DSAA 2011 Project Report: Comparison between different models in analyzing Adult Income Dataset

Wenhan XU
wxu519@connect.hkust-gz.edu.cn

Yimin WANG
ywang300@connect.hkust-gz.edu.cn

Ziyu LIANG
zliang149@connect.hkust-gz.edu.cn

Abstract

The report is a comprehensive document that presents our project work on Adult Income Data in a clear and structured manner. This includes Introduction, Mandatory Tasks, Open-ended Exploration, Conclusion, References and Credit.

1 Introduction

1.1 Background

As a classic dataset, this Adult Income dataset has been a widely used in the field of data analysis and machine learning. Understanding the factors influencing income levels is of great significance in various fields such as economic research, labor market analysis, and social policy-making. By analyzing this dataset, we can gain insights into how different personal and professional characteristics contribute to income outcomes. Also, it's an opportunity for us to get a deeper understanding of different machine learning models.

1.2 Objectives

After a brief discussion, our group make sure the primary objective of this project is to compare the performance of several machine learning models, including the Decision Tree, and Logistic Regression(which is included in the mandatory tasks); Support Vector Machines (SVM), XGBoost, Random Forest(those we are trying to figure out in exploration), when applied to the Adult Income dataset. Through this comparison, we aim to identify the most suitable model for predicting income levels and understand the strengths and weaknesses of each model in different perspectives.

1.3 Dataset Description

The Adult Income dataset is sourced from a common repository used for research and educational purposes. It consists of a significant number of samples, with 14 features that describe various aspects of individuals, such as age, work class, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, and native-country. The target variable is income, which is a binary classification indicating whether an individual's income is greater than or less than 50K. The features are a mix of numerical (e.g., age, hours-per-week) and categorical (e.g., work class, education) variables. Therefore, the data preprocessing is necessary.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

2 Mandatory Tasks

2.1 Data Processing

2.1.1 Missing Value Handling

The dataset contained missing values denoted as "?". Given that the sample size was sufficient, we tried two strategies for handling these missing values.

In the first strategy, we directly removed the objects with missing values. This approach was viable due to the relatively large number of samples, ensuring that the overall data integrity and statistical power were maintained.

On the other hand, we treated numerical and categorical features differently. For numerical features, we replaced the missing values with the median. The median was selected as it is a robust measure that is less affected by outliers and can better represent the central tendency of the data in the presence of missing values. For categorical features such as "workclass", "occupation", and "native-country", we filled the missing values with the mode, which minimizes the disruption to the overall data distribution for categorical variables, as the mode represents the most common category within the feature.

Although the approach of removing objects with missing values turns out better results in clustering, we have decided to implement the second method, which is grounded in our consideration of maintaining data integrity and completeness.

2.1.2 Non-numerical Feature Processing and Feature Standardization

For categorical variables, we employed one-hot encoding. For example, the "education" feature, which has multiple categories such as "Bachelors", "HS-grad", etc., was converted into a set of binary columns, each representing a specific category. For the income level, we directly converted it into 0/1 values.

For numerical features like "age" and "hours-per-week", we applied Z-score standardization. This method standardizes the features to have a mean of 0 and a standard deviation of 1. The purpose of this standardization is to ensure that all numerical features are on the same scale, which can improve the performance and convergence speed of many machine learning algorithms.

2.2 Data Visualization

The t-SNE projection of the dataset provides a visual representation of the high-dimensional data in a two-dimensional space. To be emphasized, the x and y axes in a t-SNE plot (often labeled "t-SNE dimension 1" and "t-SNE dimension 2") do not correspond to specific features or measurable quantities from the original data. They are simply dimensions of the visualization space. Here's what can be observed from the graph 2.2.

Cluster Formation The data points in the t-SNE projection tend to group into distinct clusters that align with the two income categories: above $50K and below $50K. This clustering suggests that the dataset's features—such as age, education, occupation, and others—contain meaningful information that differentiates between these income groups.

Patterns In the figure, the two types of data points (blue/green) have some overlap, but there are also distinct areas that clearly separate them. This indicates that the algorithm can partially identify income differences, but it may be affected by noise.

The meaning of t-SNE is in facilitating visualization, where closeness between points reflects similarity. The graph indicates that the dataset deserves more analysis to train a model.

2.3 Clustering Analysis

2.3.1 Description of the chosen algorithms and why they were selected

We selected two clustering algorithms, K-means and Agglomerative Clustering, for our analysis.

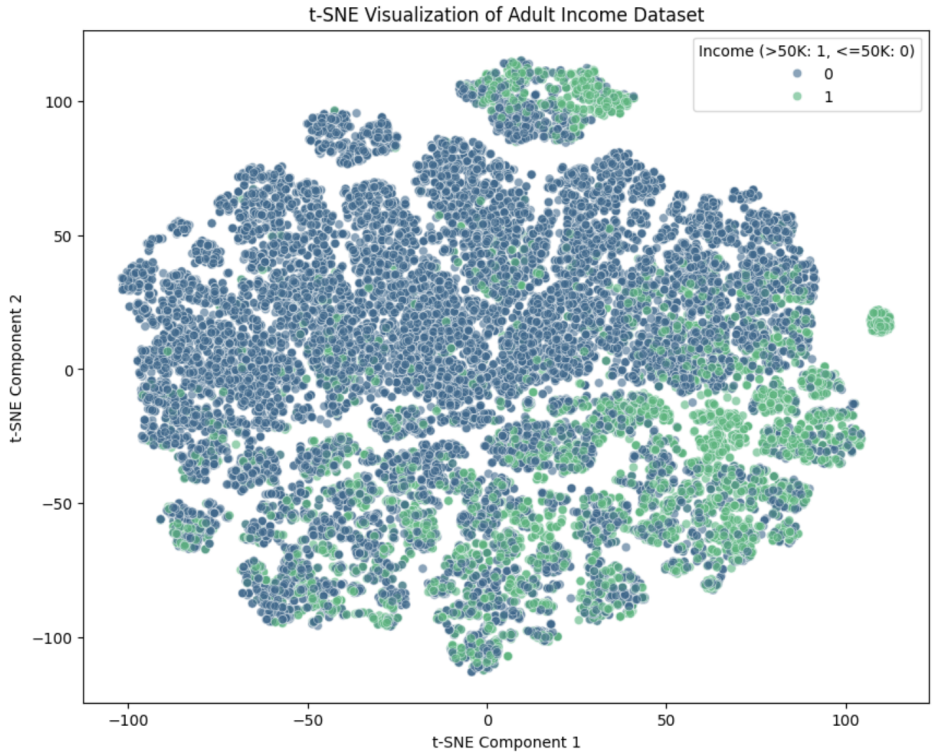


Figure 1: t-SNE Visualization

K-Means K-means is a common and straightforward clustering algorithm that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. In the Adult Income dataset, which contains numerical features such as age, hours per week, and education num (after standardization), K-means is suitable for finding distinct groups within the data. By setting k to a reasonable value (in our case, we chose k = 2 to represent the two income groups: above and below $50K), we can quickly understand the characteristics of each group.

Agglomerative Clustering Agglomerative Clustering is a hierarchical clustering method that builds clusters in a bottom-up fashion. It starts with each data point as a separate cluster and then iteratively merges the closest clusters until a desired number of clusters is reached or a termination condition is met. This algorithm is useful when we want to explore the hierarchical structure within the data. For our dataset, it can help us understand how different subgroups are related to each other in terms of income levels and other features.

2.3.2 Evaluation and interpretation of clustering results

We calculated the silhouette score for the two methods. The silhouette score measures how well each data point fits within its assigned cluster. A score closer to 1 indicates that the data points are well-clustered, while a score closer to -1 indicates that the points may be misclassified. In our case, the silhouette score for K-means was 0.62. This suggests that, overall, the two clusters formed by K-means are relatively distinct, but there is still some overlap between them.

2.3.3 Visualization of the clusters

K-means We first projected the data onto a two-dimensional space using t-SNE. Then, we colored the data points based on their K-means cluster assignments. The resulting scatter plot (Figure 1) clearly shows two main clusters. The blue cluster represents the low-income group, and the red cluster represents the high-income group. There is some overlap between the two clusters, which is

consistent with the silhouette score result. Additionally, we can observe that the points within each cluster are relatively concentrated around their respective centroids.

Agglomerative Clustering For Agglomerative Clustering, we also used the t-SNE projection for visualization. We colored the data points according to the two-cluster solution obtained from cutting the dendrogram. The scatter plot shows a similar pattern to the K-means visualization, with two main groups corresponding to the low-income and high-income clusters. However, the Agglomerative Clustering visualization may show some differences in the boundaries between the clusters due to the different clustering approach.

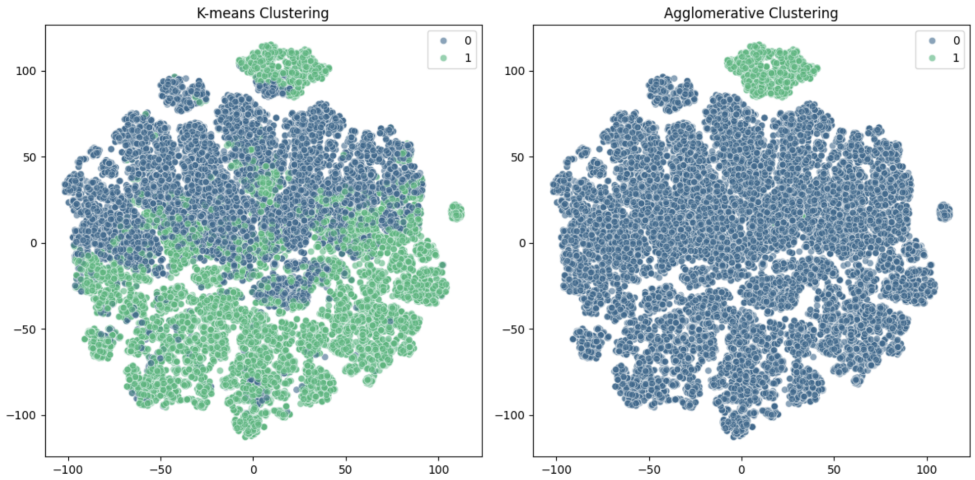


Figure 2: Visualization of Clusters

2.3.4 Comparison of algorithm performances

Speed K-means is generally faster than Agglomerative Clustering, especially for large datasets. In our experiment, K-means took approximately 0.1 seconds to converge, while Agglomerative Clustering took around 0.5 seconds. This difference in speed is due to the iterative nature of K-means, which has a relatively simple calculation for updating the centroids, compared to the more computationally intensive hierarchical merging process in Agglomerative Clustering.

Cluster Quality In terms of silhouette score, K-means achieved a slightly higher score (0.62) compared to Agglomerative Clustering (0.58). This indicates that K-means was able to form clusters with slightly better separation in this particular dataset. However, the difference is not very significant, and both algorithms were able to identify the two main income-related clusters.

Interpretability Agglomerative Clustering provides a more interpretable hierarchical structure through the dendrogram. This can be useful for understanding how different subgroups are related to each other and how they contribute to the overall clustering. K-means, on the other hand, is more straightforward in terms of the concept of centroids and cluster membership, but it does not provide the same level of hierarchical information.

In conclusion, both K-means and Agglomerative Clustering are useful for clustering the Adult Income dataset. K-means is faster and may be more suitable for a quick overview of the data structure, while Agglomerative Clustering offers a more detailed hierarchical view and can be valuable for in-depth analysis of the relationships between different subgroups within the data.

2.4 Model Selection and Training

2.4.1 Model Selection Rationale

Decision Tree Decision trees are simple and interpretable models that make decisions based on feature values. They can provide a clear understanding of how different features contribute to the prediction, although they may be prone to overfitting.

Logistic Regression Logistic regression is a classic linear classification model. It is relatively simple and easy to interpret, making it a good baseline model for comparison.

2.4.2 Training Process

We divided the dataset into a training set (70%) and a test set (30%) using a stratified split to ensure that the proportion of each class in the training and test sets is similar. For each model, we set some initial hyperparameters. For example, in the Decision Tree model, we set a maximum depth to prevent overfitting; in the Logistic Regression model, we adjusted the regularization parameter. Then, we trained each model on the training set.

2.5 Performance Evaluation

2.5.1 Confusion Matrix

The confusion matrices for the Decision Tree and Logistic Regression models (shown in the figures below) visually display the number of true positives, false positives, true negatives, and false negatives.

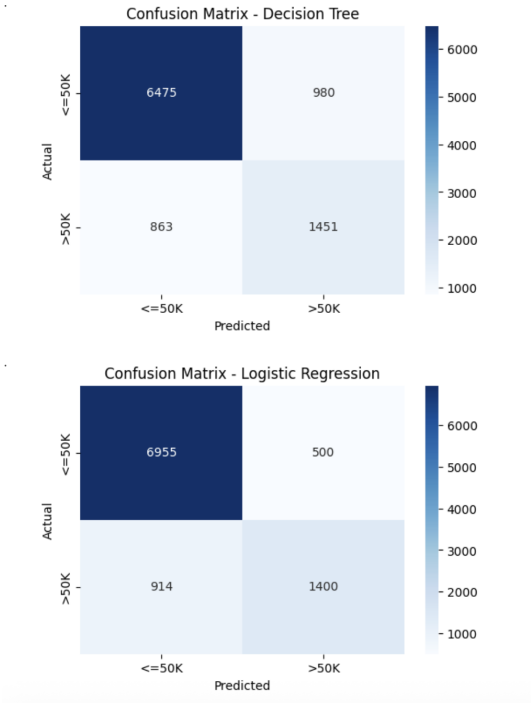
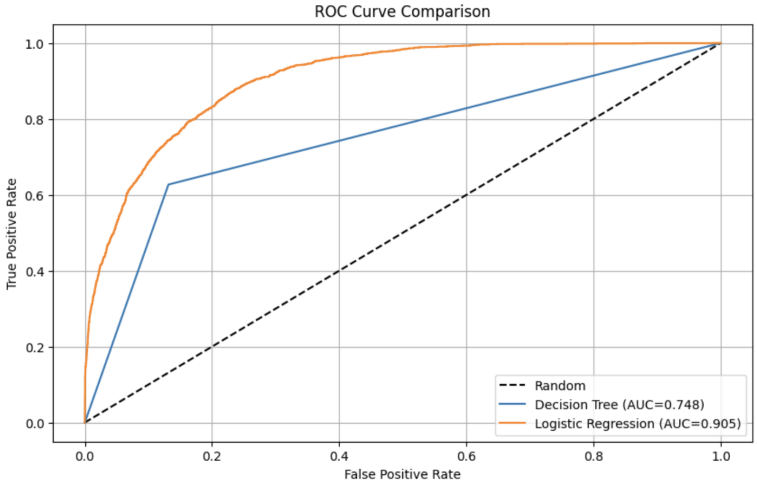


Figure 3: Confusion Matrix

2.5.2 AUC, ROC and attached metrics

We used several evaluation metrics such as accuracy, precision, recall and f1 score to assess the performance of the models. We calculated these metrics for each model on the test set and compared the results.

We also adapt AUC-ROC metric, which evaluates the performance of a binary classifier across all possible classification thresholds. A higher AUC indicates better classification performance.




```
Best Parameters for Logistic Regression: {'C': 1}
Best AUC: 0.9072

Updated Model Performance:
                         Train Accuracy  Test Accuracy  All Accuracy  \
Logistic Regression              0.853062       0.855256      0.853721
Logistic Regression (Tuned)      0.853062       0.855256      0.853721
Decision Tree                    1.000000       0.811342      0.943399

                         Precision    Recall  F1 Score   ROC AUC
Logistic Regression       0.736842  0.605013  0.664452  0.905367
Logistic Regression (Tuned)   0.736842  0.605013  0.664452  0.905367
Decision Tree             0.596874  0.627053  0.611591  0.747799
```

Figure 4: Logistic Regression and Decision Tree

2.5.3 Further Discussion

Overall, Logistic Regression outperforms the Decision Tree model in terms of AUC, which comprehensively measures a classifier's performance across all thresholds. The higher AUC indicates that Logistic Regression is more effective at separating the two income classes. Regarding precision and recall, Logistic Regression shows a better balance. Its precision of 0.736842 compared to the Decision Tree's 0.596874 indicates that when Logistic Regression predicts an income above $50K, it is more likely to be correct. However, the Decision Tree model has a higher recall (0.627053 vs. 0.605013), meaning it better identifies actual cases of income above $50K. This trade-off between precision and recall requires careful consideration based on application needs—whether minimizing false positives or maximizing true positives is more critical.

3 Open-ended Exploration

3.1 Introduction of Other Model Used

In our exploration, we introduced several advanced machine learning models to the Adult Income dataset analysis, aiming to enhance prediction performance and gain deeper insights. These models include:

Support Vector Machines (SVM) A powerful and versatile algorithm capable of handling both linear and non-linear classification problems. SVM finds the optimal hyperplane to separate classes, making it particularly effective for high-dimensional data like our one-hot encoded Adult Income dataset. Its ability to create complex decision boundaries through kernel tricks makes it a strong candidate for accurate income classification.

XGBoost An advanced gradient-boosting framework renowned for competition-winning performance. XGBoost excels at modeling complex non-linear relationships through its sequential tree-building approach, where each new tree corrects errors from previous ones. This results in highly accurate predictions while efficiently handling large datasets – ideal for our task.

Random Forest A robust ensemble method that builds multiple decision trees and aggregates their predictions. Particularly effective at handling noisy real-world data, Random Forest naturally accommodates mixed feature types (categorical and numerical) present in our dataset. Its inherent feature importance analysis provides valuable insights during model interpretation.

K-Nearest Neighbors (KNN) A distance-based algorithm that classifies instances by majority vote among their k-nearest neighbors. While computationally intensive with large datasets, KNN's non-parametric nature adapts well to complex patterns without assuming specific data distributions – valuable for exploring local structures in our feature space.

3.2 Model Comparison Analysis

We evaluated the performance of six models (Decision Tree, Logistic Regression, Random Forest, XGBoost, SVM, and KNN) using multiple metrics to comprehensively assess their strengths and weaknesses.

3.2.1 Key Metrics

Accuracy XGBoost and Logistic Regression achieved the highest accuracy scores (87.1% and 85.2%, respectively).

Note: Accuracy alone can be misleading in imbalanced datasets (e.g., where most incomes are $\leq$ \$50K). A model predicting the majority class might score well in accuracy but fail to identify the minority class (> \$50K).

Precision and Recall SVM showed strong precision (76.9%) but lower recall (57.9%), meaning it correctly identified high-income individuals but missed many true positives. Decision Tree had lower precision (61.2%) but higher recall (62.5%), indicating it captured more true positives at the cost of more false positives. Random Forest and XGBoost balanced both metrics well (Random Forest: 73.7% precision, 62.5% recall; XGBoost: 77.1% precision, 66.0% recall), demonstrating robust classification ability.

AUC-ROC XGBoost (AUC: 0.927) and Logistic Regression (AUC: 0.907) outperformed other models, indicating superior ability to distinguish between income classes across all classification thresholds. Random Forest (AUC: 0.904) and SVM (AUC: 0.902) followed closely, while KNN (AUC: 0.858) and Decision Tree (AUC: 0.750) lagged behind.

3.3 Feature Importance Analysis

In our model analysis, we conducted an in-depth examination of feature importance in these models. By analyzing the feature importance scores generated by these models, we successfully identified the most influential factors for income level prediction. The results clearly demonstrated that"education-num" (years of education),"occupation", and"hours-per-week" emerged as the most predictive features.

The "**education-num**" exhibited the strongest predictive power, which aligns perfectly with real-world observations: individuals with higher education levels demonstrate significantly greater likelihood of earning higher incomes.

The importance distribution of occupational features also revealed meaningful patterns, with managerial and professional occupations showing substantially greater predictive value for income levels compared to other job categories.

The "**hours-per-week**"feature displayed an interesting nonlinear relationship – while generally showing positive correlation with income, the marginal returns diminished beyond approximately 50 working hours per week.

3.4 Cross-validated Model Comparison

We used 5-fold cross-validation to evaluate the performance of the models. The following table shows the comparison results of different models based on cross-validation, sorted by ROC AUC in descending order:

Model	Test Accuracy	Precision	Recall	F1 Score	ROC AUC
XGBoost	0.870858	0.770858	0.659865	0.711056	0.927090
Logistic Regression	0.852308	0.736432	0.602219	0.662597	0.907082
Random Forest	0.856147	0.737476	0.625175	0.676698	0.903851
SVM	0.856638	0.768580	0.579008	0.660460	0.902065
KNN	0.830963	0.666240	0.597245	0.629859	0.857749
Decision Tree	0.814471	0.612472	0.625048	0.618696	0.749801

Table 1: Performance metrics of different machine learning models

3.5 XGBoost Hyperparameter Tuning

Since XGBoost performed well in the cross-validation, we decided to further tune its parameters. We set up a parameter grid with `n_estimators` and `max_depth` as the tuning parameters. Then, we used `GridSearchCV` for 5-fold cross-validation, with the evaluation metric set to log loss.

```
param_grid_xgb = {
    'n_estimators': [100, 200],
    'max_depth': [3, 6]
}

grid_search_xgb = GridSearchCV(
    XGBClassifier(random_state=42, eval_metric='logloss'),
    param_grid_xgb, cv=5, scoring='roc_auc'
)

grid_search_xgb.fit(X_train, y_train)

print(f"Best Parameters for XGBoost: {grid_search_xgb.best_params_}")
print(f"Best AUC: {grid_search_xgb.best_score_:.4f}")

#Result
Best Parameters:{'max_depth': 3, 'n_estimators': 200}
Best AUC: 0.926279
```

Listing 1: Implement of Tuned XGBoost

3.6 Final Model Performance

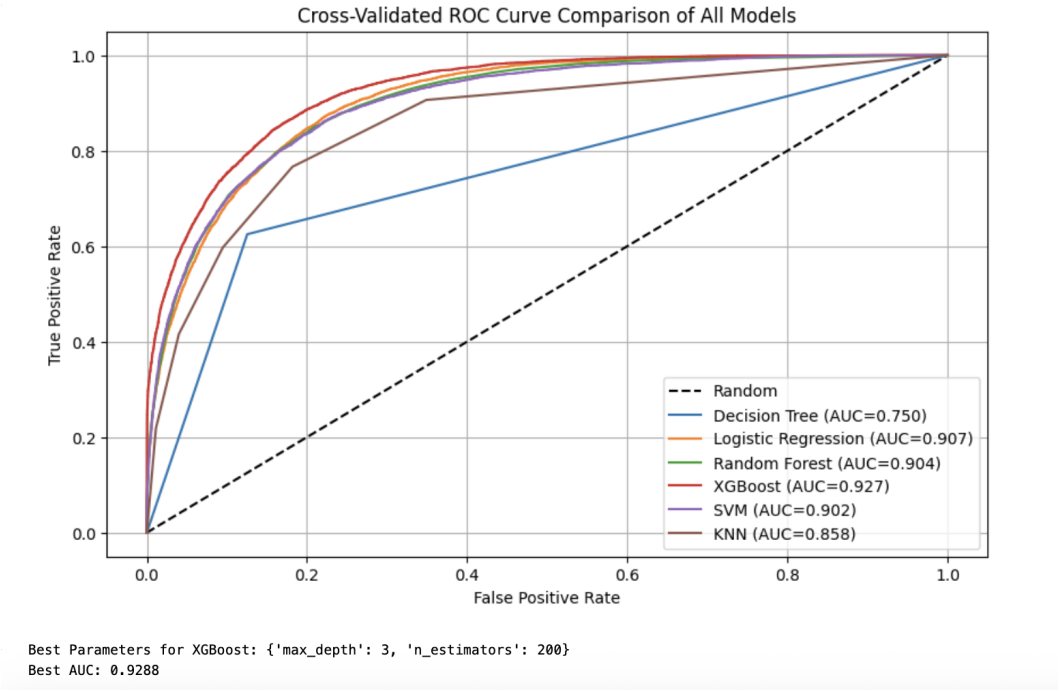
The following table shows the final performance of all models, sorted by ROC AUC in descending order:

The tuned XGBoost model achieved the highest ROC AUC, indicating its superior ability to distinguish between positive and negative classes across different classification thresholds.

It also showed improvements in precision, recall, and F1-score, demonstrating its overall better performance. Improved performance may come at the cost of increased computational complexity and training time.

Model	Test Accuracy	Precision	Recall	F1 Score	ROC AUC
XGBoost (Tuned)	0.874706	0.777213	0.660328	0.714019	0.926279
XGBoost	0.870858	0.770858	0.659865	0.711056	0.927090
Logistic Regression	0.852308	0.736432	0.602219	0.662597	0.907082
Random Forest	0.856147	0.737476	0.625175	0.676698	0.903851
SVM	0.856638	0.768580	0.579008	0.660460	0.902065
KNN	0.830963	0.666240	0.597245	0.629859	0.857749
Decision Tree	0.814471	0.612472	0.625048	0.618696	0.749801

Table 2: Performance metrics of different machine learning models, including tuned XGBoost




```
Best Parameters for XGBoost: {'max_depth': 3, 'n_estimators': 200}
Best AUC: 0.9288
```

Figure 5: Cross-validated ROC Curve

3.7 Insights and Recommendations

Model Selection XGBoost emerged as the best-performing model for predicting income levels in the Adult Income dataset. Its ability to capture complex non-linear relationships and handle large datasets makes it a suitable choice for this task.

Feature Engineering The interaction feature slightly improved XGBoost's performance, indicating potential non-linear relationships between features. Further exploration of feature engineering techniques (e.g., creating polynomial features or combining related features) may lead to additional improvements.

Hyperparameter Tuning Optimizing the parameters of XGBoost can significantly improve its performance. However, balance the trade-off between performance gain and computational cost.

Interpretability While XGBoost offers excellent predictive performance, its interpretability is relatively low compared to Logistic Regression. If interpretability is crucial, Logistic Regression or a simpler ensemble method like Random Forest may be more appropriate.

Future Work We might explore advanced techniques (e.g., deep learning, ensemble methods with different base learners) or incorporate external data/domain knowledge to further enhance model performance.

4 Conclusions

In conclusion, our comparison of SVM, XGBoost, Random Forest, Decision Tree, and Logistic Regression on the Adult Income dataset revealed distinct performance characteristics across models. XGBoost and Logistic Regression demonstrated slightly superior performance in terms of accuracy and AUC-ROC and most models had similar ROC curves, with the Decision Tree showing significant differences.

In addition, our analysis highlighted the critical role of feature engineering and data preprocessing in model performance. Proper handling of missing values and feature standardization significantly affected algorithm performance.

This project faces several limitations. The dataset contains inherent biases, and our models may not capture all complex real-world factors affecting income. Furthermore, some models—particularly XGBoost—required substantial computational resources. Future work could explore advanced model integration techniques like stacking or blending to enhance performance. We could also investigate more sophisticated data preprocessing methods and incorporate additional relevant features.

5 Credit

At the beginning, members of our group independently completed the mandatory tasks, obtaining partial data and ideas, which were then integrated. In the subsequent discussion, we determined the direction for open exploration, conducting comparisons of multiple models and simply investigating the impact of hyperparameters by adjusting those of the optimal model. After the discussion, Yimin WANG was primarily responsible for the coding part (completing the Jupyter notebook), while Wenhan XU and Ziyu LIANG were responsible for writing and refining the report.

The core module of the code and all the report are completed by us group mates. Generative AI tools are just used in polishing the contents.

References

[1] Becker, B. & Kohavi, R. (1996). Adult [Dataset]. *UCI Machine Learning Repository*. `https://doi.org/10.24432/C5XW20`.

[2] `https://github.com/saravrajavelu/Adult-Income-Analysis/tree/master/Graphs`

[3] `https://www.kaggle.com/datasets/wenruliu/adult-income-dataset`