statement

- Decision trees struggle with uneven data distributions
- Dense regions → excessive branching → overfitting
- Sparse regions → shallow trees → underfitting
- Need for a method that balances generalizability and interpretability

Proposed Solution

- Adaptive Decision Tree (ADT) with Density-Based Pruning
- Inspired by K-Nearest Neighbors (KNN) to dynamically adjust tree depth
    - Utilized a variant that uses radius rather than K
- Prevents excessive branching in sparse areas while allowing deeper splits in dense areas
- Results in better accuracy and generalization

Related Works

Various methods address decision tree adaptability:

- Depth control methods (global, lacks local adaptability)
- KNN-inspired local adaptability (better local patterns, but computationally expensive)
- Hybrid pruning techniques (reduces overfitting, but parameter tuning is tedious)

Our approach balances global structure and local adaptability without sacrificing interpretability

Dataset & Features

Dataset: Traffic Accident Prediction Dataset (Kaggle)

Class: **Accident_Severity** (Low, Moderate, High)

- Low was the most common, High was least common

Features:

- Weather, Road Type, Time of Day, Speed Limit
- Number of Vehicles, Road Condition, Vehicle Type
- Driver Age, Driver Experience, Road Light Condition

Preprocessing

- First, the attributes such as speed limit, number of vehicles, driver age, and driver experience were discretized into defined ranges

Weather	Road_Type	Time_of_Day	Speed_Limit	Number_of_Vehicles
Clear → 0 Rainy → 1 Foggy → 2 Snowy → 3	Highway → 0 City Road → 1 Rural Road → 2 Mountain Road → 3	Morning → 0 Afternoon → 1 Evening → 2 Night → 3	(-inf - 75.75] → 0 (75.75 - 121.5] → 1 (167.25 - inf) → 2	(-inf - 4.25] → 0 (4.25 - 7.5] → 1 (7.5 - 10.75] → 2 (10.75 - inf) → 3

Road_Condition	Vehicle_Type	Time_of_Day	Driver_Experience	Road_Light_Condition
Dry → 0 Under Construction → 1 Wet → 2 Icy → 3	Bus → 0 Truck → 1 Car → 2 Motorcycle → 3	(-inf - 30.75] → 0 (30.75 - 43.5] → 1 (43.5 - 56.25] → 2 (56.25 - inf) → 3	(-inf - 24] → 0 (24 - 39] → 1 (39 - 54] → 2 (54 - inf) → 3	Daylight → 0 Artificial Light → 1 No Light → 2

Preprocessing

- Instances with missing class values were removed
- Other missing values were replaced by their respective modes
- All the data values were then changed to numeric categories for computation such as Euclidean Distance
- Stratified train-test set with a 70-30 ratio to preserve the class distribution across the subsets

Methods - Control & Weka J48 Pruned Decision Trees

Regular Decision Tree

- Splits nodes recursively using Gain Ratio
- Prone to overfitting in dense areas
- Fails to generalize well in sparse regions
- Continues to split until a pure state is reached or attributes have all been used

J48 Pruned Tree (Weka)

- Java implementation of C4.5 decision tree algorithm
- Implements pruning to reduce overfitting and excessive branch length
- Still lacks local adaptability

Methods - KNN & Adaptive Decision Tree

K-Nearest Neighbors (KNN)

- Classifies based on closest data points in feature space
- Good for local patterns but expensive for large datasets

Adaptive Decision Tree (Our Model)

- Dynamically adjusts depth based on local density
- Uses density thresholding:
  - High density → deeper splits
  - Low density → early stopping (pruning)
- Strikes a balance between interpretability and flexibility

Experiment Setup

Models compared:

- Control Decision Tree (Baseline)
- J48 Pruned Tree (Benchmark)
- Adaptive Decision Tree (Our Model)

Evaluation Metrics:

- Accuracy, Precision, Recall, F1-Score
- Confusion Matrices

Experiment setup

Hyperparameters:

- Max depth = 8
- Minimum density threshold = 2
- Radius for density estimation = 14.0

Results - Accuracy Comparison

Model	Training Accuracy	Testing Accuracy
Control Decision Tree	99.5%	49.2%
Weka J48 Pruned Tree	74.0%	54.6%
Adaptive Decision Tree	52.0%	73.3%

Results - Accuracy Comparison

- Control Decision Tree: Overfits (high training accuracy, poor generalization)
- J48 Pruned Tree: Better, but struggles with class overlap
- Adaptive Decision Tree: Best testing accuracy (73.3%) due to density-based pruning

Results - Control Decision Tree Confusion Matrices

	Predicted		
Actual	**Low**	**Moderate**	**High**
Low	333	0	1
Moderate	2	167	0
High	0	0	55

Control Decision Tree - Training

	Predicted		
Actual	**Low**	**Moderate**	**High**
Low	92	34	18
Moderate	40	25	7
High	15	8	1

Control Decision Tree - Testing

Results - Weka J48 Pruned Decision Tree Confusion Matrices

	Predicted		
Actual	Low	Moderate	High
Low	309	25	0
Moderate	75	91	3
High	37	5	15

Weka J48 Decision Tree - Training

	Predicted		
Actual	Low	Moderate	High
Low	117	24	3
Moderate	55	14	3
High	16	8	0

Weka J48 Decision Tree - Testing

Results - Adaptive Decision Tree Confusion Matrices

	Predicted		
Actual	Low	Moderate	High
Low	269	43	22
Moderate	139	18	12
High	47	5	3

Adaptive Decision Tree - Training

	Predicted		
Actual	Low	Moderate	High
Low	131	6	7
Moderate	38	31	3
High	9	1	14

Adaptive Decision Tree - Testing

Confusion Matrix Analysis

- Control Decision Tree: Poor classification of Moderate/High categories
- J48 Pruned Tree: Some improvement, but still misclassified overlapping cases
- Adaptive Decision Tree: More balanced classification, with significantly better recall for Moderate cases

Results - Precision, Recall, F1-Score

Model	Precision	Recall	F1-Score
Control Decision Tree	0.345	0.343	0.344
Weka J48 Pruned Tree	0.309	0.336	0.322
Adaptive Decision Tree	0.712	0.641	0.675

- Adaptive Decision Tree outperforms other models in handling imbalanced classes

Discussion - Key Takeaways

Trade-off between Training Accuracy and Generalization

- Control Decision Tree overfits, while Adaptive Decision Tree prioritizes generalization

Density-based pruning prevents overfitting in sparse regions

Improves interpretability while maintaining accuracy

Limitations:

- Struggles with very sparse data cases
- Some edge cases may still require finer splits

Conclusion & Future Work

Key Contribution:

- Introduced a density-aware decision tree that balances local adaptability and global structure
- Achieved a 20% improvement in test accuracy over traditional models

Future Work:

- Optimize density thresholds for better performance
- Expand to other real-world applications (e.g., healthcare, finance)
- Test on larger, more diverse datasets to ensure scalability

References

[1] I. Chaabane et al., “Adapted Pruning Scheme for the Framework of Imbalanced Data-sets,” Procedia Computer Science, vol. 112, Jan. 2017, pp. 1542–53. Available: https://doi.org/10.1016/ 6 j.procs.2017.08.060.

[2] S. M. Mazinani and K. Fathi, “Combining KNN and Decision Tree Algorithms to Improve Intrusion Detection System Performance,” International Journal of Machine Learning and Computing, vol. 5, no. 6, Dec. 2015, pp. 476–79. Available: https://doi.org/10.18178/ijmlc.2015.5.6.556.

[3] I. Frias-Blanco et al., “Online Adaptive Decision Trees Based on Concentration Inequalities,” Knowledge-Based Systems, vol. 104, Apr. 2016, pp. 179–94. Available: https://doi.org/10.1016/ j.knosys.2016.04.019.

[4] View of Integrating Decision Tree and KNN Hybrid Algorithm Approach for Enhancing Agricultural Yield Prediction. Available: https://cims-journal.net/index.php/CN/article/view/17/17.

[5] V. G. Costa and C. E. Pedreira, “Recent Advances in Decision Trees: An Updated Survey,” Artificial Intelligence Review, vol. 56, no. 5, Oct. 2022, pp. 4765–800. Available: https://doi. org/10.1007/s10462-022-10275-5.

[6] Kuznetz, Den. Traffic Accident Prediction. Kaggle, 2024, https://www.kaggle.com/datasets/ denkuznetz/traffic-accident-prediction

[7] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[8] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). Available: https://doi.org/10.1038/s41586-020-2649-2

[9] The pandas development team. (2024). pandas-dev/pandas: Pandas (v2.2.3). Zenodo. https:// doi.org/10.5281/zenodo.13819579