Submitted: February 2025

Machine Learning Project Report

Arnab Biswas, Francesca Negri, Any Das, Farabi Issa, Tahira Rezaie

Analyzing and Predicting Adult Census Income

Università degli Studi di Milano-Bicocca

CdLM in Data Science

Keywords

Prediction, classification, data elaboration, machine learning

Email Correspondence

a.biswas4@campus.unimib.it
f.negri17@campus.unimib.it
any.das@campus.unimib.it
i.farabi@campus.unimib.it
t.rezaie@campus.unimib.it

Abstract

Given a set of data about US Census Incomes, this project aims to assess (by prediction) whether an adult individual earns more than $50K annually. We started by carefully processing the dataset, ensuring it was clean and well-structured. This involved handling missing values, standardizing numerical features to maintain consistency, and converting categorical features into an appropriate format for model training. Once the preprocessing was complete, we conducted an in-depth exploratory analysis, examining the distribution of numerical variables and identifying key patterns through pair plots to better understand the relationships between different features.

Following this analysis, we trained three distinct Machine Learning models—Logistic Regression, Decision Tree Classifier, and Random Forest Classifier—each offering different advantages in terms of interpretability and predictive power. Our goal was to compare their performance and determine which model was best suited for our specific task.

To conclude, we rigorously evaluated the effectiveness of our models by assessing key performance metrics, with a particular focus on accuracy and prediction quality. This final step allowed us to gain valuable insights into the strengths and limitations of each approach, helping to refine our methodology for future improvements.

Contents

Introduction

Context

Annual income represents the total earnings accumulated over a year before taxes are deducted. It encompasses various sources such as salary, bonuses, overtime pay, commissions, and tips. Understanding an individual's income can be valuable in multiple sectors, including marketing, insurance, banking, and taxation. For example, income plays a crucial role in determining the amount of taxes an individual owes. Governments can leverage this data to estimate overall tax revenues and plan public expenditures accordingly. Beyond its relevance in public finance, income information is also highly significant in the business world. Companies that focus on high-income consumers can utilize Machine Learning techniques to identify and target individuals who meet specific financial criteria. This approach allows businesses to improve the efficiency of their marketing strategies, increasing the conversion rate while optimizing costs and resources. However, financial details are often considered highly sensitive, and individuals are generally reluctant to disclose their earnings. As a result, income remains a challenging attribute to ascertain directly. This project aims to develop a classification model capable of predicting an individual's income based on readily available socioeconomic characteristics.

Dataset

The **Adult Census Income** dataset [1] extracted from the US Census Bureau of 1994, is widely used for classification tasks in machine learning. The objective is to predict whether an individual's income exceeds $50,000 per year based on various socio-economic and demographic attributes. This dataset could be particularly useful for applications in areas such as *market analysis*, *fiscal policy*, *banking*, and *consumer segmentation*.

The dataset contains approximately 33K observations and 15 variables (some categorical and some numerical).

Variable	Description
age	Age of the individual (in years).
workclass	Type of employment (e.g., "Private", "Self-emp", "Government").
fnlwgt	Final weight assigned to represent the population.
education	Level of education (e.g., "Bachelors", "Masters").
education-num	Number of years of education completed.
marital-status	Marital status (e.g., "Married", "Never-married").
occupation	Type of occupation (e.g., "Tech-support", "Sales").
relationship	Relationship to the head of household (e.g., "Husband", "Wife").
race	Ethnic group (e.g., "White", "Black", "Asian-Pac-Islander").
sex	Gender of the individual ("Male" or "Female").
capital-gain	Capital gains (positive value if any).
capital-loss	Capital losses (positive value if any).
hours-per-week	Number of hours worked per week.
native-country	Country of origin (e.g., "United-States", "Mexico").
income	Income class: ">50K" or "≤50K". This is the target variable.

Table 1. Dataset Variables

Dataset Statistics

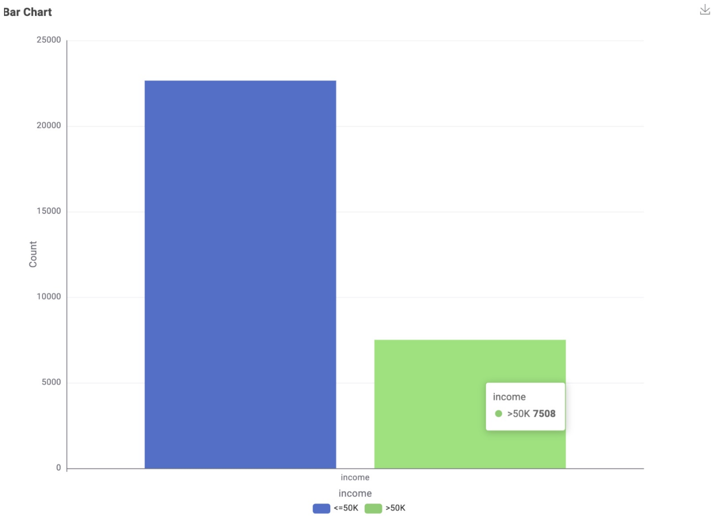

Figure 1. The bar chart compares income distribution between two categories: individuals earning ≤ 50K (blue bar) and those earning > 50K (green bar). The blue bar is significantly taller, indicating that the majority of people fall into the lower income group, while only 7,508 individuals earn more than 50K. This suggests a skewed income distribution where higher earnings are less common, possibly due to factors such as limited high-paying job opportunities, economic disparity, or workforce composition. Overall, the data highlights that most individuals earn below 50K, with a smaller proportion surpassing this threshold.

The dataset offers various features that are useful to determine the distribution of the population sample in the dataset. The majority of records is, for instance, composed by white, young male individuals. The Final Weight mean is around 18K. Figure **2** shows how the income relates to gender and workclass, while Figure **1** show the distribution of the records between "lower income" category and "higher income" category. In Table **2** the statistics about some demographic features of the dataset are shown.

Table 2. Analysis of the distribution of records in the dataset's features age, gender, ethnicity and final weight. Regarding age, the most of the population is shown to be younger than 41 years, while only few records are over 65. Male individuals are more present than female individuals, and white Ethnicity is the largest group in Ethnicity category, while black and "other" are a much smaller percentage, but the value between the two is quite similar. The final weight is the category with the largest difference between the groups, with the first group (12K - 50K) alone reaching almost 100%.

Age	16-41 *61.4%*	41-65 *35.8%*	65-90 *2.8%*
Gender	Male *67.5%*	Female *32.5%*	
Ethnicity	White *86%*	Black *9.3%*	Other *4.7%*
fnlwgt	12K-50K *98.9%*	50K-100K *1%*	100K+ *0.1%*

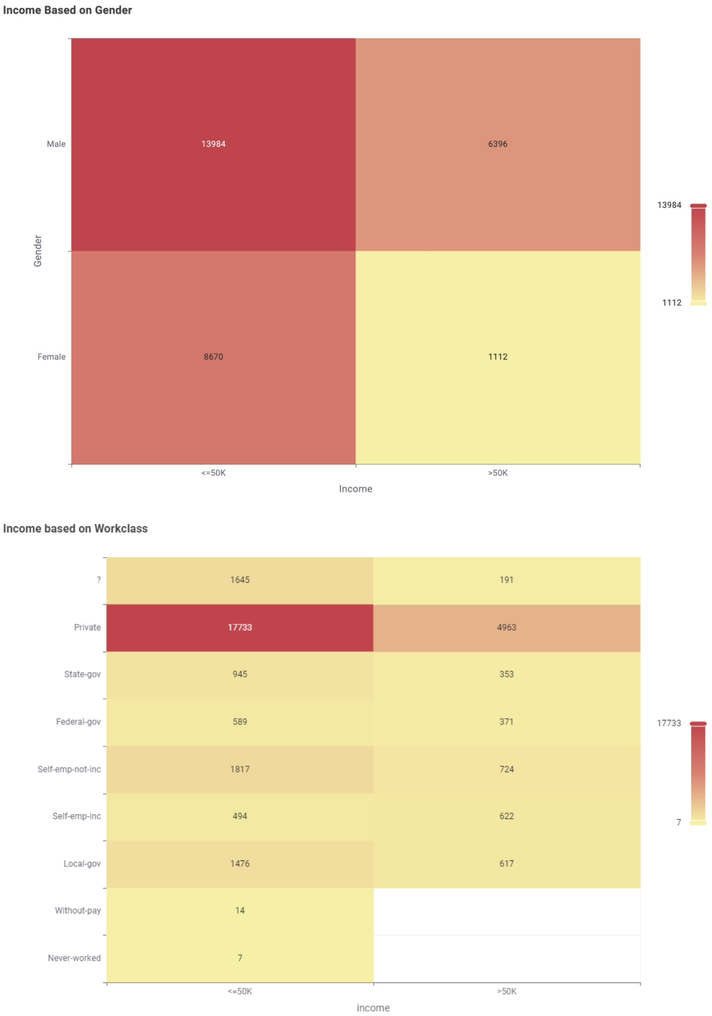

Figure 2. The two images are heatmaps that visualize income distribution based on gender and workclass, with darker shades representing higher numbers and lighter shades representing lower numbers. The first heatmap categorizes individuals into four groups: males and females earning either ≤50K or >50K. The largest group is males earning ≤50K (13,984), followed by females in the same category (8,670). Fewer individuals earn above 50K, with 6,396 males and only 1,112 females in this group. This heatmap highlights that more men earn higher incomes compared to women, while the majority of both genders fall into the ≤50K income range. The second heatmap, instead, has many more categories that divide themselves between lower income and higher income. We can see that the majority of records fall into the private workclass, which is, in turn, more populated for ≤50K income (17.733). The second and third largest group are the "self-emp-not-inc" and "Local-gov" ones, both of them for the lower income category. This heatmap tells that the population in the dataset is more focused on workers in the private sector that has an income ≤50K.

Data Elaboration

Handling Missing Values

The dataset initially contained several missing values, distributed randomly across both rows and columns. Interestingly, these missing values were not represented as `null` values but were instead marked with a single ”?” in the respective cells. When deciding how to handle these missing data, which, naturally, had an impact on various statistical measures, we considered their overall contribution to the total number of values in the dataset.

After converting all occurrences of the ”?” character into `NaN` with python (pandas), we calculated that the missing data were distributed as follows: 1,836 instances in the `workclass` variable, 1,843 in the `occupation` variable, and 583 in the `native-country` variable. These 4,262 missing values, out of a total of 32,561 entries, represented a very small percentage of the dataset.

Given the relatively small proportion of missing data, we decided to remove the rows containing these missing values. This approach allowed us to maintain a clean dataset with over 30,000 valid rows, ensuring consistency and reliability in the remaining data for further analysis.

Feature Scaling and Encoding

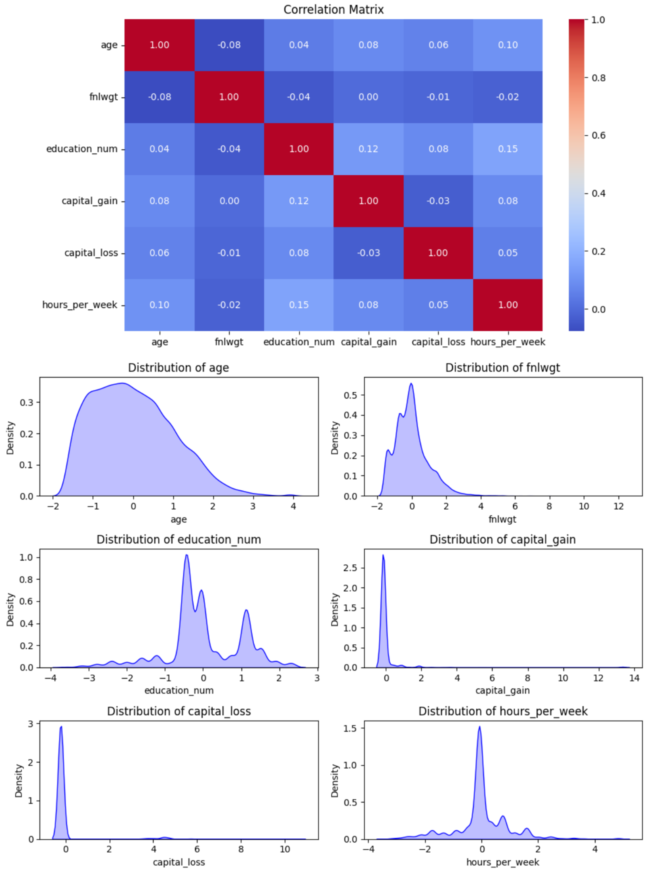

Figure 3. Correlation Matrix and distribution of the numerical values.

In figure **3** some visuals that have been made to evaluate the quality of the numerical categories. As one can see, there is no correlation between them and the distribution is heterogeneous. Given that the `age` category is the most distributed, we decided to normalize it, with 0 being age 17 and 1 meaning 90 years.

The normalization was carried out by a node in the KNIME workflow, using the **Z-Score normalization method**.
This method transforms a data point x into a standardized value using the formula:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the dataset. This method ensures a mean of 0 and a standard deviation of 1, making it useful for comparisons across different scales. The obtained results were compared to the ones we achieved with a Python (pandas) code written with the same purpose. First, the code identified categorical and numerical columns. Next„ categorical variables were encoded using `LabelEncoder`, converting each unique category into an integer representation. The fitted encoders were stored in a dictionary for potential inverse transformation. Finally, numerical features were standardized using the same Z-score normalization method. In the end, the results were compatible with each other.

Model Assessment Methods

Evaluating the performance of a machine learning model is a critical step in the development process, as it determines the model's ability to generalize to unseen data. Two widely used validation techniques were applied in this project: the Holdout method and k-Fold Cross Validation[2]. Each of these methods has distinct characteristics that impact the reliability and efficiency of model evaluation. KNIME was used to handle the total workflow (Figure **4**).

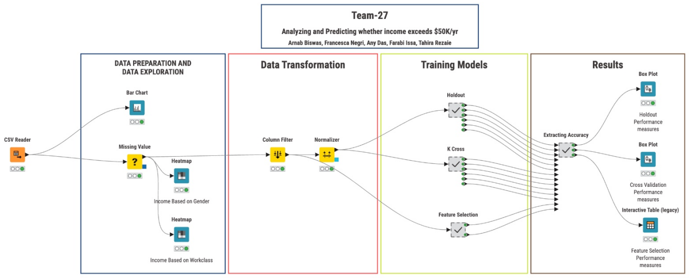
Team-27
Analyzing and Predicting whether income exceeds $50K/yr
Arnab Biswas, Francesca Negri, Any Das, Farabi Issa, Tahira Rezaie
DATA PREPARATION AND DATA EXPLORATION
Data Transformation
Training Models
Results
CSV Reader
Bar Chart
Missing Value
Heatmap
Income Based on Gender
Heatmap
Income Based on Workclass
Column Filter
Normalizer
Holdout
K Cross
Feature Selection
Extracting Accuracy
Box Plot
Holdout Performance measures
Box Plot
Cross Validation Performance measures
Interactive Table (legacy)
Feature Selection Performance measures

Figure 4. The total workflow on KNIME. There are two metanodes, for Holdout and K Cross

Holdout

The first strategy employed was the **Holdout** method, implemented by randomly dividing the dataset into two separate subsets. To ensure reproducibility across different runs, a fixed random seed was used.

Approximately **80%** of the data was allocated for training, allowing the model to learn patterns and relationships from the available features. The remaining **20%** was reserved as a test set during partition, ensuring that the model was evaluated on previously unseen data.

This approach closely mimics a real-world production environment, where trained models must generate predictions for new, unseen inputs. Additionally, all features retained after the preprocessing phase were included in the model training process, maximizing the information available for learning.

Cross Validation

The effectiveness of a classification model is closely tied to the method used to split the dataset into training and testing sets. In this case, we applied the **K-Fold Cross Validation** technique with **k = 5**. Compared to traditional holdout and iterated holdout methods, this approach reduces bias by minimizing the influence of outliers.

Additionally, k-fold cross validation ensures that each data point appears in the training set multiple times and is used for testing exactly once. The dataset is divided into **k mutually exclusive and exhaustive subsets**, each containing an equal number of samples. The model undergoes **k iterations**, with a different subset serving as the test set in each round.

Once all iterations are complete, the final performance metrics are derived by averaging the results from each fold, providing a more reliable estimate of the model's generalization ability.

Classification

After applying both the Holdout method and k-Fold Cross Validation, several machine learning models [3] were employed to classify the dataset. Below a description of the selected algorithms follows.
Each of these models has unique advantages, making them suitable for different classification tasks. The selection of the best model depends on factors such as dataset characteristics, computational efficiency, and the importance of interpretability versus accuracy, as we will see in the next section. We used *J48, Logistic Regression, Naïve Bayes, SMO, GBT and Random Forest* for the Holdout method, *J48, Grading, Logistic Regression, Naïve Bayes, SMO and Random Forest* for the Cross Validation.

J48 - Decision Tree

J48 is an implementation of the C4.5 decision tree algorithm. It constructs a hierarchical tree structure by recursively splitting the dataset based on the most informative features. The resulting tree consists of decision nodes that guide the classification process, making the model highly interpretable and suitable for datasets with categorical and numerical attributes.

Logistic Regression

Despite its name, Logistic Regression is a classification algorithm rather than a regression technique. It estimates the probability that a given input belongs to a specific class by applying the logistic function to a linear combination of the input features. This method is particularly effective when the relationship between independent variables and the target class is approximately linear.

SMO (Sequential Minimal Optimization)

Sequential Minimal Optimization (SMO) is an efficient algorithm for training Support Vector Machines (SVMs) by breaking the quadratic optimization problem into a series of smaller subproblems. At each step, SMO selects two Lagrange multipliers to optimize analytically while keeping the others fixed, significantly reducing computational complexity compared to traditional methods.

Naïve Bayes Classifier

The Naïve Bayes classifier is a probabilistic model based on Bayes' theorem. It assumes that all features are independent given the class label, which simplifies computations and makes the model highly efficient, particularly for text classification tasks such as spam detection.

Gradient Boosted Trees (GBT)

Gradient Boosting is an ensemble technique that builds multiple weak decision trees sequentially, with each tree correcting the errors of its predecessors. This iterative improvement process allows the model to capture complex relationships in the data, resulting in high predictive accuracy. However, the computational cost is higher compared to Random Forest.

Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve stability and accuracy. Each tree is trained on a random subset of the data and votes on the final classification. This reduces over-fitting and enhances generalization compared to a single decision tree.

Grading

Grading is a meta-learning technique that combines the predictions of multiple base classifiers to enhance overall performance. It assigns different weights to models based on their reliability in different scenarios, effectively acting as a "judge" to select the most accurate classification.

Evaluation metrics

It's very important to evaluate the performance of the multiple machine learning models we implemented, in order to to determine the most effective classifier 4. For achieving this goal, we employed various evaluation metrics: as accuracy, precision and F_1-measure. **Accuracy** quantifies the model's capability to correctly classify new data points and is expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{1}$$

This metric is generally reliable when the classes are evenly distributed with respect to the target variable. However, in cases where the dataset is imbalanced, additional evaluation metrics are required 5. In particular, when one class (positive) is significantly less frequent than the other (negative), precision, recall, and F_1-measure become essential. **Precision** measures the proportion of correctly identified positive instances out of all instances predicted as positive:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

A higher precision value indicates a lower number of false positives (FP).

The $\mathbf{F_1} - \textbf{measure}$ is the harmonic mean of precision and recall:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

where recall is calculated as

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

A high F_1-measure indicates a strong balance between precision and recall.

Performance Analysis

Holdout Method Analysis

For the Holdout method, the models tested include Gradient Boosting, J48 (C4.5), Logistic Regression, Naïve Bayes, Random Forest, and SMO.

Gradient Boosted Tree

The Gradient Boosting model balances precision and recall well but demands higher computational resources. It performs strongly in the ≤50K class (92% precision, 87.1% recall, 89.5% F_1-measure) but struggles in the >50K class (57.2% precision, 69.3% recall, 62.6% F_1-measure). With 83.6% accuracy and a Cohen's Kappa of 0.522, it proves reliable but may need refinements for better high-income classification.

J48 - DecisionTree

The J48 model achieves 83.0% accuracy, balancing interpretability with performance but prone to over-fitting. It excels in the ≤50K class (86.6% precision, 91.8% recall) but struggles with the >50K class (68.1% precision, 55.2% recall). With a Cohen's Kappa of 0.502, it shows moderate agreement, requiring optimization for better high-income classification.

Logistic Regression

Logistic Regression, a baseline classifier, performs well for linearly separable data but struggles with complex patterns. It achieves 86% precision and 92.3% recall in the ≤50K class, ensuring strong classification (F_1-measure: 89.1%). In the >50K class, it attains 68.5% precision and 52.7% recall, reflecting difficulty in identifying high-income cases (F_1-measure: 59.6%). With 82.8% accuracy and a Cohen's Kappa of 0.489, it provides a reasonable benchmark but lacks the adaptability of ensemble models.

Naïve Bayes

Naïve Bayes efficiently classifies categorical data but is limited by its feature independence assumption. It achieves 87.8% precision and 86.7% recall in the ≤50K class, ensuring balanced performance (F_1-measure: 87.3%). For the >50K class, it attains 59.7% precision and 62.1% recall, indicating moderate effectiveness (F_1-measure: 60.9%). With 80.8% accuracy and a Cohen's Kappa of 0.481, it demonstrates strong predictive reliability despite its constraints.

Random Forest

Random Forest excels in classification by aggregating decision trees, enhancing generalization while maintaining computational complexity. It achieves 97.6% precision and 96.6% recall in the ≤50K class, ensuring high accuracy (F_1-measure: 97.1%). For the >50K class, it attains 89.2% precision and 92.2% recall, balancing performance (F_1-measure: 90.7%). With 95.6% accuracy and a Cohen's Kappa of 0.878, it demonstrates strong predictive reliability across income groups.

SMO

SMO offers strong classification performance, particularly in high-dimensional data, but is sensitive to hyperparameters and computationally intensive. It achieves 86.5% precision and 92.5% recall in the ≤50K class (F1-score: 89.4%). For the >50K class, it attains 69.6% precision and 54.5% recall, balancing accuracy (F1-score: 61.1%). With 83.3% accuracy and a Cohen's Kappa of 0.507, SMO proves reliable but requires optimization for better performance.

Holdout models Comparative Analysis

The performance of these six classification models was assessed using the Holdout method. The table below presents a comparative evaluation based on key performance metrics.

Model	Accuracy (%)	Cohen's Kappa	$\leq 50K$			>50K			Specificity (%)
			Precision	Recall	F_1-measure	Precision	Recall	F_1-measure	
Random Forest	95.6	0.878	97.6	96.6	97.1	89.2	92.2	90.7	96.6
Gradient Boosting	83.6	0.522	92.0	87.1	89.5	57.2	69.3	62.6	87.1
SMO	83.3	0.507	86.5	92.5	89.4	69.6	54.5	61.1	54.5
J48 Decision Tree	83.0	0.502	86.6	91.8	89.1	68.1	55.2	60.9	91.8
Logistic Regression	82.8	0.489	86.0	92.3	89.1	68.5	52.7	59.6	92.3
Naïve Bayes	80.8	0.481	87.8	86.7	87.3	59.7	62.1	60.9	86.7

Table 3. Comparative Analysis for Holdout

While Random Forest offers the best accuracy, Gradient Boosting provides competitive performance with better adaptability at a computational cost. SMO is sensitive to hyperparameters, J48 is prone to over-fitting, and Logistic Regression, while simple, struggles with complex data. The best model depends on the trade-off between accuracy, interpretability, and computational efficiency.

Cross Validation Analysis

For the K-Cross Validation method, the models tested include Gradient Boosting, J48 (C4.5), Logistic Regression, Random Forest, Naïve Bayes and SMO.

Gradient Boosted Tree

The Gradient Boosting model has an accuracy of 75.9%, showing strong performance for the ≤50K class with 75.9% precision and 100% recall, resulting in an F_1-measure of 86.3%. However, it completely fails to classify the >50K class, with 0% precision, 0% recall, and 0% F_1-measure, indicating no instances of this class are correctly identified. The Cohen's Kappa score of 0.000 indicates no agreement beyond chance, highlighting severe model bias toward the majority class. This imbalance suggests the need for techniques such as resampling, class weighting, or alternative models to improve performance.

J48 - Decision Tree

The J48 model achieves 83.5% accuracy, with strong performance on the <=50K class (91.6% recall, 87.3% precision, F_1-measure: 89.4%). However, for the >50K class, recall is lower at 57.9%, with 68.7% precision (F_1-measure: 62.9%). The Cohen's Kappa score of 0.524 indicates moderate agreement beyond chance. While better than Gradient Boosting, improvements in feature selection and balancing techniques could enhance minority class predictions.

Logistic Regression

The Logistic Regression model achieves 83.2% accuracy, with strong performance on the <=50K class (92.3% recall, 86.5% precision, F_1-measure: 89.3%). However, for the >50K class, recall is lower at 54.6%, with 69.3% precision (F_1-measure: 61.1%). The Cohen's Kappa score of 0.506 indicates moderate agreement beyond chance. While the model performs well overall, improvements in handling class imbalance could enhance minority class predictions.

Random Forest

The Random Forest model achieves an accuracy of 80.4% and a Cohen's Kappa of 0.466, indicating moderate agreement between the predicted and actual classes. For the ≤50K class, the model has 88.4% recall and 86.1%

precision, resulting in an F_1-measure of 87.2%. For the >50K class, it has 54.9% recall and 60.1% precision, with an F_1-measure of 57.4%. The model performs better for the ≤50K class but struggles with recall and precision for the >50K class, indicating challenges in identifying the minority class.

Naïve Bayes

The Naïve Bayes model achieves 81.4% accuracy, slightly lower than Logistic Regression. The <=50K class has 86.9% recall and 88.4% precision (F_1-measure: 87.7%), while the >50K class has 64.1% recall and 60.9% precision (F_1-measure: 62.5%). Cohen's Kappa is 0.501, indicating moderate agreement. The model slightly improves minority class recall but struggles with precision.

SMO

The SMO model achieves accuracy statistics as shown in the table. The overall accuracy is 81.9%, and Cohen's Kappa is 0.485, indicating moderate agreement between the predicted and actual classes. For the ≤50K class, the model has 89.8% recall and 86.8% precision (F_1 score: 88.3%). For the >50K class, it has 56.8% recall and 63.9% precision (F_1 measure: 60.1%). The model performs better for the ≤50K class but struggles with recall and precision for the >50K class, indicating challenges in identifying the minority class.

Cross Validation models Comparative Analysis

The performance of these six classification models was assessed using the K-Cross Validation method. The table below presents a comparative evaluation based on key performance metrics.

Model	Accuracy (%)	Cohen's Kappa	$\leq 50K$			>50K		
			Precision	Recall	F_1-measure	Precision	Recall	F_1-measure
J48	83.5	0.524	87.3	91.6	89.4	68.7	57.9	62.9
Logistic Regression	83.2	0.506	86.5	92.3	89.3	69.3	54.6	61.1
SMO	81.9	0.485	86.8	89.8	88.3	63.9	56.8	60.1
Naïve Bayes	81.4	0.501	88.4	86.9	87.7	60.9	64.1	62.5
Random Forest	80.4	0.446	86.1	88.4	87.2	60.1	54.9	57.4
Gradient Boosting	75.9	0	75.9	100	86.3	0	0	0

Table 4. Comparative Analysis for K-Cross Validation

The comparative analysis highlights key performance differences across the six models.

Feature Selection

Using all the attributes present in the dataset is not an efficient choice, because it can alter the classification. To fix this, it is very important to make a proper *feature selection* in order to to discover attributes that are irrelevant, or redundant. This process reduces the total number of input variables taken into account, as well as the computational cost. Hence, there's an improving both in performance and in model comprehension. To proceed with the feature selection, among the various approaches that can be used, we chose the Filter one and the Joiner (Wrapper) one. An explanation of this part of the workflow in KNIME follows.

- **Multivariate Filter (Correlation Feature Selection)**
  - After applying **Equal Size Sampling** with undersampling of the minority class, the data is sent to a **multivariate filter**;
  - This method, known as *Correlation Feature Selection (CFS)*, selects only those features that have a strong correlation with the target variable but a low correlation with each other. The goal is to remove redundant or irrelevant features to enhance data quality without losing essential information for the model;
  - As output, a subset of relevant features from the original dataset is given.
- **Joiner for Merging Performance Metrics**
  - After feature selection, multiple machine learning models were executed, and their results were evaluated;
  - Then, the **Joiner** is used to combine the different performance measures into a single dataset, allowing for a structured comparison of the models.).

The following table (5), meanwhile, shows the performance recorded after Feature Selection.

	J48 (%)	Logistic	SMO	Naïve Bayes	Random Forest	Multilayer Perceptron
Recall	0.819	0.837	0.815	0.837	0.776	0.818
Precision	0.781	0.777	0.775	0.768	0.747	0.773
F_1 measure	0.8	0.806	0.794	0.801	0.761	0.795
Accuracy	0.795	0.798	0.789	0.792	0.757	0.789
AUC (Area Under Curve)	0.856	0.876	0.789	0.876	0.826	0.869

Table 5. Comparative Analysis after Feature Selection

Results Evaluation

As mentioned previously, this project aims to estimate an individual's income based on various socioeconomic factors.
To accomplish this, several classification algorithms were reported to have been implemented and analyzed under different conditions.To ensure robust model evaluation, a 5-fold cross-validation technique was employed, allowing the dataset to be systematically split into training and testing subsets.

Initially, classifiers were trained and assessed using only complete observations; next, models were reapplied after performing an **undersampling procedure on minority class**. This adjustment provided insight into how class distribution influences model performance. Finally, a **feature selection** strategy was introduced to refine the analysis.

Comparing Accuracy

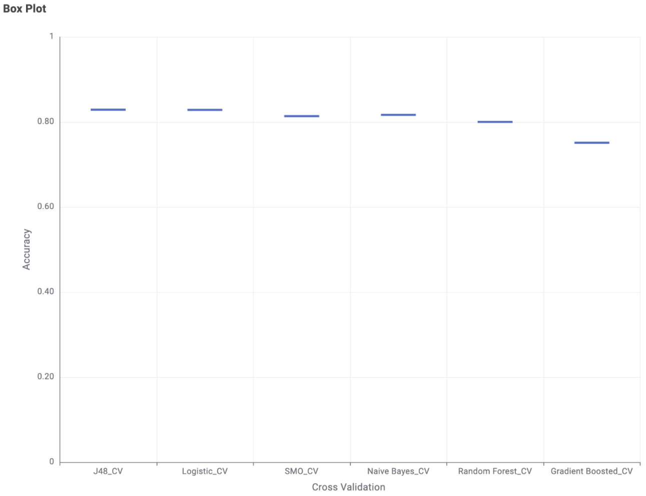

Figure 5. The chart compares the accuracy of different machine learning models using cross-validation. J48, Logistic Regression, SMO, Naïve Bayes, and Random Forest all achieve similar accuracy, around 80%, while Gradient Boosting performs the worst, falling below this mark. There is little variation among the top models, suggesting they are equally effective for this dataset. However, Gradient Boosting may require further tuning or may not be well-suited for this data.

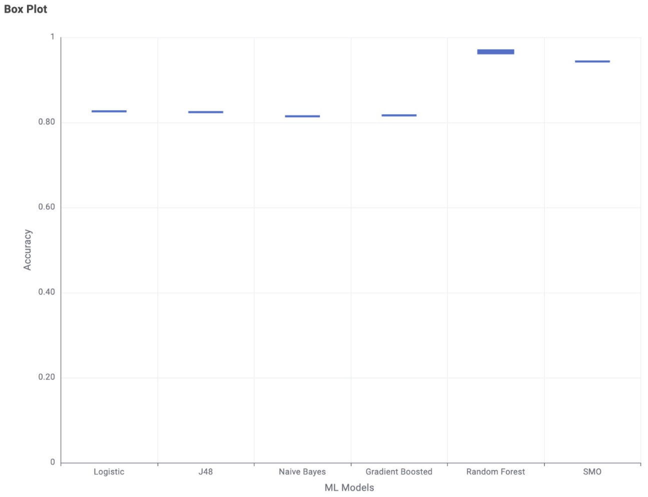

Figure 6. The chart compares the accuracy of different machine learning models on a holdout dataset. The y-axis represents accuracy, while the x-axis lists models like Logistic Regression, J48, Naïve Bayes, Gradient Boosted Trees, Random Forest, and SMO. Among them, Random Forest performs the best, followed closely by SMO, while Naïve Bayes has the lowest accuracy. The other models (Logistic Regression, J48, and Gradient Boosted Trees) show similar but slightly lower performance. Overall, Random Forest is the top choice for accuracy, but factors like speed and interpretability should also be considered when selecting a model.

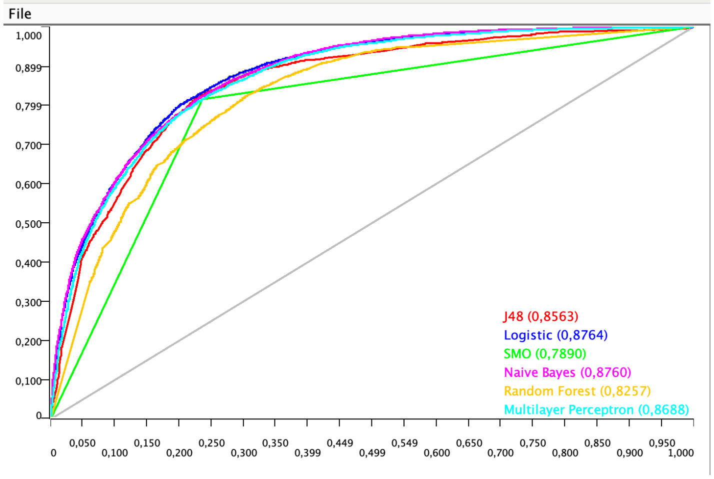

Figure 7. The ROC curve illustrates the performance of the six classification models in predicting the target variable. The area under the curve (AUC) values, indicated in parentheses, provide insight into the effectiveness of each model. The Logistic Regression and Naïve Bayes models achieve the highest AUC, followed closely by the Multilayer Perceptron The J48 decision tree also performs well, while Random Forest and SMO exhibit slightly lower discriminative power. The diagonal reference line represents random classification, reinforcing the models' predictive capabilities.

The two box plot and the ROC curve offers a visual comparison among the various machine learning models implemented for Cross Validation (Figure **5**), Holdout (Figure **6**) and Feature Selection (Figure **7**). The comparison of Holdout, k-Fold Cross-Validation, and Feature Selection highlights key trade-offs in model evaluation. Holdout validation produced the highest accuracy (Random Forest: 95.6%) but risked overfitting due to a single test split. In contrast, k-Fold Cross-Validation (k=5) provided more reliable generalization, reducing variance but lowering accuracy (Random Forest: 80.4%. Gradient Boosting struggled under Cross-Validation (75.9%), revealing sensitivity to class imbalance. Feature Selection improved efficiency and interpretability, with Logistic Regression achieving the highest AUC (0.876). Overall, k-Fold Cross-Validation proved the most robust method, Feature Selection enhanced efficiency, and Holdout, despite its high accuracy, was less reliable for final model selection.

Conclusions

This project aimed to develop a classification model capable of estimating income of individuals, using readily available socioeconomic variables. The task was approached as a binary classification problem, addressing the issue of class imbalance through an undersampling technique. Additionally, a feature selection process was applied to identify the most significant attributes. The classification models assessed in this study include *J48, Random Forest, Logistic Regression, Multi-Layer Perceptron, Naïve Bayes and SMO*. When we used Holdout method, we found out that J48 performed the best with an accuracy of 83.5% and a Cohen's Kappa of 0.524, showing strong agreement with actual values, especially for the ≤50K class. Logistic Regression followed closely with 83.2% accuracy and 0.506 Kappa, performing well for the ≤50K class but slightly weaker for the >50K class. SMO and Naïve Bayes both performed well overall, with Naïve Bayes excelling in recall for the minority class. Random Forest dropped to 80.4% accuracy under K-Fold, showing potential over-fitting compared to the Holdout method. Gradient Boosting had the lowest performance with 75.9% accuracy and 0% recall for the >50K class, suggesting it requires improvements in handling class imbalance. In conclusion, J48 was the most balanced model, while Gradient Boosting and Random Forest showed weaknesses in certain areas.

Meanwhile, when we used the k-Fold Cross Validation, it was clear that Random Forest achieved the highest accuracy (95.6%) and Cohen's Kappa (0.878), demonstrating robust performance across both income classes. Gradient Boosting (83.6%) and Naïve Bayes (80.8%) followed, with Naïve Bayes performing well despite its assumption of feature independence. Class-wise, Random Forest performed best for ≤50K income (97.6% precision, 96.6% recall) and >50K income (89.2% precision, 92.2% recall). Gradient Boosting and SMO exhibited lower recall values for high-income instances, suggesting difficulties in capturing all relevant cases.

Regarding the issue of class imbalance, the sampling-based strategy and the feature selection that came after led to a substantial change in model performance, leading to discover that the model with best accuracy and F-measure, in this case, is the logistic one.

In the end, the classification process confirmed that the study achieved its goal, also demonstrating that reliable knowledge is available from these data, providing interesting results that could be explored in more depth and with several application fields.

References

1. Adult Census Income Dataset ; https://www.kaggle.com/datasets/uciml/adult-census-income
2. Kavyasrirelangi. From Hold-Out to K-Fold: Understanding Cross-Validation Methods in Machine Learning ; https://medium.com/%40kavyasrirelangi100/from-hold-out-to-k-fold-understanding-cross-validation-methods-in-machine-learning-37402f406759
3. Pranckevicius, T. & Marcinkevičius, V. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing* **5** (2017). 10.22364/bjmc.2017.5.2.05
4. McGill Science Undergraduate Research Journal. *This is a book* (MSURJ, Montreal, 2022).
5. Google. Classification: Accuracy, recall, precision, and related metrics ; https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall