Assignment: Decision Tree Classifier Implementation

CAP 4630 - Artificial intelligence

Objective

In this assignment, you will implement a Decision Tree Classifier using the `sklearn.tree.DecisionTreeClassifier` from the `scikit-learn` library. You will load and preprocess a real-world dataset, train the Decision Tree model, and evaluate its performance using various metrics. You will also visualize the Decision Tree and interpret feature importance. There is a seprate file for training and test dataset. Training data is with name of "census-income.data" and testing data is with the name of "census-income.test"

Download the dataset from: Census-Income (KDD) Dataset.

Or use the following link:

https://archive.ics.uci.edu/dataset/117/census+income+kdd

Dataset Information

In this task, you will build a decision tree classifier using a real-world dataset called Census-Income Data Set, available publicly. This dataset contains weighted census data extracted from the 1994 and 1995 Current Population Surveys conducted by the U.S. Census Bureau. The data contains demographic and employment-related variables.

Basic Statistics:

- Number of instances in training data: 199,523
- Duplicate or conflicting instances: 46,716
- Number of instances in test data: 99,762

- Duplicate or conflicting instances in test data: 20,936
- Class probabilities for income in the test set:
  - Probability for income $\leq$ 50K: 93.80%
  - Probability for income $>$ 50K: 6.20%
- Majority class accuracy: 93.80% for income $\leq$ 50K
- Number of attributes: 40 (7 continuous, 33 nominal)

Detailed Information about Attributes:

- 91 distinct values for attribute #0 (age) — continuous
- 9 distinct values for attribute #1 (class of worker) — nominal
- 52 distinct values for attribute #2 (detailed industry recode) — nominal
- 47 distinct values for attribute #3 (detailed occupation recode) — nominal
- 17 distinct values for attribute #4 (education) — nominal
- 1,240 distinct values for attribute #5 (wage per hour) — continuous
- 3 distinct values for attribute #6 (enroll in educational institution last week) — nominal
- 7 distinct values for attribute #7 (marital status) — nominal
- 24 distinct values for attribute #8 (major industry code) — nominal
- 15 distinct values for attribute #9 (major occupation code) — nominal
- 5 distinct values for attribute #10 (race) — nominal
- 10 distinct values for attribute #11 (Hispanic origin) — nominal
- 2 distinct values for attribute #12 (sex) — nominal
- 3 distinct values for attribute #13 (member of a labor union) — nominal

- 6 distinct values for attribute #14 (reason for unemployment) — nominal
- 8 distinct values for attribute #15 (full or part-time employment status) — nominal
- 132 distinct values for attribute #16 (capital gains) — continuous
- 113 distinct values for attribute #17 (capital losses) — continuous
- 1,478 distinct values for attribute #18 (dividends from stocks) — continuous
- 6 distinct values for attribute #19 (tax filer status) — nominal
- 6 distinct values for attribute #20 (region of previous residence) — nominal
- 51 distinct values for attribute #21 (state of previous residence) — nominal
- 38 distinct values for attribute #22 (detailed household and family status) — nominal
- 8 distinct values for attribute #23 (detailed household summary in household) — nominal
- 10 distinct values for attribute #24 (migration code — change in MSA) — nominal
- 9 distinct values for attribute #25 (migration code — change in region) — nominal
- 10 distinct values for attribute #26 (migration code — move within region) — nominal
- 3 distinct values for attribute #27 (lived in this house one year ago) — nominal
- 4 distinct values for attribute #28 (migration previous residence in sunbelt) — nominal

- 7 distinct values for attribute #29 (number of persons worked for employer) — continuous
- 5 distinct values for attribute #30 (family members under 18) — nominal
- 43 distinct values for attribute #31 (country of birth — father) — nominal
- 43 distinct values for attribute #32 (country of birth — mother) — nominal
- 43 distinct values for attribute #33 (country of birth — self) — nominal
- 5 distinct values for attribute #34 (citizenship) — nominal
- 3 distinct values for attribute #35 (own business or self-employed) — nominal
- 3 distinct values for attribute #36 (filled income questionnaire for veterans admin) — nominal
- 3 distinct values for attribute #37 (veterans benefits) — nominal
- 53 distinct values for attribute #38 (weeks worked in year) — continuous
- 2 distinct values for attribute #39 (year) — nominal

The data was split into a training set and a test set using approximately a $\frac{2}{3} : \frac{1}{3}$ proportion using MineSet's MIndUtil tool.

Tasks and Marks Distribution

1. **Data Loading and Preprocessing (20 Marks)**
   Load the dataset using appropriate column names and clean the data. Handle missing values, if any, and ensure that categorical features are converted into a suitable format (e.g., one-hot encoding or label encoding). Write the code to load the data, preprocess it, and display the first few rows after preprocessing.

Hint: You can load the dataset using the `pandas` library. Below is an example of how to define the column names and read the dataset. Figure out all the colum names by yourself.

```
columns = [
   'age', 'class_of_worker', ..... 'income'
]

data = pd.read_csv('census-income.data', header=None, names=columns)
```

After loading the dataset, ensure to clean the data and apply appropriate transformations, especially handling categorical features (e.g., one-hot encoding or label encoding).

2. **Splitting the Data and Training the Model (30 Marks)**
   First, separate the dataset into features (`X`) and labels (`y`). The features will include all the columns except the target label, while the label column will contain the income classification (e.g., `-50000` or `50000+`).

   Use the `train_test_split` function from `sklearn.model_selection` to divide the features (`X`) and labels (`y`) into training and test sets with a ratio of 70:30. This will result in four variables: `X_train`, `X_test`, `y_train`, and `y_test`. Use the training split (`X_train`, `y_train`) to train the Decision Tree Classifier, and the test split (`X_test`, `y_test`) to evaluate its performance.

   Train the Decision Tree classifier with a maximum depth of 3 using the `DecisionTreeClassifier` from `sklearn.tree`. After training the model on the training data, evaluate it on the test data by calculating and reporting the following metrics:

   - Accuracy
   - Precision
   - Recall
   - F1-score

   Use the `sklearn.metrics.classification_report` and `accuracy_score` functions to compute and display these metrics.

3. **Visualizing the Decision Tree (10 Marks)**
   Print the decision tree and display all the features in tree form using `plot_tree` or any other visualization technique from `sklearn.tree`.

4. **Feature Importance (20 Marks)**
   Print the importance score for each feature using the `feature_importances_` attribute of the Decision Tree classifier. Discuss why one feature might appear as the most important, providing an intuitive explanation based on the dataset.

5. **Evaluation on the Test Set (20 Marks)**
   Use the trained model to predict labels for the separate test set (census-income.test). You will again need to split the features and labels. Calculate and display the accuracy, precision, recall, and F1-score on the test set, again using `classification_report` and `accuracy_score`.

Submission Instructions

Submit your implementation as a Jupyter notebook file (`.ipynb`). Structure your notebook into clearly labeled sections or tabs for each part of the assignment. Each part should contain both code and outputs. Ensure that the code for each task runs correctly and generates the required outputs (e.g., printed decision trees, evaluation metrics, etc.).

For part 4 (Feature Importance), create a separate text field in the Jupyter notebook and provide a detailed discussion on why one feature might appear as the most important. Provide an intuitive explanation based on the nature of the dataset and its features.

Save the Jupyter notebook file with your full name as the filename (e.g., `Firstname_Lastname_DecisionTree.ipynb`). Ensure that your code is well-documented with comments explaining each step of the process. Submit this file to Webcourse.

Submission checklist:

- The notebook should be divided into clear sections, with code and output for each part.
- Each section must show some output relevant to the task (e.g., accuracy, decision tree visualization, etc.).

- A separate text field should be included for the discussion of feature importance in part 4.
- Save the notebook using your full name.
- Submit the `.ipynb` file on the designated platform.

Total Marks: 100