CS303 Project 2: Adult Census Income

12312110 Xuanran Li, *Department of Mathematics, SUSTech*

***Abstract*—This report presents for Project 2 of CS303 Artificial Intelligence. The objective is to predict whether an individual's annual income exceeds $50K based on the given data. We suppose Logistic Regression, Multi-Layer Perceptron and Random Forest, analyzing their performances and introducing data preprocessing strategies.**

I. Introduction

This project studies the Adult Census Income prediction problem, which is derived from the 1994 U.S. Census dataset and is widely used as a benchmark in machine learning. The task is to predict whether an individual's annual income exceeds $50K based on a set of employment-related attributes, such as age, education level, occupation, working hours, etc. From a machine learning perspective, this problem can be formulated as a binary classification task on structured tabular data containing both numerical and categorical features. The purpose of this project is to apply machine learning techniques learned in the course to a real-world dataset. Specifically, we aim to build a complete classification pipeline that includes data preprocessing, feature encoding, model training, and evaluation. Through this project, we seek to better understand how different components of a machine learning system work together and to gain hands-on experience in evaluating classification models on practical data.

II. Preliminary

In this project, we consider the Adult Census Income prediction task as a supervised learning problem, specifically a binary classification problem. Denote $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ as the training dataset, where $x_i \in \mathbb{R}^d$ represents the feature vector of the i-th individual, and y_i is the corresponding income label. Each feature vector x_i consists of both numerical attributes (e.g., age, capital gain, hours worked per week) and categorical attributes (e.g., education, occupation, marital status). The label y_i is a binary variable, which means $y_i \in \{0, 1\}$, where $y = 0$ indicates that the individual's annual income is less than or equal to $50K, and $y = 1$ indicates that the income is greater than $50K.

The goal of this project is to learn a classification function

$$f_\theta : \mathbb{R}^d \to \{0, 1\},$$

parameterized by θ, that maps input features to income labels. The parameter θ is learned from the training data by minimizing the empirical risk on the dataset, which is equivalent to maximizing the prediction accuracy on unseen data.

Model performance is evaluated using standard classification metrics, such as accuracy, precision rate, recall rate, and F1 score. Their definitions are as below:

$$\begin{aligned}
&\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_i = y_i),\\
&\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},\\
&\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},\\
&\text{F1 score} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned}$$

where TP, FP, and FN represent True Positives, False Positives, and False Negatives, respectively, and

$$\mathbf{1}(\hat{y}_i = y_i) = \begin{cases} 1, & \hat{y}_i = y_i, \\ 0, & \hat{y}_i \neq y_i. \end{cases}$$

III. Methodology

A. General Workflow

The overall procedure can be divided into the following three main steps:

1) **Data preprocessing**: The original dataset consists of 14 features, including 6 numerical features and 8 categorical features. The dataset contains 22,792 training samples and 9,769 test samples. Both the training and testing datasets contain missing values, denoted by the symbol "?". For categorical features, missing values are handled using mode imputation, where each missing entry is replaced with the most frequent category observed in the training data. For numerical features, missing values are also imputed using the most frequent value. All imputation statistics are learned exclusively from the training set and consistently applied to the test set to avoid data leakage. The training data includes 5,489 positive samples and 17,303 negative samples.
2) **Model training**: Supervised classification models are trained on the processed training data using labeled examples.
3) **Evaluation and prediction**: The trained model is evaluated using cross-validation on the training set and then applied to the test set to generate final predictions.

This workflow ensures that the same preprocessing steps are consistently applied during both training and inference, resulting in a reliable end-to-end prediction pipeline.

B. Detailed Algorithm and Model Design

We use Logistic Regression (LR) as a baseline and test the performances of MLP and Random Forest.

1) Logistic Regression: Logistic Regression[1] serves as a strong and interpretable baseline for binary classification. Instead of directly predicting class labels, LR estimates the posterior probability of the positive class through a parametric linear function followed by a sigmoid transformation. Formally, given an input vector $\mathbf{x} \in \mathbb{R}^d$, the model computes

$$p(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x} + b),$$

where $\mathbf{w}$ and b denote the learnable weight vector and bias term, respectively, and $\sigma(\cdot)$ is the sigmoid function.

Model parameters are obtained by minimizing the regularized binary cross-entropy objective using iterative gradient-based optimization. Due to the convexity, LR guarantees convergence to a global optimum and provides stable performance even under limited data regimes. However, its reliance on a linear decision boundary restricts its ability to capture higher-order feature interactions commonly present in tabular data.

Algorithm 1 Logistic Regression Training

```
Input: Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, learning rate $\eta$,
       number of iterations $T$
Output: Optimized weight vector $\mathbf{w}$ and bias $b$
Initialize $\mathbf{w}$ and $b$
for $t = 1$ to $T$ do
    Compute predictions $\hat{y}_i = \sigma(\mathbf{w}^\top x_i + b)$ for all samples
    Compute gradients of the binary cross-entropy loss
    Update parameters: $\mathbf{w} \leftarrow \mathbf{w} - \eta(\nabla_\mathbf{w} L + \lambda \mathbf{w})$
    $b \leftarrow b - \eta \nabla_b L$
end
return $\mathbf{w}, b$
```

2) Multi-Layer Perceptron: The Multi-Layer Perceptron (MLP)[2] models the conditional distribution $P(y \mid \mathbf{x})$ through a hierarchy of non-linear transformations. Denote $\mathbf{x} \in \mathbb{R}^d$ as the input feature vector. A L-layer MLP defines a function $f_\theta : \mathbb{R}^d \to (0, 1)$ parameterized by weights and biases $\theta = (\mathbf{W}^{(l)}, \mathbf{b}^{(l)})_{l=1}^L$.

The forward propagation is defined recursively as

$$\mathbf{a}^{(0)} = \mathbf{x}, \quad \mathbf{a}^{(l)} = \phi\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \ldots, L-1,$$

where $\phi(\cdot)$ denotes an activation function (ReLU in our implementation). The output layer applies a sigmoid function to produce a probabilistic prediction:

$$\hat{y} = f_\theta(\mathbf{x}) = \sigma\left(\mathbf{W}^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)}\right).$$

Training is formulated as the minimization of the empirical risk over the dataset $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^N$:

$$\min_\theta \frac{1}{N}\sum_{i=1}^N \ell\left(y_i, f_\theta(\mathbf{x}_i)\right) + \lambda \mathcal{R}(\theta),$$

where $\ell(\cdot, \cdot)$ is the binary cross-entropy loss and $\mathcal{R}(\theta)$ denotes a regularization term (e.g., ℓ_2-norm or weight decay).

Gradients $\nabla_\theta \ell$ are computed via backpropagation using the chain rule, and parameters are updated with stochastic gradient-based optimizers such as Adam[4]. This formulation enables the MLP to approximate highly non-linear decision boundaries, at the cost of increased sensitivity to optimization dynamics and model capacity.

Algorithm 2 MLP Training with Backpropagation

```
Input: Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, learning rate $\eta$
Output: Trained network parameters $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^L$
Initialize all weights and biases randomly
repeat
    Forward pass: compute activations layer by layer  Compute
      loss between predictions and ground-truth labels  Back-
      ward pass: compute gradients using chain rule  Update
      parameters using gradient-based optimization
until convergence;
return Optimized network parameters
```

3) Random Forest: Random Forest[3] constructs an ensemble predictor by aggregating a collection of randomized decision trees. Each tree corresponds to a function

$$h_t : \mathbb{R}^d \to 0, 1, \quad t = 1, \ldots, T,$$

trained on a bootstrap sample of the original dataset.

Let $\mathcal{D}_t$ denote the dataset obtained by sampling N instances from $\mathcal{D}$ with replacement. Each tree h_t is learned by recursively partitioning the feature space. At each internal node, a subset $\mathcal{S} \subset 1, \ldots, d$ of size m is drawn uniformly at random, and the optimal split is selected by minimizing an impurity criterion such as the Gini index $\mathrm{Gini}(\mathcal{D}) = 1 - \sum_{c \in \{0,1\}} p_c^2$, where p_c denotes the empirical class proportion in the node.

Formally, the Random Forest classifier is defined as

$$F(\mathbf{x}) = \mathrm{mode}\big(h_1(\mathbf{x}), \ldots, h_T(\mathbf{x})\big),$$

which approximates the Bayes optimal classifier by reducing variance through ensemble averaging.

From a probabilistic perspective, Random Forest can be interpreted as a Monte Carlo estimator over a distribution of randomized trees:

$$\mathbb{E}_{h \sim \mathcal{H}}[h(\mathbf{x})] \approx \frac{1}{T}\sum_{t=1}^T h_t(\mathbf{x}),$$

where randomness arises from both bootstrap sampling and feature subsampling. This dual source of randomness decorrelates individual trees and yields strong generalization performance, particularly on high-dimensional tabular data with complex feature interactions.

Algorithm 3 Random Forest for Binary Classification

```
Input: Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, number of trees
       $T$, number of features considered at each split $m$
Output: Trained Random Forest classifier $F$
for $t = 1$ to $T$ do
    Draw a bootstrap sample $\mathcal{D}_t$ from $\mathcal{D}$  Train a decision tree
      $h_t$ on $\mathcal{D}_t$  while tree $h_t$ is not fully grown do
        Randomly select $m$ features from all features  Choose
          the best split among the selected features according
          to an impurity measure (e.g., Gini index)  Split the
          node and grow the tree
    end
end
return $F(x) = \text{majority\_vote}\{h_t(x)\}_{t=1}^T$
```

C. Analysis

We analyze Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Random Forest (RF) in terms of optimality, computational complexity, and the key factors affecting their performance.

1) Computational Complexity: The training complexity of LR is $O(ndi)$, where n is the number of samples, d is the feature dimension, and i is the number of iterations, making it the most computationally efficient method.

RF has a higher complexity of approximately $O(nTd \log n)$, where T is the number of trees; the $\log n$ term arises from recursive split searching.

For MLP, the training cost is dominated by backpropagation and can be expressed as $O(nEP)$, where E is the number of epochs and P the total number of trainable parameters, resulting in the highest computational overhead among the three models.

2) Optimality: LR optimizes a convex objective function and therefore guarantees convergence to a global optimum. However, its optimality is confined to linear decision boundaries. RF does not solve a global optimization problem; each tree is constructed via greedy local impurity minimization. Nevertheless, the ensemble achieves strong generalization through variance reduction as the number of trees increases. MLP training corresponds to a non-convex optimization problem and converges only to local optima, with performance highly dependent on initialization, optimization strategy, and regularization.

3) Performance Factors: LR performs well when the data are close to linearly separable and is strongly influenced by feature scaling and regularization strength. RF performance is mainly determined by ensemble size, tree depth, and feature subsampling, and it excels at capturing non-linear and rule-based patterns in tabular data. MLP performance depends on model capacity and optimization stability; inappropriate architecture or insufficient regularization can easily lead to overfitting.

IV. Experiments

A. Setup

We conducted our experiments on Macbook Air (M3, 16GB RAM), with the environment `Python 3.12`.

B. Result and Analysis

The performance of the three models—Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Random Forest (RF)—was evaluated using accuracy, precision, recall, and F1-score. Table I summarizes the best experimental results achieved by each model on the test set.

1) Logistic Regression Analysis: As shown in Fig. 1, we analyzed the impact of the regularization strength C on the LR model. We observed that when C is very small (strong regularization), the model suffers from underfitting, resulting in lower recall and F1-score. As C increases, the performance metrics stabilize, achieving an accuracy of approximately 82.6%. The relatively low recall compared to precision indicates that the linear decision boundary of LR struggles to identify all high-income individuals in this imbalanced dataset.

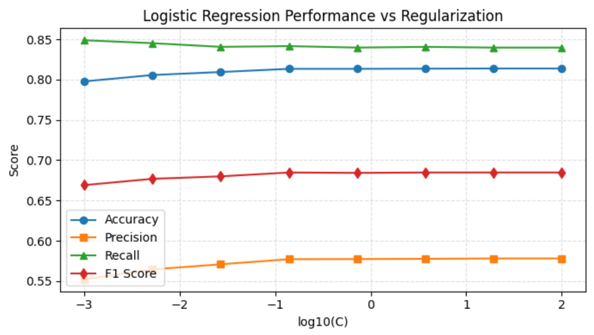

Fig. 1. Fig 1 illustrates how accuracy, precision, recall, and F1 score vary with the regularization strength C in logistic regression. The results reflect the trade-off between model bias and variance.

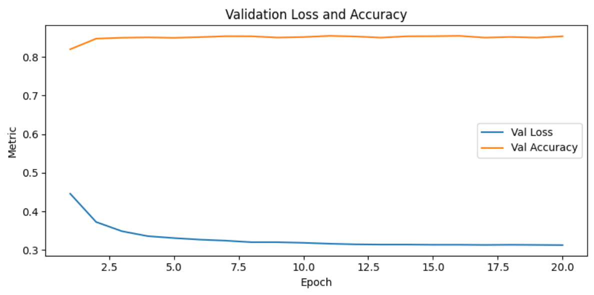

Fig. 2. Fig 2 shows that the MLP converges quickly, with validation loss decreasing sharply in the early epochs and then stabilizing, while validation accuracy consistently improves and remains high, indicating stable training and good generalization performance.

2) MLP Convergence: The training process of the MLP is illustrated in Fig. 2. The validation loss decreases rapidly within the first 10 epochs and plateaus after 30 epochs, suggesting efficient convergence. The gap between training and validation accuracy remains narrow, indicating that the dropout and L_2 regularization effectively prevented significant overfitting. The MLP outperforms the LR baseline by capturing complex non-linear interactions between features like education and occupation.

3) Random Forest Robustness: Fig. 3 demonstrates the relationship between the number of trees and model accuracy. The accuracy improves significantly as the ensemble size increases from 1 to 40, after which the performance gains diminish and converge around 85.8%. Random Forest achieved the highest overall performance among the three models. This superiority is attributed to its ability to handle both categorical and numerical data effectively while reducing variance through bootstrap aggregating.

C. Summary

In summary, while Logistic Regression provides a computationally efficient and interpretable baseline, ensemble methods RF and deep learning models MLP offer superior predictive power for the Adult Census dataset. Random Forest is identified as the optimal model for this task due to its robustness against noise and its ability to model complex feature dependencies without extensive hyperparameter tuning.

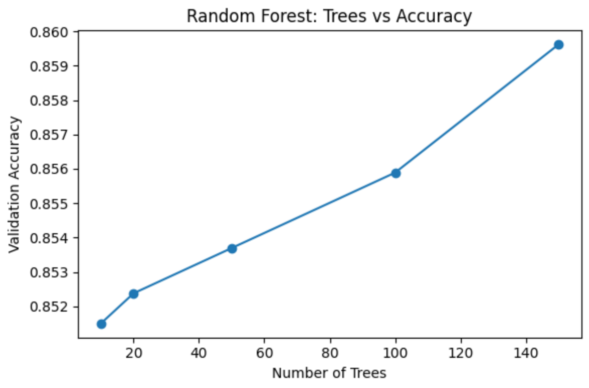

Fig. 3. Fig 3 shows the accuracy with respect to the number of trees in the random forest. The performance improves with increasing ensemble size and stabilizes once sufficient trees are used.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT MODELS

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Logistic Regression	82.63%	0.7121	0.4682	0.5651
MLP	85.12%	0.7435	0.5824	0.6531
Random Forest	**85.84%**	**0.7512**	**0.6015**	**0.6681**

V. CONCLUSION

In this project, we compared Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Random Forest (RF) on the Adult Census dataset. Experimental results (Figs. 1-3) align with our analysis: RF achieved the highest accuracy and F1-score, proving that ensemble tree-based models excel at capturing discrete decision boundaries in tabular data compared to the linear baseline of LR and the optimization-sensitive MLP. We realized that efficient implementation relies on vectorized operations and rigorous preprocessing, such as handling missing values and scaling. While the models performed well, the relatively lower recall suggests that future work should focus on addressing class imbalance via implementing automated hyperparameter tuning to further refine the predictive power of non-linear models.

REFERENCES

[1] D. R. Cox, “The regression analysis of binary sequences,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.
[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[3] L. Breiman, “Random forests,” *Machine learning*, vol. 45, pp. 5–32, 2001.
[4] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.