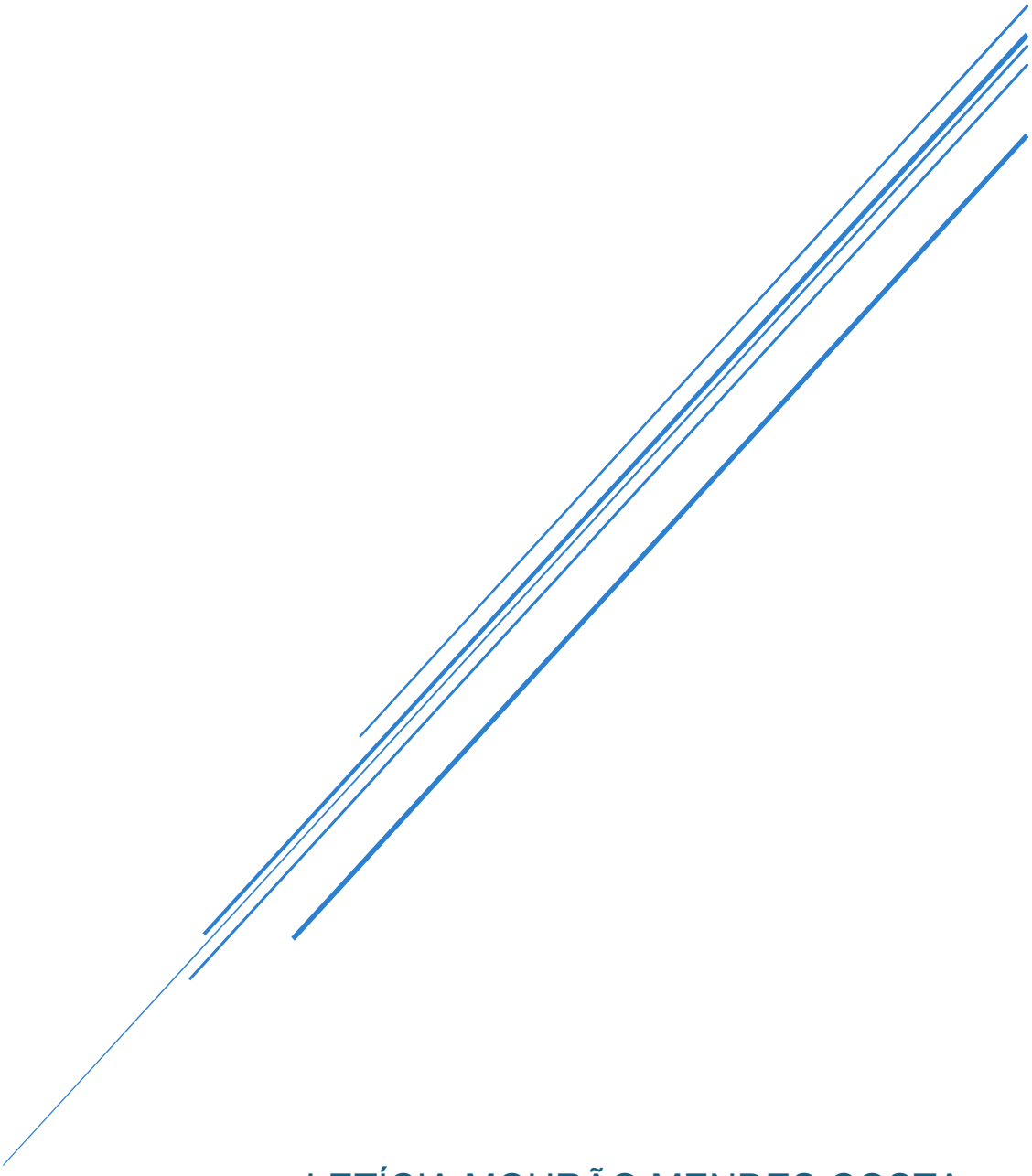
REPORT

ADULT CENSUS INCOME CLASSIFICATION



LETÍCIA MOURÃO MENDES COSTA

1. Introduction

The objective of this project is to apply Data Science concepts to the Adult Census Income dataset from the UCI Machine Learning Repository. The main goal is to predict whether an individual's annual income exceeds $50K per year based on demographic and socioeconomic attributes. The target variable is income, with two classes: ≤50K and >50K.

The main tasks include:

- Exploratory Data Analysis (EDA):
- Preprocessing and preparation of the dataset
- Training three different classification models
- Evaluating model performance through multiple metrics
- Performing two statistical hypothesis tests related to income proportions

The exploratory data analysis (EDA) section includes descriptive statistics for numerical and categorical variables, handled missing values and identified outliers, explored feature distributions and class imbalance, and relevant visualizations to support insights.

Regarding the data preparation, it covers the encoding of categorical variables, the normalization/standardization of numerical features, and the train/test split, along with a brief justification for these preprocessing decisions.

In the modeling phase, three classifiers were tested, with detailed justification of the selected hyperparameters and a comparison of their performance. Evaluation was conducted using standard classification metrics - Accuracy, Confusion Matrix, Precision, Recall, F1-score, and ROC-AUC - and included an analysis of model behavior under class imbalance as well as the consequences of false positives and false negatives.

Furthermore, two statistical hypothesis tests were performed to investigate differences between groups with respect to the income variable. The first test compared proportions to determine whether the proportion of women earning more than 50K is equal to that of men. The second test was an A/B test to investigate whether the proportion of individuals earning more than 50K differs between those who work more than 40 hours per week and those who work 40 hours or fewer.

💡 Workflow: Data science pipeline from EDA to modeling and statistical testing, with insights guiding preprocessing and model decisions.

2. Exploratory Data Analysis (EDA)

The dataset was imported following the instructions provided by the UCI Machine Learning Repository, using the ucimlrepo library, which enables fetch datasets directly from the UCI repository through Python. Two pandas dataframes were created: X = adult.data.features, containing the feature columns (such as age, education, etc.), and y = adult.data.targets, containing the target variable (income). The income column was then added to the feature DataFrame by concatenating both objects using df = pd.concat([X, y], axis=1).

The df.head() command shows the first five lines of the dataframe and allows us to check if the data was imported correctly, understand briefly the data, and maybe detect some inconsistencies or outliers. The dataset contains a mix of numerical and categorical variables. The education levels vary (Bachelors, HS-grad, 11th), and the education-num values seem to be related to these category. The individuals also represent different races, sexes, and countries of origin, showing the dataset's demographic diversity. Additionally, the occupations vary across categories such as Adm-clerical, Exec-managerial, Handlers-cleaners, and Prof-specialty, indicating a diverse workforce represented in the data.

The df.info() command provides a technical summary of the dataframe. It shows the total number of rows and columns, the number of non-null values in each column, the data type of each column, the total of the data types found, and memory usage of the dataframe. The dataset contains 48842 observations and from the Non-Null Count information, we observe that several columns contain missing values: workclass (47879), occupation (47876), and native-country (48568). The Dtype information shows that the dataset includes six numerical columns (int64): age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week; and nine categorical columns (object): workclass, education, marital-status, occupation, relationship, race, sex, native-country, and income.

The df.describe() command provides information about the numerical columns. It gives us different statistics, such as the number of non-NA/null observations, the mean, standard deviation, minimum and maximum values of each column, as well as the 25th, 50th and 75th percentiles of each numerical variable. We can observe with count that all numerical columns have 48842 values, indicating no missing data. Age ranges from 17 to 90 years old, with an average of 38.6 years, representing a typical working-age adult population. The number of years of education (education-num) ranges from 1 to 16, with an average of approximately 10 years, representing a high-level education population. Capital-gain values range from 0 to 99999, with an average of approximately 1079, and capital-loss ranges from 0 to 4356, with an average of about 87, highlighting the presence of extreme outliers in both columns. Hours worked per week vary from 1 to 99, with an average of roughly 40 hours, thus some extreme values exist. The high standard deviations of capital-gain and capital-loss further confirm the presence of outliers. Observing the quartiles values we can see that: age is reasonably distributed, slightly concentrated between 28–48; most individuals have high school or undergraduate education; most have no capital gains or losses; and most work around 40 hours per week. Overall, the dataset is largely consistent with expectations, though some columns contain extreme values that may require careful consideration during analysis

The df.describre(include='all') command includes the categorical columns, where unique indicates the number of unique observations, top shows the most frequently occurring value, and freq provides the frequency of that value. In our dataset, the income column contains 48842 values, with the most frequent value being <=50K and occurring 24720 times. This means that a little more than half of the individuals earn less than or equal to 50K. However, the unique shows that income has four distinct values, indicating inconsistencies in the string (maybe extra spaces). These inconsistencies need to be corrected before conducting the analysis. From the count values, we can also confirm that some categorical columns (workclass, occupation, and native-country) contain

missing values. Analyzing the most frequent values, we find that education is most commonly HS-grad (15784 occurrences), marital-status is Married-civ-spouse (22379), relationship is Husband (19716), race is White (41762), and sex is Male (32650).

To ensure that the dataset is consistent and clean, a categorical list and a numerical list were created, and their unique values verified. We can observe that there are some columns with "?" that need to be treated and also that the "income" column has 4 different values: '<=50K', '>50K', '<=50K.', '>50K.'. To fix these values we use df['income']=df['income'].str.rstrip('.') to remove the extra "." and df['income'].unique() again to confirm that the column now contains only the two correct values: <=50K and >50K. To replace '?' with proper NaN, we use df.replace('?', np.nan, inplace=True) and df.isnull().sum() to verify that the values were replaced.

We have observed earlier that "workclass", "occupation" and "native-country" columns contain missing values. We can create a categoricals_with_nans list containing categorical columns with missing values and verify the percentage of missing values in each column. The results show that workclass has 5.73% missing, occupation has 5.75% missing, while native-country only have 1.75% missing. To treat these missing values we can use different strategies. For the workclass, we could remove the rows using df.dropna(), but it may delete useful information from our dataset. Alternatively, we can fill the missing values with the most frequent value of each column by running the code df['column']=df['column'].fillna(df['column'].mode()[0]). This fills the NaN values in the selected column with its mode. The [0] only selects the first mode value, in case of multiple modes. We use the df.info() code to confirm that there are no more missing values in the dataset. For the occupation, were used a group-based imputation, where the function fills missing values using the most common occupation within the same education level. This gives a more realistic approach to the data. For the native-country, missing rows were dropped with df.dropna(subset=["native-country"]), which is acceptable when the percentage of missing values is small. In the end, by using df.info() we observe that there are no more missing values in the dataset.

Regarding the exploration of the target variable, the income distribution graph shows that the large majority of individuals earn 50K or less, indicating a significant class imbalance. Using df['income'].value_counts(normalize=True), which counts the number of occurrences of each unique value in the income column and return their proportions (normalize=True), we find that approximately 76% of individuals earn 50K of less, while only 24% earn more than 50K. This imbalance suggests that models trained on this dataset may be biased toward predicting the majority class if appropriate evaluation metrics are not selected.

The age distribution histogram shows that most people in both income categories are concentrated in the age range of 20 to 50 years, indicating more people in younger age groups. The density of the ≤50K population peaks at younger ages and decreases gradually. This suggests that most young adults (who are early in their careers or in entry-level jobs) fall into this lower income category. The higher-income (>50K) population peaks at slightly older ages, suggesting that income tends to increase with age and experience. To visualize outliers in the dataset, an age boxplot was created and it shows that most individuals are between 28 and 48 years old, with a median age of around 37, although there are outliers representing much older individuals.

The hours-per-week distribution plot shows that most people work around 40 hours per week, which is visible at the peak of both curves. Individuals earning more than 50K generally work more hours per week than those earning less or equal than 50K, with a concentration around 40–60 hours. Additionally, these group also displays longer tails at higher hours (>50–60 hours per week), which may reflect individuals in more demanding roles or with multiple jobs. Regarding outliers, the hours-per-week boxplot shows the median around 40 hours/week, consistent with the KDE plot. There are extreme outliers in both ends, ranging from individuals who work only 1 hour per week to those working as many as 99 hours per week.

For the capital-gain data, a log1p transformation (log(1 + x)) was applied to reduce the impact of extreme outliers. This step is crucial since these data have the majority of values equal to zero and the command compresses the scale for large gains, making the distributions easier to compare. From the KDE and Boxplot graphs, we can observe that the majority of individuals in both income classes have no capital gains. The people earning more than 50K tend to have higher capital-gain values. This can be confirmed by the little peak around 8-10 bigger than the <=50K group peak in the KDE graph, and by the higher outliers in the Boxplot.

Regarding the capital-loss plots, a log transformation was also applied. We can analyze that the majority of people in both classes have no capital losses. People with more than 50K income generally experience slightly higher capital-loss values, but the difference is less pronounced than with capital-gain.

To analyze the relationship between education and income, the educational categories were first ordered to facilitate the plot visualization. We notice that although having a high school degree or some college education is the most frequent education level, advanced degrees (Bachelors, Masters, Doctorate) are the most effective predictors for achieving a higher income.

The analysis of the sex distribution plot reveals a significant gender-based income difference. Although the majority of both sexes earn less or equal than 50K, we can observe that men have a higher income in both groups (<=50K and >50K). Additionally, the proportion of people earning >50K is higher among men than women.

Upon analyzing the race distribution, we notice that the 50K-or-less class predominates for all races. The White group concentrates the majority of people for both incomes (<=50K and >50K), reflecting the composition of the population sampled in the census data. These group also shows the highest proportion of high earners, indicating a greater likelihood of earning more than 50K.

3) Data Preparation:

To use Machine Learning Models, we need to encode categorical variables because these models can only process numerical data. There is no single best method for encoding these variables. The choice depends principally on the nature of the categorical variable (nominal, ordinal, or binary) and the specific machine learning algorithm planned to use. The One-Hot Encoding (OHE) method is generally the safest and most common default for nominal variables.

The initial steps of data preparation include encoding the target variable using LabelEncoder(). Specifically, le = LabelEncoder() initializes the encoder, and y_encoded = le.fit_transform(df['income']) converts the <=50K values to 0 and the >50K values to 1. In the feature set (X), the income column is dropped to prevent data leakage, along with non-predictive or redundant columns such as fnlwgt (a sampling weight) and education-num (which is redundant with education). Additionally, log transformation was applied to the capital-gain and capital-loss features to reduce skewness and the impact of extreme outliers, improving the model performance. Then, lists of categorical and numerical columns were created.

The train/test split was performed, where 20% of the data was selected for testing; the command stratify=y maintains the proportion of classes in training and testing, which is essential because of the imbalance; and random_state=42 ensures reproducibility. The code may raise a SettingWithCopyWarning, thus were created copies to suppress the issue.

Furthermore, Capping is a technique used to treat extreme outliers in the dataset. It replaces any data point that falls outside a defined range with the boundary value itself. In essence, the most extreme 1% of the data is capped at the 1st and 99th percentile thresholds. This method was used particularly for features like age, hours-per-week, and the log-transformed capital-gain/capital-loss.

Different transformations were applied using a ColumnTransformer pipeline. Numerical features were standardized using StandardScaler(). This is a process of scaling numerical features so that they all have a mean of 0 and a standard deviation of 1, preventing large-scale features from dominating the model. Categorical variables were transformed with One-Hot Encoding, dropping the first category to avoid multicollinearity and ignoring unknown categories in the test set. The preprocessing was fit only on the training data and then applied to the test data to avoid data leakage. The output confirms the 80% train / 20% test split and the class imbalance of the dataset. The result is a processed dataset, where numerical features are scaled and categorical features are encoded, making it suitable for training machine learning models.

4) Modelling: Test of 3 Different Classifiers

- **Logistic Regression (LR):**

Logistic Regression is a fundamental linear classifier often used as a baseline for binary classification problems. It models the probability that a given input belongs to a specific class (income > 50K in this dataset) using the logistic (sigmoid) function. This function finds a linear combination of features that separate the classes and is particularly effective for linearly separable data.

The selected hyperparameters were: max_iter=1000, to ensure convergence given the large dataset and the high dimensionality introduced by one-hot encoding; C=1.0, which represents the standard regularization strength and provides a balance between overfitting and underfitting; and solver='lbfgs', an efficient and stable optimization algorithm suitable for medium to large datasets with high-dimensional feature spaces.

- **Random Forest Classifier (RF):**

Random Forest is an ensemble tree-based model known for its high accuracy and robustness to noisy data. Each tree is trained on a random subset of features and samples. For a prediction, the final class is determined by the majority vote of all individual trees in the forest. This reduces the high variance typically found in a single decision tree and improves generalization.

The selected hyperparameters were: n_estimators=200, which provides stable performance by reducing variance through a larger number of trees; max_depth=10, which restricts tree growth to prevent overfitting while still capturing meaningful non-linear patterns; and random_state=42, ensuring full reproducibility of the results

- **Gradient Boosting Classifier:**

Gradient Boosting is an ensemble tree-based model and it builds a strong predictive model by sequentially combining predictions from many models, where each new tree corrects the errors of the previous ones. It uses gradient descent to minimize the loss function and is powerful for structured/tabular data.

The selected hyperparameters were: n_estimators=200, providing boosting rounds to capture meaningful patterns; learning_rate=0.1, a standard choice that balances model performance and overfitting risk; and random_state=42, ensuring reproducibility of the results

5) Evaluation Using Classification Metrics:

Accuracy measures the proportion of correct predictions. However, in imbalanced datasets this metric can be misleading. Thus, additional metrics are important to evaluate both classes correctly. Confusion Matrix shows the counts of: True Positives (TP) - correctly predicted positives; True Negatives (TN) - correctly predicted negatives; False Positives (FP) - incorrectly predicted positives; and False Negatives (FN) - incorrectly predicted negatives.

Precision measures, out of all the instances the model predicted as positive, how many were actually positive. Useful when false positives are costly. Recall measures, out of all the actual positive cases, how many the model correctly identified as positive. Useful when false negatives are critical. F1-score is the harmonic mean of precision and recall, and is useful for imbalanced datasets.

ROC-AUC measures how well the model can distinguish between the positive and negative classes across all possible classification thresholds. ROC curve plots the true positive rate against the false positive rate at different thresholds and AUC summarizes the curve into a single number between 0 and 1, where higher values indicate better discrimination. It is threshold-independent, meaning it evaluates the model's ability to separate classes regardless of the cutoff used to classify predictions.

The dataset has many more people earning ≤50K than >50K. This imbalance can cause problems, such as majority class bias (models may predict the majority class (≤50K) most of the time, ignoring the minority class) and misleading accuracy (models can have high overall accuracy just by predicting the majority class). To get a true picture

of performance, it's important to look at precision, recall, F1-score, and ROC-AUC, which better capture how the model performs on both classes.

For all three models we can observe from the confusion matrix that class 0 support (income<=50K) is equal to 7304, while class 1 support (income >50K) is 2293. This confirms the dataset is imbalanced, with ~76% class 0 and ~24% class 1, making it easier for the models to correctly classify low earners than high earners.

Summary of Model Performance:

	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	0.840784	0.705094	0.573485	0.632516	0.893953
Random Forest	0.854642	0.817090	0.504579	0.623888	0.906299
Gradient Boosting	0.867771	0.784129	0.616223	0.690110	0.919907

Upon analyzing the results, we observe that accuracy is highest for Gradient Boosting, followed by Random Forest and Logistic Regression. Precision is generally higher than recall for Random Forest and Gradient Boosting, indicating these models are better at predicting high-income correctly when they do predict it. Recall is lower than precision for all models, highlighting the challenge of detecting high-income individuals in a highly imbalanced dataset. ROC-AUC values indicate good discrimination ability across thresholds, with Gradient Boosting performing best.

When comparing the models, we observe that Gradient Boosting has the highest accuracy, best F1-score, and highest ROC-AUC. This confirms that boosting ensembles are generally the strongest performers on this type of structured data. Random Forest model shows very high precision but the lowest recall, meaning that when it predicts high income, it is 84% of the time. However, it misses half the actual high earners. Logistic Regression has the lowest overall metrics but also provides a decent balance, with an F1-score close to the Random Forest model.

False Positives (FP) occur when model predicts >50K, but the person earns ≤50K, while False Negatives (FN) occur when model predicts ≤50K, but the person earns >50K. When comparing the models, we observe that Logistic Regression shows more FN than FP. This occurs because the model is linear and conservative, underestimating the minority class and it often predicts ≤50K, leading to more missed high-income individuals. Random Forest has fewer FP but more FN, since ensemble trees are very precise in predicting high-income, but class imbalance causes the model to miss many high-income cases, increasing FN. Gradient Boosting shows a balanced FP/FN ratio, particularly because these models sequentially correct errors from previous trees, improving detection of high-income individuals (reduces FN) while keeping FP moderate. This is why Gradient Boosting achieves the best F1-score and ROC-AUC.

Model	Advantages	Disadvantages
Logistic Regression	- Simple and easy to interpret - Fast to train - Good for linearly separable data	- Limited to linear relationships - Not good for complex patterns - More FN in imbalanced datasets
Random Forest	- Captures non-linear relationships - Robust to outliers - Good for high-dimensional data	- Can overfit if not tuned - Less interpretable than LR - May produce more FP
Gradient Boosting	- Typically the best accuracy - Good with imbalanced data - Strong modelling power	- Slower to train - Sensitive to hyperparameters - Can overfit if not tuned

6) Statistical Tests - Comparing proportions and A/B Testing:

To examine whether the proportion of individuals earning more than 50K differs between men and women, a two-proportion z-test was conducted. The null hypothesis (H_0) states that the proportion of women earning >50K is equal to that of men, while the alternative hypothesis (H_1) proposes that these proportions differ. A two-proportion z-test is appropriate because we are comparing proportions from two independent groups with large sample sizes. The results show a p-value <0.05, indicating strong evidence against the null hypothesis. This suggests that the difference in high-income proportions between men and women is statistically significant. Specifically, men show a higher proportion of individuals earning more than 50K compared to women. In conclusion, the analysis suggests that gender is associated with income levels within this dataset.

A weekly working hours A/B test was performed to investigate whether individuals working more than 40 hours per week (Group A) have a different proportion of high-income earners compared to those working 40 hours or less (Group B). The null hypothesis (H_0) states that the proportion of individuals earning >50K is the same in both groups, while the alternative hypothesis (H_1) is that these proportions differ. As before, a z-test for two proportions was also applied for the same reasons. The test returned a p-value <0.05, indicating a statistically significant difference between the two groups. Individuals working more than 40 hours per week have a higher proportion of high-income earners than those working 40 hours or less. In conclusion, weekly working hours appear to be associated with income levels, suggesting that longer working hours may correlate with higher earnings.

Both gender and weekly working hours show statistically significant differences in the proportion of individuals earning more than 50K. Since the dataset is large and the outcomes are binary, two-proportion z-tests were appropriate for these comparisons. The results reinforce patterns observed during exploratory data analysis, providing additional evidence that gender and working hours are factors associated with income levels.

7) Conclusion:

In this project, we analyzed the Adult Census Income dataset with the objective of predicting whether an individual earns more than $50K per year. The analysis followed a complete data science workflow, including data exploration, preprocessing, feature engineering, statistical testing, model training, and evaluation.

During data preparation, categorical variables were encoded using One-Hot Encoding, numerical features were standardized, and skewed variables such as capital-gain and capital-loss were log-transformed to reduce the impact of extreme values. Outlier capping was applied using training-only statistics to avoid data leakage. This ensured that the final dataset was suitable for machine learning models and that comparisons between models were fair and reliable.

Three classification models were evaluated: Logistic Regression, Random Forest, and Gradient Boosting. Logistic Regression served as a strong baseline, providing interpretability and stable performance, but showed limitations in detecting high-income individuals due to the class imbalance. Random Forest achieved higher precision, meaning fewer false positives, but at the cost of lower recall, missing a larger number of high-income cases. Gradient Boosting achieved the best overall performance, with the highest F1-score and ROC-AUC, demonstrating a better balance between precision and recall and superior discrimination capability.

Because the dataset is imbalanced, accuracy alone proved insufficient for model comparison. Metrics such as precision, recall, F1-score, ROC-AUC, and confusion matrices were essential to fully understand model behavior. In particular, Gradient Boosting produced fewer false negatives compared to the other models, making it more suitable when correctly identifying high-income individuals is important.

Statistical hypothesis tests further supported the findings. Significant differences were observed between income proportions across gender and weekly working-hour groups, reinforcing the relevance of demographic and work-related features in income prediction.

Overall, Gradient Boosting was the most effective model for this task, offering the best trade-off between predictive performance and robustness under class imbalance. Future work could include hyperparameter optimization, cost-sensitive learning, or fairness analysis to further improve model performance and interpretability.