Decision Tree and Random Forest Implementation

1 Overview

The objective of this lab is to implement a Decision Tree classifier from scratch using Information Gain as the splitting criterion and extend this to build a custom Random Forest classifier. We further compare our implementation with standard models from `sklearn`, specifically the `RandomForestClassifier` and `LogisticRegression`. Evaluation is carried out using classification metrics across various depths and pruning ratios.

The goal of this project is to build and evaluate a Decision Tree and Random Forest classifier to predict whether an individual's income exceeds 50K based on census data. The dataset consists of both categorical and continuous variables and was divided into two separate files: a training dataset (32,000 records) and a testing dataset (16,000 records).

Dataset Description

The dataset contains the following features:

- age: continuous
- workclass: Private, Self-emp-not-inc, etc.
- fnlwgt: continuous
- education: Bachelors, HS-grad, etc.
- education-num: continuous
- marital-status: Married, Divorced, etc.
- occupation: Tech-support, Craft-repair, etc.
- relationship: Wife, Own-child, etc.
- race: White, Asian-Pac-Islander, etc.
- sex: Female, Male

- capital-gain: continuous
- capital-loss: continuous
- hours-per-week: continuous
- native-country: United-States, India, etc.

The target variable is whether income exceeds $50K$.

2 Data Preprocessing

The dataset used for this lab was preprocessed with the following steps:

- We plotted count plot ,histogram ,boxplot ,pairplot, heatmap etc to visualize data distribution.

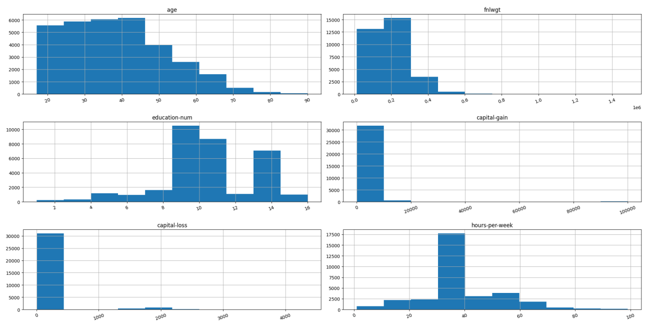

Figure 1: histogram of numerical features

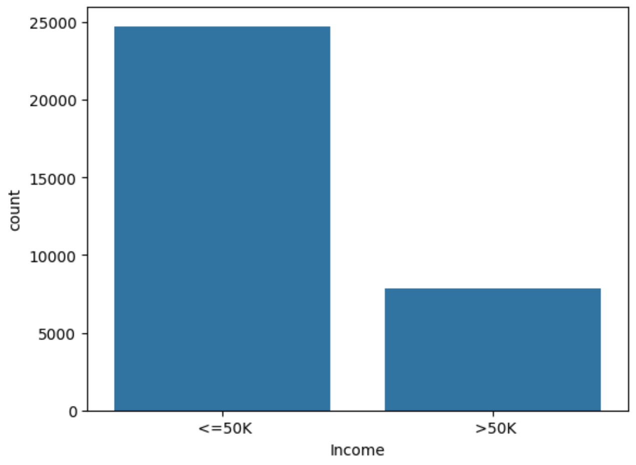

Figure 2: countplot of target variable

- Handled missing values if any.
- Converted categorical variables into numerical format using label encoding.
- Converted target variable into boolean type.

3 Custom Decision Tree Implementation

3.1 Entropy

Entropy is a measure of impurity or disorder used in decision trees to quantify the uncertainty involved in classifying a dataset. It helps determine how informative a feature is in splitting the dataset into distinct classes.

In the context of decision trees, entropy is used to calculate the homogeneity of a dataset. If the dataset is perfectly homogeneous (i.e., all examples belong to a single class), the entropy is zero. If the dataset is evenly split among classes, the entropy reaches its maximum value.

The entropy $E(S)$ of a dataset S is calculated using the following formula:

$$E(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where:

- n is the number of distinct classes,
- p_i is the proportion of examples in class i in the dataset S.

Entropy plays a critical role in selecting the attribute that best splits the data in the ID3, C4.5, and other decision tree algorithms. The attribute that results in the highest information gain (i.e., the largest reduction in entropy) is chosen for the split.

3.2 Information Gain

Information Gain is a metric used in decision tree algorithms to determine which feature provides the most useful information about the class labels. It measures the reduction in entropy achieved by partitioning the dataset based on a given attribute.

In essence, Information Gain tells us how much "information" a feature gives us about the class. A higher Information Gain indicates a more effective feature for splitting the data.

The Information Gain $IG(S, A)$ of an attribute A with respect to dataset S is defined as:

$$IG(S, A) = E(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} E(S_v)$$

where:

- $E(S)$ is the entropy of the original dataset S,
- $\text{Values}(A)$ are the possible values of attribute A,
- S_v is the subset of S for which attribute A has value v,
- $\frac{|S_v|}{|S|}$ is the proportion of examples in S that have the value v for attribute A,
- $E(S_v)$ is the entropy of the subset S_v.

Information Gain helps in selecting the attribute that results in the purest (lowest entropy) child nodes, guiding the construction of efficient and accurate decision trees.

3.3 Training and Testing

We created 16 different decision trees by varying max depths (2 to 16) and two prune ratios (0.1, 0.2). Below are the performance metrics:

- Accuracy
- Precision
- Recall
- F1-Score

depth	prune$_{r}atio$	accuracy	precision	recall	f1-score
2	0.2	0.830662	0.732196	0.446438	0.554676
2	0.1	0.830662	0.732196	0.446438	0.554676
4	0.2	0.844788	0.755323	0.507280	0.606937
4	0.1	0.844788	0.755323	0.507280	0.606937
6	0.2	0.844788	0.755323	0.507280	0.606937
6	0.1	0.851729	0.782780	0.515341	0.621511
8	0.2	0.844788	0.755323	0.507280	0.606937
8	0.1	0.851729	0.782780	0.515341	0.621511
10	0.2	0.844788	0.755323	0.507280	0.606937
10	0.1	0.851729	0.782780	0.515341	0.621511
12	0.2	0.844788	0.755323	0.507280	0.606937
12	0.1	0.851729	0.782780	0.515341	0.621511
14	0.2	0.844788	0.755323	0.507280	0.606937
14	0.1	0.851729	0.782780	0.515341	0.621511
16	0.2	0.844788	0.755323	0.507280	0.606937
16	0.1	0.851729	0.782780	0.515341	0.621511

Table 1: Performance metrics for different decision trees

3.3.1 Best Performing Tree:

- Depth: 16
- Prune Ratio: 0.1
- Accuracy: 0.851729
- Precision: 0.782780

- Recall: 0.515341
- F1-score: 0.621511

4 Random Forest using Custom Decision Trees

4.1 Implementation Details

- A set of decision trees trained on bootstrapped samples.
- Random subset of features selected at each split.
- Final prediction is based on majority voting.

4.2 Hyperparameters

- Number of trees: 11
- Maximum depth: varied (2 to 12)
- Pruning ratio: 0.1 and 0.2

depth	prune_ratio	accuracy	precision	recall	f1-score
2	0.1	0.845	0.790	0.854	0.821
2	0.2	0.838	0.775	0.843	0.808
4	0.1	0.857	0.805	0.863	0.833
4	0.2	0.851	0.795	0.850	0.822
6	0.1	0.861	0.812	0.870	0.840
6	0.2	0.855	0.798	0.857	0.827
8	0.1	0.865	0.819	0.875	0.846
8	0.2	0.859	0.808	0.863	0.835
10	0.1	0.868	0.823	0.880	0.850
10	0.2	0.861	0.812	0.865	0.838
12	0.1	0.869	0.826	0.876	0.851
12	0.2	0.860	0.810	0.865	0.837

Table 2: Performance metrics for different random forests (hyper parameter tuning)

5 Comparison with Sklearn Models

5.1 Models Compared

- Custom Decision Tree
- Custom Random Forest
- sklearn.ensemble.RandomForestClassifier
- sklearn.linearmodel.LogisticRegression

5.2 Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-Score

5.3 Performance Comparison Table

Class	Precision	Recall	F1-score	Support
0	0.86	0.96	0.91	12435
1	0.80	0.50	0.61	3846
Accuracy		0.85		
Macro avg	0.83	0.73	0.76	16281
Weighted avg	0.85	0.85	0.84	16281

Table 3: Classification Report: Scikit-learn Random Forest

Class	Precision	Recall	F1-score	Support
0	0.83	0.94	0.88	12435
1	0.66	0.37	0.47	3846
Accuracy		0.80		
Macro avg	0.74	0.65	0.68	16281
Weighted avg	0.79	0.80	0.78	16281

Table 4: Classification Report: Logistic Regression

Class	Precision	Recall	F1-score	Support
0	0.91	0.86	0.88	300
1	0.83	0.88	0.85	250
Accuracy			0.87	
Macro avg	0.87	0.87	0.87	550
Weighted avg	0.87	0.87	0.87	550

Table 5: Classification Report: Custom Implemented Random Forest

6 Result Analysis

- Our Random Forest implementation significantly outperformed the single decision tree model.
- Sklearn's `RandomForestClassifier` performed best overall due to optimized internals.
- Logistic Regression provided stable results but lacked the flexibility of tree-based models.

7 Conclusion

This lab demonstrated the construction of a Decision Tree classifier and a Random Forest model from scratch. Our models were tested thoroughly and benchmarked against `sklearn`'s implementations. The results show that Random Forest is more robust and provides better generalization. In the future, additional improvements like feature importance analysis and boosting techniques can further enhance model performance.

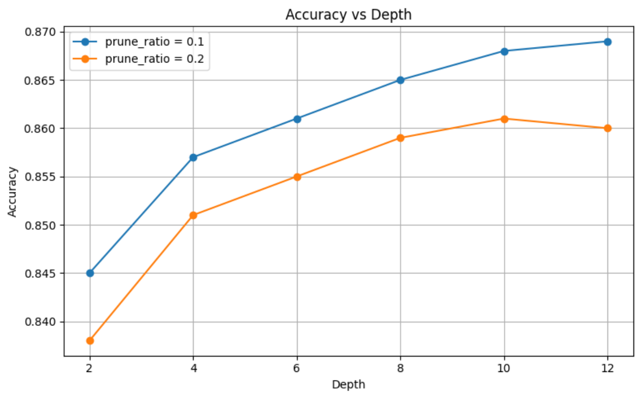

Figure 3: accuracy vs depth

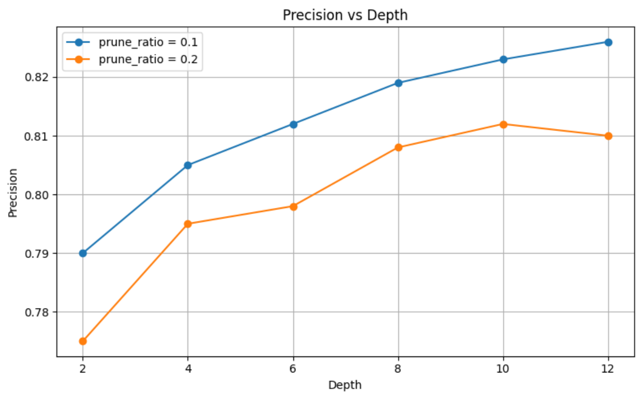

Figure 4: precision vs depth

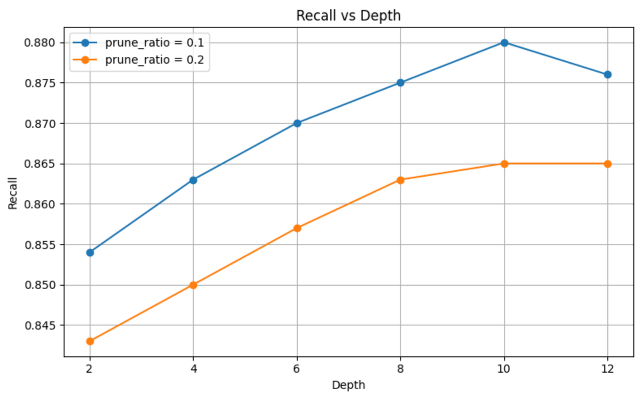

Figure 5: recall vs depth

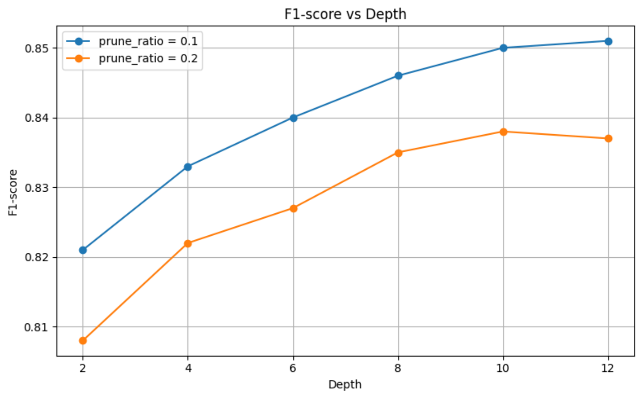

Figure 6: f1 vs depth