Report - CADT

Introduction

This report summarizes the implementation and evaluation of decision tree models for credit approval classification on a public dataset. The objectives were to:

1. Implement the decision tree algorithm from scratch using information gain/gini index
2. Add reduced error pruning to reduce overfitting
3. Compare performance against scikit-learn's decision tree
4. Analyze the impact of pruning and choice of scoring metric

The dataset contains 690 instances with 15 features and a binary approved/denied decision. A training/test split of 80/20 is used for evaluation.

Data Preprocessing

The following data preprocessing steps were taken:

- Missing values were encoded as NaN
- Continuous features were converted to numerical features
- Median imputation was used for missing numerical values
- Continuous features were categorically binned into 3 equal width intervals
- Categorical features one-hot encoded
- Features and label separated

ID3 Decision Tree Implementation

A custom `DecisionTree` class is implemented with the following key methods:

`fit()`: grows full decision tree recursively
`_best_split()`: finds the best split at each node using chosen scoring metric
`predict()`: makes predictions on data by traversing decision tree
`prune()`: prunes tree using reduced error pruning
`accuracy()`: calculates accuracy score

Both entropy and gini index were implemented as interchangeable scoring metrics. Scikit-learn's `DecisionTreeClassifier` is used later for comparison.

The tree growing works by choosing the best split that maximizes information gain / Gini reduction at each node recursively until leaf nodes are pure or maximum depth is reached.

Post pruning is done by reverting splits that decrease accuracy on a validation set. The tree is traversed recursively after learning, with splits reverted if accuracy improves without them. This helps prevent overfitting.

Experiments and Results

The custom tree was first evaluated without pruning using both scoring metrics:

Decision Tree (Information Gain)

Training Set:

	precision	recall	f1-score	support
+	0.91	0.95	0.93	237
-	0.96	0.93	0.95	315
accuracy			0.94	552
macro avg	0.94	0.94	0.94	552
weighted avg	0.94	0.94	0.94	552

Test Set:

	precision	recall	f1-score	support
+	0.87	0.74	0.80	70
-	0.77	0.88	0.82	68
accuracy			0.81	138
macro avg	0.82	0.81	0.81	138
weighted avg	0.82	0.81	0.81	138

Decision Tree (Gini Index)

Training Set:

	precision	recall	f1-score	support
+	0.94	0.95	0.94	237
-	0.96	0.95	0.96	315
accuracy			0.95	552
macro avg	0.95	0.95	0.95	552
weighted avg	0.95	0.95	0.95	552

Test Set:

	precision	recall	f1-score	support
+	0.88	0.74	0.81	70
-	0.77	0.90	0.83	68
accuracy			0.82	138
macro avg	0.83	0.82	0.82	138
weighted avg	0.83	0.82	0.82	138

The high training accuracy indicates overfitting. After reduced error pruning:

Decision Tree (Information Gain)

Training Set:

	precision	recall	f1-score	support
+	0.94	0.92	0.93	237
-	0.94	0.95	0.95	315
accuracy			0.94	552
macro avg	0.94	0.94	0.94	552
weighted avg	0.94	0.94	0.94	552

Test Set:

	precision	recall	f1-score	support
+	0.90	0.74	0.81	70
-	0.78	0.91	0.84	68
accuracy			0.83	138
macro avg	0.84	0.83	0.83	138
weighted avg	0.84	0.83	0.82	138

Decision Tree (Gini Index)

Training Set:

	precision	recall	f1-score	support
+	0.94	0.94	0.94	237
-	0.96	0.96	0.96	315
accuracy			0.95	552
macro avg	0.95	0.95	0.95	552
weighted avg	0.95	0.95	0.95	552

Test Set:

	precision	recall	f1-score	support
+	0.88	0.74	0.81	70
-	0.77	0.90	0.83	68
accuracy			0.82	138
macro avg	0.83	0.82	0.82	138
weighted avg	0.83	0.82	0.82	138

Pruning helped reduce overfitting and improve generalization by lowering training performance. The test accuracy is more indicative of real world performance.

For final comparison, Scikit-learn's DecisionTreeClassifier was evaluated:

scikit-learn DecisionTreeClassifier:

Train Set:

	precision	recall	f1-score	support
+	0.95	0.99	0.97	237
-	0.99	0.96	0.98	315
accuracy			0.97	552
macro avg	0.97	0.98	0.97	552
weighted avg	0.97	0.97	0.97	552

Test Set:

	precision	recall	f1-score	support
+	0.84	0.74	0.79	70
-	0.76	0.85	0.81	68
accuracy			0.80	138
macro avg	0.80	0.80	0.80	138
weighted avg	0.80	0.80	0.80	138

The sklearn tree achieves the best test performance. This is likely due to the sophistication of its implementation as well as automatic hyperparameter tuning.

Analysis

The custom decision tree achieved reasonably good performance, with >82% test accuracy after pruning. This meets the goal of building an accurate credit approval model.

However, Scikit-learn still performed better, indicating room for improvement via tuning hyperparameters like tree depth, leaf samples and pruning parameters. More advanced ensemble methods like random forests may also help.

One limitation is that reduced error pruning uses a validation set created by splitting the training data. A better approach would be an independent pruning dataset.

The difference between scoring metrics was minor, with <1% variance in results. Both successfully produced effective trees. There was no clear better metric.

Conclusion

In summary, the objectives were achieved by developing a custom decision tree model that performed credit approval classification fairly accurately after pruning.

There is certainly further room for improvement in performance compared to Scikit-learn. But this program provides a solid baseline and framework to build upon with extra enhancements.

The implementation demonstrated core concepts like recursive tree induction, overfitting reduction via pruning and fundamental metrics like information gain and gini impurity.

Next steps for future work include hyperparameter optimization, ensemble techniques, alternate pruning methods and additional extensions to improve accessibility.