CS550-Machine Learning
Homework Assignment #2

Contact: a.yildirim@bilkent.edu.tr

Deadline: 30.03.2025 - 23:59

1 Introduction

In this homework, you will use and implement a decision tree classifier. All experiments are performed on the **Credit Risk dataset**, which is provided to you via Moodle. The assignment is divided into three parts, each focusing on different aspects of decision tree construction and evaluation. **Please review the guidelines in Section 4 before beginning your implementations.**

2 Dataset Description

The dataset is provided in two separate files:

- Training set: `train_credit_data.csv` (3772 instances)
- Test set: `test_credit_data.csv` (3428 instances)

Each instance consists of 12 features and 1 class label. The features are:

1. **Income** (in thousands)
2. **DebtToIncome** (in percent)
3. **CreditHistory** (in years)
4. **LatePayments** (integer count)
5. **Employment** (years in current job)
6. **CreditLines** (integer count)
7. **LoanAmount** (in thousands)
8. **HomeOwnership** (binary indicator)
9. **Age** (in years)
10. **Savings** (in thousands)

11. **Education** (ordinal: 1–5)
12. **ReliabilityScore** (dependent feature computed from `CreditHistory` and `LatePayments`)

There are 3 classes in total: Good, Average, and Poor. In Part 3, you will also consider feature extraction costs provided in `feature_costs.csv`.

3 Assignment Tasks

Part 1: Toolbox-Based Decision Tree

Use a machine learning toolbox of your choice (e.g., scikit-learn, PRTools, Weka, etc.) to construct a decision tree classifier. In this part:

- Train a decision tree on the training set by fine-tuning the hyperparameters `criterion`, `max_depth`, and `min_samples_split` (using either cross-validation or by splitting the training data into training and validation sets).
- Draw the decision tree corresponding to the best configuration found.
- Obtain training and test set accuracies. Report the class-based accuracies as well as the confusion matrices for both the training and test sets.
- Compare the training and test set class-based accuracies when pruning is applied versus when it is not. Note that the fine-tuned model from the first item can be considered a pre-pruned model. For this item, you are required to train a decision tree without any hyperparameter restrictions (i.e., with no pre-pruning) to disable pruning.
- Compare the training and test set class-based accuracies when normalization is applied versus when it is not. Is there any difference? Interpret these results.
- The training dataset has imbalanced class distributions. Compare the results when using the dataset as is and when you balance the number of classes by applying undersampling to reduce the number of samples in the majority classes to match the minority class. Is there any difference? Interpret these results.

Part 2: Decision Tree Implementation from Scratch

Implement your own decision tree classifier that uses pre-pruning. In your implementation:

- Train the model on the training instances using your selected splitting criterion and pre-pruning technique (found after fine-tuning the hyperparameters in Part 1); provide details of your selection.

- Explain your implementation of model training, including key calculations.
- Draw the decision tree learned on the training instances (with the best configuration of parameters that you have selected). For visualization purposes, you may use libraries (e.g., Graphviz) to draw the tree, but the training must be implemented from scratch.
- Obtain training and test set accuracies. Report the class-based accuracies and the confusion matrices for both the training and test sets.

Part 3: Cost-Sensitive Decision Tree

Extend your implementation from Part 2 so that it also considers the cost of using a feature as a splitting criterion (in conjunction with the purity of a split). Note that no additional cost is incurred for the **ReliabilityScore** feature if its constituent features (**CreditHistory** and **LatePayments**) have been extracted; otherwise, you must pay for the cost of the unextracted features. In this part:

- Provide the explicit form of your new splitting criterion (by Tan 93, in the lecture slides).
- Train a decision tree model on the training samples with a constraint of a maximum depth of 3.
- Explain your implementation of model training, including key calculations.
- Draw the decision tree. You may use visualization libraries (e.g., Graphviz) to draw the tree.
- Obtain training and test set accuracies. Report the class-based accuracies and the confusion matrices for both the training and test sets.
- Compute the cost of classifying each instance with this decision tree on the test set and report the average cost of classifying instances separately for each class (based on the actual classes of instances).
- In the report, compare the decision changes between the models trained with (Part 3) and without (Part 2) considering cost.

Note:

During the tree creation process, if a feature is used at a decision node for the first time, its extraction cost is taken from the feature cost table. However, if the same feature is used again in a later split, its cost should **default to 1.0 in the splitting criterion** to prevent double-counting and avoid zero division by ensuring a minimum cost value.

During inference, the total classification cost is computed by summing only the costs of the features listed in the feature cost table, based on whether they were used in the decision path. Repeated usage of the same feature should not be counted multiple times.

4 Implementation Guidelines and Submission

- **Part 1**: You may use any machine learning toolbox (e.g., PRTools, Weka, scikit-learn, etc.).
- **Parts 2 and 3**: You must implement the decision tree classifier by **writing your own code**. Use of any machine learning package for training is not allowed. (Libraries for visualizing/drawing the decision tree, such as Graphviz, are permitted.)
- You are allowed to use general-purpose libraries for data manipulation and numerical computation (e.g., Pandas, NumPy, etc.) in every part.
- You may use any programming language of your choice.
- The format, structure, and quality of tables and figures will contribute to your grade.
- **You are required to use the IEEE conference template for your report.**
- Do not include any screenshots in your report.
- Ensure that all questions specified in each part are addressed.
- Your report should be **no longer than six pages**. However, this is not a strict requirement; as long as all questions are answered as requested, having fewer pages will not affect your grade.

Submission: Submit the PDF of your report along with the source code of your implementation to Moodle by the deadline.