ELL409 - MACHINE INTELLIGENCE & LEARNING

Assignment 2 - Decision Tree

About Dataset

The dataset contains information related to direct marketing campaigns (via phone calls) conducted by a private banking institution. The objective is to predict whether a client will subscribe to a term deposit (variable y) based on various demographic and campaign-related features.

Input Variables:

- **age:** (numeric)
- **job**: type of job (categorical: "admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar" , "self-employed", "retired", "technician", "services")
- **marital** : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
- **education** (categorical: "unknown","secondary","primary","tertiary")
- **default**: has credit in default? (binary: "yes","no")
- **balance**: average yearly balance, in euros (numeric)
- **housing**: has a housing loan? (binary: "yes","no")
- **loan**: has a personal loan? (binary: "yes","no")
- **contact**: contact communication type (categorical: "unknown","telephone","cellular")
- **day**: last contact day of the month (numeric)
- **month**: last contact month of year (categorical: "jan", "feb", "mar", …, "nov", "dec")
- **duration**: last contact duration, in seconds (numeric)
- **campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- **pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
- **previous**: number of contacts performed before this campaign and for this client (numeric)
- **poutcome**: outcome of the previous marketing campaign (categorical: "unknown", "other", "failure", "success")

Output variable:

y: has the client subscribed to a term deposit? (binary: "yes","no")

Task(s) Description:

Task 1: Implementing a Decision Tree from Scratch

In this task, you will implement a **Decision Tree Classifier** from scratch using Python and the dataset provided. Follow the steps outlined below to build the decision tree, incorporating **pruning techniques** to improve generalization and reduce overfitting.

- **Data Preprocessing:** Load the dataset and perform any necessary preprocessing steps, including handling categorical variables, dealing with missing values, and scaling numeric features as needed.

- **Implement Key Components of the Decision Tree**:

    - **Entropy**: Write a function to calculate the entropy of a dataset.
    - **Gini Index**: Write a function to calculate the Gini impurity.
    - **Information Gain**: Implement a function to calculate the information gain for splitting on a particular feature.
    - **Tree Node Creation**: Implement the logic for creating nodes, including deciding when to stop splitting (e.g., maximum depth, minimum samples per node, or no further information gain).
    - **Split Criterion**: Implement a function that selects the best feature and threshold to split the dataset.
    - **Tree Construction**: Combine the above functions to construct the decision tree using a recursive algorithm.
    - **Prediction**: Implement a method that traverses the trained decision tree to make predictions for new samples.

- **Pruning:** Implement post-pruning (cost complexity pruning) to prevent overfitting. You should write code to,
    - Calculate the complexity of each subtree.
    - Prune branches that do not improve performance significantly on a validation set.
    - Use cross-validation to determine the optimal tree depth or subtree structure.

Task 2: Model Evaluation

Evaluate the performance of your decision tree on the test set using the following metrics:

- **Accuracy**: The percentage of correctly classified instances.

- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.
- **Recall**: The ratio of correctly predicted positive observations to all actual positives.
- **F1-Score**: The weighted average of precision and recall.

You should implement the calculation of these metrics from scratch (without using pre-built functions from libraries like Scikit-learn).

Task 3: Report on Pruning and Model Performance

After training and pruning your decision tree, provide a detailed observation in your report. Discuss:

- How pruning affected the size of the tree.
- The differences in model performance (e.g., accuracy, precision, recall, F1-score) before and after pruning.
- Whether pruning helped in mitigating overfitting and improving generalization.

Deliverables:

1. Python Code: A fully functional Python script or notebook that implements:
    - Decision tree with entropy/Gini calculations.
    - Pruning (post-pruning).
    - Model evaluation (accuracy, precision, recall, F1-score).
2. Report: A concise (2-3 page) report summarizing your approach, pruning strategy, and observations about the model's performance before and after pruning.