Predicting Income Using Classical Machine Learning Models, XGBoost, and a Neural Network

Nakul Mody, Jesse Oh

1 Introduction and Project Statement

The goal of this project is to evaluate how different machine learning models perform on structured, tabular data. Specifically, the study uses the UCI Adult Census Income Dataset to predict whether an individual earns more than $50,000 per year based on demographic and employment characteristics.

The central research question guiding the project is:

> Do classical tree-based machine learning algorithms—such as Random Forests and XGBoost—outperform feed-forward neural networks on tabular data, and why?

Tree-based models are often regarded as the strongest performers on structured datasets due to their ability to naturally capture nonlinearities and interactions between features. Neural networks, on the other hand, typically excel in high-dimensional or unstructured domains such as images, text, and audio. This project systematically compares both approaches using consistent preprocessing, evaluation metrics, and hyperparameter tuning.

2 Data Sources and Technologies Used

2.1 Data Source

The dataset used for this project is the UCI Adult Income Dataset, sourced from the U.S. Census Bureau (1994). It contains approximately 48,000 individuals with 14 features, including:

- Numerical variables: age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week
- Categorical variables: marital-status, occupation, workclass, education, race, sex, native-country

- Target variable: income ($\leq$ 50K or $>$ 50K)

The dataset was downloaded from the UCI Machine Learning Repository. The project notebook merges the provided `.data` and `.test` files into a unified `adult.csv` file.

2.2 Technologies Used

The following technologies were used throughout the project:

- Python
- scikit-learn (Random Forests, preprocessing tools, GridSearchCV)
- XGBoost (if available in the environment)
- TensorFlow (feed-forward neural network)
- pandas and numpy (data manipulation)

2.3 Tools and Frameworks

- **OneHotEncoder** for categorical variables
- **StandardScaler** for neural network preprocessing
- **train_test_split** for reproducible dataset partitioning
- Jupyter Notebook in a TACC Docker environment

3 Methods Employed

3.1 Data Preprocessing

- One-hot encoding was applied to all categorical features using `OneHotEncoder`.
- Numerical features were passed through unchanged for tree-based models.
- Neural networks required feature standardization using `StandardScaler`.
- Missing or unknown values (denoted "?") were removed or imputed according to common dataset practice.

3.2 Model Families Tested

3.2.1 Classical Machine Learning Models

- Random Forest Classifier (baseline)
- Tuned Random Forest Classifier using GridSearchCV
- XGBoost Classifier
- Neural Network

3.2.2 Neural Network Model

A feed-forward Multilayer Perceptron (MLP) was implemented using TensorFlow/Keras. The architecture included:

- Dense hidden layers with ReLU activations
- Sigmoid output layer
- Adam optimizer

Hyperparameters tuned included number of layers, layer width, learning rate, batch size, and number of training epochs.

3.3 Evaluation Metrics

All models were evaluated using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion matrix

The dataset was split using an 70/30 train-test split with a fixed random seed.

3.4 Hyperparameter Tuning

- Classical models were tuned using **GridSearchCV**.
- Neural network hyperparameters were tuned manually through controlled experiments.

4 Results

4.1 Model Performance Summary

Typical performance values observed during experimentation are shown in Table 1.

Model	Accuracy	Notes
Random Forest (baseline)	~ 0.849	Minor improvement after tuning
Random Forest (Tuned)	~ 0.855	Minor improvement after tuning
XGBoost (GridSearch CV)	~ 0.86	Best overall performance
Neural Network (MLP)	0.84–0.85	Competitive but slightly lower

4.2 Analysis

Tree-based models outperformed the neural network, consistent with expectations for structured tabular datasets. XGBoost achieved the highest performance due to its ability to capture nonlinear feature interactions and robustness to unscaled inputs.

The neural network required significant tuning and computational time but did not surpass the performance of the tuned Random Forest or XGBoost. This result supports the widely observed trend that neural networks are not always optimal for traditional structured datasets.

4.3 Feature Importance

Tree-based model interpretations revealed that key predictors included:

- education-num
- marital-status
- capital-gain
- age
- hours-per-week

These features align with known socioeconomic factors influencing income.

5 References

1. UCI Machine Learning Repository, Adult Dataset.
`https://archive.ics.uci.edu/ml/datasets/adult`

2. Breiman, L. "Random Forests." *Machine Learning*, 2001.

3. Chen, T. and Guestrin, C. "XGBoost: A Scalable Tree Boosting System." *KDD*, 2016.

4. TensorFlow Documentation.
   `https://www.tensorflow.org/api_docs`

5. scikit-learn Documentation.
   `https://scikit-learn.org/stable/`