COMP 6721-Project Proposal: Adult Census Income Analysis

Dina Omidvar
Concordia University
40239883

Niloofar Tavakolian
Concordia University
40220767

Nastaran Naseri
Concordia University
40215694

SeyedehMojdeh Haghighat Hosseini
Concordia University
40171693

ABSTRACT

This report presents a study on income classification using a supervised approach utilizing a decision tree model, a semi-supervised approach, and a deep neural network model. The objective is to classify samples based on a dataset containing 15 features.

The decision tree model employs a hierarchical structure to make decisions based on specific features,[1] while the DNN model consists of interconnected layers with nonlinear activation functions. In addition, a semi-supervised learning algorithm is integrated, allowing the utilization of unlabeled data.

The dataset used in this study comprises 48,842 observations, capturing various socio-economic factors.[2] Among these features, the income variable is the target, indicating whether an individual's income exceeds a certain threshold.

The semi-supervised algorithm and DNN models demonstrate significant accuracy in classifying income levels. The accuracy obtained from the decision tree model, semi-supervised algorithm, and the DNN model are respectively, 81.12%, 73.24%, and 85.76%.

1 INTRODUCTION

Income classification is a critical task in various domains, such as finance, social sciences, and market research. The primary focus of this study is the classification of income based on a set of features.

During this project, we faced challenges in handling missing data, categorical data, and an unbalanced dataset during the preprocessing stage. We encountered missing values in the dataset, which we addressed by removing instances with missing values, ensuring complete data for training and minimizing potential bias.[3] Categorical features were effectively handled through one-hot encoding, enabling accurate capture of categorical information.[4] Additionally, we tackled class imbalance by employing a downsampling technique to balance the class distribution.[5]

In terms of model implementation, we encountered specific issues. The decision tree model exhibited overfitting, which we mitigated by adjusting hyperparameters such as tree depth and minimum samples.[6] By carefully adjusting parameters like max_depth, min_samples_leaf, and min_samples_split, we were able to prevent the decision tree from excessively fitting the training data, achieving a better balance between model complexity and generalization, and improving its performance on unseen data.

In terms of the DNN model, its high time complexity posed challenges in terms of training and inference speed. To mitigate this, we explored architectural modifications such as reducing the number of layers, which helped streamline the computational requirements of the model. By finding the right balance between model complexity and computational efficiency, we were able to enhance the training and inference speed of the DNN model without sacrificing its classification accuracy.

Overall, by addressing these challenges and fine-tuning our models, we aimed to achieve robust and efficient income classification results.

1.1 High-level Abstract Explanation of Methodology, Implementations, and Results

1.1.1 Decision Tree Model. A decision tree model was implemented to classify income based on the provided features. The architecture of the decision tree was designed to recursively split the data based on feature thresholds, resulting in a hierarchical structure for decision-making. The decision tree model provided interpretable rules for income classification.

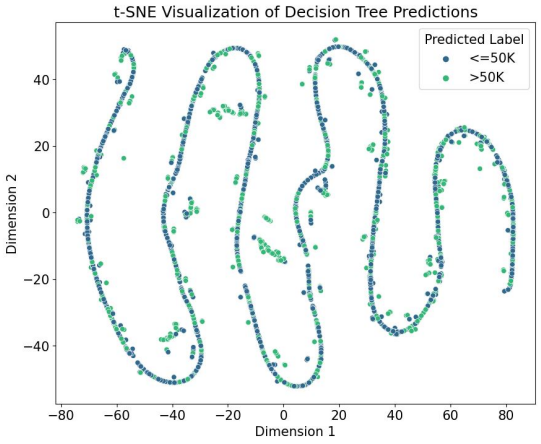


Figure 1: t-SNE Visualization of Decision Tree Predictions [7] [8]

1.1.2 Semi-Supervised Learning Model. In addition to the decision tree, a semi-supervised learning approach was employed to leverage both labeled and unlabeled data. The initial step of the semi-supervised learning process involved training a supervised learning model using the available labeled data. This model was then applied to the unlabeled data, and pseudo-labels were assigned to each unlabeled instance based on the model's predictions.[9]

To improve the quality of the pseudo-labeled data, only instances with high confidence were considered and were selected and combined with the original labeled data to form a new labeled subset. This was determined by evaluating the predicted probabilities of the model.

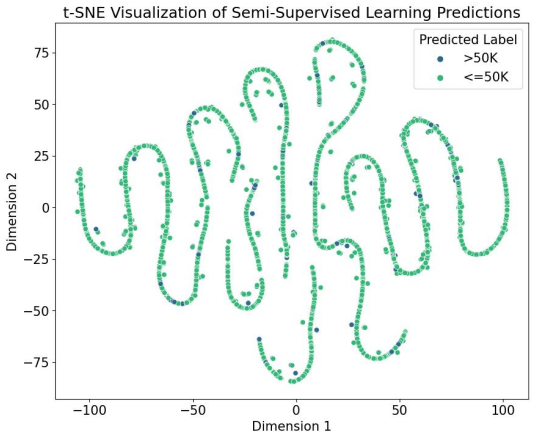


Figure 2: t-SNE Visualization Semi-Supervised Learning Predictions [7] [8]

1.1.3 Deep Neural Network (DNN) Model. A deep neural network model was also utilized to classify income. The architecture of the selected DNN was designed with multiple layers, allowing for the extraction of complex patterns and representations from the data. The DNN model leveraged non-linear transformations and learned hierarchical features to enhance the classification accuracy.

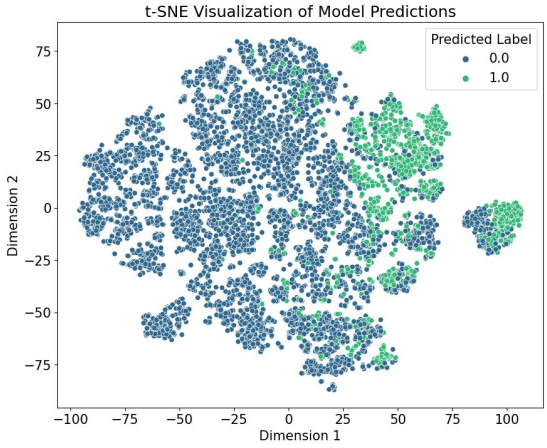


Figure 3: t-SNE Visualization of DNN Predictions [7] [8]

The performance of each model was evaluated using appropriate metrics such as accuracy, precision, recall, and F1-measure. The decision tree model provided interpretable rules but had limitations in capturing complex relationships. It achieved an accuracy of 81.12% on the test dataset. The DNN model, with its ability to extract intricate patterns, achieved the highest accuracy among the three models with 85.76%. Despite the promising results of the decision tree and DNN models, the semi-supervised learning approach yielded a slightly lower accuracy of 73.24%.

1.2 Related Works

The paper "Classification of Adult Income Using Decision Tree" by Roland Fiagbe from the University of Central Florida explores the application of decision tree methodology for predicting an individual's income and determining the factors that contribute to their income level. The study utilizes the Adult Income dataset from the UCI and applies classification metrics to evaluate the model's performance. The results demonstrate that the decision tree model performs well, achieving a good accuracy rate. From the model, the training accuracy is 85.85 and the test accuracy is 85.29%.[10]

2 METHODOLOGY

Our methodology involves explanations of implementing and evaluating our three models, namely Decision Tree, Semi-Supervised learning, and DNN, to address the income prediction problem.

2.1 Datasets

As discussed before, the dataset used for this project is the "Adult Census Income" dataset obtained from the UCI Machine Learning Repository on the Kaggle website.[2] It contains information about individuals, including various demographic and socio-economic features. The dataset consists of nearly 50,000 observations, and was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics).

The dataset includes the following columns: Age, Workclass (Indicates the type of workclass the individual belongs to), Fnlwgt (Stands for "final weight" and is a sampling weight assigned to each observation in the dataset. This feature represents how many people have the same list of features.), Education, Education.num, Marital.status, Occupation (Specifies the type of occupation the individual is engaged in), Relationship (Represents the relationship status of the individual in the household), Race, Sex, Capital.gain (Indicates the capital gains for the individual.), Capital.loss (Represents the capital losses for the individual.), Hours.per.week, Native.country, Income (income level, which is the target variable has two classes. Values ">50K" indicating an income above $50,000 per year and "<=50K" indicating an income of $50,000 or below per year.)

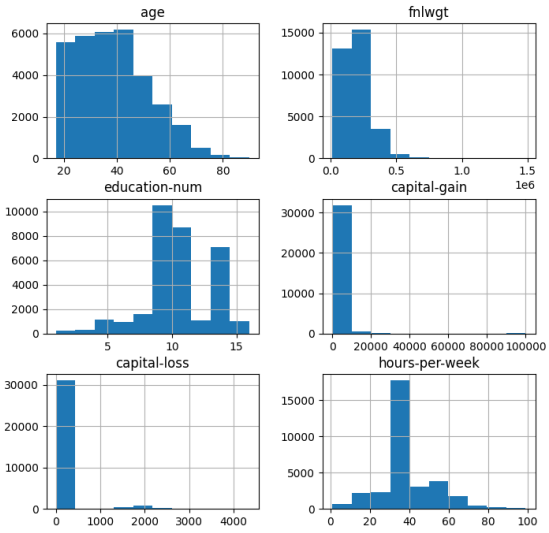
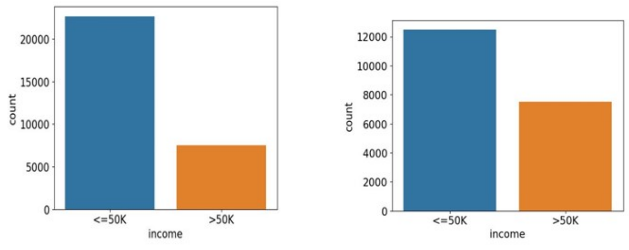


Figure 4: Visualizing the numerical features of the dataset using histograms [8]

2.1.1 Challenges during the pre-processing step. As mentioned before, one of the challenges encountered in this study was handling missing values in the dataset.[3] During data exploration, it was observed that some features contained missing values. To determine the appropriate strategy for dealing with missing values, the percentages of missing values were examined. Since the percentage of missing values was found to be less than 3% (specifically 1.79%), these instances were removed from the dataset. By removing the instances with missing values, we ensure the availability of complete data for model training and minimize any potential bias or misinterpretation in the training process.

Another challenge involved dealing with categorical features in the dataset. Categorical features, such as education, occupation, relationship, etc require appropriate encoding to be utilized effectively in the models. In this study, one-hot encoding was employed to transform categorical features into a binary vector representation.[4] This encoding technique allows for a comprehensive representation of categorical data, enables the models to capture the categorical information accurately, and avoids any ordinal assumptions that might bias the predictions.

Class imbalance is another issue commonly encountered in income classification tasks, where one class (high-income individuals) may be significantly underrepresented compared to the other class (low-income individuals). To address this challenge, a downsampling technique was employed to balance the class distribution. Downsampling involves randomly removing instances from the majority class until the class distribution becomes more balanced. By applying this technique, we aim to ensure that the models are not biased towards the majority class and can effectively learn from both classes.



(a) Imbalance dataset **(b) Balanced dataset**

Figure 5: Comparison of datasets [8]

Considering the existing solutions, each approach used in this study has its own advantages and disadvantages. For instance, removing instances with missing values (less than 3%) helps maintain a clean dataset and avoids potential biases caused by imputing missing values. However, this approach may result in a reduction in the overall dataset size and potential loss of information. Or, one-hot encoding of categorical features allows the models to capture the non-linear relationships within these features accurately and understand the relationships between different categories.[4] Nevertheless, it leads to an increase in the dimensionality of the dataset, which can affect the computational complexity and model training time. Also, downsampling the majority class helps address the class imbalance problem and prevent the models from being biased towards the majority class. However, downsampling may result in the loss of some information from the majority class and may lead to a decrease in overall model performance and potentially reducing the model's ability to generalize accurately.

By considering these pros and cons, we aimed to strike a balance between data quality, model complexity, and overall performance to ensure reliable income classification results.

2.2 Decision Tree Model

2.2.1 Architecture of Supervised Learning Model. For the Decision Tree part, we first identified the target variable as 'income' and selected a set of features to train the classifier. The dataset was split into training, validation, and test sets with a distribution of 60%, 20%, and 20% respectively. One-hot encoding was applied to the feature matrix to convert categorical variables into numerical representations. Next, a Decision Tree classifier was created and trained on the training set using the specified hyperparameters. The maximum depth was set to 11, while the minimum number of samples per leaf and for splitting were set to 4 and 2 respectively.

For evaluation, the trained classifier was used to make predictions on the validation set. The accuracy of the model on the validation set was calculated and reported. Subsequently, predictions were made on the test set, and the accuracy of the model on the test set was also calculated. To provide visual insights into the decision tree structure, the plot_tree function from scikit-learn was utilized to generate a visualization. Furthermore, a classification report was printed, including precision, recall, and F1-score for each class, as well as the overall accuracy.

2.2.2 Architecture of Semi Supervised Learning Model. The semi-supervised learning algorithm follows a two-step process to leverage labeled and unlabeled data. Initially, the dataset is divided into high-income and low-income groups, ensuring equal representation in both labeled and unlabeled datasets through a 50-50 split for each group. These datasets are then one-hot encoded to enable numerical processing.

In the iterative semi-supervised learning loop, a decision tree classifier is trained on the labeled data and subsequently used to predict labels for the unlabeled data. Confident predictions are identified using a predefined confidence threshold. The confident samples are added to the labeled data, updating the encoding, and improving the model's understanding. The process is repeated for a specified number of iterations, evaluating the accuracy of the combined dataset at each step. The final classification report provides comprehensive metrics, including precision, recall, F1-score, and accuracy, to assess the model's performance on the unlabeled data. [9]

2.3 Architecture of Deep Neural Network

The dataset was preprocessed by handling missing values, renaming columns, and cleaning up the occupation column. Categorical variables were one-hot encoded, while numerical variables were standardized. The dataset was split into train and test sets, with 70% and 30% of the data, respectively. A neural network model was then constructed using the PyTorch framework. The model consisted of four fully connected layers with ReLU activation and a dropout layer to prevent overfitting. The model was trained using

the training set for 10,000 epochs, with a stochastic gradient descent optimizer and a binary cross-entropy loss function.[11] [12]

The performance of the model was evaluated using classification metrics such as precision, recall, and F1-score. The results showed promising performance on the test set with an accuracy of 85.76%. The model demonstrated good precision and recall for both income classes, indicating its ability to effectively classify individuals into the two income categories. In conclusion, the developed neural network model successfully classified adult census data into income categories. The model's performance highlights its potential in assisting decision-making processes related to socioeconomic factors.

Also, during the training of the Deep Neural Network (DNN) model, we measured two important aspects: the number of floating-point operations (FLOPs) and the training run time. FLOPs refer to the number of mathematical operations, specifically floating-point operations, required to perform the computations within the DNN model.

To calculate the FLOPs, we consider the number of operations involved in each layer of the DNN model, including matrix multiplications, additions, activations, and other mathematical operations. By summing up the operations across all layers, we obtain an estimation of the total number of FLOPs required for one complete forward and backward pass through the model. In our specific implementation, the DNN model utilized a total of 18,881 FLOPs, reflecting the complexity of the network architecture and the computational demands of the dataset.

In addition to FLOPs, we also measured the training run time for each epoch, which provides insights into the computational efficiency of the training process. The training run time for the DNN model for each epoch was determined to be on average about 14 seconds and in total 1,068.73 seconds, equivalent to approximately 17 minutes and 48 seconds.

It is important to note that the FLOPs and training run time may vary depending on the specific hardware, software, and optimization techniques employed during the training process. These metrics offer valuable information about the computational complexity and efficiency of the DNN model, allowing us to assess its performance and scalability.

2.4 Optimization Algorithm

In this study, we evaluated the performance of two optimization algorithms, Adam and Stochastic Gradient Descent (SGD), on our Adult census income dataset.[13] [14] Our findings revealed that SGD outperformed Adam in terms of classification accuracy. This can be attributed to several factors inherent to our dataset. Firstly, the moderately sized nature of our dataset allowed SGD to effectively navigate the parameter space and converge to a better solution. Secondly, SGD's stochastic nature, updating parameters based on mini-batches of data, potentially facilitated better exploration of the solution space. Moreover, the moderate dimensionality of our dataset, coupled with SGD's suitability for such scenarios, likely contributed to its superior performance.

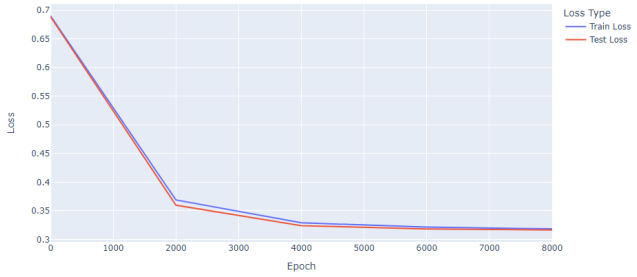


Figure 6: Training and Testing's Losses per Epoch of DNN with SGD Optimizer

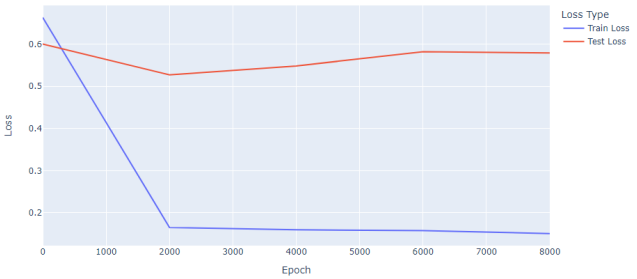


Figure 7: Training and Testing's Losses per Epoch of DNN with Adam Optimizer

3 RESULTS

For the Decision Tree part, we conducted experiments to optimize the hyperparameters and architecture of the model. Specifically, we focused on parameters such as the maximum depth (max_depth), the minimum number of samples required to be at a leaf node (min_samples_leaf), and the minimum number of samples required to split an internal node (min_samples_split).

To determine the best configuration, we performed a trial-and-error process, systematically varying these hyperparameters. We tested different values for max_depth, min_samples_leaf, and min_samples_split, evaluating the performance of the Decision Tree classifier on the test set for each combination. After careful experimentation, we found that the optimal hyperparameter values for the Decision Tree model were as follows: max_depth=11, min_samples_leaf=4, and min_samples_split=2. These values were determined based on their ability to achieve the highest accuracy and generalization performance on the validation set.

By selecting these specific hyperparameter values, we ensured that the Decision Tree model had an appropriate depth, effectively captured the underlying patterns in the data, and avoided overfitting. These optimized hyperparameters allowed the model to make accurate predictions on the test set, achieving superior performance compared to other configurations we tried.

Table 1: Comparative Performance at Different max_depth Levels

max_depth	5	8	11
Accuracy	79.1	80.75	81.12

Table 2: Evaluation of Model's Performance (Supervised learning Classification with Decision Trees) [8]

	Precision	Recall	F1-score	Support
≤50K	0.88	0.82	0.85	2516
>50K	0.72	0.81	0.76	1484
Accuracy			0.81	4000
Macro Avg	0.80	0.81	0.80	4000
Weighted Avg	0.82	0.81	0.81	4000

In the semi-supervised learning approach, we explored various hyperparameters and thresholds to optimize the performance of the Decision Tree classifier. Specifically, we focused on adjusting the maximum depth (max_depth) of the decision tree as well as the confidence threshold (confidence_threshold) used to identify confident predictions.

Regarding the confidence threshold, we investigated the effect of it on the performance of the semi-supervised learning algorithm. We experimented with different threshold values, both high and low values, to understand their impact on the quality of the confident predictions.

Based on our experimentation, we determined that a confidence threshold of 0.99 yielded the most favorable results. This threshold allowed the algorithm to identify confidently predicted samples without being overly restrictive or lenient, striking a balance between precision and recall. By continually updating the labeled dataset with confident predictions, the model improved its understanding of the data and achieved higher accuracy and performance on the unlabeled data. The classification report provided comprehensive evaluation metrics, including precision, recall, F1-score, and accuracy, offering a comprehensive assessment of the model's performance.

Table 3: Comparative Performance at Different confidence_threshold

confidence_threshold	0.70	0.80	0.99
Accuracy	50.46	58.90	73.24

Table 4: Evaluation of Model's Performance (Semi-supervised learning Classification with Decision Trees) [8]

	Precision	Recall	F1-score	Support
≤50K	0.72	0.93	0.81	4605
>50K	0.77	0.40	0.53	2738
Accuracy			0.73	7343
Macro Avg	0.75	0.66	0.67	7343
Weighted Avg	0.74	0.73	0.71	7343

In our investigation of the DNN model, we experimented with different model architectures and explored the impact of varying the learning rate [15]. By adjusting the learning rate from 0.06 to 0.1, we observed an improvement in performance, with the best results obtained at a learning rate of 0.1. The increase in learning rate [15] likely facilitated faster convergence and allowed the model to explore the parameter space more effectively, leading to better optimization and higher accuracy.

Furthermore, we compared the performance of two different optimizers: Adam and SGD. As we discussed previously, the SGD optimizer yielded better results compared to Adam, indicating that the stochastic gradient descent optimization approach was more suitable for our specific classification task.

Table 5: Comparative Performance at Different Learning Rate

Learning Rate	0.1	0.5	0.6
Accuracy	85.76	85.12	84.97

Table 6: Evaluation of Model's Performance (DNN) [8]

	Precision	Recall	F1-score	Support
≤50K	0.8845	0.9345	0.9088	7415
>50K	0.7488	0.6155	0.6757	2354
Accuracy			0.8576	9769
Macro Avg	0.8167	0.7750	0.7922	9769
Weighted Avg	0.8518	0.8576	0.8526	9769

REFERENCES

[1] scikit-learn developers, "Decision trees," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/tree.html.

[2] , "Adult census income," *kaggle*, https://www.kaggle.com/datasets/uciml/adult-census-income?resource=download.

[3] scikit-learn developers, "Handling missing data," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/impute.html.

[4] J. Brownlee, "Why one-hot encode data in machine learning?" 2020, https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[5] scikit-learn developers, "Resampling methods," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/classes.html#module-sklearn.utils.

[6] ——, "Hyperparameter tuning," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection.

[7] ——, "Understanding the impact of learning rate on neural network performance," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[8] D. Omidvar, N. Tavakolian, N. Naseri, and S. H. Hosseini, "Project github," https://github.com/dinaomidvartehrani/Applied-AI-.git.

[9] J. Brownlee, "Semi-supervised learning with label propagation," 2020, https://machinelearningmastery.com/semi-supervised-learning-with-label-propagation/.

[10] R. Fiagbe, "Classification of adult income using decision tree," 2023.

[11] V. K. Solegaonkar, "Introduction to pytorch," 2019, https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3.

[12] OLCAY_BOLAT, "Deep learning classification with pytorch," 2023, https://www.kaggle.com/code/olcaybolat1/deep-learning-classification-w-pytorch.

[13] scikit-learn developers, "Stochastic gradient descent," *scikit-learn documentation*, 2023, https://scikit-learn.org/stable/modules/sgd.html.

[14] J. Brownlee, "Gentle introduction to the adam optimization algorithm for deep learning," 2021, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.