Data Mining

Authors

Karolina Wypych 262333, Mateusz Stasiak 262339

Politechnika Wrocławska

Faculty of Pure and Applied Mathematics
May 2024

Contents

1 Introduction

The main goal of the project is to use data mining methods introduced during Data Mining course to perform a complete analysis of Adults Income Dataset, related to a specific practical problem. This report is just the first part of the project and covers Exploratory Data Analysis and classification.

1.1 Data

The following report is based on Adult Income Dataset also known as Census Income Dataset. It consists of characteristics of United States citizens in 1994. The dataset is commonly used for classification problems. A default prediction task is to determine whether a person makes over $50K$ a year. To compare it to the reality information on income in the most popular place in the dataset — the USA was found. According to information at census website mean income in 1994 was equal to 43130$, so those who earn more than $50K$ can be treated as rich ones.

Table 1: Characteristics of Adult Income Dataset.

Observations	Features	Target value	Missing values
48842	14	Income	Yes

Source: Own elaboration.

1.2 Dataset source

This data was provided by donors Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics) to 1994 Census database. Nowadays it can be found at archive.ics.uci.edu site. The dataset can also be imported directly in python from *ucimlrepo* library:

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
adult = fetch_ucirepo(id=2)

# data (as pandas dataframes)
X = adult.data.features
y = adult.data.targets
```

Figure 1: Importing data in python.

1.3 Dataset variables

Adult Income Dataset consists of 15 columns of different types. Their characteristics are described in Table 2.

2 Preliminary analysis and data cleaning

Studying the data before applying a specific probabilistic model is a crucial step of data mining process. EDA is a stage where simple statistical techniques (including summary measures) and graphical tools are used to learn its most important properties, understand its specifics, etc. In short terms EDA includes descriptive analysis and data visualisation.

Main steps of Exploratory Data Analysis are:

1. Getting familiarity with data
2. Basic data preparation
3. Analysis of individual variables (univariate analysis)
4. Analysis of relationships (correlations) between variables
5. Interpretation of results

In order to preserve the realistic behavior while working on the given dataset, it was first split into train and test subsets with ratio $4:1$ respectively. All analyses were conducted on the created this-way treining set.

2.1 Class imbalance

Class imbalance is a familiar problem occurring in classification when a dataset has a severely unequal ratio of data points in a target class.

Most common ways of dealing with imbalanced data:

- collecting more data,
- random under/oversampling,
- creating synthetic data (e.g. SMOTE),
- cost-sensitive learning.

This problem concerns the analyzed dataset too— the group of people whose annual income is lower than or equal to $50K$ is nearly three times than the group of those who don't. The disparity in class sizes is not devastating but significant enough to require the use of methods for dealing with imbalanced datasets.

Table 2: Characteristics of variables.

Variable Name	Role	Type	Demographic	Description	Missing Values
age	Feature	Integer	Age	N/A	no
workclass	Feature	Categorical	Income	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.	yes
fnlwgt	Feature	Integer		final weight, fnlwgt variable is designed to represent the total number of people in the population that each respondent represents or "stands for." This number helps to balance out the sample and make it more representative of the entire population.	no
education	Feature	Categorical	Education Level	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.	no
education-num	Feature	Integer	Education Level		no
marital-status	Feature	Categorical	Other	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.	no
occupation	Feature	Categorical	Other	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.	yes
relationship	Feature	Categorical	Other	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.	no
race	Feature	Categorical	Race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.	no
sex	Feature	Binary	Sex	Female, Male.	no
capital-gain	Feature	Integer			no
capital-loss	Feature	Integer			no
hours-per-week	Feature	Integer			no
native-country	Feature	Categorical	Other	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.	yes
income	Target	Binary	Income	$\leq$50K, >50K	no

Source: Own elaboration.

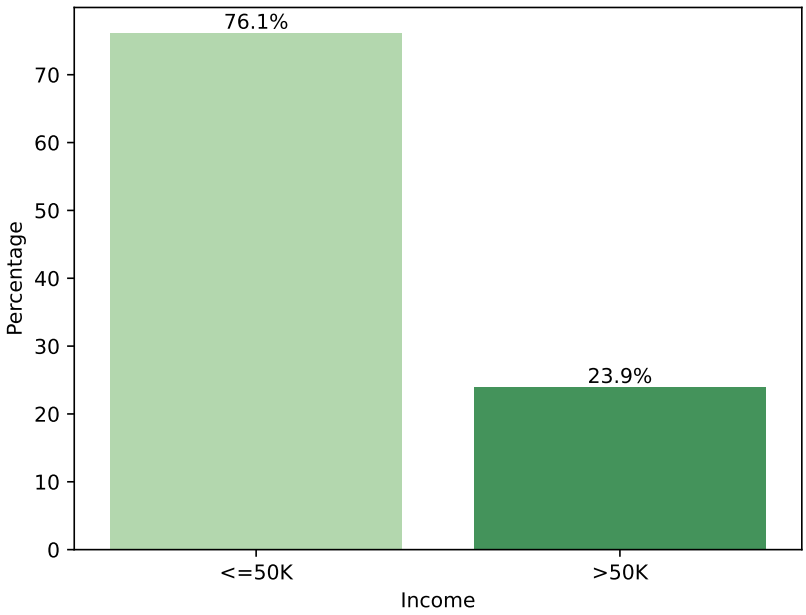


Figure 2: Income percentage distribution for Adult Income Dataset.

2.2 Missing values

Most of the algorithms require data to have no missing values. That is why identifying potential missing values and dealing with them is a crucial step in exploratory data analysis.

The Adults Income Dataset has quite an unusual notation for missing values. They are denoted in two ways by " ?" or by " ". That is why before further analysis they should be converted to so-called not-number values.

There are not many missing values (they appear in only 7.4% of rows), but they should be filled in to effectively perform further data operations. Features with missing values are shown in Figure 3.

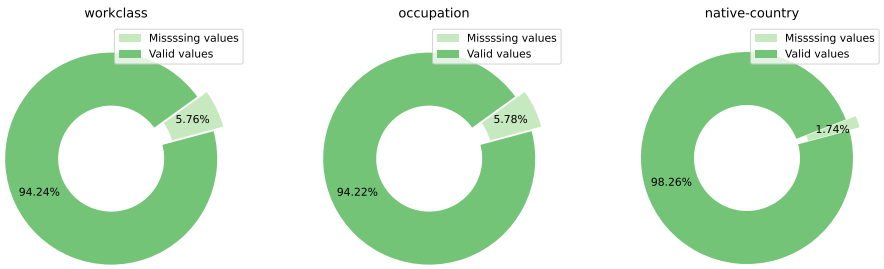


Figure 3: A percentage share of missing values for raw data.

Missing values appear in only 3 columns of the dataset. Their share is close to 2% for *native-country* and 6% for *workclass* and *occupation*.

Before handling missing values, it's better to review the remaining data, as certain groupings or other operations may lead to the creation of new nulls that will need to be addressed.

2.3 Default test subset notation

The dataset found in *ucimlrepo* was prepared to be split into train and test subsets accordingly to authors' will. Last 30% of the data has a dot after its *income* values. In this report a different division was planned so these dots had to be removed.

```
# Delete dots from test subset
df['income'] = df['income'].replace({"<=50K.":"<=50K", ">50K.":">50K"})
```

Figure 4: Removal of dots.

2.4 Selection of education variables

Educational information was provided in two columns, using qualitative nominal (*education*) and qualitative ordinal (*education-num*) variables. As they yielded the same information, it was decided to get rid of the first one. Although it was a more descriptive and easily interpretable variable, the order of the educational variable was ruled to be the most important factor for further analysis.

2.5 Capital balance

Capital information was provided in two columns, using quantitive discrete variables. In order to simplify further analysis, they were replaced with a variable called *capital* representing their balance.

```python
# Capital bilans
df["capital-gain"] = df["capital-gain"] - df["capital-loss"]
df = df.rename(columns = {"capital-gain":"capital"})
df.pop("capital-loss")
```

Figure 5: Replacing two capital information columns with a variable representing their balance.

2.6 Initial data visualization

Due to data visualization, it is possible to gain familiarity with considered data and notice some suspicious values that should be taken into consideration during further analysis.

Looking at histograms generated for numerical variables (Figure 6) one can observe that plots of *capital* and *hours-per-week* have dominant values. While for hours this can be easily explained (accorring to statista.com 40 represents full-time employment), in the case of capital, it is puzzling that the majority of surveyed individuals report capital equal to 0. So, it is necessary to analyze this situation further in the subsequent section of the report.

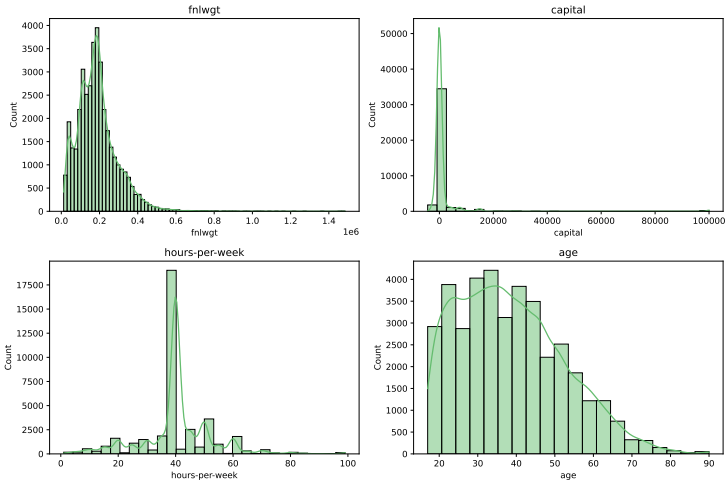


Figure 6: Histograms for numerical variables

There are more categorical variables than numerical ones. At first glance, it is evident that several features are imbalanced: *workclass*, *race* and *region* (Figrue 7).

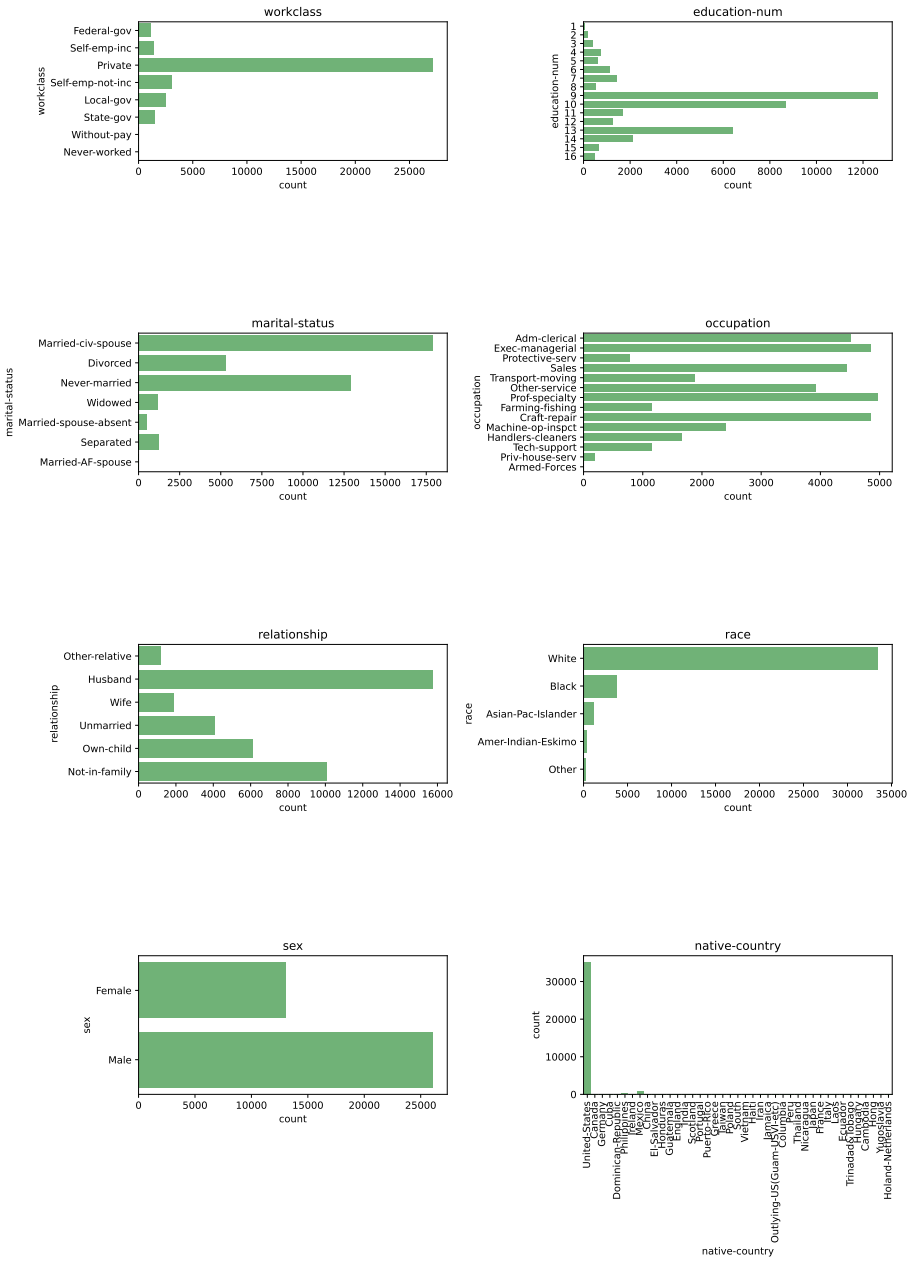


Figure 7: Frequency distribution of categorical variables.

The biggest disparity is observed for the United States. Such a pronounced dominance of this country makes it impossible to read the values assigned to the other ones. In order to mitigate this problem, a frequency distribution after

the United States removal was generated.

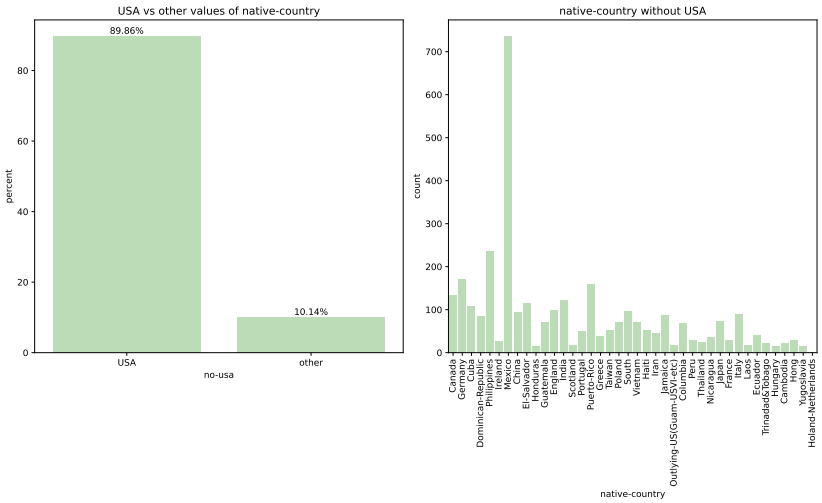


Figure 8: Frequency distribution concerning *region* feature.

Figure 8 shows that the percentage of people coming from the USA is nearly ten times as high as individuals born in other countries (including unknown places). In the plot on the right one can observe that Mexico is the second most popular country of origin.
It is undeniable that the *region* consists of many classes, with only a small percentage of observations belonging to most of them. Therefore, a good idea is to group rare categories.

2.7 Age groups creation

When classifying people based on annual income, small age differences do not play a significant role. For example, a person aged 54 does not differ much in terms of job seniority and experience from someone who is 55. That is why feature *age* was grouped into age ranges. The result of this operation is shown in Figure 9.

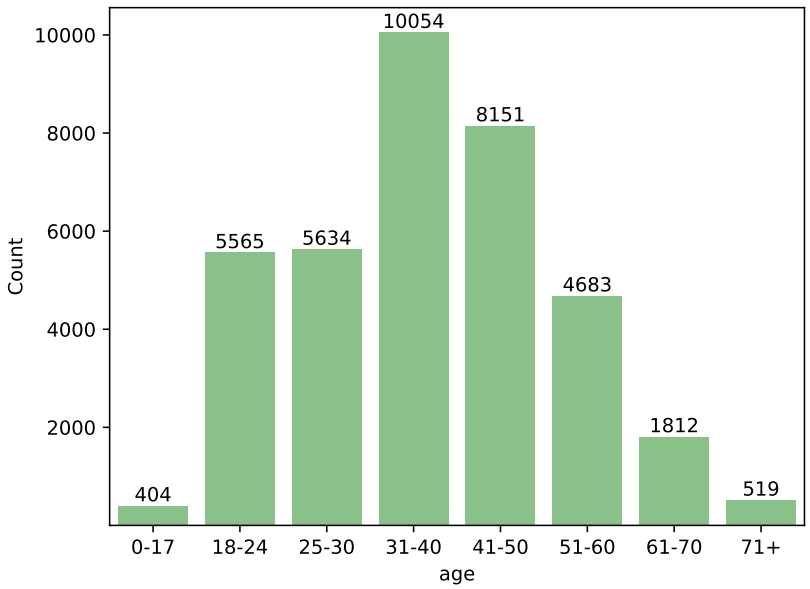


Figure 9: Distribution concerning *age* feature.

2.8 Mapping countries

According to observations based on Figure 8 there is a severe imbalance in *native-country* variable. In order to moderate this phenomenon, countries were grouped by geographical location. Those with a significantly higher occurrence were left unaltered. Moreover, some ambiguous values such as *South* and *Hong* were found among *native-country* values. Each one of them was replaced with *NaN*. Such mapping was presented in Table 3.

Because of the fact that this operation has introduced some additional *NaN* values to the dataset, handling them earlier would not have been the best choice.

Table 3: The mapping of *native-country* variable into new *region* variable.

Country	**Region**
United-States	United-States
Cuba	Middle America
Jamaica	Middle America
India	South Asia
Mexico	Mexico
South	*NaN*
Puerto-Rico	Middle America
Honduras	Middle America
England	West Europe
Canada	Canada
Germany	Central Europe
Iran	West Asia
Philippines	Southeast Asia
Italy	Central Europe
Poland	East Europe
Columbia	South America
Cambodia	Southeast Asia
Ecuador	South America
Holand-Netherlands	Central Europe
Dominican-Republic	Middle America
Trinadad&Tobago	South America

Country	**Region**
Haiti	Middle America
Portugal	West Europe
El-Salvador	Middle America
France	West Europe
Guatemala	Middle America
China	East Asia
Japan	East Asia
Yugoslavia	East Europe
Peru	South America
Outlying-US	Outlying-US
Scotland	West Europe
Greece	South Europe
Nicaragua	Middle America
Vietnam	Southeast Asia
Hong	*NaN*
Ireland	West Europe
Hungary	East Europe
Laos	Southeast Asia
Taiwan	West Asia
Thailand	Southeast Asia

Source: Own elaboration.

2.9 Suspicious values

Some values in the dataset seem to be suspicious. To understand their origin better and decide whether they can be kept or not some analyses were conducted.
These values were found while investigating the maximum and minimum values of some features in the data set. The applied approach was based on finding 10 smallest or largest values and later on looking for and analyzing reasons for those boundary values.

```
def check_largest(df, column, n):
    x = heapq.nlargest(n, df[column].unique())
    print(x)

def check_smallest(df, column, n):
    x = heapq.nsmallest(n, df[column].unique())
    print(x)
```

Figure 10: Functions used to find extreme values.

2.9.1 hours-per-week

One of the most surprising was *hours-per-week* indicating the number of hours per week a person works. For 421 (approximately 1% of the whole dataset) of people, it exceeds or equals 80 which at the time of data collection was equivalent to two full-time jobs (sprawdzić czy to prawda). This also means that assuming someone works 7 days a week, they would have to work at least approximately 12 hours a day to achieve such a result. To understand better this small anomaly more specific analyses were made.
"Firstly, charts were generated taking into account income and occupation for individuals working at least 80 hours per week. On the second subplot, one can observe that these values appeared among most of the representatives of various professions. Looking at each of them individually helps to understand this phenomenon.

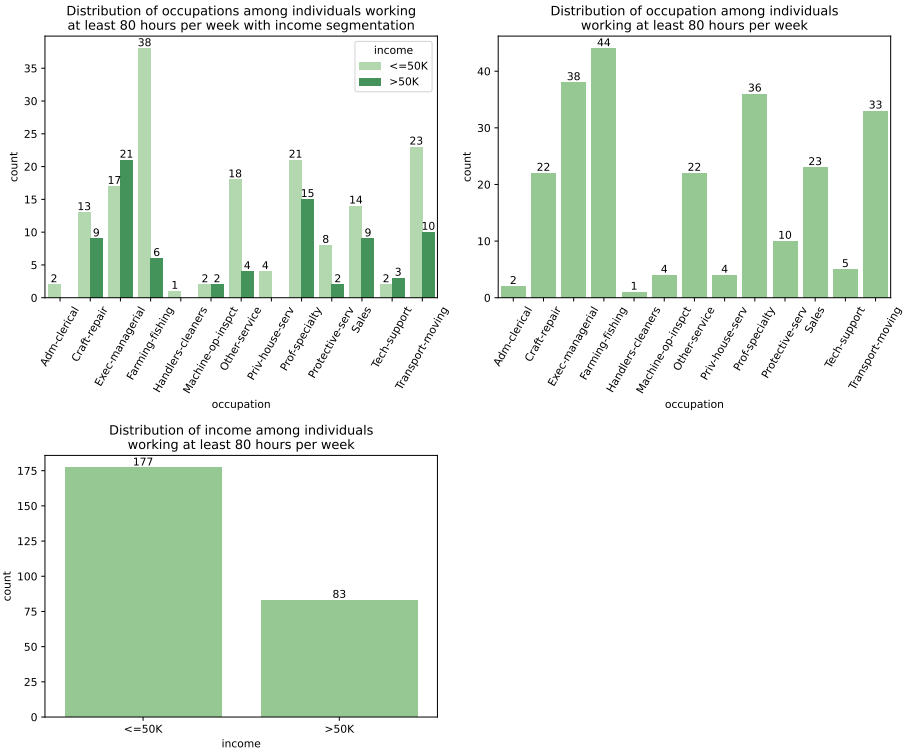
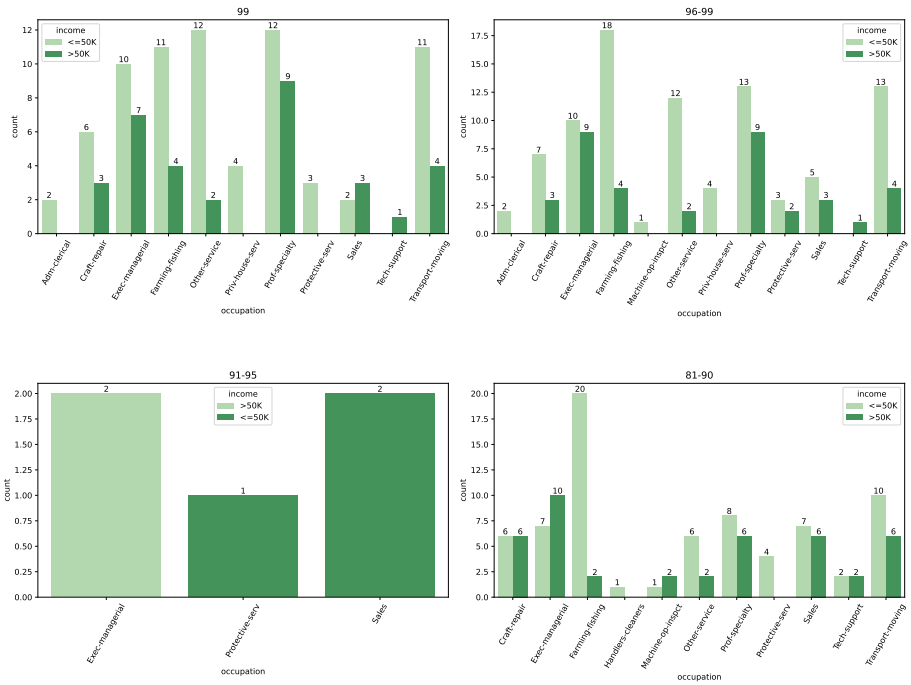


Figure 11: Figures representing income and occupation of people working at least 80 hours per week.

Generally for all observations one could assume that the long hours spent at work result from being the owner of a company with a certain profile, who often is compelled to work longer hours than other employees. Another argument in favor of keeping these values in a dataset is the fact that the findings are (coherently with intuition) dominated by professional specialists, executive managers, and farmers and fishers. The latter have to take care of their livestock and crops from early morning till late evening. The first one group is very diverse and consists of i.a. doctors and lawyers who sometimes spend at work 24 or 48 hours without any break or solve their clients' problems late hours. Whereas executive managers are sometimes obliged to spend more time at work to ensure everything is taken care of.
What may be surprising, in the majority of cases working a lot doesn't lead to being rich. Generally over twice as many people earn below $50K$ annually. The most significant disparity is observed in the farming-fishing sector, where the number of workers with lower incomes surpasses the rich ones. They usually have smaller farms that can't generate high earnings but still require taking care of the livestock for the whole day. Only a small group has big farms that can be high-income businesses, what the height of the bars on the plot reflects.



r

Figure 12: Professions distribution with income segmentation for hard-working people for selected hours per week intervals.

Additionally, specific ranges of hours were considered. Figure 12 presents that analyzed people usually come from different professions. Looking at these subplots one can conclude that the smallest group of hard-working people works between 91-95 per week while more observations were reported in other ranges. The fact that the group of people who work 99 hours is numerous may indicate that during the survey, it was not possible to enter a three-digit number of hours and in reality, some people work 100 or more.

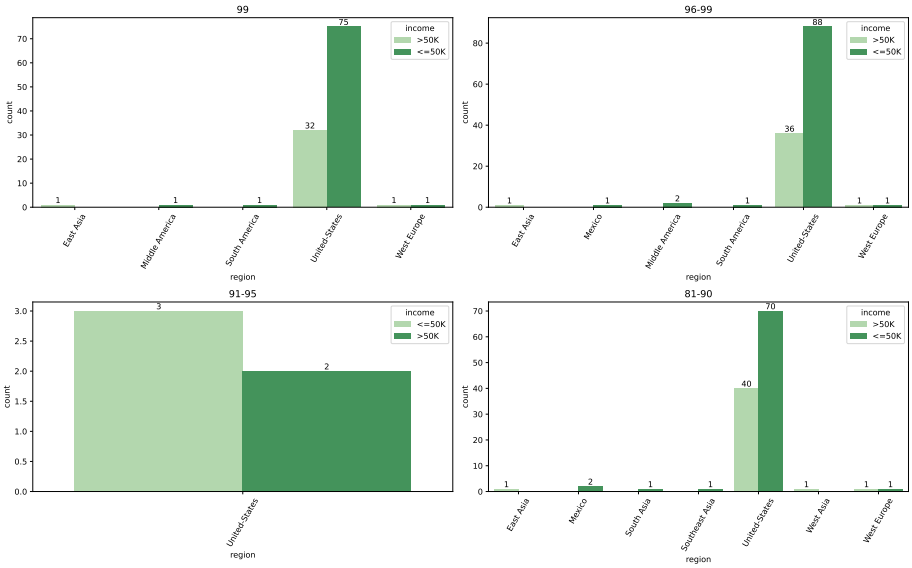


Figure 13: Region distribution with income segmentation for hard-working people for selected hours per week intervals.

The analysis also examined how earnings in various subgroups vary across different regions of the world 13. Each time, the United States dominates (as it predominates in the dataset), but with minor values, other areas have also appeared.
All this analysis proved that these extreme values are reasonable so the decision was made to to keep all records with extremely high values of *hours-per-week*.

Some remarkably low values were observed too. They appeared for people of different professions and achieved the highest level for other services and prof-specialty. Initially, such a small number of hours worked seems to be ridiculous but after some considerations possible explanations of this issue can be found.

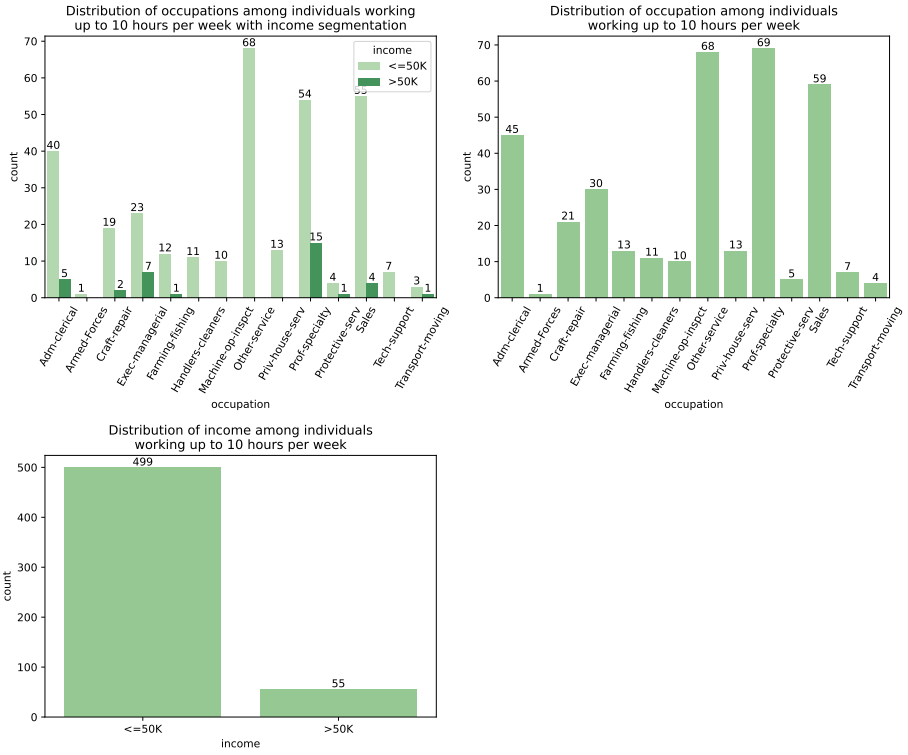


Figure 14: Figures representing income and occupation of people working up to 10 hours per week.

Right after individuals whose occupation is unknown, the largest groups consist of representatives of other services and professional specialties. The first one can include individuals working casually, even for just one day per week, for example, providing tutoring to one or two students (they don't earn a lot). Sometimes somebody can be assigned to a given occupation while in reality they just slightly help with something.
High incomes for individuals who don't work much appear for several categories i.a. executive managers and professionals that were mentioned above. These observations may cover business owners who developed their companies enough to limit their work to just controlling them occasionally (they may earn a lot). In general professionals are the largest group in the entire dataset. The term *Prof-specialty* covers different domains so it is also very diverse. That is why it is a popular occupational group both among those working long hours and those working few hours. For example, they may include both doctors who work a lot in the hospital and those who have their private clinics and personally conduct only several but very expensive visits per month.

Moreover distribution of income in this case was calculated. As presumed, the

overwhelming majority of this group earns less than 50K.

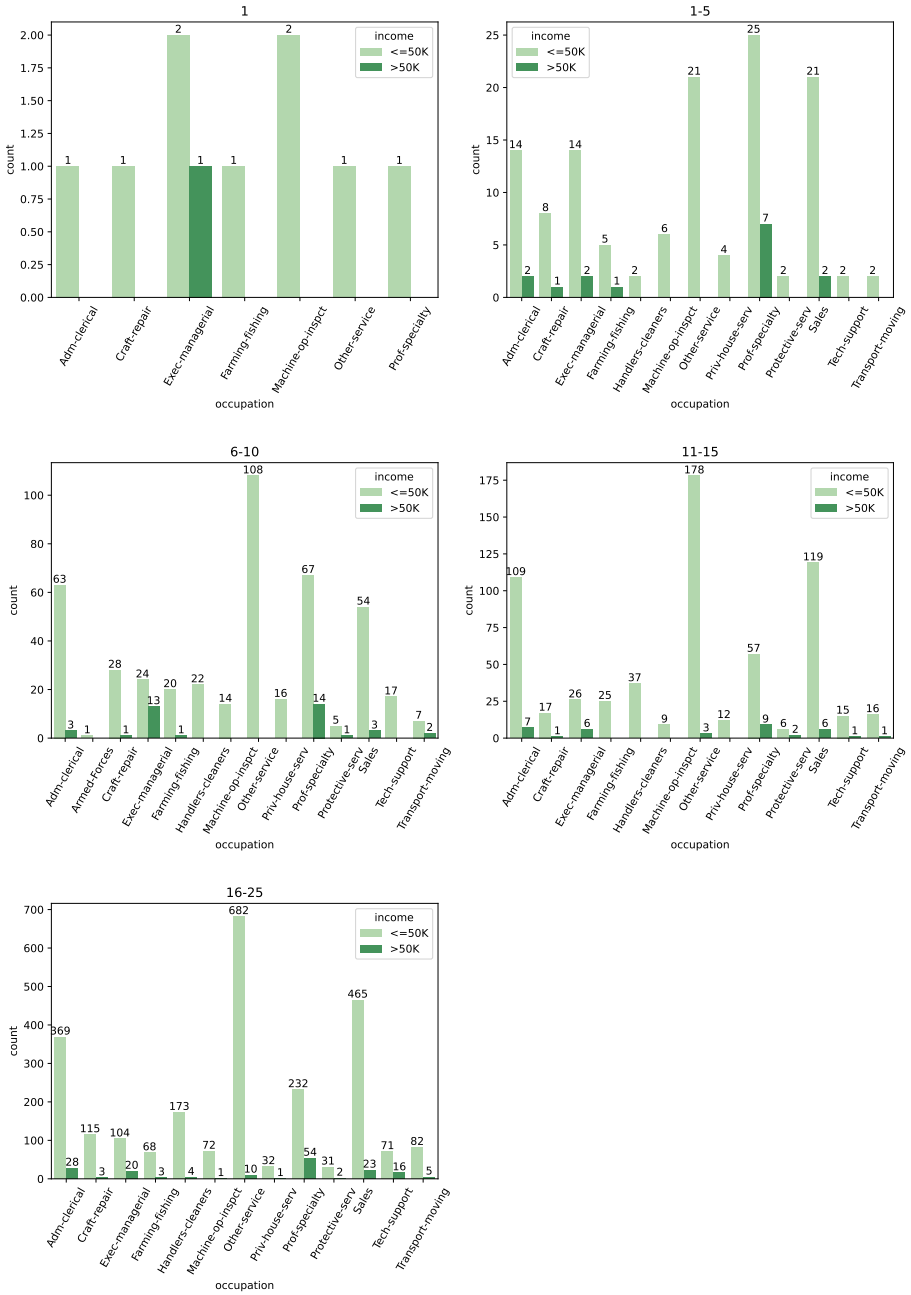
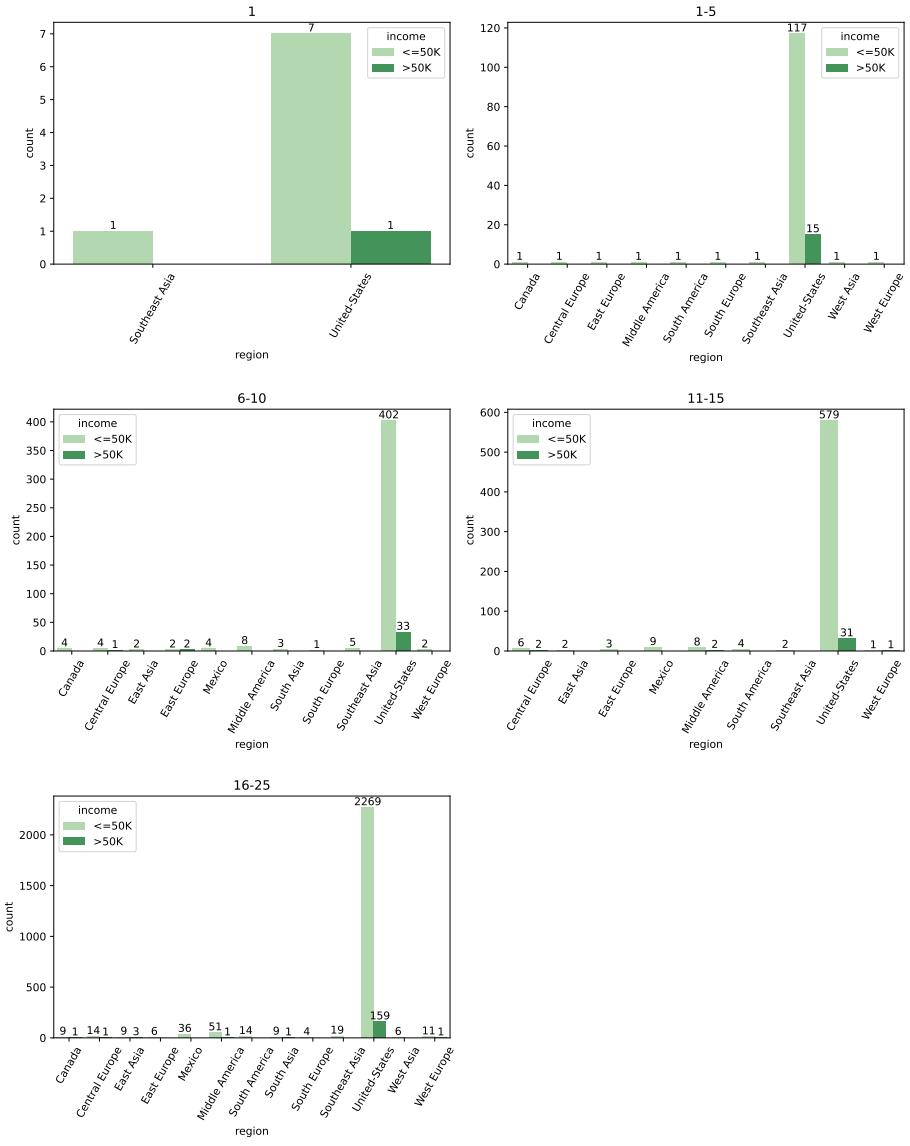


Figure 15: Professions distribution with income segmentation for people working little for selected hours per week intervals

There are no people in the dataset that reported 0 hours. There's a small group who worked for just 1 hour. Generally, such low values concern all occupations.



r

Figure 16: Region distribution with income segmentation for people working little for selected hours per week intervals

Again some ranges of hours were considered (Figure 16). Similarly to the case discussed above (13) the chart presents a clear dominance of the most numerous

group in the dataset - the United States. However, the numerical values on the charts are much higher than in the case of the other extreme. This is because it is definitely easier to work 30 hours less than 40 hours more.

Compared to the entire dataset, these extreme values seem to be reasonable, so they are kept in the dataset for further analysis.

2.9.2 capital

Some interesting phenomenon is observed also in a *capital* column. The largest value is 99999, with the second largest being only 41310. Initially, it was considered to treat it as an outlier but the fact that it appears 199 times suggests that it might be a valid value representing individuals earning six-figure income. As presented on Figure 17 this situation concerns representatives of many professions but is the most popular among professional specialists, executive managers and sellers (they are popular also when it comes to working a lot Figure 11).

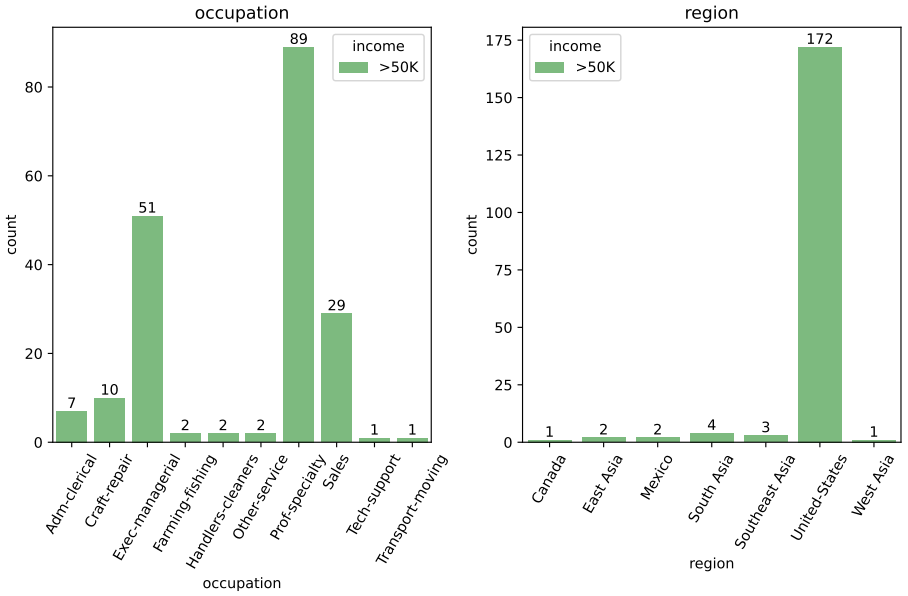


Figure 17: Plots showing distributions of occupation and region with income segmentation for people earning 99999

It is not surprising that all of these people have income greater than $50K$. In less lucrative professions like farming, fishing or technical support earning a lot is rare so the distribution of occupations is sensible. Additionally, *region* feature is dominated by the USA.
In conclusion, the analysis of the charts lets conclude that all 99999 values are valid and do not need to be removed.

2.10 Fnlwgt

Fnlwgt is a bit problematic variable. It was designed to represent the total number of people in the population that each respondent represents or "stands for". As data provider did not specify explicitly the equation for this measure, some own research was developed. Main approach was to calculate and compare mean *fnlwgt* for subsets of data grouped with other variables. Unfortunately no sensible conclusion was obtained as mean *fnlwgt* was not varying that much across all classes among sex, race, relationship, region variables.

Moreover, it is of vital importance to notice the influence of grouping records by *native-country* and *age* on *fnlwgt* interpretability. The measure was obviously disturbed as now grouped records are more universal and therefore more common than *fnlwgt* suggests.

Furthermore, when it comes to classification, the definition of *fnlwgt* implies it should have no influence on determining a person's income. As this variable is an integer with no upper bound it would also require some data preprocessing before being fed to a model.

That is why in this report it was decided to get rid of the *fnlwgt* variable.

3 Handling missing values with KNNImputer

Operations conducted in the previous section resulted in the appearance of new missing values. Thus they are visualized one more time in Figure 18 and the exact counts are recorded in Table 4.

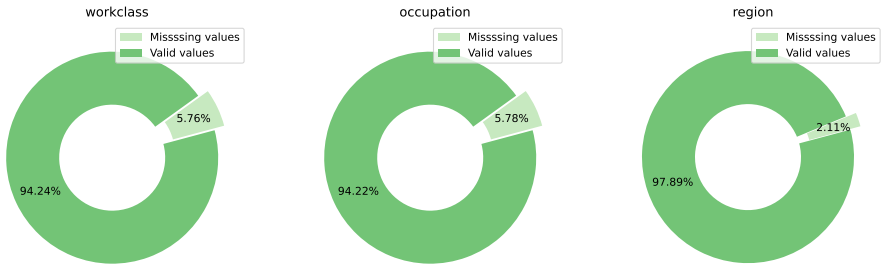


Figure 18: A percentage share of missing values for data after some transformations.

By grouping countries into regions, the number of missing values for the third feature increased as expected.

Table 4: Number of rows with a given number of missing values

missing values	**rows**
0	36043
1	779
2	2198
3	53

Source: Own elaboration.

Over 2200 rows have at least 2 missing values and it seemed to be a good idea to remove such cases from the dataset. To ensure that it doesn't significantly affect the distribution of the target variable *income*, the percentage share of each class was compared before and after removing records (Figure 19).

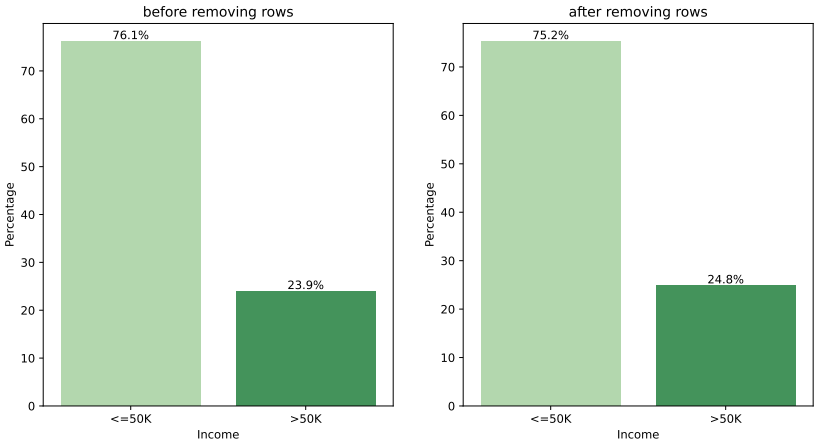


Figure 19: A percentage share of missing values for cleaned data.

The fact that the percentage share of classes changed only by 0.9 percentage points (which represents 3.8% of the minority class), confirms that removing rows with 2 or 3 missing values is a good approach.

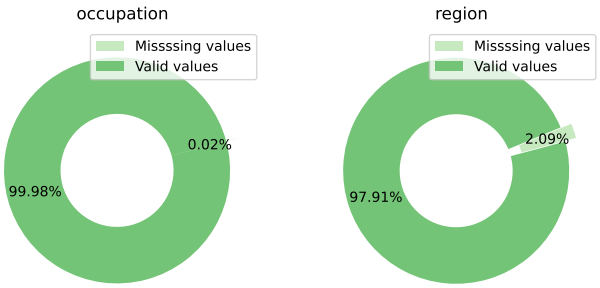


Figure 20: A percentage share of missing values after removal of some rows.

Row removal completely solves missing value problem for a *native-country* column, reduces it for *occupation* and slightly amplify percentage share for *region*. It can be repaired by handling with remaining 208 missing values being single in a row. One of ways of doing it is using KNN Imputer.

Handling missing values is a crucial step that needs to be performed before putting data into a model. To achieve the desired goal the KNNImputer is employed. This is a scikit-learn class used to fill in or predict missing values in a dataset. This method works on the basic approach of the KNN algorithm that predicts a value for a given observation by considering the K instances that are the most similar to it (more details in a subsection 5.3.2). This option is better than the naive approach of filling all the values with the mean or the median.

However KNNImputer accepts numerical data while most of the features in Adults Income Dataset is categorical. That is why all categorical features have to be transformed into numerical ones. It can be achieved by using a preprocessing tool LabelEncoder that enables machine learning models to work with categorical data by assigning numerical values to categorical labels. This approach is not perfect because assigning numerical values to categories, such as regions, in consecutive order, implicitly imposes an order on them. Thus, regions 1 and 2 could be considered closer to each other than regions 1 and 5, even though this may not be the case in reality. However, given that label encoding is a frequently employed solution for such transformations during missing value imputation, it was decided to stick with this approach.

During the imputation of missing values remaining after previous operations, the following values were assigned to the parameters:

- $n\_neighbors = 5$ — 5 neighboring samples are used for imputation,
- $weights = 'distance'$ — observations are weighted by the inverse of their distance, tzn. closer neighbors of observation with a missing value will have a greater influence than neighbors which are further away,
- $metric = 'nan\_euclidean'$ — Euclidean distance measure that is NaN aware.

After using such configured imputer, there are no missing values left in the dataset.

4 Features and target variable analysis

During the data cleaning process some useless variables have been identified: *fnlwgt* has been completely removed while *capital-gain* and *capital-loss* have been transformed into one variable called *capital* (subsection 2.5). Furthermore, all missing values were addressed and the issue of class imbalance was recognized. Potential inconsistencies in the data, namely unusual feature values, were identified, justified, and subsequently retained as valid values. Moreover, during the analysis, attention was given to how income and working hours were structured within individual countries (particularly focusing on the United States) and occupations during the period covered by the dataset.
After data preparation, the dataset has different characteristics from the original. They are listed in Table 5.

Table 5: Characteristics of transformed Adult Income Dataset.

Categorical variables	**Numeric variables**	**Number of observations**	**Missing cells**
9	4	36822	0

Source: Own elaboration.

4.1 Age

Currently, this column consists of 8 distinct string values that are duplicated many times. The exact number of occurrences is presented in Figure 21. The most numerous group covers individuals between 31 and 40 years old and the smallest one comprises teenagers up to 17 years old (which is unsurprising as most of them do not work but fully dedicate themselves to education and their hobbies, making them not particularly interesting for researchers collecting this data).

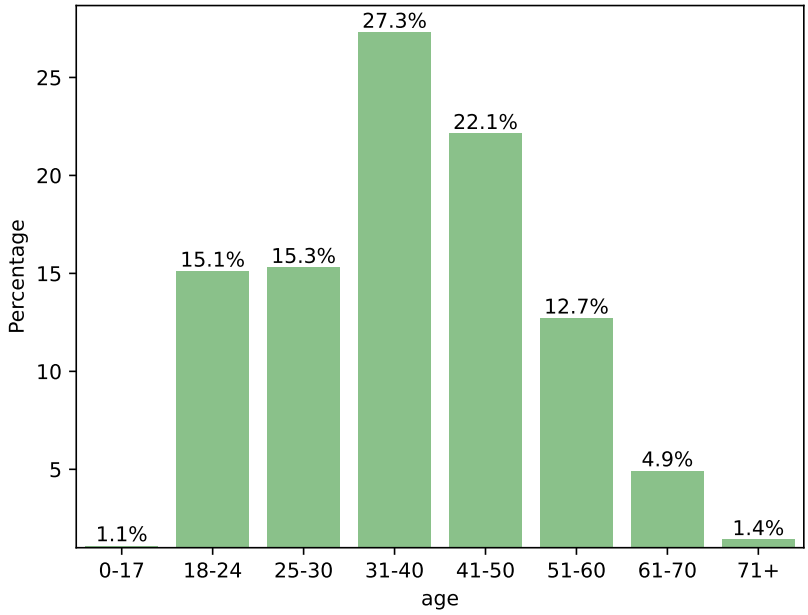


Figure 21: The population size of each age group.

The highest ratio between people with high income and low income is observed for those in their forties and fifties. For individuals younger than 25 and older than 71 high income does not appear at all.

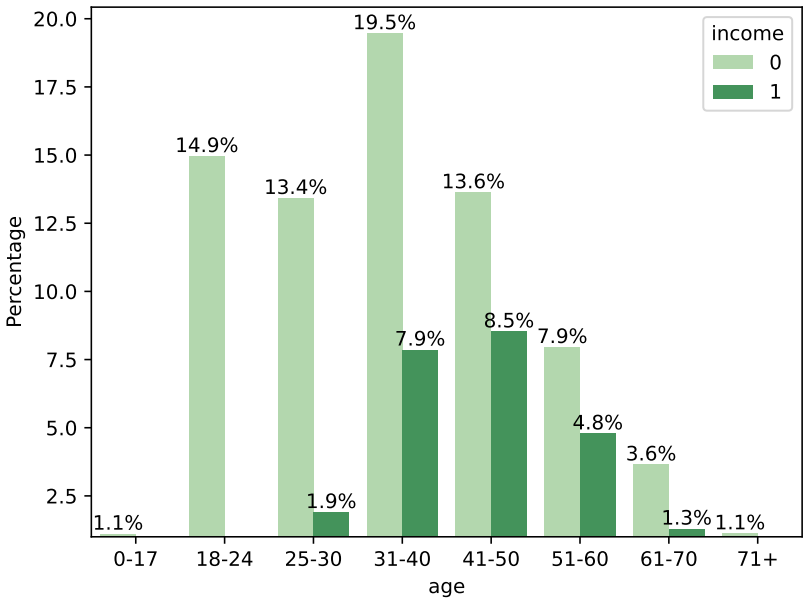


Figure 22: The population size of each age group with income segmentation.

4.2 Workclass

This column is a highly imbalanced one, which is visible in Figure 23. *Private* group containing nearly three-quarters of the observations dominated the other 7. The less popular are *Withoutpay* and *Never-worked* that appeared respectively 17 and 9 times. Having these kinds of classes in a dataset focused on income is a little bit surprising, so these observations are worth paying attention to in the subsequent section of the report.

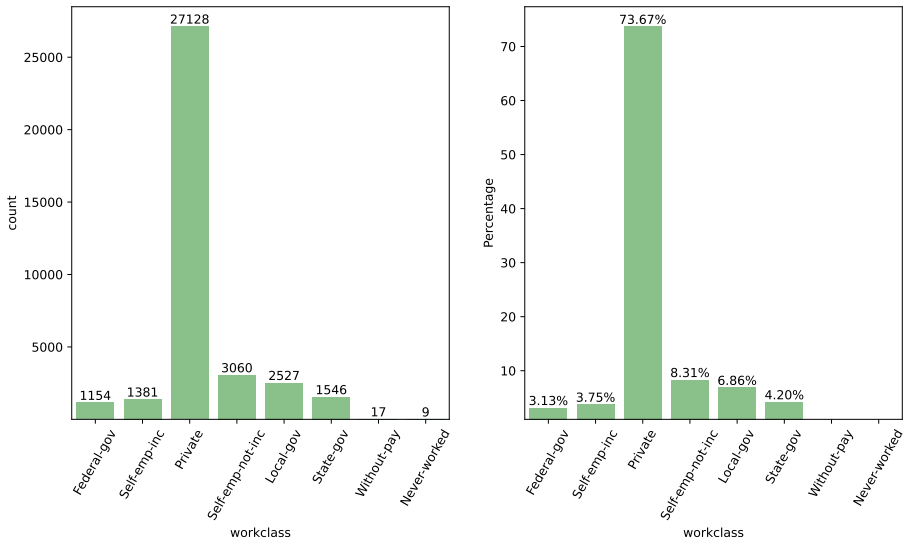


Figure 23: *Workclass* group distribution and percentage.

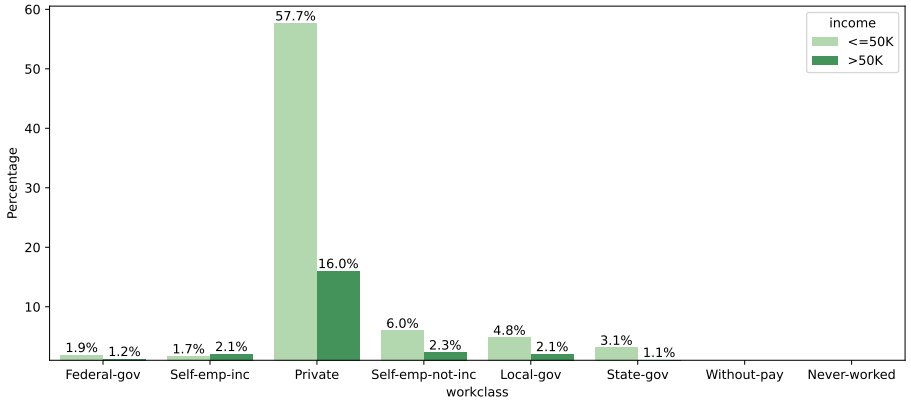


Figure 24: *Workclass* percentage distribution with income segmentation.

4.3 Education-num

The *education-num* column consists of repeatedly occurring 16 values ranging from 1 to 16. The majority of individuals described in the dataset have completed 9 years of education. This is an equivalent of six years in elementary school and three in junior high school that were parts of educational system in the United States in the 20th century (this value is compared to the USA because it is the majority of the dataset).

Table 6: *Education-num* statistics.

Mean	10.14
Std	2.56
Minimum	1
Maximum	16
Median	10

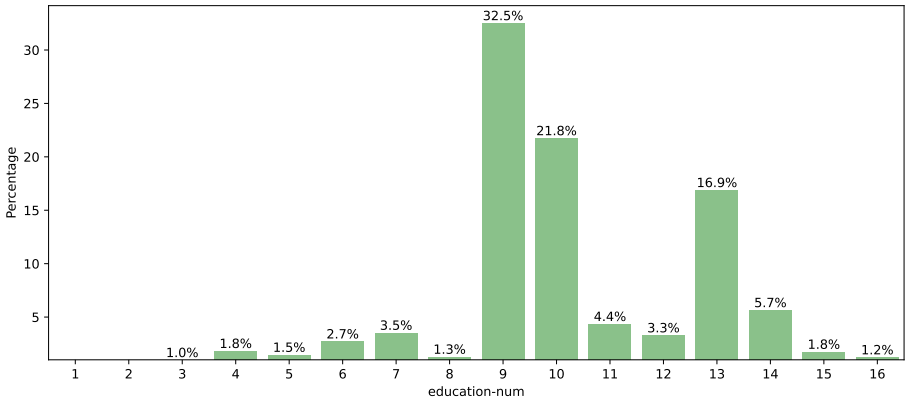


Figure 25: *Education-num* percentage distribution

Apart from the 9, for the studied group 10 and 13 were common too. Additionally, a small percentage (about 3.4%) had less than 5 years of learning. The mean of time spent at school equals 10.14 while standard deviation is 2.56 and interquartile range equals 4.

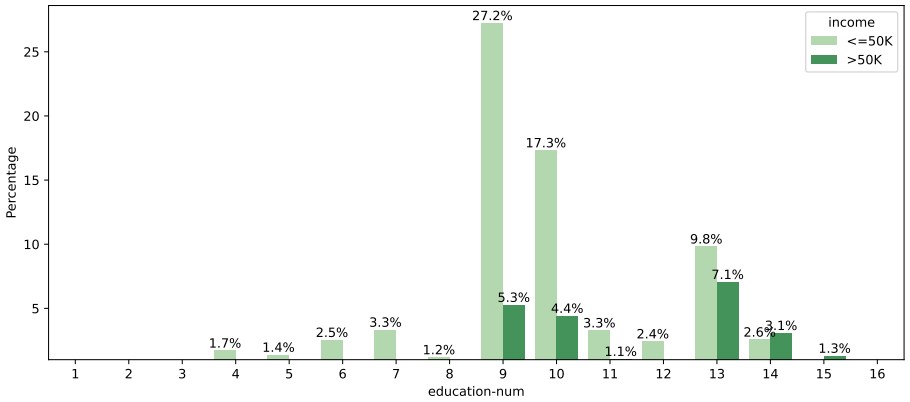


Figure 26: *Education-num* percentage distribution with income segmentation

Values in Figure 26 show that people who were learning for less than 9 years are not rich. The ratio of those earning above $50K$ to the rest is highest for 13.

4.4 Marital-status

In the dataset, there are 7 different marital status categories. Looking at Figure 27 it is visible that *Married-civ-spouse* group covers nearly half of the observations in the dataset. The second largest group consists of those who have never been married, and the least numerous (just approximately 0.1%) is *Married-AF-spouse*. A small percentage of widowed people results from the fact that the significant majority of the surveyed individuals is under 50 years old, among whom the death of a spouse occurs much less frequently.
The low number of widowed individuals results from the significant majority in the dataset being individuals under 50 years old, among whom the death of a spouse statistically occurs much less frequently than for individuals over 70, who constitute only a few percent."

Married-civ-spouse concerns individuals who are married and live together as a civilian couple in the same house.
Married-AF-spouse category refers to these marriages where one of the spouses serves in the Armed Forces.
Married-spouse-absent refers to individuals who are legally married but whose spouses are not currently present in the household. This could be due to various reasons such as work-related travel, or other circumstances that require the spouses to live separately for some time.

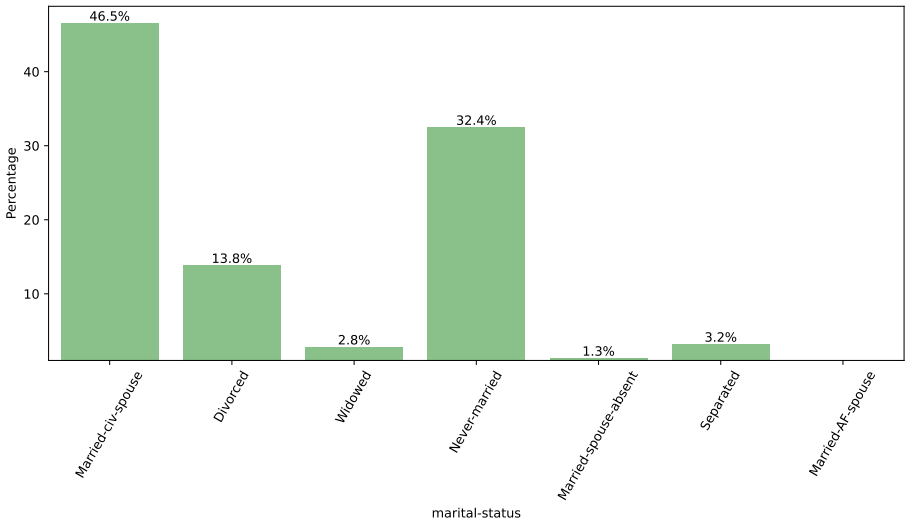


Figure 27: *Marital-status* percentage distribution.

Interestingly, those who are never married rarely earn above $50K$ annually, whereas for *Married-civ-spouse* the bars for both income variants are of similar heights.

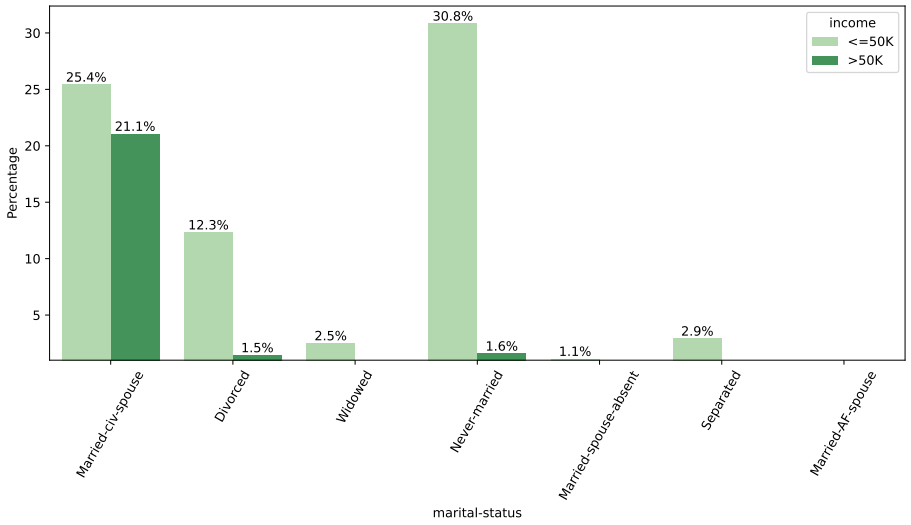


Figure 28: *Marital-status* percentage distribution with income segmentation.

4.5 Occupation

This feature covers 14 different occupations. One can say that there are no one dominant category because the most popular ones: *exec-managerial*, *prof-specialty* and *craft-repair* have almost the same percentage share in the whole dataset. The occupations in which the minority people work are *Armed-Forces* and *Priv-house-serv*.

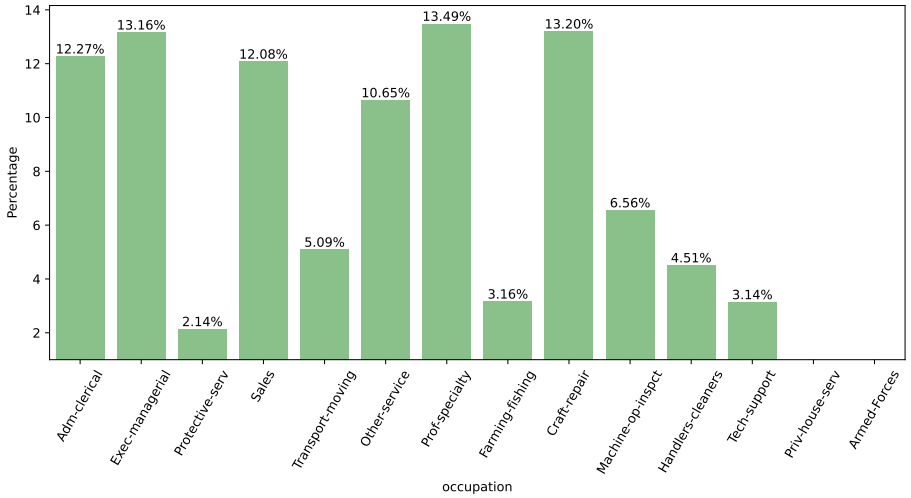


Figure 29: *Occupation* percentage distribution.

Moreover high income occurs for executive managers nearly as often as low. The proportion is also really optimistic for professional specialists.

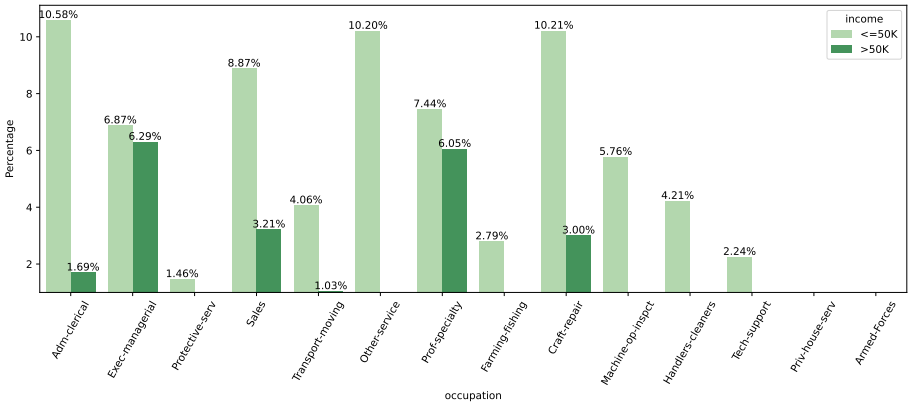


Figure 30: *Occupation* percentage distribution with income segmentation.

4.6 Relationship

Figure 31 presents the percentage share of every relationship class in the dataset. The most common is being a husband while the second place is occupied by *not-in-family*. The largest number of husbands corresponds to the fact that in this dataset men overwhelmingly outnumber women.

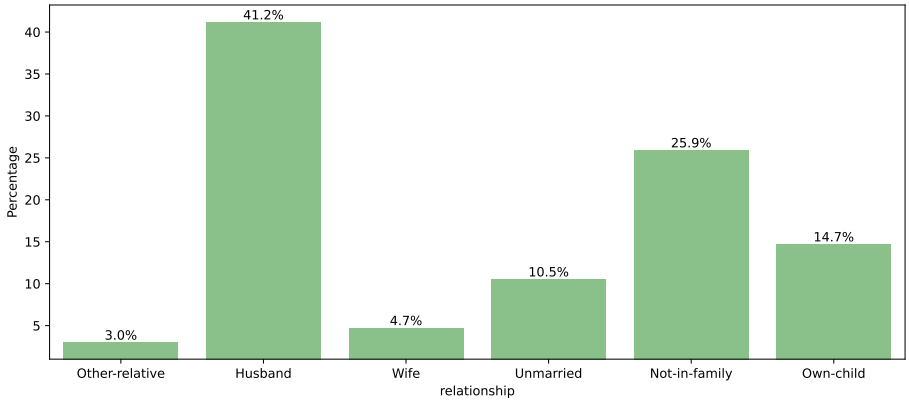


Figure 31: *Relationship* percentage distribution.

When it comes to income the most balanced relationship type is *wife*. A high percentage of rich people is observed in *husband* category too (Figure 32).

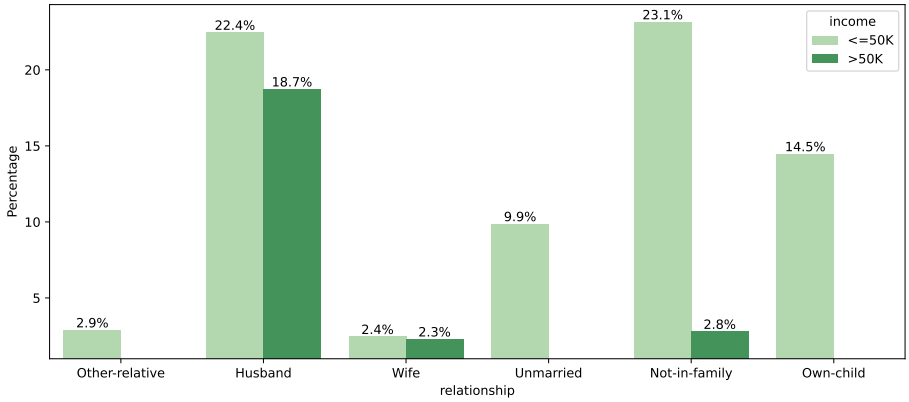


Figure 32: *Relationship* percentage distribution with income segmentation.

4.7 Race

The *race*, alongside the *workclass*, is another example of a highly imbalanced feature. The overwhelming majority of surveyed people is white. Nearly 10% of the whole dataset is covered by black ones and less than 5% is covered by the representatives of other races.

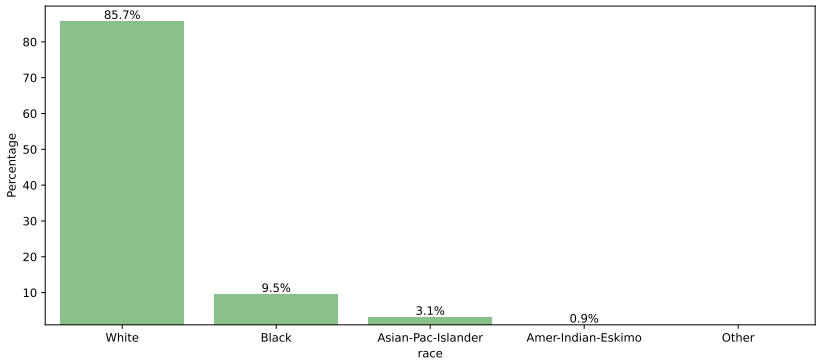


Figure 33: *Race* percentage distribution

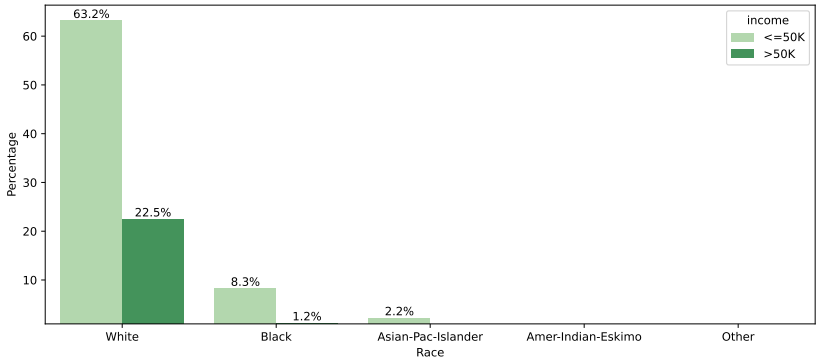


Figure 34: *Race* percentage distribution with income segmentation

Figure 34 shows that the ratio of those earning over $50K$ to those earning below is comparably high for the white ones and Asian-Pacific Islanders and much lower for the individuals of the black race.

4.8 Sex

In the Adults Income Dataset men is twice the number of women and they constitute the majority in the group of individuals earning over 50 thousand. But what is noteworthy is that the percentage of women earning well is significantly lower than the percentage of rich men. It is not a surprising result, as even in these days, women earn less on average than men.

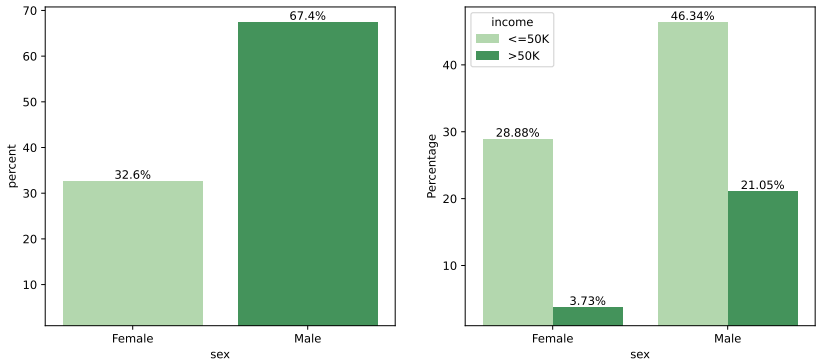


Figure 35: *Sex* percentage distribution and percentage distribution with income segmentation

4.9 Capital

Another highly imbalanced feature is *capital*. For nearly 87% observations it equals 0. So having other value of *capital* is quite rare.

Table 7: *Capital* statistics.

Mean	1041.47
Std	7673.81
Minimum	-4356
Maximum	99999
Median	0

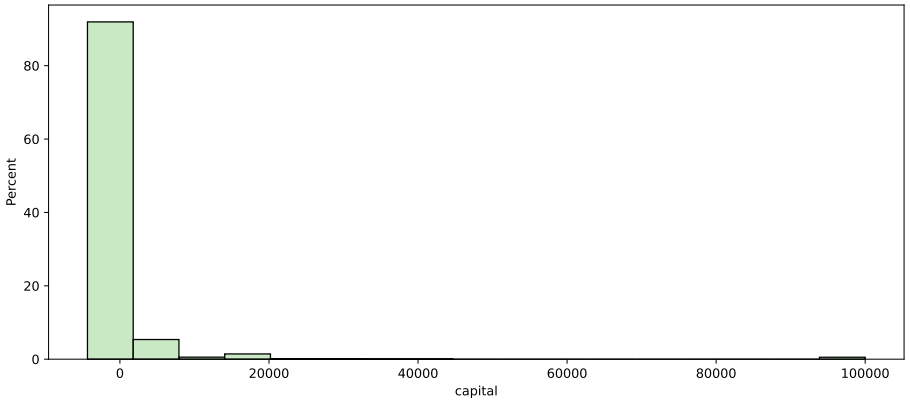


Figure 36: *Capital* percentage distribution.

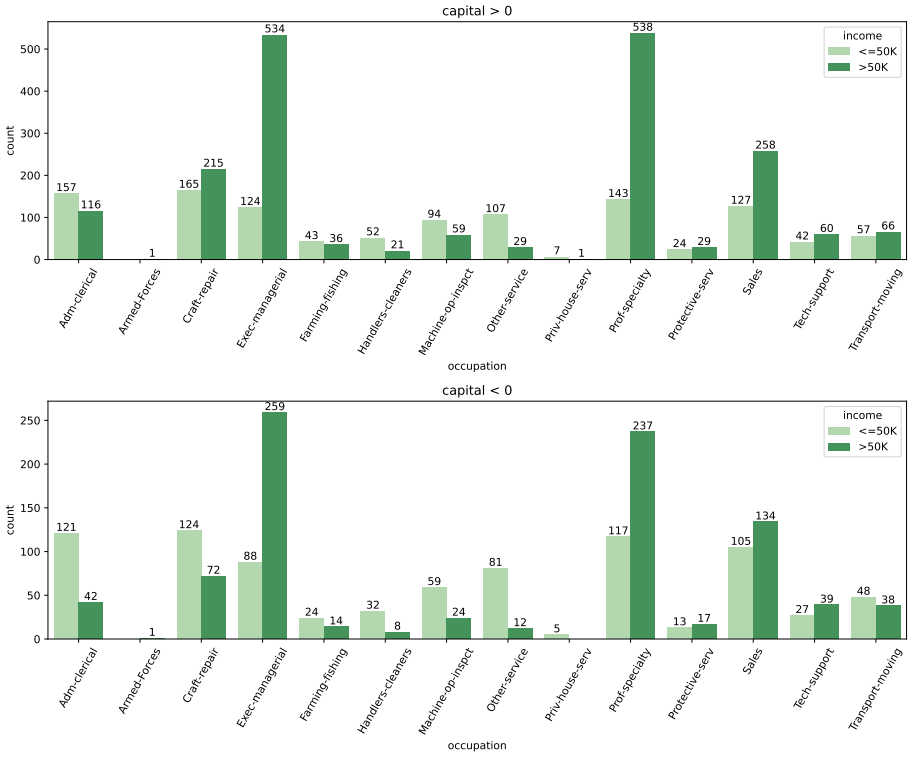


Figure 37: *Capital* percentage distribution for observations where *capital* is not 0.

For most of the observations where *capital* is not 0 the income exceeds $50K$. The majority of people who earn so much are executive managers and professional specialists while the smallest groups are people working in the Armed Forces and private house services.

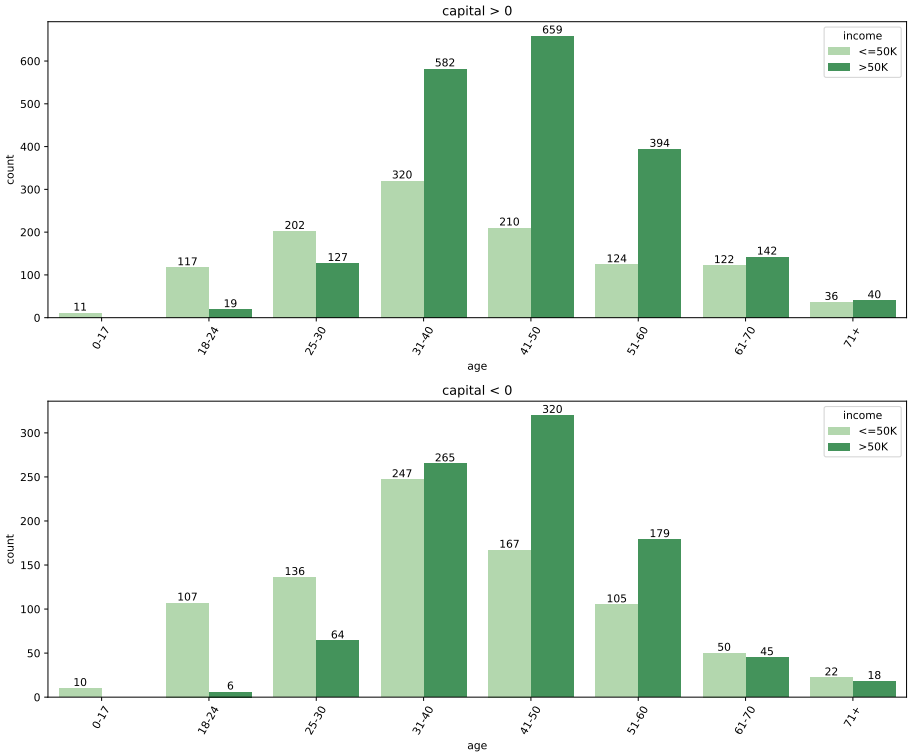


Figure 38: *Age* percentage distribution for observations where *capital* is not 0.

Generally for both *occupation* and *age* proportions between classes are kept regardless of the negative or positive value of the *capital*. The analysis of an extreme capital value 99999 was conducted in section 2.9.2.

4.10 Hours-per-week

hours-per-week column includes 95 distinct values varying from 1 to 99 from which the most popular is 40 (nearly 50%). The mean for the whole dataset equals 40.95 and the standard deviation is 12.03. More specific analyses concerning extreme values of this variable were made in a section 2.9.1.

Table 8: *Hours-per-week* statistics.

Mean	40.95
Std	12.03
Minimum	1
Maximum	99
Median	40

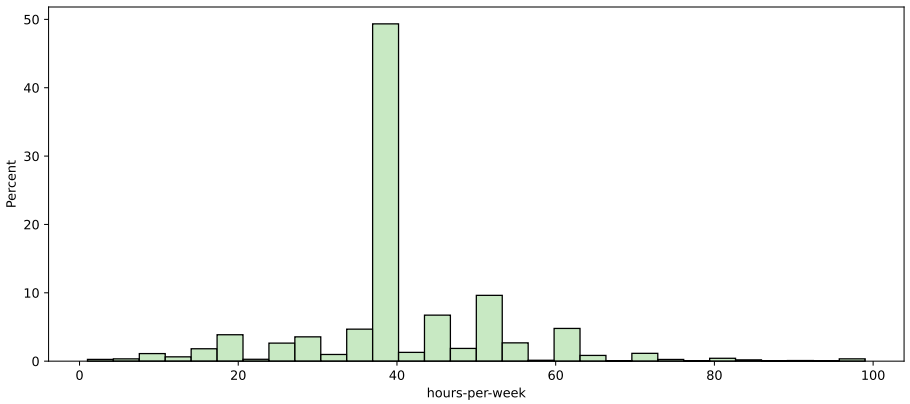


Figure 39: *Hours-per-week* percentage distribution.

Moreover, the average time spent at work every week was calculated for all work classes and occupations.

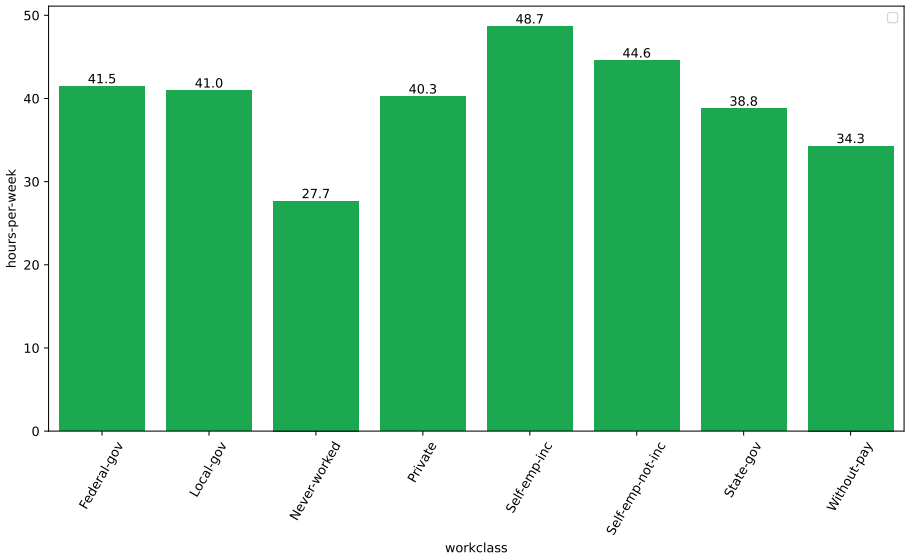


Figure 40: Mean *hours-per-week* for every work class.

As it was mentionted in the subsection 4.2 *Never-worked* and *Without-pay* classes demands more analyses. Generally finding this kind of category in the dataset concerning income is a little bit surprising but what is really suspicious is the fact that those who are described as never working, worked on average nearly 28 hours per week. Moreover, people who work without payment do it for 34 every week. These phenomena can be explained by diving into data for

these observations.

Never-worked status is assigned only to young individuals (under the age of 25) from the United States. They probably worked casually somewhere, but without signing a contract, which is why they were assigned to this class.

The *Without-pay* class mainly consists of people over 60 years old and under the age of 25. For people at these stages of life, it is easier to do something voluntarily because often they do not have to go to work and take care of children.

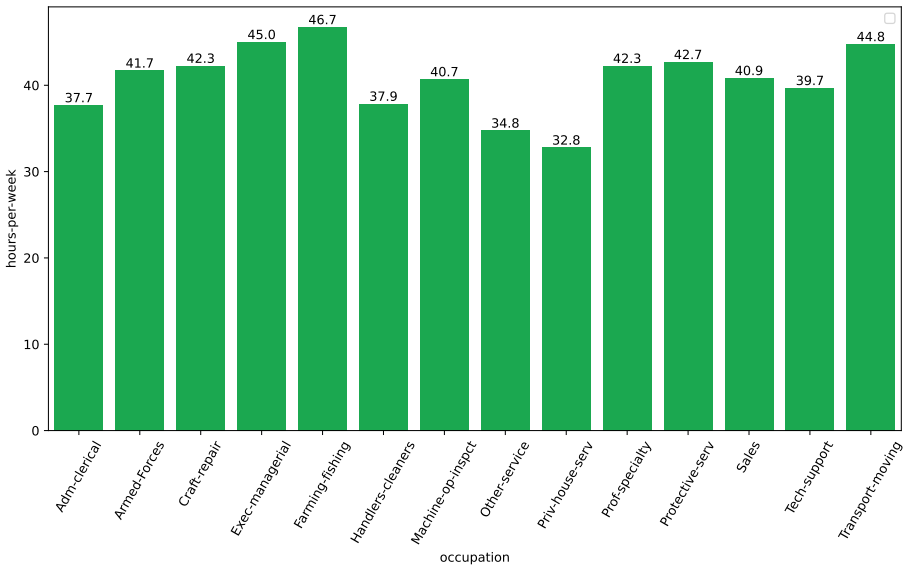


Figure 41: Mean *hours-per-week* for every occupation.

Then some interesting conclusions can be drawn from Figure 41. The average time spent at work oscillates around 40 hour per week. Farmers and fishermen work the most, while those employed in private housing services work the least. Taking results from 2.9.1 into consideration one can say that even though there are some extreme values of *hours-per-week* feature, people of every profession on average work full-time.

4.11 Region

The last imbalanced feature is *region* which includes 13 unique values. In Figure 42 the share of the United States is so high that for most classes, no values can be noticed.

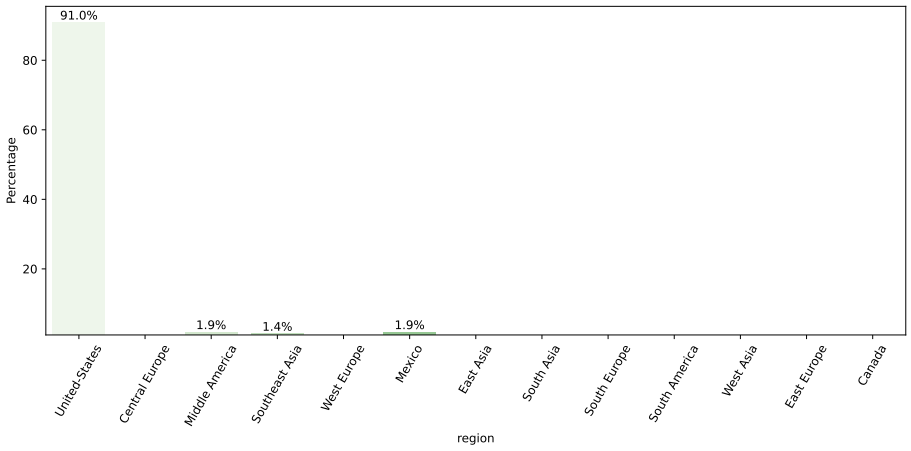


Figure 42: *Region* percentage distribution.

Figure 43 presents the share of individual regions after excluding the USA. This way it is easy to observe that among them the most common are Middle America and Mexico. The smallest group represents South Asia and there are no representatives from Africa and Australia.

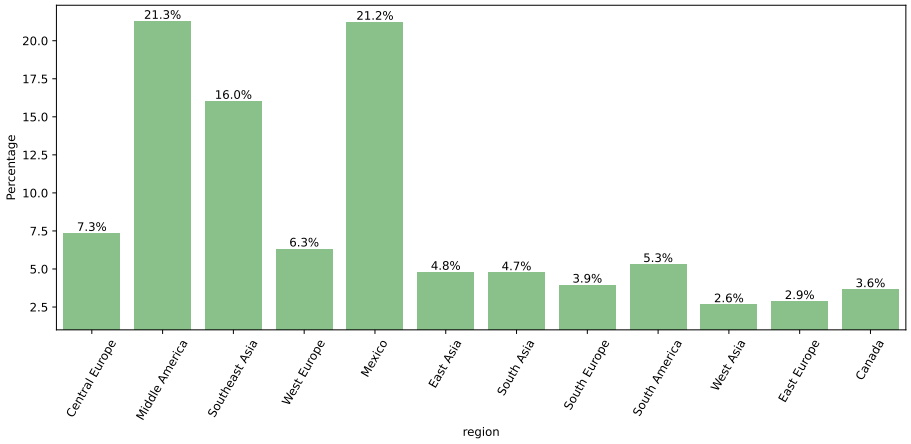


Figure 43: *Region* percentage distribution without the USA.

Looking at Figure 44 one can observe that a quarter of the USA citizens have a capital higher than $50K$. In other regions of the world, the percentage of those earning well and in some places like for example East Europe there are no such observations (Figure 45.

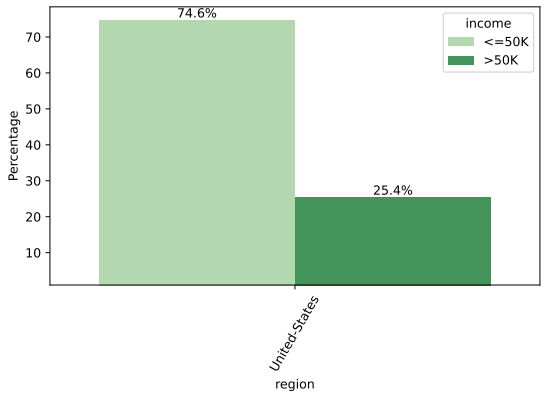


Figure 44: *Income* percentage distribution in the USA.

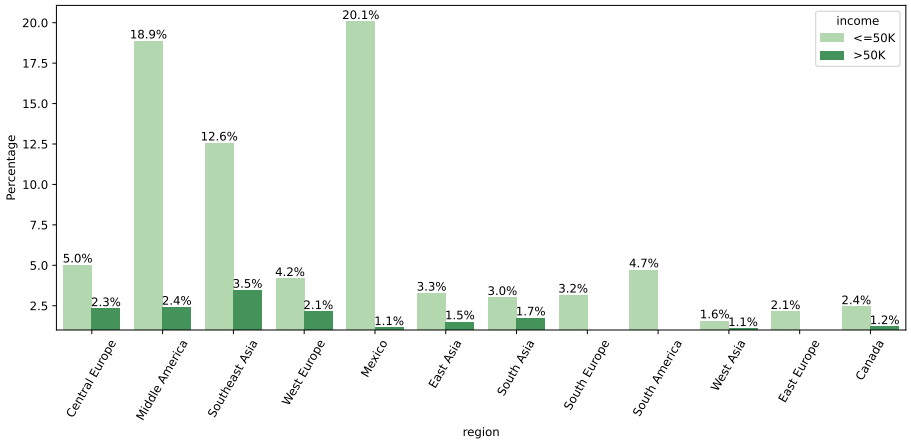


Figure 45: *Region* percentage distribution without the USA with income segmentation.

4.12 Income

This is a target variable whose values will be predicted in the next section of the report. As it was mentioned before *income* is imbalanced and class $' <= 50K'$ covers three-quarters of all observations.

4.13 Correlations between variables

Correlations between numerical variables are presented in the correlation matrix in Figure 46. Income was transformed in such a way that 1 corresponds to $> 50K$ and 0 corresponds to $<= 50K$.

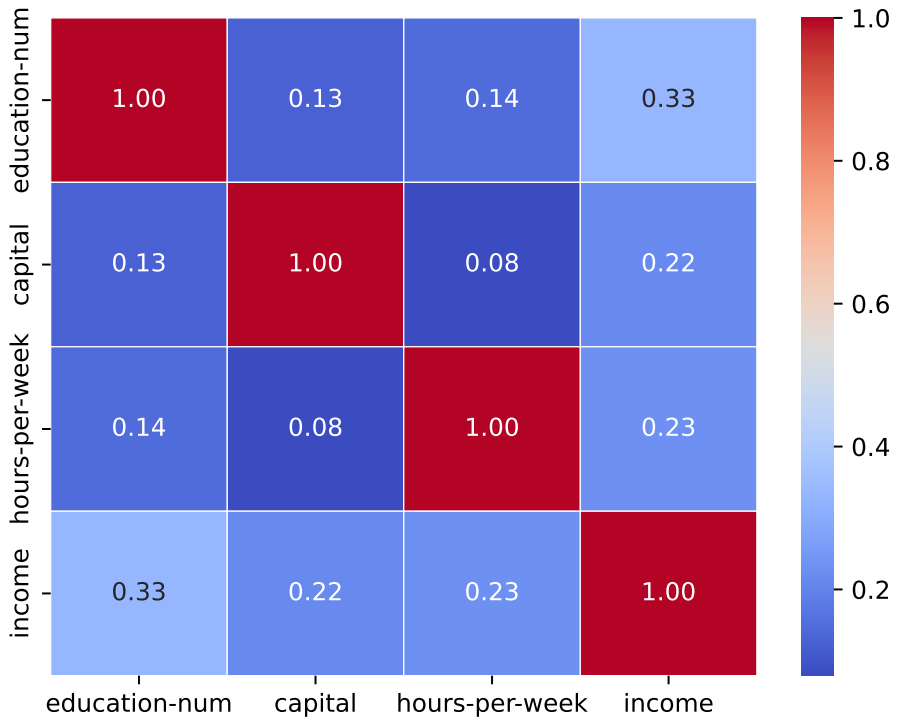


Figure 46: Correlation matrix of numerical variables.

There is only slight linear correlation between *income* and *education-num*. Values for other variables show that they are not correlated.

5 Classification

All of the above sections provided necessary information about the Census Dataset. It was used to explore, alter, correct, impute and even delete some records. Now, having cleaned data, it is possible to make a much more reliable prediction if a person makes more than 50K or not.

5.1 Preprocessing data

Classification requires strictly obeying some rules before feeding any data to a chosen model. Firstly, it is of vital importance not to mix training and testing data. That is why these were kept in separate files through the whole research and now they are read directly from them. Of course, these are the imputed datasets as some classification algorithms do not accept missing values.

Afterwards, the *pd.get_dummies* is applied to training subset. This function is used for converting categorical variables into dummy/indicator variables, which are numerical representations suitable for machine learning algorithms.

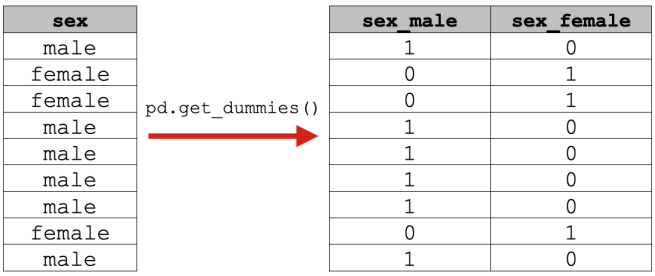


Figure 47: Get dummies scheme.

Similar conversion to binary values also has to be done on test data although now all the information has to be stored in one single column. Since there are only two classes in target variable it can be easily done by denoting "≤50K" as 0 and ">50K" as 1. The whole code for preprocessing the data before classification was presented below.

```
def my_read_data():
    # Reads already cleaned and imputed data from .csv

    file_path_train = Path('adults_data','adults_imputed_train_data.csv')
    file_path_test = Path('adults_data','adults_imputed_test_data.csv')

    df_train = pd.read_csv(file_path_train)
    df_test = pd.read_csv(file_path_test)

    X_train = df_train.iloc[:,:-1]
    X_test = df_test.iloc[:,:-1]
    X_train = pd.get_dummies(X_train, dtype=float)
    X_test = pd.get_dummies(X_test, dtype=float)

    Y_train = df_train.iloc[:,-1]
    Y_test = df_test.iloc[:,-1]

    Y_train = Y_train.replace("<=50K", 0)
    Y_train = Y_train.replace(">50K", 1)
    Y_test = Y_test.replace("<=50K", 0)
    Y_test = Y_test.replace(">50K", 1)

    return X_train, X_test, Y_train, Y_test
```

Figure 48: Function used to read and preprocess data before classification.

5.2 Evaluation metrics

A confusion matrix is a fundamental tool used in machine learning and statistics to evaluate the performance of a classification model. It provides a summary of the predictions made by a model versus the actual known outcomes or labels in the dataset. It is typically represented as a table, where the rows correspond to the actual classes or labels and the columns correspond to the predicted classes by the model.

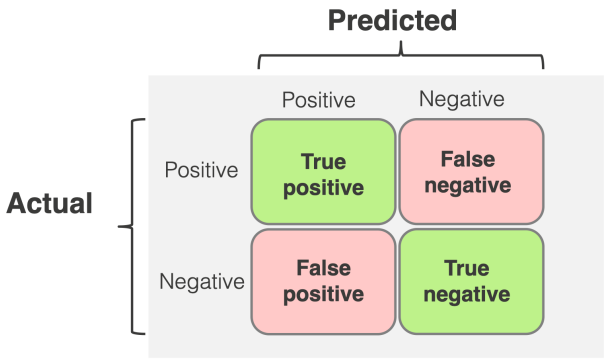


Figure 49: Confusion matrix scheme.

Provided such representation of confusion matrix it is possible to compute metrics that allow researchers to evaluate effectiveness of performed predictions. The most common ones were presented below.

Accuracy shows how often a classification ML model is correct overall.

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

Precision shows how often an ML model is correct when it classifies object to the minority class.

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

Recall shows whether an ML model can find all objects of the target class. It is particularly useful when target class is of great importance, e.g. fatal disease.

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

The F1 score is the harmonic mean of the precision and recall. It thus symmetrically represents both precision and recall in one metric.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

5.3 Models

Different types of datasets and problem scenarios can benefit from various modeling techniques. That is why it is recommendable to try multiple classification approaches. Usually they vary between assumptions, constraints, interpretability, computation time and of course performance. In practice, the choice of model often involves experimentation and evaluation of different techniques to determine which one performs best for a particular dataset and problem context.

5.3.1 XGBoost

XGBoost is an abbreviation for Extreme Gradient Boosting. The name comes from the gradient of the function marked as ∇f. It is a vector which consists of all the partial derivatives of this function. Geometrically, it is a vector indicating its growth direction at a given point. That is why $-\nabla f$ is often used in various algorithms to find minimums. XGBoost is one of the most popular and most efficient implementations of Gradient Boosted Trees algorithms for both regression and classification problems. In general, it relies on optimizing a prediction function by minimizing a chosen loss function and using regularization.

5.3.2 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a straightforward yet effective supervised learning algorithm used for classification tasks. t relies on the principle of similarity - the assumption that similar things exist in close proximity. New data points are classified based on the majority class among their K nearest neighbours in the feature space. The closeness or distance between data points is typically calculated using metrics like Euclidean distance, Manhattan distance, or other distance measures, depending on the nature of the data.

5.3.3 Logistic regression

Logistic Regression is a fundamental statistical method used for binary classification tasks. The term "logistic" stems from the logistic function used in this model, which maps any real-valued number into a value between 0 and 1. This mapping is crucial for interpreting the output as a probability that a given instance belongs to a particular class. This estimation is achieved by fitting the coefficients of the logistic function to the training data. The objective is to find the best-fitting line (or hyperplane in higher dimensions) that separates the classes in feature space.

Logistic regression is widely used due to its simplicity, interpretability, and efficiency. It optimizes the parameters by minimizing a chosen loss function, typically the logistic loss (or cross-entropy loss), using optimization techniques like gradient descent.

5.3.4 Neural network

Neural networks are a class of machine learning models inspired by the structure and functioning of the human brain. They consist of interconnected nodes organized into layers, where each node performs a simple computation. Neural networks are capable of learning complex patterns and relationships in data, making them versatile for various tasks including classification, regression, and pattern recognition.

Training a neural network involves optimizing its parameters (weights and biases) to minimize a chosen loss function, typically using backpropagation. Backpropagation calculates the gradients of the loss function with respect to each parameter.

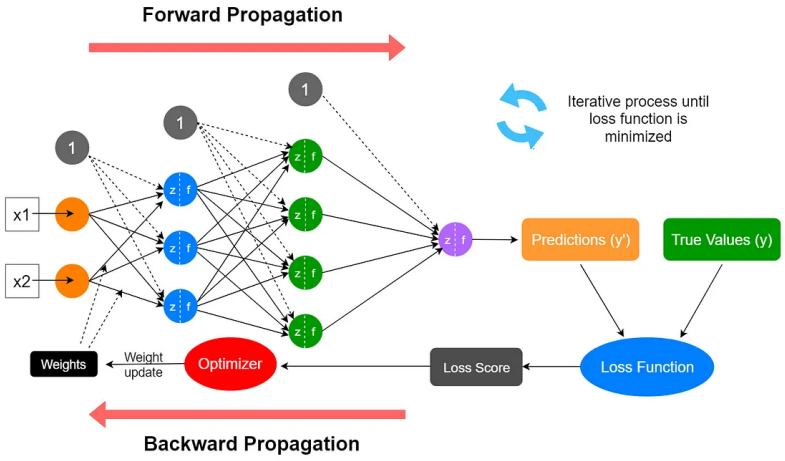


Figure 50: Neural network scheme.

5.4 Optimization

In order to get the best possible results, optimization techniques were conducted for some of the models. Provided research includes balancing the dataset and tuning the hyperparameters.

5.4.1 Balancing the dataset

Accordingly to section 2.1, there is a severe imbalance in the dataset. To deal with this problem two solutions were proposed and compared: random oversampling and SMOTE. Of course, these methods were applied only to training data.

```python
def balance_data(X_train, Y_train, type="ros"):

    if type == "ros":
        ros = RandomOverSampler(random_state=5)
        X_train, Y_train = ros.fit_resample(X_train, Y_train)
    elif type == "smote":
        smote = SMOTE(random_state=5, k_neighbors=5)
        X_train, Y_train = smote.fit_resample(X_train, Y_train)
    else:
        print("Balancing type not found")

    return X_train, Y_train
```

Figure 51: Function used for balancing the training data.

5.4.2 Hyperparameter tuning

In order to get the best possible results an optimization of parameters was conducted for some of the models. The chosen approach was to perform a grid search using *GridSearchCV* from *sklearn.model_selection* library. The grid of the parameters had to be provided manually. F1 Score was chosen as the main metric during optimization.

```python
model_xgb = XGBClassifier()

param_grid = {'gamma': [0,0.1,0.2,1.6],
      'learning_rate': [0.01, 0.03, 0.1, 0.2],
      'max_depth': [7,12,15],
      'n_estimators': [35,50,80,115],
      'reg_alpha': [0,0.1,0.2,1.5],
      'reg_lambda': [0,0.05,0.1,0.3]}

# Define & fit grid search
grid_search = GridSearchCV(model_xgb, param_grid, cv=5, scoring='f1')
grid_search.fit(X_train, Y_train)

print("Best Hyperparameters:", grid_search.best_params_)
print("Best F1 score:", grid_search.best_score_)

# Create the best model
model_xgb = XGBClassifier(gamma=grid_search.best_params_['gamma'],
                          learning_rate=grid_search.best_params_['learning_rate'],
                          n_estimators=grid_search.best_params_['n_estimators'],
                          reg_alpha=grid_search.best_params_['reg_alpha'],
                          reg_lambda=grid_search.best_params_['reg_lambda'])
```

Figure 52: Performing grid search to find the best parameters among those manually provided in the grid.

5.5 Classification results

5.5.1 Random oversampling results

Table 9: XGBoost random oversampling (ROS) results.

	Target	Accuracy	Precision	Recall	F1 Score
Default XGBoost	≤50K	0.88	0.90	0.95	0.92
	>50K		0.81	0.69	0.74
Default XGBoost + ROS	≤50K	0.85	0.96	0.84	0.89
	>50K		0.65	0.88	0.75
Default XGBoost + SMOTE	≤50K	0.86	0.93	0.90	0.91
	>50K		0.71	0.79	0.75

Source: Own elaboration.

Table 10: KNN random oversampling (ROS) results.

	Target	Accuracy	Precision	Recall	F1 Score
Default KNN	≤50K	0.88	0.91	0.94	0.92
	>50K		0.79	0.71	0.75
Default KNN + ROS	≤50K	0.88	0.94	0.90	0.92
	>50K		0.73	0.83	0.77
Default KNN + SMOTE	≤50K	0.85	0.94	0.86	0.90
	>50K		0.67	0.84	0.75

Source: Own elaboration.

Table 11: Logistic regression random oversampling (ROS) results.

	Target	Accuracy	Precision	Recall	F1 Score
Default logistic regression	≤50K	0.84	0.88	0.93	0.90
	>50K		0.73	0.60	0.66
Default logistic regression + ROS	≤50K	0.80	0.94	0.79	0.86
	>50K		0.57	0.85	0.68
Default logistic regression + SMOTE	≤50K	0.81	0.94	0.80	0.86
	>50K		0.58	0.84	0.68

Source: Own elaboration.

Table 12: Neural Network random oversampling results.

	Target	Accuracy	Precision	Recall	F1 Score
Default neural network	≤50K	0.85	0.88	0.94	0.91
	>50K		0.77	0.63	0.69
Default neural network + ROS	≤50K	0.85	0.93	0.87	0.90
	>50K		0.67	0.82	0.73
Default neural network + SMOTE	≤50K	0.85	0.93	0.86	0.90
	>50K		0.66	0.81	0.73

Source: Own elaboration.

In all of the above cases random oversampling and SMOTE proved to be good ways of handling the class imbalance problems. Every model increased its recall and f1 score when it comes to minority class.

5.5.2 Hyperparameter tuning results

Grid search approach for XGBoost showed that the best tested parameters are: {'gamma': 0.08, 'learning_rate': 0.6, 'max_depth': 70, 'n_estimators': 1000, 'reg_alpha': 0.5, 'reg_lambda': 0.5}. What is worth mentioning is that maximum depth and number of estimators had the biggest impact on the results. As expected, selecting high values for these parameters led to improving the scores significantly at the cost of computation time. Further increase of the mentioned parameters did not provide worthwhile results.

Table 13: XGBoost classification results.

	Target	Accuracy	Precision	Recall	F1 Score
Default XGBoost + ROS	≤50K	0.85	0.96	0.84	0.89
	>50K		0.65	0.88	0.75
Optimized XGBoost + ROS	≤50K	0.89	0.97	0.88	0.92
	>50K		0.72	0.92	0.81

Source: Own elaboration.

Grid search approach for KNN showed that the best parameters are: {'algorithm': 'auto', 'n_neighbors': 11, 'p': 1, 'weights': 'distance'}. Parameter 'p' equal to 1 stands for using manhattan distance instead of euclidean distance.

Table 14: KNN classification results.

	Target	Accuracy	Precision	Recall	F1 Score
Default KNN + ROS	≤50K	0.88	0.94	0.90	0.92
	>50K		0.73	0.83	0.77
Optimized KNN + ROS	≤50K	0.92	0.96	0.93	0.94
	>50K		0.80	0.88	0.84

Source: Own elaboration.

In both cases hyperparameter tuning was indeed beneficial. It increased values across all of the calculated metrics.

5.5.3 Comparison of the models

Table 15: Comparison of the models.

	Target	Accuracy	Precision	Recall	F1 Score
Optimized XGBoost + ROS	≤50K	0.89	0.97	0.88	0.92
	>50K		0.72	0.92	0.81
Optimized KNN + ROS	≤50K	0.92	0.96	0.93	0.94
	>50K		0.80	0.88	0.84
Default logistic regression + ROS	≤50K	0.80	0.94	0.79	0.86
	>50K		0.57	0.85	0.68
Default neural network + ROS	≤50K	0.85	0.93	0.87	0.90
	>50K		0.67	0.82	0.73

Source: Own elaboration.

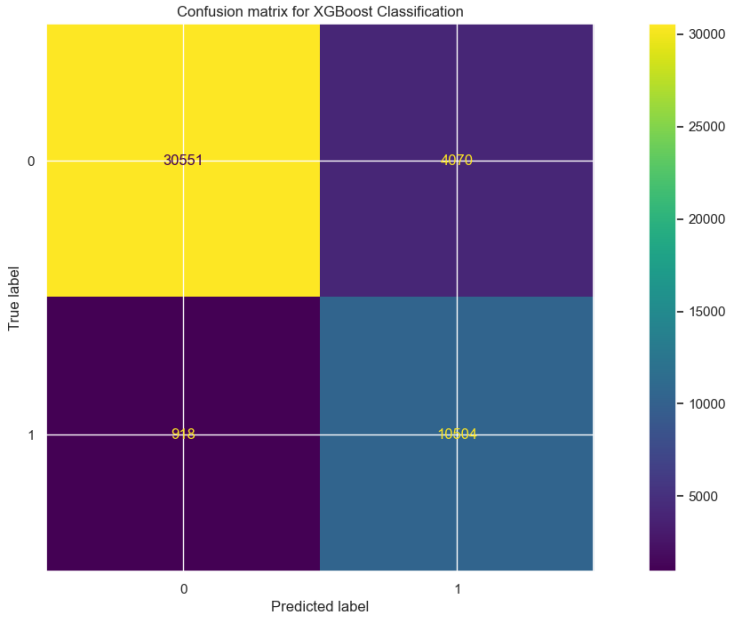


Figure 53: Confusion matrix for Optimized XGBoost + ROS.

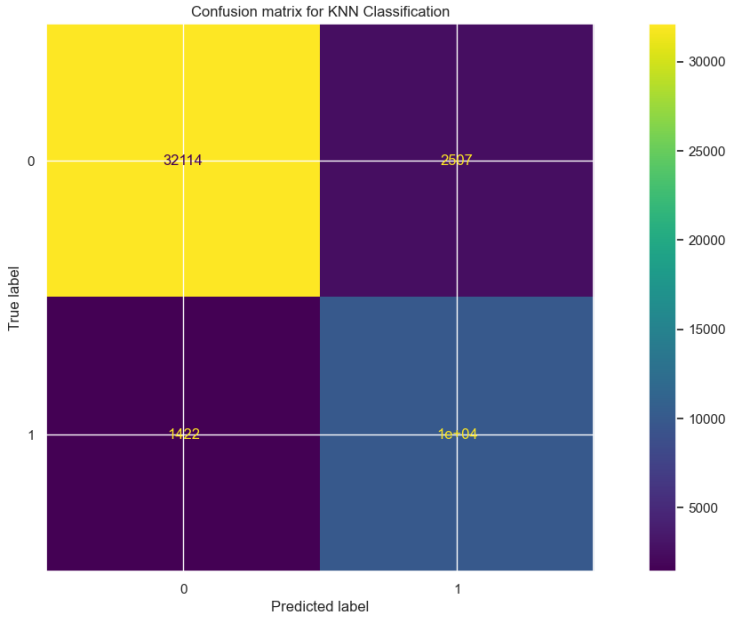


Figure 54: Confusion matrix for Optimized KNN + ROS.

The best results were obtained clearly by optimized models. XGBoost and KNN have by far the highest metrics. Both of them were adjusted to deal well with the minority class and indeed they managed to get a high recall for this group. Between those two, KNN seems to be the better model due to higher precision and f1 score. Its proximity is also noticeable on confusion matrixes above.

Results for neural networks are a bit disappointing. This family of models have huge potential which probably was not fully exploited in this report.

6 Conclusion

This report completely covers the assumptions of the first part of the Data Mining project. Some useless variables were identified and removed, while other one were grouped or relabelled. Additionally, some grouping was done and missing values were handled. Moreover, a lot of attention was paid to suspicious values and justification of their occurrence. Every variable was analyzed individually and linear correlation between them was also checked.

In the second part of this report binary classification problem was approached. Some neccessary basic preprocessing was introduced and applied to the data. Afterwards, there was a brief theoretical description of evaluation metrics and characteristics of used models. Two optimization techniques were also addressed. Research has shown that both balancing the dataset using random oversampling or SMOTE and hyperparameter tuning using grid search turned out to be beneficial for the results. In the last section the best results for each model were gathered and compared to each other. The highest results were obtained clearly by optimized XGBoost and KNN. The best model overall was decided to be KNN + ROS with parameters {'algorithm': 'auto', 'n_neighbors': 11, 'p': 1, 'weights': 'distance'}.