Supervised Learning On Income Around The World

Beatriz Mesquita
*DETI**
University of Aveiro
Machine Learning Foundations
Prof. Petia Georgieva
*DETI: Department of Electronics, Telecommunications and Informatics

***Abstract*—This project concerns a supervised learning problem. Using the UCI Adult dataset [1], a database with 14 variables, we investigate how deep and machine learning approaches can be used to estimate if a person's salary is above or below $50,000. This dataset includes demographic and employment-related data for a sample of individuals from the US Census of 1994. To forecast a person's income, we train a number of deep learning models, including a Multi-layer Perceptron Classifier (MLP) [17], where we got an accuracy of 79.4%, and a Long short-term memory (LSTM) [9], where we got an accuracy of 80.7%. We also train a machine learning model for comparison, Random Forest (RF) [12], where we got an accuracy of 82.2%. Before applying these models we study appropriate data preprocessing techniques for this problem. Also, when actually applying the models, we tune them, so we can reach an ideal set of hyperparameters for each one. To finish, we compare them using appropriate evaluation metrics for this classification problem: confusion matrix [14], accuracy, precision, support, recall, and F1 score [13].**

***Index Terms*—salary, supervised learning, machine learning, deep learning, MLP, LSTM, RF, confusion matrix, accuracy, precision, support, recall, F1 score**

I. Introduction

An important subject that has ramifications for many domains and applications, including policy-making, marketing, and finance, is estimating a person's income based on demographic and employment-related data. Knowing what influences a person's income can be useful in spotting trends of inequality and informing policy choices that attempt to close gaps and encourage economic growth.

Investigating this issue is crucial since income plays a significant role in establishing a person's level of life, social standing, and access to resources. We can discover variables that contribute to income disparity and develop ways to solve these concerns by evaluating the relationship between income and demographic and employment-related data.

For instance, if we discover that women are routinely paid less than males with equivalent credentials and experience, we may use this knowledge to push legislation and policies that support gender equality in the workplace, such as parental leave regulations and equal pay laws. Similar to the previous example, if we discover that members of particular racial or ethnic groups are more likely to earn less money than those from other groups, we may use this data to support policies that combat systemic discrimination and encourage diversity and inclusion in the workforce.

Also, it might be helpful for marketing and financial applications to estimate a person's income based on demographic and employment-related data. For instance, businesses might utilize this data to target particular consumer groups with goods and services tailored to their wants and needs. These details can be used by financial organizations to determine a borrower's creditworthiness and inform their lending decisions.

As a result, it is critical to look into the issue of calculating a person's income based on demographic and employment-related data because doing so can reveal patterns of inequality, guide decision-making about public policy, and enhance marketing and financial applications. We may endeavor to create a more just and prosperous society by understanding the variables that contribute to economic discrepancies. [5]

The UCI "Adult" dataset [1] is a widely used dataset for machine learning and data mining research. It is a well-known benchmark dataset that has been applied in a great deal of research and writing. This dataset's popularity and the availability of several previous research and findings to compare this work with are some factors that simplified the choice for this study. [27] [25] [21] [34] [26]

Another factor is the dataset's abundance of instances and attributes, which make it ideal for developing and testing machine learning models. The dataset's inclusion of categorical and continuous variables also makes it possible to experiment with various data preparation and feature engineering methods.

It is important to note that while the data in the UCI "Adult" dataset is from 1994, the underlying relationships between the attributes and the target variable (income class) may still be relevant today.

In order to estimate the income class of various individuals, we set out to create three distinct models. Multilayer Perceptron (MLP) [17] and Long Short-Term Memory (LSTM) [9] were our two deep learning models. To compare the effectiveness of deep learning models with other machine learning models, we also employed the Random Forest (RF) [12] model.

We started by concentrating on feature selection and data pre-processing. The data must be cleaned up and prepared for future examination in this key step. The next step was data visualization. With the aid of this phase, we were better able to comprehend the statistical distribution of our data and spot any patterns or trends. To improve our understanding of the

data, we used a variety of visualization approaches, including histograms, box plots, and density plots.

Lastly, we put the supervised learning strategies we previously discussed into practice.

Data pre-processing, visualization, and supervised learning should always be utilized in tandem, it is vital to keep this in mind. This is due to data pre-processing, which makes sure the data is in the right format for analysis, visualization, which makes it easier to comprehend the data and see trends, and supervised learning, which uses the data to make predictions or categorize it. The analysis would not be as efficient and accurate without one of these phases. [**?**]

II. State of art review

The UCI Adult Dataset is a popular benchmark for machine learning and deep learning models. This way, existing works in the literature have investigated various aspects of income prediction. Ron Kohavi [27] proposed a new algorithm called NBTree which induces a hybrid of decision-tree classifiers and Naive-Bayes classifiers. He showed that it outperforms both methods individually on the Adult Data Set. The decision-tree nodes contain univariate splits as regular decision trees, but the leaves contain Naive-Bayesian classifiers. This approach retains the interpretability of Naive-Bayes and decision trees while resulting in classifiers that frequently outperform both constituents, especially in larger databases tested.

Kao et al. [25] proposed a decomposition method that reduces the memory and computational requirements of linear SVMs. They proposed a decomposition method with alpha seeding that is extremely useful for solving a sequence of linear support vector machines (SVMs) with more data than attributes. They found that a coordinate descent method with shrinking is the most efficient on the Adult Data Set.

Caruana and Niculescu-Mizil [21] conducted an empirical evaluation of supervised learning for ROC area, and compared neural networks, SVMs, boosted stumps, boosted trees, bagging, random forests, and naive Bayes on the Adult Data Set.

Zadrozny [34] studied the effects of sample selection bias on classifier performance, and used naive Bayes, logistic regression, C4.5, and SVMs on the Adult Data Set.

Lin and Lin [26] presented a study on reduced SVMs, which use fewer support vectors than standard SVMs, and showed that they achieve comparable accuracy and faster prediction speed on the Adult Data Set.

To sum up, these studies suggest that different algorithms perform differently on the dataset, and there is no single algorithm that performs best in all scenarios. The NBTree algorithm proposed by Kohavi [27], appears to be a promising approach, particularly for larger databases. Kao et al.'s [25] decomposition method for linear SVMs reduces the memory and computational requirements and can be particularly useful when dealing with large amounts of data. Caruana and Niculescu-Mizil's [21] evaluation of several supervised learning algorithms suggests that there is no single best algorithm for the Adult Data Set, and the choice of the algorithm may depend on the specific task and the desired performance metric. Zadrozny's [34] study on sample selection bias highlights the importance of carefully considering the data used to train and test the classifiers. Finally, Lin and Lin's [26] study on reduced SVMs shows that reducing the number of support vectors can lead to faster prediction speed without sacrificing accuracy. Overall, the studies demonstrate the importance of careful algorithm selection and data preprocessing when working with the Adult Data Set.

III. Data description and Pre-processing

Our dataset was initially donated by Ronny Kohavi and Barry Becker from the UCI Machine Learning Repository [1]. It consists of demographic information about individuals, such as age, education, marital status, occupation, race, sex, and other features. It also includes information about their income level, which is the target variable. The income level is classified as either "¿50K" or "¡=50K", which indicates whether the individual earns more or less than $50,000 per year, respectively.

The Adult Data Set contains 32561 rows/samples and 15 columns/features, including the target variable. It is often used for classification tasks, where the goal is to predict an individual's income level based on their demographic information. It is a popular dataset for evaluating and comparing the performance of different classification algorithms.

A. Data description

1) **age:** describes the age of individuals. *Continuous*;
2) **workclass:** describes the type of work an individual does. *Categorical*. Possible values are:
   - Private: the person works for a private company or business;
   - Self-emp-not-inc: the person is self-employed but not incorporated;
   - Self-emp-inc: the person is self-employed and incorporated;
   - Federal-gov: the person works for the federal government;
   - Local-gov: the person works for a local government (e.g., city or county);
   - State-gov: the person works for a state government;
   - Without-pay: the person is not paid for their work (e.g., a volunteer);
   - Never-worked: the person has never worked or has not worked for a long time.
3) **fnlwgt:** represents a statistical weight that is used to adjust for the complex sampling design used to collect the data. It's important to understand that the fnlwgt variable is not a direct measure of the number of people in the population, and should not be interpreted as such. Also, the exact formula used to calculate the fnlwgt is not publicly available, as it is considered confidential by the US Census Bureau to protect the privacy of individuals in the dataset. *Continuous*;

4) **education:** describes the highest level of education an individual has achieved. *Categorical*. Possible values are:
   - Bachelors: a bachelor's degree;
   - Some-college: some college but no degree;
   - 11th: 11th grade;
   - HS-grad: a high school diploma or equivalent;
   - Prof-school: a professional degree (e.g., MD, JD, etc.);
   - Assoc-acdm: an associate's degree in a technical or academic field;
   - Assoc-voc: an associate's degree in a trade or vocational field;
   - 9th: 9th grade;
   - 7th-8th: 7th or 8th grade;
   - 12th: 12th grade;
   - Masters: a master's degree;
   - 1st-4th: 1st, 2nd, 3rd, or 4th grade;
   - 10th: 10th grade;
   - Doctorate: a doctorate degree (e.g., PhD, EdD, etc.);
   - 5th-6th: 5th or 6th grade;
   - Preschool: a preschool education.
5) **education-num:** describes the years spent in education in numerical form. *Continuous*;
6) **marital-status:** describes the marital status of an individual. *Categorical*. Possible values are:
   - Married-civ-spouse: the person is married and a civilian spouse is present;
   - Divorced: the person is divorced;
   - Never-married: the person has never been married;
   - Separated: the person is separated;
   - Widowed: the person is widowed;
   - Married-spouse-absent: the person is married but their spouse is absent;
   - Married-AF-spouse: the person is married and their spouse is in the armed forces.
7) **occupation:** describes the occupation of an individual. *Categorical*. Possible values are:
   - Tech-support: the person provides technical support;
   - Craft-repair: the person repairs things;
   - Other-service: the person provides other services;
   - Sales: the person sells things;
   - Exec-managerial: the person manages things;
   - Prof-specialty: the person provides professional services;
   - Handlers-cleaners: the person cleans things;
   - Machine-op-inspct: the person inspects things;
   - Adm-clerical: the person provides administrative services;
   - Farming-fishing: the person farms or fishes;
   - Transport-moving: the person moves things;
   - Priv-house-serv: the person provides private household services;
   - Protective-serv: the person provides protective services;
   - Armed-Forces: the person is in the armed forces.
8) **relationship:** describes the relationship between an individual and their spouse. *Categorical*. Possible values are:
   - Wife: the person is a wife;
   - Own-child: the person is a child;
   - Husband: the person is a husband;
   - Not-in-family: the person is not in the family;
   - Other-relative: the person is some other type of relative;
   - Unmarried: the person is unmarried.

   (We can see that, although there is some overlap between the "relationship" and "marital-status" columns, they are not exactly the same thing. So, while there is some **correlation** between the two columns, they are not exactly the same thing and may provide different information for predictive modeling purposes).
9) **race:** refers to the ethnicity of the individual. *Categorical*. The possible values are:
   - White: person classified as White;
   - Black: person classified as Black or African American;
   - Asian-Pac-Islander: person classified as Asian or Pacific Islander;
   - Amer-Indian-Eskimo: person classified as American Indian or Eskimo;
   - Other: person classified as Other.

   (I might add that it is important to note that the use of racial categories can be controversial and may not accurately reflect the complexity of an individual's racial or ethnic identity. Additionally, the meaning and significance of race can vary widely across different cultures and contexts).
10) **sex:** represents the gender of an individual. *Categorical*. It can take two values:
   - Male: represents male gender;
   - Female: represents female gender.
11) **capital-gain:** describes the capital gains for an individual. *Continuous*;
12) **capital-loss:** describes the capital losses for an individual. *Continuous*;
13) **hours-per-week:** describes the average number of hours worked per week. *Continuous*;
14) **native-country:** describes the native country of an individual. *Categorical*. Possible values are: United-States, Cuba, Jamaica, India, Mexico, Puerto-Rico, Honduras, England, Canada, Germany, Iran, Philippines, Poland, Columbia, Cambodia, Thailand, Ecuador, Laos, Taiwan, Haiti, Portugal, Dominican-Republic, El-Salvador, France, Guatemala, Italy, China, South, Japan, Yugoslavia, Peru, Outlying-US(Guam-USVI-etc), Scotland, Trinadad&Tobago, Greece, Nicaragua, Vietnam, Hong, Ireland, Hungary, and Holand-Netherlands;
15) **income:** Describes whether an individual makes more or less than $50,000 per year. *Categorical*. It can take

two values:

- $\leq 50K$: the person makes less than or equal to $50,000 per year;
- $> 50K$: the person makes more than $50,000 per year. **This is the target variable**.

B. Pre-processing

Data pre-processing is a crucial step in the data analysis process. It is the process of cleaning, transforming, and organizing the data before it is fed into a deep or machine-learning model. The goal of data pre-processing is to make sure that the data is in a format that is suitable for analysis and modeling. This step is necessary to ensure that the data is accurate, consistent, and free of errors.

We divide this pre-processing into two steps, the pre-processing before visualization and the pre-processing after visualization and before the deep and machine learning models. So, in the first step, we do some feature selection. Features that don't seem interesting to us for income forecasting won't be interesting to visualize either. Then we take care of the null values and check for repeated rows. This is all we do before visualization so we can have a good view of how our features behave and how our dataset is distributed.

After visualization, we end up checking for multi-collinearity for numerical and categorical features, performing data encoding of the categorical features, and normalizing our data.

1) Pre-processing Before Visualization: As said before, our first step is to select information. Since our goal is, within these listed individuals, to predict if their income is less or greater than 50K, we will select the information that we think is important and least important for this prediction (**feature selection**).

Analyzing the meaning of each column, listed above, we decided that the "fnlwgt" column has no influence on the income prediction. As explained above, it works as a weighting factor in statistical analyses to account for the sampling design. Therefore, we will drop this column before training our models.

Then, looking at the "education" column, we can see that it is highly correlated with the "education-num" column, which contains the same information but in numerical form. Including both columns in our models could lead to multi-collinearity and overfitting, so we will drop one of them. In a first approach, we tried dropping the "education-num" column and leaving the categorical one. Then, we encoded the "education" column using one-hot encoding (which will be explained later). However, this feature has 16 possible values as we have seen before. This means that, after encoding, we would have 16 new columns, which would increase the dimensionality of our dataset. This, combined with the other categorical features that we have to encode later, could lead to overfitting and, in addition, would make our models harder to interpret. Therefore, in a second approach, we decided to drop the "education" column and keep the "education-num" column. This way, we will have a single column with the years of education, which will be easier to interpret and will not increase the dimensionality of our dataset.

Finally, we drop the "capital-gain" and "capital-loss" columns, as they are highly skewed: most of the values are 0. In fact, more than 91% of the values in these columns are 0. As a result, the distribution of values in these columns is heavily skewed towards 0, and this could negatively affect the performance of some machine learning algorithms. In addition, since most of the values are 0, these columns are unlikely to have a significant impact on predicting an individual's income level.

After this, we check for null values. After checking the information in our dataset we saw that it didn't detect any null values. We first saw that our dataset, after our data selection, had shape (32561, 11), that is, it has 32561 rows and 11 columns. Later, when checking for the dataset information, the number of entries on the Non-Null Count column is also 32561 entries, so we can conclude that the dataset contains absolutely no null values. However, a closer look tells us that there are a lot of '?' values in our dataset. We will have to replace those values by null values and then deal with them in the most appropriate way. We could also see taht "age", "hours-per-week", and "education-num" are integer columns. There are no float datatypes in the dataset. And, finally, "workclass", "marital-status", "occupation", "relationship", "race", "sex", "native-country" and "income" are of object datatypes.

After replacing the '?' with null values we decided to drop them. It felt like the best approach. Substituting the null values by the mean or median of the column would have a significant impact on the distribution of the data. After dropping the null values, our dataset has shape (30162, 11), that is, it has 30162 rows and 11 columns. Comparing this number with the original number of rows (32561), we can see that we dropped 2399 rows. This is a significant number, but it was necessary since they contained null values and wouldn't be useful for our analysis.

Regarding the repeated rows, in this case, we won't drop them. It is very common to have repeated rows in datasets like this one, that contains demographic information. If I have two individuals with very similar characteristics, it doesn't necessarily mean that one of them is a duplicate of the other. In fact, it is very likely that they are two different individuals. Therefore, we will not drop the repeated rows.

Checking for a statistical summary of the remaining dataset we found, for our numerical variables ("age", "hours-per-week" and "education-num") that:

- The minimum and maximum age of people in the dataset is 17 and 90 years respectively, while the average age is 37;
- The number of hours spent per week varies between 1 to 99 and the average being 40 hours;
- The minimum and maximum number of years of education is 1 and 16 years respectively, while the average number of years of education is 10.

For object data (e.g. strings), the result's index will include count, unique, top, and freq. The top is the most common

value. The freq is the most common value's frequency. This way, we were able to see that our target variable, "income", is extremely unbalanced. Around 22654 of the samples are of the class $\leq 50K$. This is something we have to deal with after the visualization step.

2) Pre-processing after visualization and before applying any DL or ML model: As explained before, after visualization, after having a better understanding of the dataset, further pre-processing steps were done. We still need to check for multi-collinearity, encode the categorical features and normalize our data.

Multi-collinearity is a phenomenon that occurs when two or more independent variables are highly correlated with each other. In other words, multi-collinearity exists when the independent variables are interdependent or redundant, making it difficult to determine the individual effect of each variable on the dependent variable. [23]

In the context of predicting income class, multi-collinearity can make it challenging to identify the most significant independent variables that affect the outcome. It can also lead to unstable and inaccurate parameter estimates in the classification model, making it difficult to interpret the results.

To avoid multi-collinearity, it is essential to identify and remove the highly correlated independent variables or combine them to form a new variable that captures the same information. Other techniques that can help reduce multi-collinearity include regularization methods like L1 and L2, which we will talk more about later on. [24]

Now, to check the existence of ant multi-collinearity issues in the numerical features, we plot the correlation matrix:

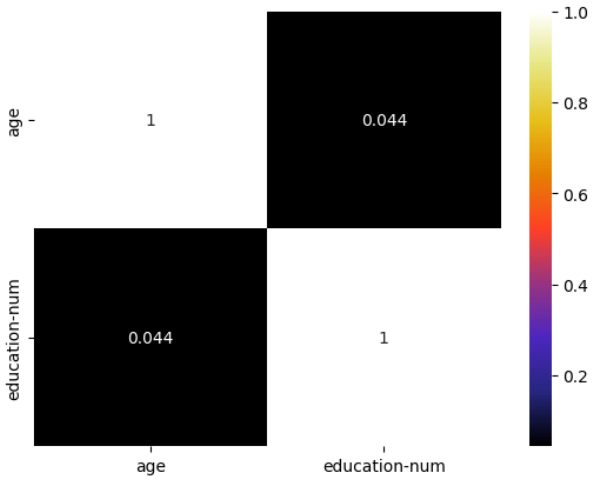


Fig. 1. Correlation Matrix for numerical features.

From Fig.1 we got a correlation matrix between two variables, 'age' and 'education-num'. The diagonal represents the correlation of each variable with itself, which is always 1. The other cells in the matrix represent the correlation between the two variables.

The correlation coefficient between 'age' and 'education-num' is 0.044, which is a very weak positive correlation. This means that there is little to no linear relationship between these two variables. Therefore, they are unlikely to have multicollinearity issues when used in a predictive model. [30]

Regarding the categorical features, one common way to check for multi-collinearity between categorical features is by using a chi-squared test of independence (also chi-square or 2 test).

The chi-squared test of independence measures the association between two categorical variables. It tells us whether there is a significant relationship between the two variables.

To perform a chi-squared test of independence between two categorical variables, we create a contingency table that cross-tabulates the two variables and then calculates the chi-squared statistic and the corresponding p-value. If the p-value is below a predetermined threshold (e.g. 0.05), we would reject the null hypothesis that the two variables are independent and conclude that there is evidence of a significant relationship between the two variables. [32]

The results showed that for all the categorical variables we tested, the p-value is less than 0.05, which means that we can reject the null hypothesis that the variables are independent at the 0.05 level of significance. This suggests that there is a statistically significant association between each of these categorical variables and the income variable. In other words, the value of these categorical variables is not independent of the value of the income variable and may be useful for predicting income. Also, when we test for all possible pairs of categorical variables, the p-value is always less than 0.05, which means that we can reject the null hypothesis that the variables are independent at the 0.05 level of significance. This suggests that there is a statistically significant association between each pair of categorical variables. In other words, the value of one categorical variable is not independent of the value of another categorical variable and there may be some multi-collinearity between the independent variables.

Multi-collinearity can cause problems when building a predictive model because it can make it difficult to determine the individual effect of each independent variable on the dependent variable. One way to deal with multi-collinearity is to remove one of the correlated independent variables from the model. However, this may result in a loss of information and a reduction in the model's predictive power. This way, we will bet on applying, later, some regularization methods like L1 and L2 (which we will talk better about later on).

After this, we need to find the best way to encode our categorical features. Regarding the "workclass", "marital-status", "occupation", "relationship", "race" and "native-country" features, we have more than two classes. We also have a new categorical feature called "hours" that will be created during data visualization based on the "hours-per-week" variable, which also has multiple classes. We have to figure out how to encode these features.

It would not make sense to use "*.LabelEncoder()*". It is a class in the scikit-learn library in Python that is used to convert categorical data or data that can be divided into categories, into numerical data. It assigns a unique integer value to each

category. [31] However, we would end up assigning codes to these features' classes randomly, and we would not be able to understand and identify what each code corresponds to. Since we don't want to do this "blindly", we will encode this differently.

We will use the One-Hot Encoding method. This method is used to convert categorical data into a form that could be provided to deep and machine learning algorithms to do a better job in prediction. One hot encoding creates a new column for each unique value in the categorical feature. The new column will have a value of 1 if the row has that value and 0 otherwise. [28]

Before applying one-hot encoding to the "native-country" feature we actually grouped these countries into regions. There were about 41 different countries. If we applied the encoding to this feature as it was we would end up summing 41 features to our dataset. This is a lot of features and, of course, it could lead to overfitting. This way, we created 5 regions to group these countries into North America, Latin America, Asia, Europe, and Other. This also makes sense from an economic-social point of view, since each of these regions has different levels of development, which can lead to quite different sets of values for the features we have.

Then for the "sex" and "income" features, encoding them is pretty straightforward, since thy have only 2 possible values/classes. This way, for the "sex" feature, we assign 1 to Female and 0 to Male. As for the "income" feature, we assign 1 to income $> 50K$ and 0 otherwise.

We already stated that our dataset is unbalanced, as our target variable has a lot more values for "¡=50K" (22654) than for "¿50K" (7508). This can cause problems for our models, so we will use the SMOTE method to balance it. SMOTE stands for Synthetic Minority Oversampling Technique. It is an oversampling technique that creates synthetic samples from the minority class. The technique works by randomly choosing a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors. [29]

Some advantages of SMOTE include:

1) It can effectively balance class distribution by oversampling the minority class;
2) It can reduce the impact of the class imbalance problem on the model's performance;
3) It can help improve the model's ability to generalize to new minority samples.

Disadvantages of SMOTE include:

1) It can lead to overfitting if the synthetic samples are too similar to the original minority samples;
2) It can be computationally expensive if the size of the dataset is large;
3) It can lead to a loss of diversity in the minority class if the synthetic samples are not diverse enough.

[33]

We will apply this later when training our dataset. It is also important to note that, it is generally recommended to only use SMOTE on the training set, and not the test or validation set, so that the model is not trained to expect the oversampled data during evaluation.

SMOTE can be used on normalized data, as it does not change the underlying distribution of the data. So, we will normalize the dataset before applying SMOTE.

Since our features are measured at different scales we have to normalize our data. This is because scales with higher numbers end up having more importance in our models, while smaller numbers end up having less, and this will end up misleading our models. Columns that are already classified as 0 and 1 make no sense to be normalized. We will then normalize the numerical features: 'age' and 'education-num'. This is done using the max-min normalization technique, which scales the values between 0 and 1. The mathematical formulation for this is:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

where x represents a single feature/variable vector.

[6]

Next, we need to select what we want to predict (our target variable). As said before, we intend to predict the "income" column. In that case, we don't want to train the "income" column, we want that column to be our predictable one. This way, we are going to drop and save the column we want to predict: "income".

Finally, we split our data into train, validation, and test sets. The purpose of this splitting is to prevent overfitting, which occurs when a model fits the training data too well and performs poorly on new, unseen data, but also to prevent underfitting, which occurs when a model is too simple to capture the underlying patterns in the data. The basic idea behind data splitting is to separate the available data into three mutually exclusive subsets:

- Training set: This is the largest subset of data and is used to fit the model. The model is trained on this subset by adjusting the model parameters to minimize the error between the predicted outputs and the true outputs;
- Validation set: This is a smaller subset of data used to tune the model hyperparameters. Hyperparameters are settings of the model that cannot be learned during training, such as the learning rate or the regularization strength. The validation set is used to evaluate the model performance on new, unseen data and adjust the hyperparameters accordingly;
- Test set: This is used to evaluate the final performance of the model. The test set should be representative of the new, unseen data that the model will encounter in the real world. The model should not be trained or tuned on this data to avoid overfitting.

The typical split ratio for data splitting is 70/15/15, where 70% of the data is used for training, 15% for validation, and 15% for testing.

Of course, like we said before, after splitting our dataset, we apply SMOTE to the training set. This way, our target feature gets balanced:

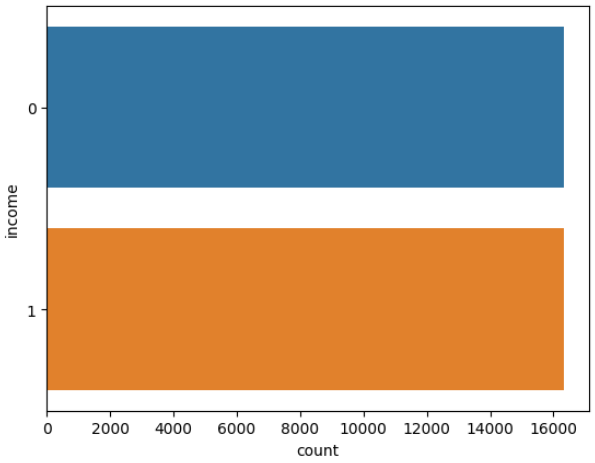


Fig. 2. Distribution of the target variable after applying SMOTE.

Now, when training our models, we will not have to worry about the unbalanced dataset and biased results. Checking for the shape of our training set, the first dimension of the shape (32688,) represents the number of samples in the training set and the second dimension (50,) represents the number of features for each sample. It's always important to have a sufficiently large training set to train a predictive model effectively.

IV. Data Visualization

In the previous section, Pre-processing, we analyzed the data provided and made a few changes in order to easily have a better approach for our solution to the problem. Now, we will provide some different types of graphics and plots for a user-friendly comprehension of the data to study as well as some statistical interpretation of our dataset. This can and will leave to some more pre-processing of the data. It's always good to work with these two together: data engineering and visualization.

A. Target Variable Distribution

We start by analyzing the distribution of the target variable, the income (before we applied SMOTE, of course).

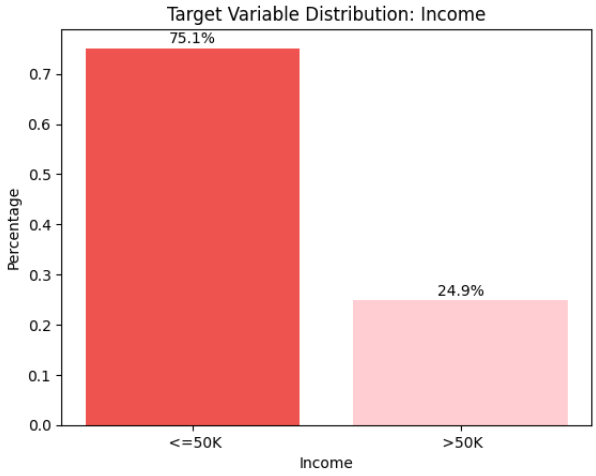


Fig. 3. Target feature distribution - Income Distribution.

Once again, we can see, from 3 how unbalanced our dataset is. We have 75.1% of the entries with income $\leq 50K$ and 24.9% with income $> 50K$. This is a piece of very important information to keep in mind when training our models and preparing our data for it.

B. Numerical Variables

Let's now have a look at the distribution of the numerical variables: "age", "education-num", and "hours-per-week".

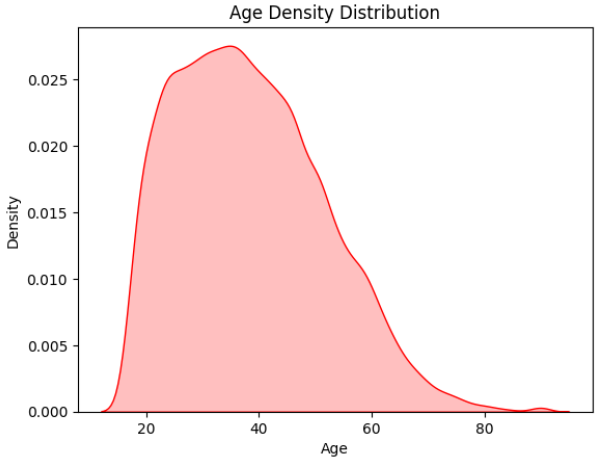


Fig. 4. Age Density Distribution.

1) Age Density Distribution: From the plot of figure 4, we can see that the distribution of age in the dataset is roughly bell-shaped with a peak around 35-40 years old, and then gradually decreases as the age increases or decreases. This may suggest that there are more middle-aged individuals in the dataset compared to younger or older individuals.

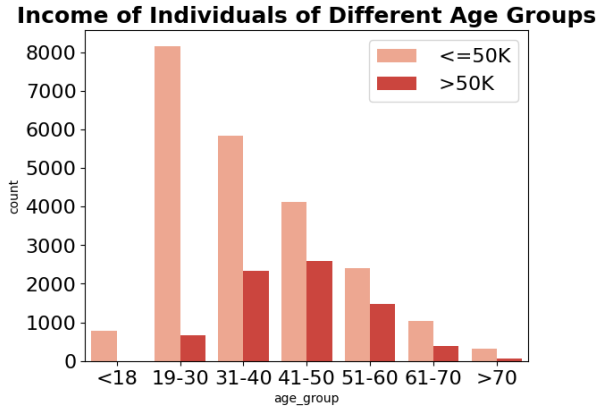


Fig. 5. Age distribution for each income level.

2) Age distribution for each income level: The graph of figure 5 shows the distribution of income across different age groups. We can see that people in the age groups of 31-50 have the highest number of individuals earning $> 50K$ income. For the people of age group 41-50 and 51-60, the number of people earning more than 50K is quite comparable to those earning less than it. The age group of age < 18 and greater than 70 have the lowest number of individuals earning $> 50K$ income. Also, we can say that the number of people earning more than 50K is quite negligible amongst people of age groups 19-30 and 61-70. Finally, we can also observe that for all age groups, there are more individuals earning $\leq 50K$ income than $> 50K$ income.

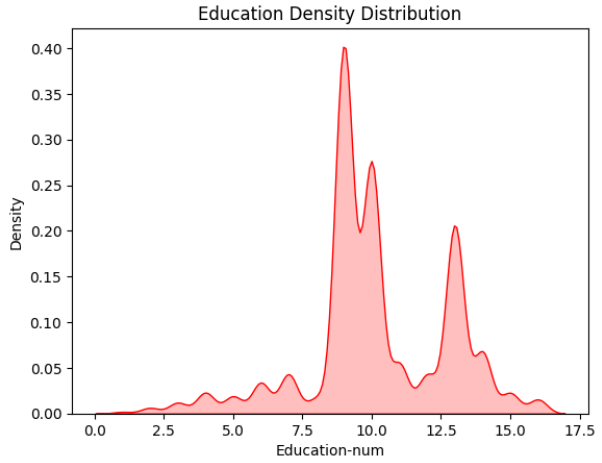


Fig. 6. Education-num Density Distribution.

3) Education-num Density Distribution: The plot of figure 6 shows that the education level is bimodal, with peaks around the values 9 and 13, indicating that there are more individuals in the dataset with education levels of 9 and 13.

Bimodal is a term used to describe a distribution with two peaks or modes. In a bimodal distribution, the frequency of values is concentrated around two different values or ranges of values, creating two distinct peaks in the histogram or density plot. A bimodal distribution is often an indication that there are two sub-populations or groups within the data that have different characteristics or behaviors. [2]

In this case, value 9 corresponds to a high school diploma or equivalent, while value 13 corresponds to a Bachelor's degree. Therefore, this plot suggests that the dataset has a large number of individuals with a high school diploma or Bachelor's degree.

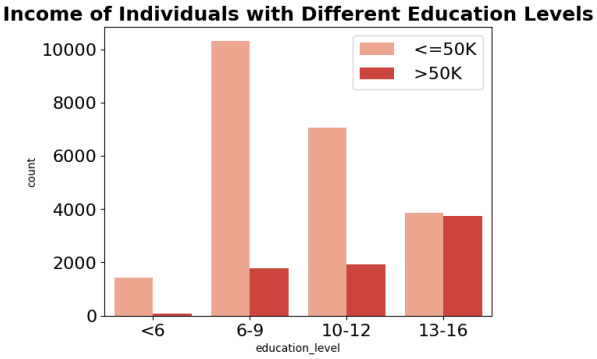


Fig. 7. Education-num distribution for each income level.

4) Education-num distribution for each income level: From the plot of figure 7, we can see that individuals with higher education levels (i.e., those in the 10-12, 13-16 education level groups) have a higher proportion of high-income earners compared to those with lower education levels. Additionally, individuals with lower education levels (i.e., those in the ¡6, 6-9 education level groups) have a higher proportion of low-income earners compared to those with higher education levels.

Overall, this plot suggests that education level is an important predictor of income, with higher education levels generally associated with higher income levels.

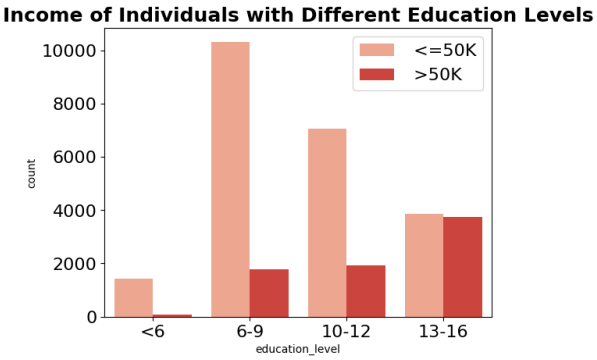


Fig. 8. Hours-per-week Density Distribution.

5) Hours-per-week Density Distribution: From the graph of figure 8, we can see that the majority of individuals work between 35 to 45 hours per week. The distribution is skewed towards the left, indicating that there are more individuals who work fewer hours per week than those who work more. There are also small bumps in the distribution around 50 and 60

hours per week, indicating that there is a subset of individuals who work longer hours.

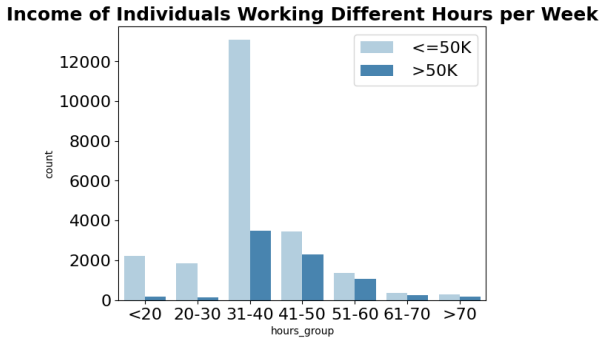


Fig. 9. Hours-per-week distribution for each income level.

6) Hours-per-week distribution for each income level: We can see, from figure 9, that the majority of individuals who work between 31-40 hours per week earn less than 50K per year. However, as the number of hours worked per week increases, the proportion of individuals earning more than 50K per year also increases. This trend is particularly noticeable for those working more than 50 hours per week, where a higher proportion of individuals earn more than 50K per year compared to those working fewer hours.

7) Hours-per-week: feature transformation: Let's have a look at the 'hours-per-week' distribution using a boxplot:

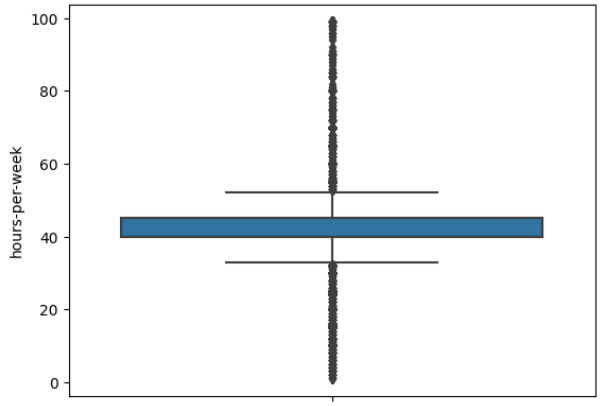


Fig. 10. Hours-per-week distribution using a boxplot.

From the plot of figure 10, we can see that the median hours-per-week value is around 40, with the middle 50% of values falling between around 35 and 50 hours per week. The plot also shows a number of outliers on the high end of the range, indicating that there are individuals in the dataset who work significantly more than the median number of hours per week. There are also a number of outliers on the low end of the range, indicating that there are individuals in the dataset who work significantly less than the median number of hours per week.

By plotting this boxplot we found that there's a better approach than using the "hours-per-week" feature directly, that is to group the values into categories.

This way, by transforming the "hours-per-week" feature into a categorical feature, we can reduce the number of final features when doing the one-hot encoding. This will be useful when we prepare the data for the models. We will check the distribution of this new categorical variable "hours" when exploring all the categorical features.

8) Checking for outliers of numerical variables: So, now, after this transformation, we only have two numerical variables: age, and education-num. To check for numerical data outliers, we will use, one more time, the boxplot. The boxplot is a very useful tool for visualizing the distribution of data, as we saw with the "hours-per-week" variable in the subsection Hours-per-week: feature transformation. It is a very simple and intuitive way to detect outliers. It presents information from a five-number summary: the minimum, first quartile, median, third quartile, and maximum. This way, it can tell us about our outliers and what their values are. It can also tell us if our data is symmetrical, how tightly our data is grouped, and if and how our data is skewed. [7]

Let's have a look at the boxplots of the numerical variables:

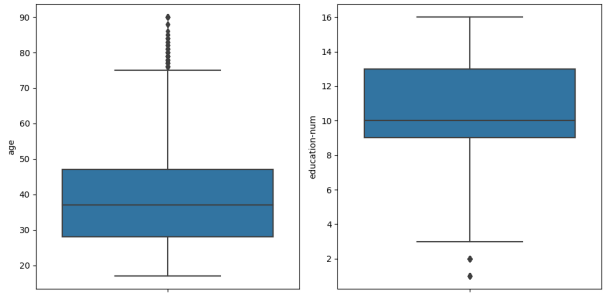


Fig. 11. Distribution of the numerical variables.

The "age" plot shows a number of outliers on the high end of the range, indicating that there are some individuals in the dataset who are significantly older than the median age.

For the "education-num" variable, the plot shows a relatively symmetrical distribution with only a small number of outliers (two) on the low end of the range. This suggests that the majority of individuals in the dataset have a relatively similar education level.

We will create a new dataset without those outliers that we identified here and, later, we will compare the results of our models with and without these outliers, to measure the effect these have on the models. This dataset with no outliers will go through the same transformations as the original dataset went through in the section Pre-processing.

One first approach we could try, for taking care of the outliers, is using the Mahalanobis distance. This is a known way of detecting outliers. The Mahalanobis distance is a measure of the distance between a point and a distribution. It is a generalization of the idea of Euclidean distance to multiple dimensions. It is used to detect outliers in multivariate data.

However, there's an important thing we need to have in mind before applying the Mahalanobis distance: it assumes that the data follows a multivariate normal distribution and that the covariance matrix is representative of the population. If these assumptions are not met, the method may not be effective in detecting outliers.

[10]

This way, let's start by verifying if our data follows a multivariate normal distribution. To check this, we will simply do a multivariate normal probability plot: we will do this for both of the numerical features ("age" and "education-num"). If the data follow a multivariate normal distribution, the points on the plot will follow a straight line:

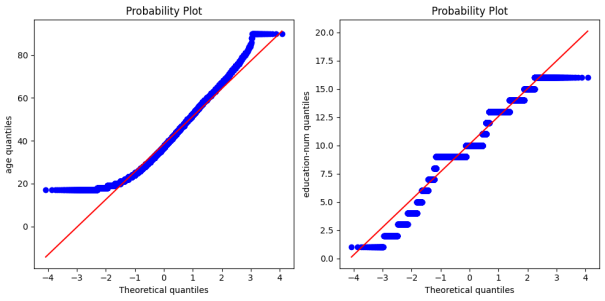


Fig. 12. Multivariate normal distribution of "age" and "education-num"

Since the probability plots for age and education-num do not follow a straight line, it suggests that the data does not follow a multivariate normal distribution, and the Mahalanobis distance may not be an appropriate method to identify outliers for these features.

With that in mind, we had to try an alternate approach to dealing with those outliers. This way, our next approach consists of winsorization and removal of outliers. [20]

Removing all outliers identified from the dataset would result in a loss of almost 3000 entries, and influence the distribution of the data. This is a significant amount of data and we would like to avoid losing as much data as possible, of course. An alternative is winsorizing the data. This involves replacing the extreme values with the nearest non-extreme value. This method can help preserve the distribution of the data and prevent distortion. [20] We will winsorize the "age" column. However, as we saw from the boxplot of the "education-num" column, there are only two outliers, so we will simply remove these ones since it won't affect the distribution of the data, nor lose a significant amount of information. To remove them we will use the Interquartile Rule (IQR rule) [19]. First, we calculate the first quartile (q1), third quartile (q3), and interquartile range (iqr) of the 'education-num' variable. Next, we calculate the lower and upper bounds for outliers using the IQR rule, which states that any data point below $q1 - 1.5 * iqr$ or above $q3 + 1.5 * iqr$ is considered an outlier. [19] To finish, we create the new dataset without the outliers that we talked about before, which we call "df_no_outliers".

Now, we can check the boxplots of the numerical variables again, to see if the outliers were removed and if the distribution of the data was preserved, which was done with success:

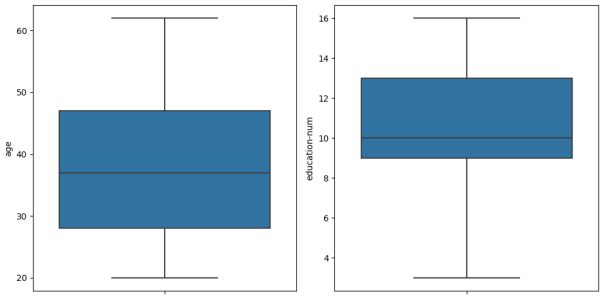
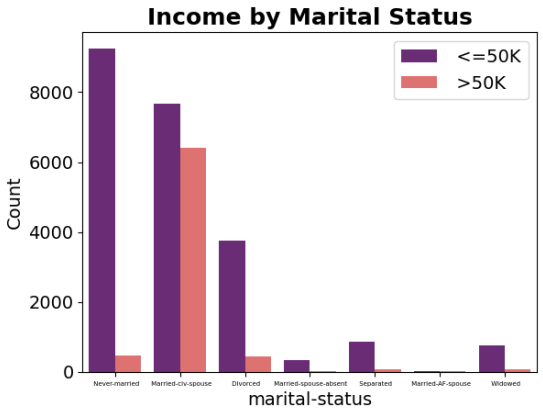


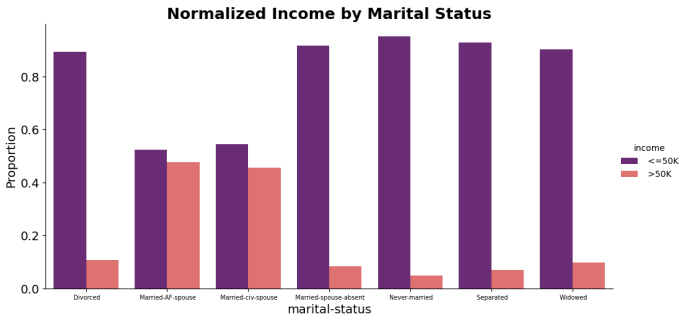
Fig. 13. Boxplot for numerical variables without outliers.

C. Categorical Variables

Now that we have analyzed the distribution of the numerical features, let's have a look at the categorical ones.



(a) Income Distribution by Marital Status (Not-Normalized)



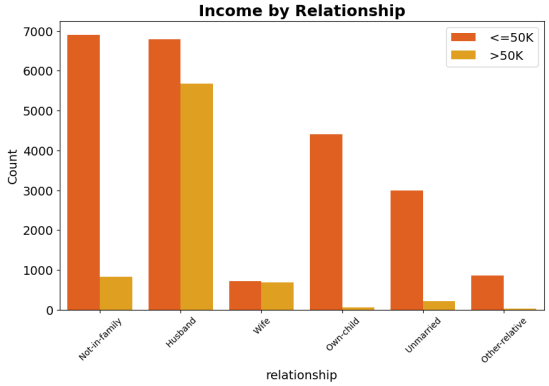
(b) Income Distribution by Marital Status (Normalized)

Fig. 14. Income Distribution by Marital Status.

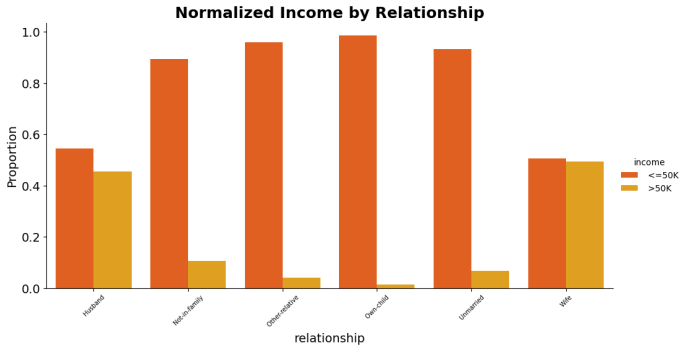
1) Income Distribution by Marital Status: From the plot of figure 14(a) it appears that married individuals have a higher proportion of individuals earning more than 50K compared to other categories. We also have to note that most of the individuals in this dataset fall into this category (about 14065). It also seems that individuals in the category "Married-spouse-absent" and "Married-AF-spouse" have the lowest proportion of individuals earning more than 50K.

Normalizing the marital status feature before plotting can help to better compare the proportion of individuals with income greater than 50K across the different categories of marital status. That's why we plot the figure 14(b). We can see that individuals who are "married-AF-spouse" have the highest proportion of individuals earning more than 50K, followed by individuals who are "married-civ-spouse". This suggests that being married is associated with higher income levels. However, we should also keep in mind that correlation does not necessarily imply causation, and there may be other factors at play. On the other hand, individuals who have never been married, or separated, and whose spouse is absent have the lowest proportion of individuals earning more than 50K.

The same figure is available in the Appendix for better interpretability.



(a) Income Distribution by Relationship (Not-Normalized)



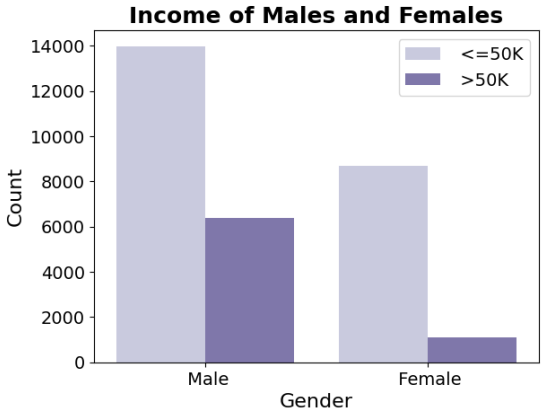
(b) Income Distribution by Relationship (Normalized)

Fig. 15. Income Distribution by Relationship.

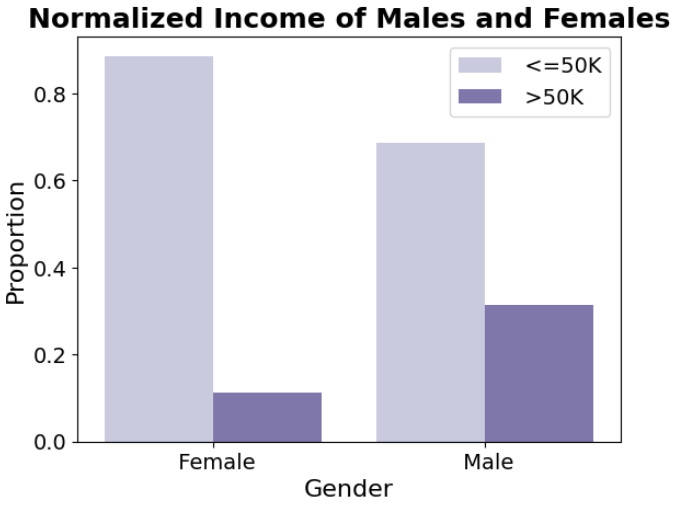
2) Income Distribution by Relationship: We can see, in figure 15(a) that the majority of individuals in the dataset are either husbands or not in a family, and the majority of individuals with an income over 50K are husbands or wives.

As usual, normalizing the "relationship" feature before plotting can help us compare the income distribution across different categories by accounting for the fact that some categories may have more individuals than others. We do this in figure 15(b). From that, we can see that some relationship categories have a higher proportion of individuals with incomes greater than 50K than others. For example, individuals classified as husbands and wives have a higher proportion of incomes greater than 50K compared to other categories such as unmarried, not in family, and other relative.

The same figure is available in the Appendix for better interpretability.



(a) Income Distribution by Gender (Not-Normalized)



(b) Income Distribution by Gender (Normalized)

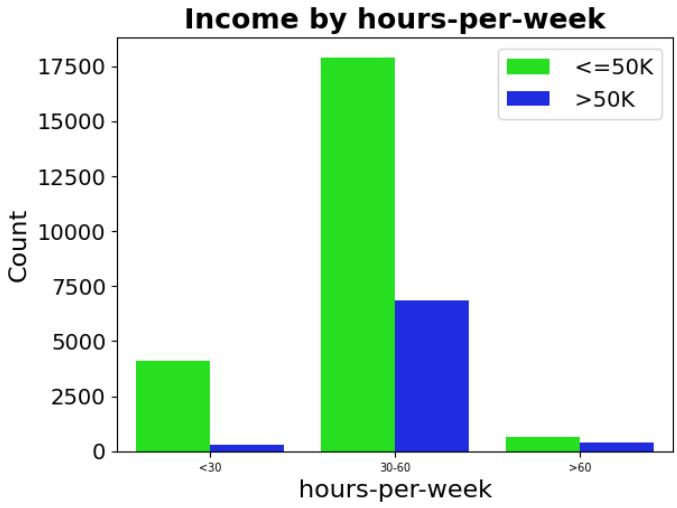
Fig. 16. Income Distribution by Gender.

3) Income Distribution by Gender: We should be aware that the graph of figure 16(a) can be misleading. When we looked at the dataset statistics, we saw that the number of males was much higher than the number of females (about 20380 out of a total of 30162 individuals). To be able to draw conclusions from the income distribution by gender and the existence of a gender income gap, we must therefore normalize our data to the "sex" feature before plotting the graph.

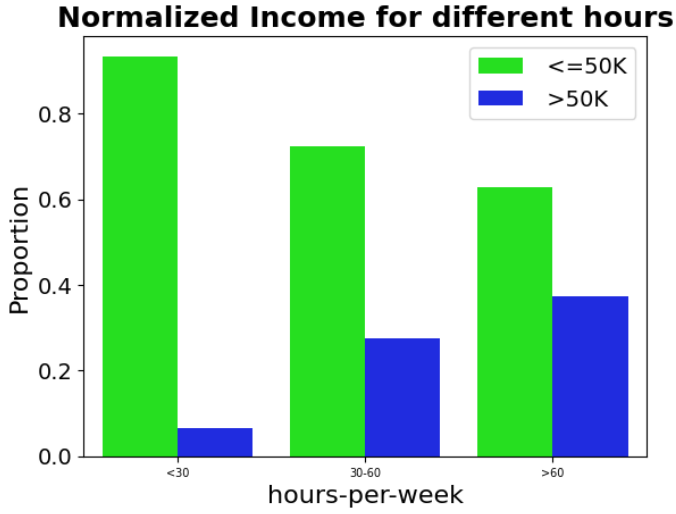
But from this one, we can definitely conclude that there is a significant difference in income between males and females in the dataset. The graph shows that there are more males than females in the high-income category but there is also a higher number of males with an income lower than 50K.

From the plot of figure 16(b), we can take more accurate conclusions about income distribution per gender. We can see that there is a larger proportion of females than males in the low-income category, and a larger proportion of males than females in the high-income category. This suggests that there may be a gender income gap. However, we cannot make any definitive conclusions about the existence or magnitude of the income gap without further analysis of the different features in the dataset. This way, from now on, to understand and outstand the magnitude of the income gender gap we will visualize the different features and then distinguish the gender for each feature.

4) Income Distribution by Hours per Week: As we saw when visualizing the numerical features, we have a new feature, "hours", that is categorical, so we will use a bar plot to visualize the income distribution by hours per week.



(a) Income Distribution by Hours per Week (Not-Normalized)



(b) Income Distribution by Hours per Week (Normalized)

Fig. 17. Income Distribution by Hours per Week.

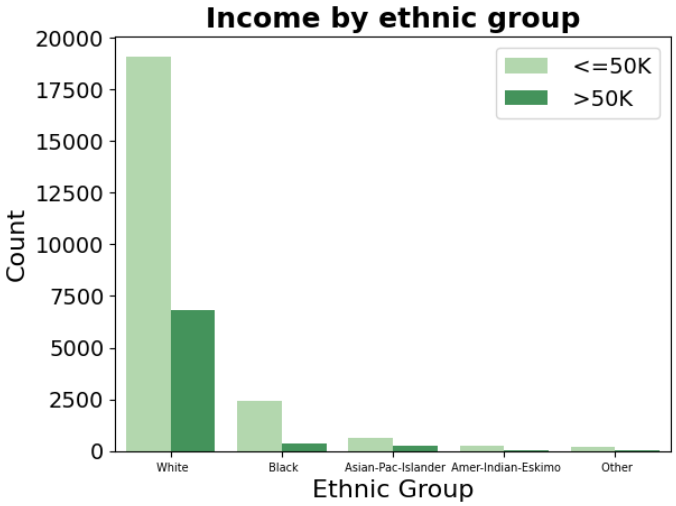
We can see, from figure 17(a) that our dataset has more individuals that work 30-60 hours per week. We can also see that the proportion of individuals with an income greater than 50K is higher for individuals working more than 30 hours per week. This suggests that working more hours per week may be associated with higher income levels. However, as in our previous visualizations, we should normalize our data before making any conclusions, as we do in figure 17(b). It shows that as the number of hours per week increases, the proportion of individuals earning more than 50K per year also increases. Additionally, for each category of hours per week, there are more individuals earning less than 50K per year than those earning more than 50K per year. This indicates that the dataset is imbalanced towards the lower income bracket, as we already stated.

Now, let's compare this also by gender. We want to check if, working the same hours per week, there's still a gap in the income between males and females. This plot is represented in figure 41 in the Appendix section. From this, it is possible to see that, for both males and females, those who work more hours per week tend to have a higher proportion of people earning more than $50K. However, there are some differences in the income distribution between genders and hours-per-week groups. That is, for males, the proportion of people earning more than $50K is higher when compared with the same "hours" group for females. These differences suggest that gender and hours per week may have an impact on income, and should be considered when building predictive models for income.

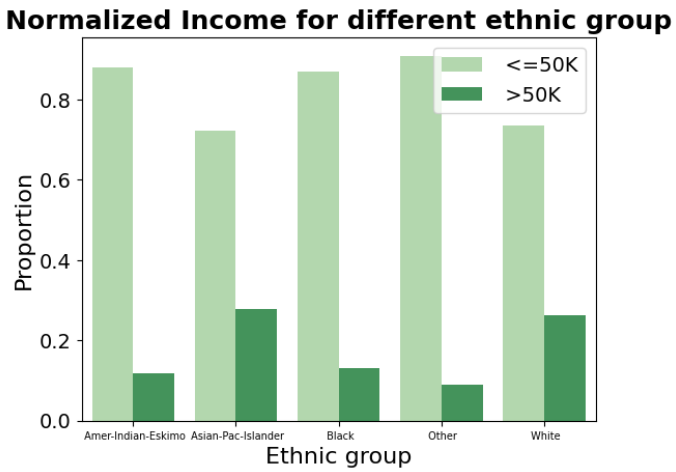
5) Income Distribution by Ethnic Group: We can observe, in figure 18(a) that the White ethnic group has the highest count of individuals earning more than 50K, followed by the Asian-Pac-Islander group. On the other hand, the Amer-Indian-Eskimo and Other ethnic groups have the lowest count of individuals earning more than 50K.

We should normalize the race feature before plotting because the counts of individuals in each ethnic group are not the same, and this could lead to a biased interpretation of the results. Normalizing the feature ensures that each ethnic group is represented proportionally, which allows for a fair comparison of the income distribution across ethnic groups. This normalized plot is presented in figure 18(b). We can more clearly see that there are significant income differences between different ethnic groups. For example, individuals from the Asian-Pac-Islander and White ethnic groups are more likely to have income greater than 50K, while individuals from the Black and Amer-Indian-Eskimo groups are less likely to have income greater than 50K.

In figure 42, in the Appendix section, we distinguish also by gender. We can see that there are significant differences in income between different ethnicities and genders. For example, among females, those in the White and Asian/Pacific Islander categories have a higher proportion of high-income earners compared to other ethnic groups. Among males, also those in the White and Asian/Pacific Islander categories have a higher proportion of high-income earners however, this proportion is

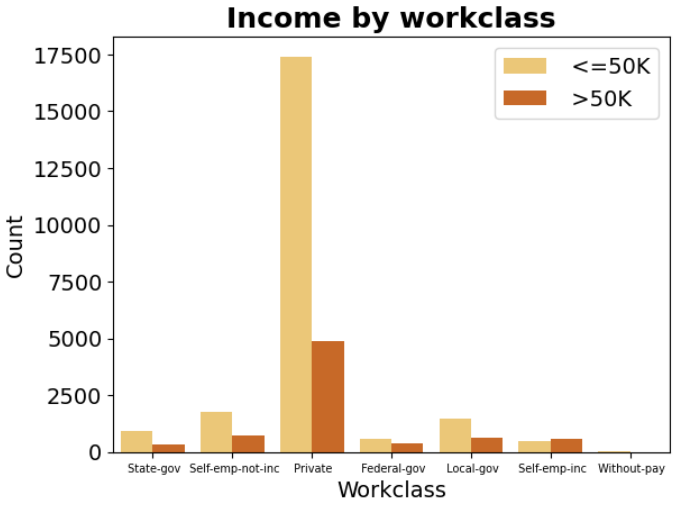


(a) Income Distribution by Ethnic Group (Not-Normalized)

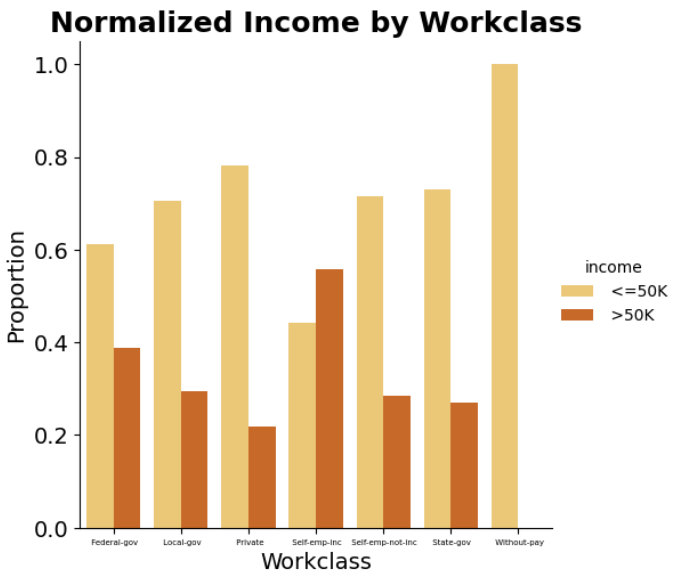


(b) Income Distribution by Ethnic Group (Normalized)

Fig. 18. Income Distribution by Ethnic Group.



(a) Income Distribution by Workclass (Not-Normalized)



(b) Income Distribution by Workclass (Normalized)

Fig. 19. Income Distribution by Workclass.

higher for males than females.

Again, can conclude that there is a significant gender gap in income, with a higher proportion of males in the high-income category compared to females. This trend is observed across all ethnic groups in the dataset. Furthermore, we can observe that the gender gap is more pronounced in some ethnic groups than others. For example, in the White and Asian/Pacific Islander groups, the gap between males and females is wider than in the other ethnic groups.

Overall, the plots suggest that there are disparities in income by both gender and ethnic group, and these disparities may intersect and compound one another.

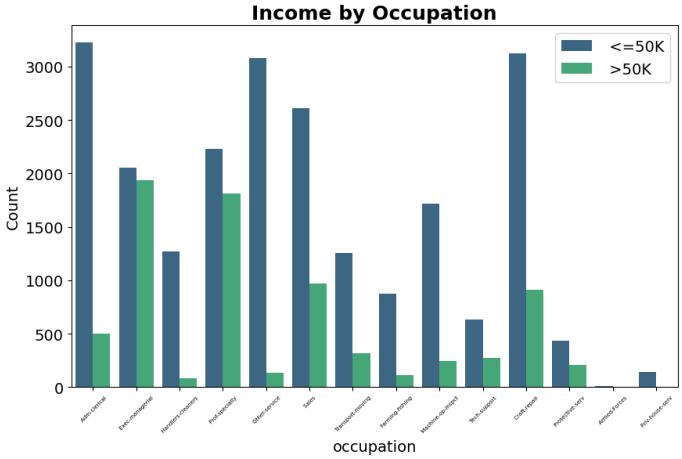
6) Income Distribution by Workclass: We can see, from figure 19(a) that the majority of individuals work in the Private workclass, and a large proportion of them earn less than or equal to 50K. On the other hand, individuals in the Self-emp-inc workclass have a higher likelihood of earning over 50K. We can clearly see that there is a substantial income disparity between different workclasses.

Again, normalizing the workclass feature allows us to compare the income distribution of each workclass on a level playing field, without being biased towards workclasses that have a larger number of samples. This makes it easier to compare and draw conclusions about the income distribution of different workclasses. We do this in figure 19(b). Now, we can conclude that there are significant differences in income levels among different workclasses. We can see that

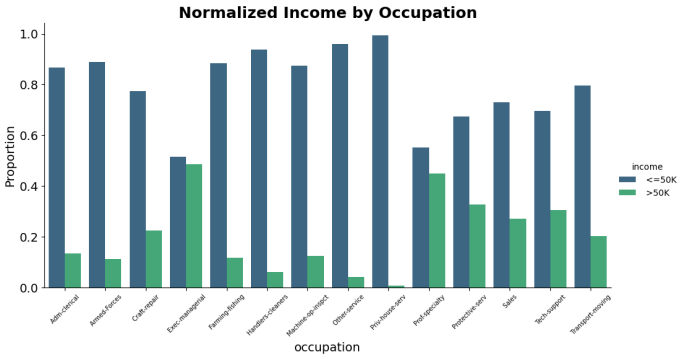
some workclasses such as 'Self-emp-inc' and 'Federal-gov' have a higher proportion of people with income above 50k, while others like 'Private' and 'Self-emp-not-inc' have a lower proportion.

As before, we distinguish by wokclass but also gender. We can see this in figure 43, in the Appendix section. One more time, we can see that there are significant disparities. For example, in almost all workclass categories, a higher proportion of males earn more than 50K compared to females. Additionally, for some workclass categories such as Private and Self-emp-not-inc, the proportion of males earning more than 50K is significantly higher than the proportion of females. This suggests that there may be gender-based income disparities in certain occupations or work environments.

We've already noticed some gender discrepancies in the previous visualizations, but comparing the gender together with different features (such as workclass, education, etc) may help us see these disparities more clearly.



(a) Income Distribution by Occupation (Not-Normalized)



(b) Income Distribution by Occupation (Normalized)

Fig. 20. Income Distribution by Occupation.

7) Income Distribution by Occupation: We can conclude, from figure 20(a) that individuals in the executive/managerial and professional specialty occupations have a higher proportion of individuals earning more than 50K. Conversely, individuals in the Armed Forces and private household service occupations have the lowest proportion of individuals earning more than 50K.

Once more, normalizing the occupation feature before plotting is important because different occupations may have significantly different numbers of individuals in the dataset. By normalizing the values, we can compare the proportion of individuals in each occupation that earn more than 50K, rather than simply comparing the raw counts. This allows for a more accurate comparison of the income distributions across different occupations. We do this in figure 20(b). We can see that individuals in executive/managerial and professional specialty occupations have a higher proportion of individuals earning more than 50K. Conversely, individuals in private household service and other-service occupations have the lowest proportion of individuals earning more than 50K.

Again, let's add the comparison between genders, as we can see in figure **??**, in the Appendix section. We can see that the proportion of individuals who earn more than 50K is generally higher among males than females across all occupations, except for "Other-Service" where the proportions are similar for both genders. Among females, the proportion of those who earn more than 50K is generally highest in the "Exec-managerial" and "Prof-specialty" occupations, as it is for the males. The "Farming-fishing" and "Priv-house-serv" occupations have the highest proportion of individuals who earn less than or equal to 50K across both genders.

8) Income Distribution by Native Country: We can observe, in figure 45(a) in the Appendix section, that the majority of people in the dataset are from the United States. We already saw this when we did a brief statistical analysis of the dataset (about 27504 individuals are from the US). The majority of people in each country have an income of less than 50K.

Normalizing the "native-country" feature will help us with further conclusions. Having a look at figure 45(b) in the Appendix section, we can see more clearly that there are significant differences in the income distribution across different native countries. For example, countries like Iran, India, and Taiwan have a higher proportion of people with an income greater than or equal to 50K compared to countries like the Outlying US (Guam-USVI-etc), Dominican Republic, and Columbia.

Once again, we also add the comparison between genders as we can see in figure 46, in the Appendix section. We can see that there are significant gender differences in income distribution across most native countries. In particular, males generally have a higher proportion of individuals with income greater than 50K compared to females. This suggests, one more time, that there is a gender gap in income, where males tend to earn more than females. However, the magnitude of this gender gap varies across different native countries. For example, in some countries, such as Vietnam and China, the gender gap in income is relatively small. In contrast, in other countries, such as Cambodia and Puerto Rico, the gender gap in income is much larger.

After this visualization process, we already have a much better understanding of the data we are dealing with, as well

as the relationships of our variables. We are then able to apply our supervised learning models.

V. Implemented Models

After having pre-processed the data we proceeded to the implementation of our deep and machine learning algorithms.

For our solution, we implemented one machine learning algorithm:

*1) **RandomForestClassifier**:* It is a type of machine learning algorithm that combines the output of multiple decision trees to reach a single result. It is an ensemble method that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve predictive accuracy and control overfitting. [12] [18] This can be a good choice for the UCI Adult Dataset because of this ability to reduce overfitting and improve predictive accuracy by averaging the results of multiple decision trees.

It is always a good idea to try multiple machine learning algorithms and compare their performance to determine which one is the best fit for a specific dataset. In this project, we pretend to compare with deep learning algorithms to see if they can get better performance than Random Forest. This way, for our solution we implemented 2 deep-learning algorithms:

*2) **Multilayer Perceptrons Classifier (MLP)**:* is a type of artificial neural network that is composed of multiple layers of interconnected "neurons." The input layer receives the raw data, and each subsequent layer applies mathematical transformations to the data, allowing the network to learn increasingly complex features of the data. The final layer, known as the output layer, generates the network's predictions. [17]

They are capable of learning complex non-linear relationships between input and output data. This way, it could be a good fit for this dataset because it can learn to model the complex relationships between the input features and the target variable.

*3) **Long Short-Term Memory (LSTM)**:* is a type of artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections, making it a type of recurrent neural network (RNN). This allows it to process not only single data points but also entire sequences of data.

A common LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. [9] [22]

Applying LSTM to this dataset can be a good idea as the goal is to predict or classify the income of individuals based on their attributes. The LSTM can process the data sequentially, taking into account the temporal dependencies and patterns in the data. For example, it can learn from the education and occupation of an individual in the past to predict their future income.

VI. Application and Analysis of the Machine Learning Models

We divide the application of our machine learning models into two parts. In the first part, we apply the machine learning models using the default values for their hyperparameters, using the training set. After this, we can tune our model's hyperparameters, using the validation set, and see if there are any improvements in their performance. In the end, when we find the best values for our hyperparameters, we apply the model to our test set (we do exactly the same for the datasets with and without outliers to see if there is any improvement in removing them).

To search for the best hyperparameters for the model (hyperparameter tunning), we do a systematic approach instead of just one or several randomly chosen values. To later see what's best, we compare the training and validation accuracy. We also plot the cost function (when possible) for the validation and training sets so we can see how our model is evolving over iterations. It is safe to use accuracy as a metric for doing this because the dataset is balanced, so accuracy is a good metric for this case. After choosing the model with the hyperparameters with best the best pair of training and validation accuracies, we apply it to the test set. After this, since we are dealing with a classification problem, we present a classification report performance metrics [13], a confusion matrix [14], and a ROC curve [15].

For the classification report, we analyzed the following performance metrics:

1) **Precision:** tells us, out of all the positive predicted, what percentage is truly positive. The precision value lies between 0 and 1;
2) **Recall:** tells us, out of the total positive, what percentage are predicted as positive. It is the same as TPR (true positive rate);
3) **F1-score** that is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

Precision, recall, and F1 score are all broken down by class, and then a macro average and weighted average are given for each:

1) **Macro average:** is the usual average we're used to seeing. Just add them all up and divide by how many there were;
2) **Weighted average:** considers how many of each class there were in its calculation, so fewer of one class means that it's precision/recall/F1 score has less of an impact on the weighted average for each of those things.
3) **Support:** tells how many of each class there were.

[13]

The confusion matrix represents, on the principal diagonal, the true positives and true negatives, that is, the values that the model predicted as having income higher or lower than 50K and actually are. The remaining two values are false positives

or false negatives, that is, cases where the model predicted the wrong class. [14]

Finally, we also talk about the Receiver Operating characteristic curve. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection.The false-positive rate is also known as probability of false alarm and can be calculated as (1-specificity). Classifiers that give curves closer to the top-left corner indicate better performance. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). [15]

For each model we apply, we also do k-fold cross-validation [4] when tunning the hyperparameter. We usually apply this when we are tuning the hyperparameters using GridSearch. [16] It is a method in scikit-learn that performs an exhaustive search over specified parameter values for an estimator. It helps to find the best hyperparameters for a machine-learning model by evaluating all possible combinations of hyperparameter values using cross-validation.

K-fold cross-validation is a technique used to evaluate the performance of a model. The data is divided into k subsets of equal size. The model is trained on k-1 subsets and tested on the remaining subset. This process is repeated k times, with each subset serving as the test set once. The average performance across all k iterations is used as the overall performance measure. [4]

When using GridSearchCV with k-fold cross-validation, the data is split into k subsets and for each combination of hyperparameters, the model is trained and evaluated k times using cross-validation. The combination of hyperparameters that results in the best average performance across all k iterations is selected as the best set of hyperparameters for the model.

A. Random Forest Classifier (RF)

When building a machine learning model, it's common to start with standard or default values for the hyperparameters because these values have been shown to work well in many different scenarios. Using standard values as a starting point can help us quickly build a baseline model that we can use to evaluate the performance of the model and compare it to other models.

This way, let's start by building a baseline model with the default hyperparameter values. We will use the RandomForestClassifier class from the sklearn.ensemble module to build the model. [12]

The default parameters for the RandomForestClassifier from sklearn.ensemble are as follows:

1) n_estimators=100: The number of trees in the forest;
2) criterion='gini': The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain;
3) max_depth=None: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples;
4) min_samples_split=2: The minimum number of samples required to split an internal node;
5) min_samples_split=2: The minimum number of samples required to split an internal node;
6) min_samples_leaf=1: The minimum number of samples required to be at a leaf node;
7) min_weight_fraction_leaf=0.0: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided;
8) max_features='sqrt': The number of features to consider when looking for the best split.
9) max_leaf_nodes=None: Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes;
10) min_impurity_decrease=0.0: A node will be split if this split induces a decrease of the impurity greater than or equal to this value;
11) bootstrap=True: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
12) oob_score=False: Whether to use out-of-bag samples to estimate the generalization accuracy;
13) n_jobs=None: The number of jobs to run in parallel for both fit and predict. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors;
14) random_state=None: Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap is True) and the sampling of the features to consider when looking for the best split at each node (if max_features ¡ n_features);
15) verbose=0: Controls the verbosity when fitting and predicting;
16) warm_start=False: When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest;
17) class_weight=None: Weights associated with classes in the form class_label: weight. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y;
18) ccp_alpha=0.0: Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed;
19) max_samples=None: If bootstrap is True, the number of samples to draw from X to train each base estimator.

[12]

In this model we will not be able to plot the cost/loss function or apply regularization to it. Random forest is an ensemble learning method that constructs a multitude of decision trees at

training time. Unlike some other machine learning algorithms, such as neural networks or logistic regression, random forest does not have a single loss function that is optimized during training. Instead, each individual decision tree in the random forest is constructed by recursively splitting the data based on a cost function, also known as an impurity measure or splitting criterion.

As a result, it is not possible to plot the loss function for a random forest model as a whole. However, we can plot the out-of-bag (OOB) error rate for the random forest as a function of the number of trees in the forest. The OOB error rate is an estimate of the generalization error of the random forest and can be used to assess its performance.

This way, we create a random forest classifier with oob_score set to True and warm_start set to True (leaving the rest of parameters as default). It then fits the random forest to the training data using different numbers of trees and calculates the OOB error rate for each number of trees. Finally, it plots the OOB error rate as a function of the number of trees as we can see in figure 21.

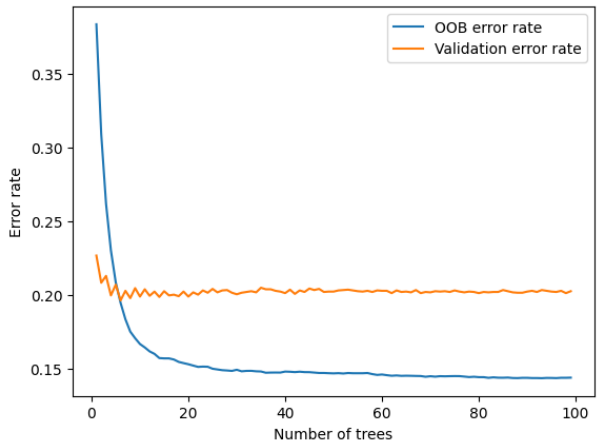


Fig. 21. OOB error for training and validation sets - Default Parameters.

Looking at the error rate as a function of the number of trees, we can see that the error rate decreases as the number of trees increases. However, the error rate does not decrease monotonically. Instead, the error rate decreases rapidly at first and then levels off. This is because the random forest is able to reduce the error rate by adding more trees, but only up to a certain point. After a certain number of trees, the error rate does not decrease any further.

The validation error decreases a little but it is always higher than the training error. This is a sign of overfitting. [8]

For this prediction, we got the confusion matrix (with the validation set, since we are still tuning our model):

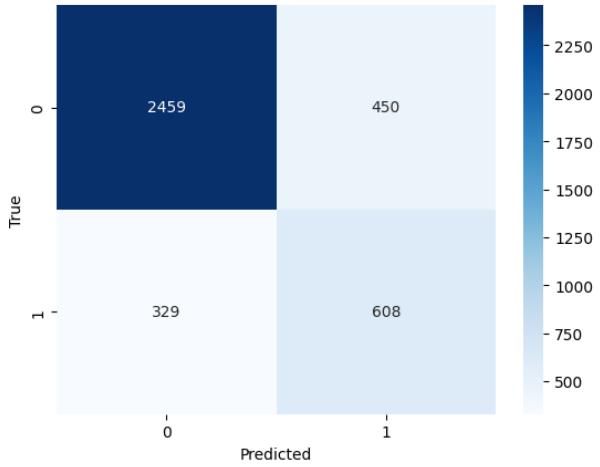


Fig. 22. Confusion Matrix for the RF - default parameters.

We can see that we got 2459 true negatives (TN), 450 false positives (FP), 329 false negatives (FN), and 608 true positives (TP).

As for the classification report:

```
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.85      0.86      2909
           1       0.57      0.65      0.61       937

    accuracy                           0.80      3846
   macro avg       0.73      0.75      0.74      3846
weighted avg       0.81      0.80      0.80      3846
```

Fig. 23. Classification report for the RF - default parameters.

The classification report of figure 23 shows several metrics to evaluate the performance of a classification model.For this case, we can see that the model has an overall accuracy of **0.80**, meaning that it correctly classified 80% of the samples.

For class 0, the model has a precision of 0.88 and a recall of 0.85. The F1-score is the harmonic mean of precision and recall and is 0.86 for class 0.

For class 1, the model has a precision of 0.57 and a recall of 0.65, with an F1-score of 0.61. This means that there's is still margin for improvement.

We also show the ROC curve, and we got:

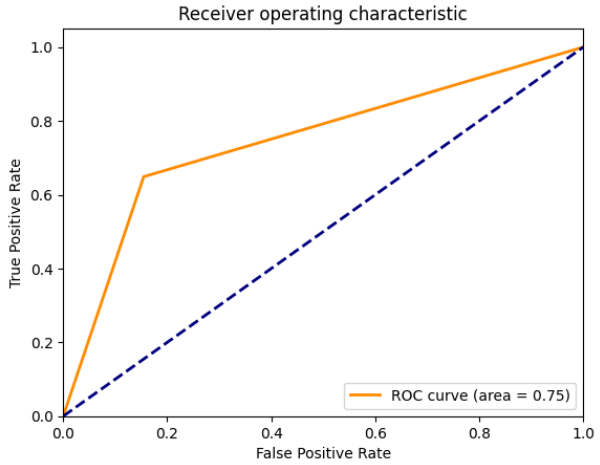


Fig. 24. ROC Curve - default parameters.

An AUC of 0.75 indicates that our model has a moderate ability to distinguish between positive and negative classes.

Now, we tune the hyperparameters of our model. For this, we will use GridSearch, as we already talked before. Having in mind the shape and complexity of our data, we searched for the following hyperparameters, as these are commonly used to tune random forest models:

- n_estimators
- max_depth
- min_samples_split
- max_features

We tunned the values for our hyperparameters three times. We were able to do this systematic approach by plotting how the hyperparameters of a model affect its performance. Having in mind the shape and complexity of our data, we will search for the following hyperparameters, as these are commonly used to tune random forest models:

- n_estimators
- max_depth
- min_samples_split
- max_features

We can see the steps we went through to get to the final model in the Appendix, in the sections Hyperparameter tuning using GridSearchCV (1st try), Hyperparameter tuning using GridSearchCV (2nd try) and Hyperparameter tuning using GridSearchCV (3rd try). In this last section, we can actually see, in figure 49 that the score of the model is pretty much stable as the hyperparameters change. This way, we will stop our search here and return the best hyperparameters found by the grid search.

This way, we got, as final hyperparameters for this model:

1) 'max_depth': 30
2) 'max_features': 'auto'
3) 'min_samples_split': 5
4) 'n_estimators': 100

Now that we tuned our model and found the optimal hyperparameter's values for the random forest model, we apply this model to our test set. This way, we got the following results:

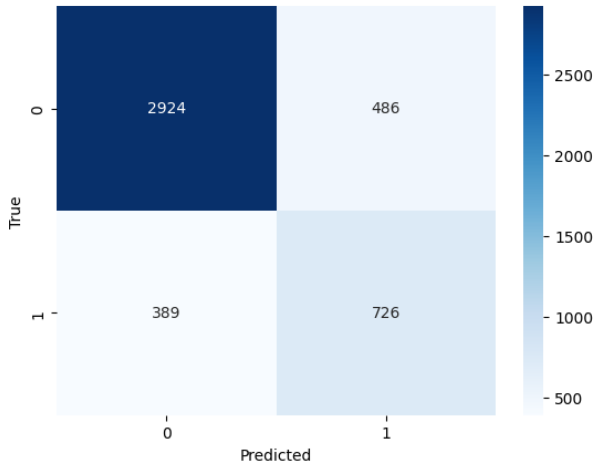


Fig. 25. Confusion Matrix - Tuned Model.

Classification report:	precision	recall	f1-score	support
0	0.88	0.86	0.87	3410
1	0.60	0.65	0.62	1115
accuracy			0.81	4525
macro avg	0.74	0.75	0.75	4525
weighted avg	0.81	0.81	0.81	4525

Fig. 26. Classification Report - Tuned Model.

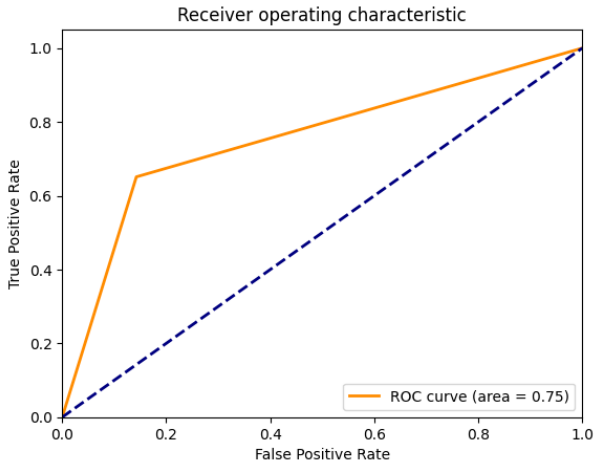


Fig. 27. ROC Curve - Tuned Model.

From the confusion matrix of figure 25, we can see that:

1) The model correctly predicted 2924 instances of class 0 (true negatives) and 726 instances of class 1 (true positives).

2) The model incorrectly predicted 486 instances of class 0 as class 1 (false positives) and 389 instances of class 1 as class 0 (false negatives).

From the classification report of figure 26, we can see that the model has an overall accuracy of 0.81, meaning that it correctly classified 81% of the samples.

For class 0, the model has a precision of 0.88 and a recall of 0.86. The F1-score is the harmonic mean of precision and recall and is 0.87 for class 0.

For class 1, the model has a precision of 0.60 and a recall of 0.65, with an F1-score of 0.62.

Again, ROC curve area of 0.75.

We concluded that the model has a moderate ability to distinguish between positive and negative classes. The performance of the random forest classifier remains the same after hyperparameter tuning as it was with the default parameters. After this, another thing we tried was applying the exact same thing but for the dataset without outliers that we defined earlier. However we did not see any improvement at all. Therefore, we will not use this dataset without outliers again in the following deep learning models. We experimented with random forest classifier because it is a machine learning model of binary classification, faster to run and less computationally expensive. Since we saw that it does nothing to change the performance of the model, we know that it will have no effect on the other models either. Anyways, the results we got are in the Appendix in the section GridSeachCV for dataset without outliers.

VII. Application and Analysis of the Deep Learning Models

After applying the machine learning algorithm the next step was to implement Deep Learning (DL). Again the approach was to start with the default values of the algorithms and then choosing a set of hyper-parameters for our models and see if there were any improvements in their performance.

A. Multi Layer Perception

The procedure was very similar for both of the deep learning algorithms implemented. In a first part test three different values for recurrent units and then test different regularizations using the number recurrent units which had the best performance in the previous step. For the MLP the baseline was 3 Dense Layers which included the output whose activation function was sigmoid.

1) Using 8 Recurrent Units with no Regularization: In this hypothesis we tested the MLP using 8 recurrent units which lead to a training accuracy of 82.7% and a validation accuracy of 79.1%.

2) Using 16 Recurrent Units with no Regularization: The next approach was to test it with 16 units. In this attempt we got a training accuracy of 82.6% and a validation accuracy of 79.09%.

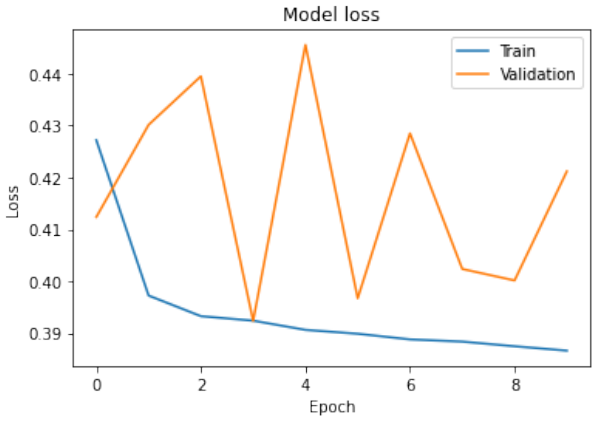


Fig. 28. Loss Function for 32 recurrent units

3) Using 32 Recurrent Units with no Regularization: On the Figure 28 we can see the loss function for the execution of the MLP with 32 units. The training and validation accuracy were 82.4% and 79.4%, respectively.

Now, it was time to compare the results which are presented in the following table. Even though the model with 32 units has the worst training accuracy we need to take account the existence of over fitting and, as we can see, contrarily it has the highest accuracy for the validation set.

TABLE I
Result Analysis

Model	Training Accuracy	Validation Accuracy
8 MLP	0.827642	0.791472
16 MLP	0.826023	0.790952
32 MLP	0.792512	0.792512

4) Using 32 Recurrent Units with L1 Regularization: Here we introduce GridSearch to test the L1 factor. The values tested were [0.001, 0.01, 0.1]. In the end the 0.001 factor became the best parameter leading to a training accuracy of 81.9% and a validation accuracy of 80.29%.

5) Using 32 Recurrent Units with L2 Regularization: This attemp was very similar to the previous one. Even the values tested were the same and the best L2 factor was also 0.001. The accuracy was 82.1% and 80.86% for training and validation, respectively.

6) Using 32 Recurrent Units with Dropout: To test the dropout we used this values [0.01,0.1, 0.2, 0.3, 0.4, 0.5] for the dropout rate. After doing a GridSearch the best parameter was 0.01 and its training accuracy was 81.7% and validation accuracy 81.53%. The following Figure shows the variation of mean test score according to the tested dropout rates.

29

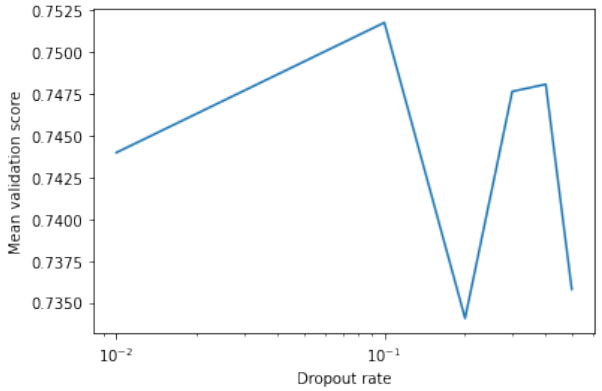


Fig. 29. Mean test score variation through dropout rate

7) Using 32 Recurrent Units with Early Stopping: The final attempt was to do the same but for Early Stopping. Here we tested different values for the parameter patience ([1,2,3]).The best parameter lead to a training accuracy of 82.5% and a validation accuracy of 79.9%.

8) Result Analysis: Analysing the following table we can see that the results are very similar with a slight advantage for both, training and validation, accuracy with early stopping.

TABLE II
RESULT ANALYSIS

Regularization	Training Accuracy	Validation Accuracy
L1	0.819659	0.802912
L2	0.821584	0.808632
Dropout	0.818712	0.815393
Early Stopping	0.825707	0.819012

Then on the test set we got 79% accuracy. The roc curve and confusion matrix can be found above.

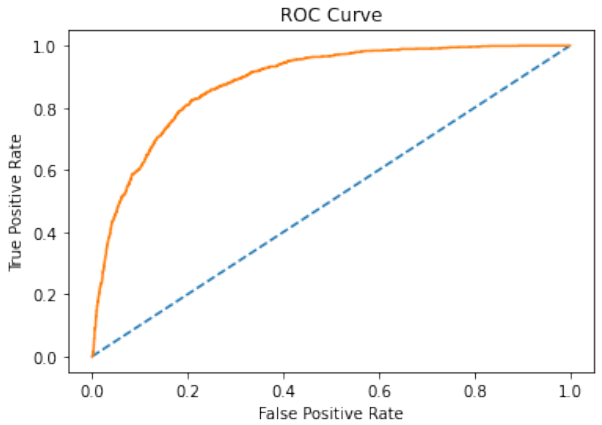


Fig. 30. Roc Curve for MLP

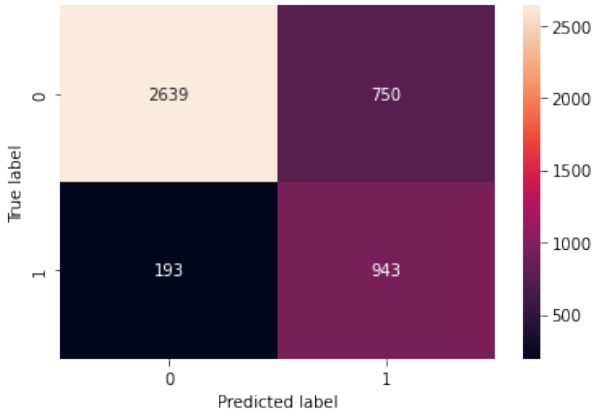


Fig. 31. Confusion Matrix for MLP

B. Long Short-Term Memory

As mentioned before both deep learning processes were very similar. For the LSTM the baseline was 2 LSTM Layers and the output whose activation function was sigmoid.

1) Using 8 Recurrent Units with no Regularization: We used 8 recurrent units to test the MLP for this hypothesis, and the results showed training accuracy of 83.2 percent and validation accuracy of 80.3 percent. In Figure 32 we can see the loss function for the execution of the MLP with 8 units.

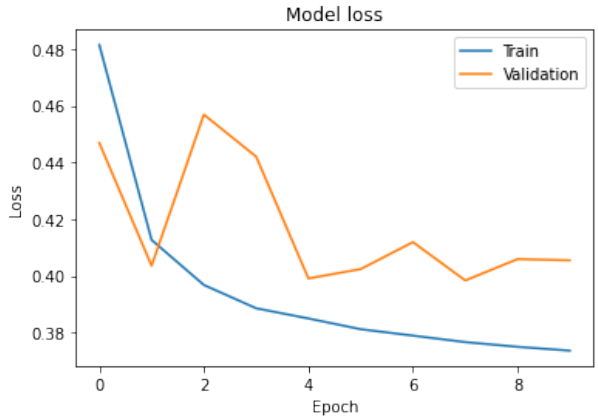


Fig. 32. Loss function for LSTM using 8 units

2) Using 16 Recurrent Units with no Regularization: The following strategy was to test it using 16 units. We achieved training accuracy of 82.1 percent and validation accuracy of 82.2 percent in this effort.

3) Using 32 Recurrent Units with no Regularization: The final attempt with different recurrent units was with number 32. Here we got 83.3 percent in training and 79 percent in validation.

The results, which are shown in the following table, might now be compared. We must consider over fitting may exists on the other hypothesis because the model with 32 units has the worst training accuracy even if, as we can see, it has the highest accuracy for the validation set.

TABLE III
RESULT ANALYSIS

Model	Training Accuracy	Validation Accuracy
8 MLP	0.832427	0.803432
16 MLP	0.820911	0.821633
32 MLP	0.833466	0.795112

4) Using 32 Recurrent Units with L1 Regularization: Just like in the previous model we introduced hyperparameter tuning with regularizations. The values tested were [0.001, 0.01, 0.1]. Here we saw that the the slope of the function was decreasing right from the beginning of the plot (represented in Figure 33). So we did another GridSearch increasing the parameters list with lower numbers,[1e-20,1e-15,1e-10,1e-3]. The final result is in Figure 34.

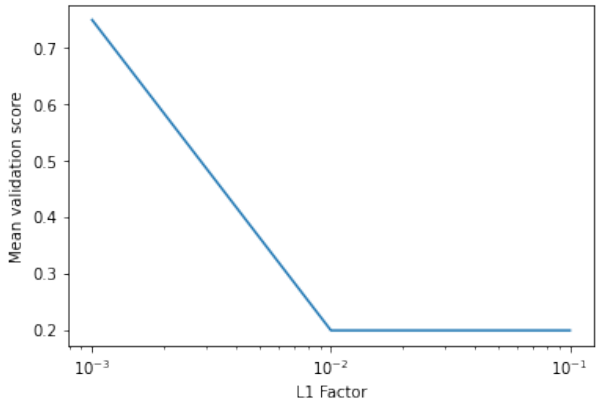


Fig. 33. Mean test score variation through L1 rate

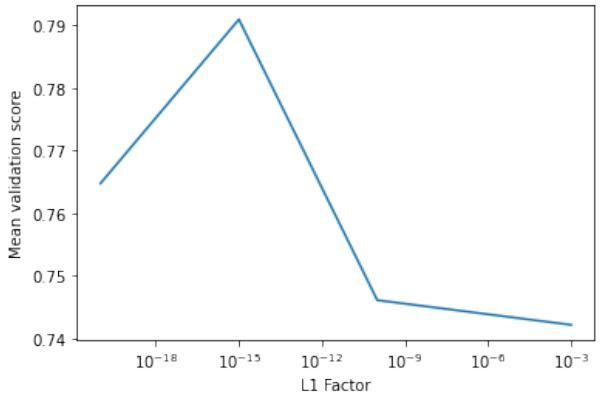


Fig. 34. Mean test score variation through L1 rate

5) Using 32 Recurrent Units with L2 Regularization: The prior attempt was remarkably similar to this one. Again the values tested were the same, but, comparing with the MLP, here we found another coincidence. We had the need to do Multiple GridSearch's due to the event referred that happened in the previous L1. So we have to test 3 different sets of values to find one which was good enough. Its training accuracy was 81.7% and validation accuracy 81.53%.

6) Using 32 Recurrent Units with Dropout: We utilized the same dropout rate values to test the dropout. The best parameter after doing a GridSearch was 0.3, with training accuracy of 82.8% and validation accuracy of 80.2%.

7) Using 32 Recurrent Units with Early Stopping: The final attempt was, again, to do the same but for Early Stopping. A training accuracy of 82.7 percent and a validation accuracy of 74.9 percent were achieved with the best parameter.

8) Result Analysis: The results are relatively comparable, with accuracy with early halting showing a little advantage for both training and validation, according to the analysis of the accompanying table.

TABLE IV
RESULT ANALYSIS

Regularization	Training Accuracy	Validation Accuracy
L1	0.834321	0.781071
L2	0.835329	0.797192
Dropout	0.828120	0.801872
Early Stopping	0.827143	0.749090

Once more, we calculated the roc curve and confusion matrix, which are represented above. The final accuracy for the test set was: 76%.

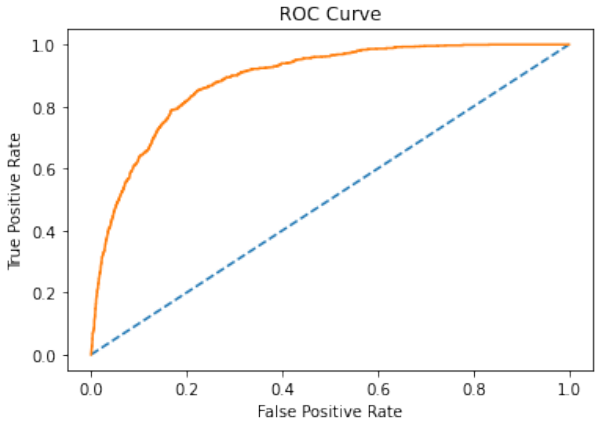


Fig. 35. Roc Curve for LSTM

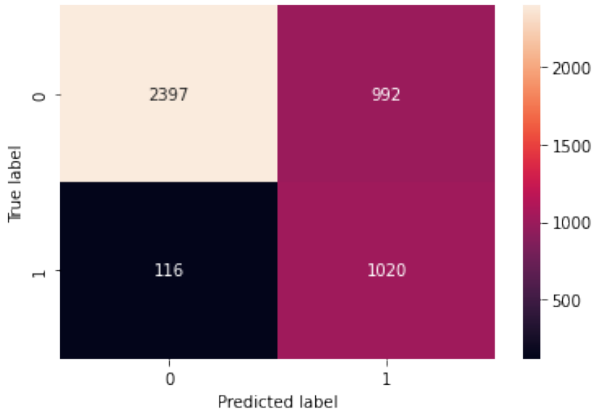


Fig. 36. Confusion Matrix for LSTM

VIII. Future work and Novelties

In this following part, we will discuss potential research fields and identify fresh and creative avenues that could be explored in light of the findings and recommendations of the current study.

Example 1

In AliAkbar Badri [3] approach, one of the early perceptions that we have looking at the notebook is an insufficient lack of data exploration and visualization. In fact, in this work the pre-processing of data starts right from the beginning of the notebook which might be a bad approach since it's always important to know what we are dealing with before do anything to the data. Also a different choice was how the encoding was done, since in the referred work label encoder was done whereas in this project we used one-hot encoding. Then, when it comes to the models, using only one model and not attempt to see how different parameters work on the neural network is also not a good approach too. However, the use of some of the utilities of keras such as model summary, for example, can be an helper to see the model architecture.

Example 2

The work of Mohammad-Reza Azizi [11], also includes a deep learning approach using MLP on the same dataset. Again the we do not have any data exploration that can be relevant and have a later impact of the results, depending on the findings we have. Like in the previous reference it could have further efforts to experiment with alternative parameters. On the other hand, its activation function is user defined whereas in our work that not and tuning this hyper-parameter could improve, or not, the results of presented in this paper.

IX. Conclusions

In this paper, we presented a detailed analysis of individual's attributes for income prediction. We have covered every required criteria content. First, the dataset was pre-processed, all data cleaning and pre-processing revealed to be very important to improve our model's performance. We continue with data visualisation, all the plots and graphics were a major assistance to decide how to manipulate and interpret the data. Data visualization helps people to see, interact with, and better understand the data. Whether simple or complex, the right visualization can bring everyone on the same page, regardless of their level of expertise.

Supervised learning techniques such as random forest classifier (RF) was applied to the data, and evaluated based on its performance on the test set. Additionally, deep learning techniques such as MLP and LSTM were also implemented and their performance were compared with the traditional machine learning model (RF):

Fig. 37. Comparison between MLP, LSTM and RF.

	Model	Test Accuracy
0	LSTM	0.806630
1	MLP	0.794033
2	Random Forest	0.822099

Fig. 38. Comparison between MLP, LSTM and RF.

We can see that RF was the best algorithm (82% of accuracy) followed by the deep learning algorithms, LSTM and MPL, with 80% and 79% of accuracy, respectively. The algorithm with the worst score was the MLP, although the accuracies didn't vary much.

Concluding, this project highlights the importance of using data pre-processing, visualization and supervised learning together in order to achieve the best performance in predicting the income class and also how to tune the models to get the best accuracy and try to prevent them from overfitting.

References

[1] Adult income dataset. https://archive.ics.uci.edu/ml/datasets/adult. Accessed: 2023-03-27.

[2] Bimodal distribution: What is it? https://www.statisticshowto.com/what-is-a-bimodal-distribution/. Accessed: 2023-04-07.

[3] Classification Adult DataSet using MLP (Multi-Layer Perceptron). https://github.com/AliAkbarBadri/mlp-classifier-adult-dataset/blob/master/mlp.ipynb. Accessed:2023-04-07.

[4] Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html. Accessed:2023-04-07.

[5] Demographics: How to collect, analyze, and use demographic data. https://www.investopedia.com/terms/d/demographics.asp. Accessed: 2023-04-07.

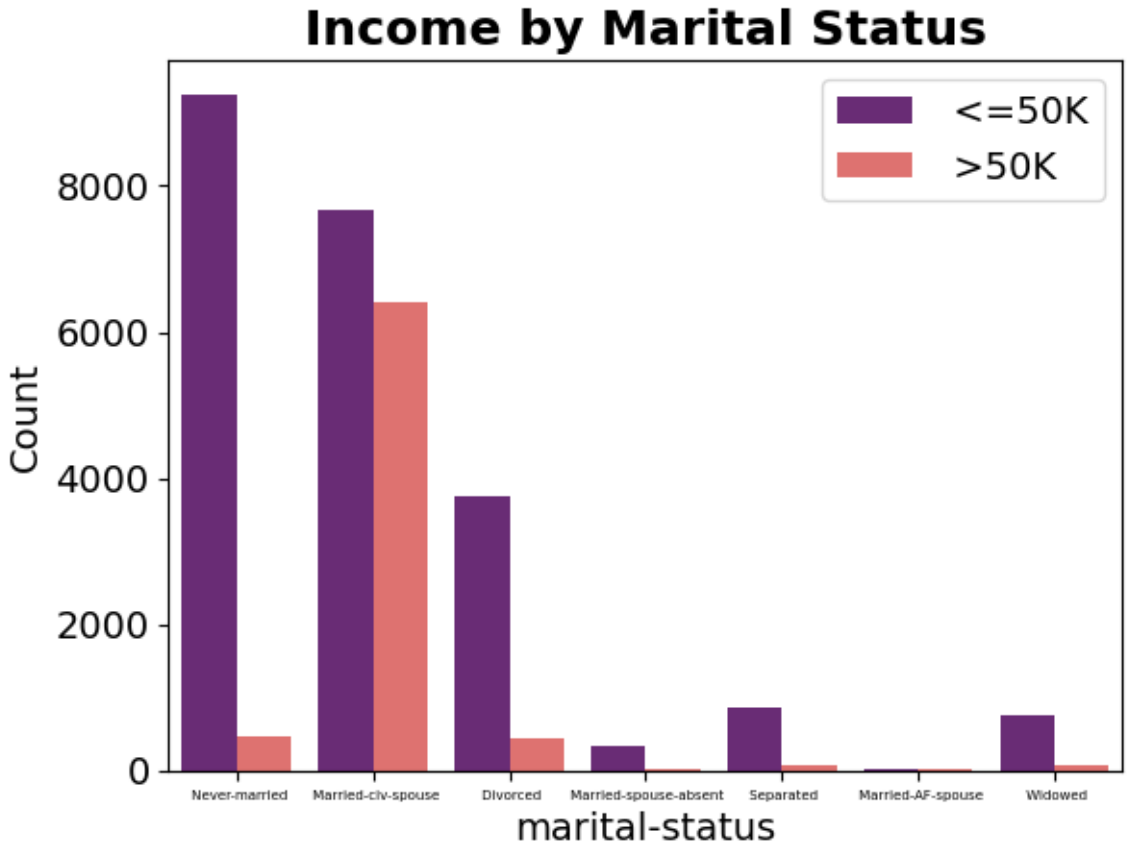
[6] Everything you need to know about min-max normalization: A python tutorial. https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79. Accessed: 2023-04-07.

[7] How to understand and compare box plots. https://mathsathome.com/understand-and-compare-box-plots/. Accessed: 2023-04-07.

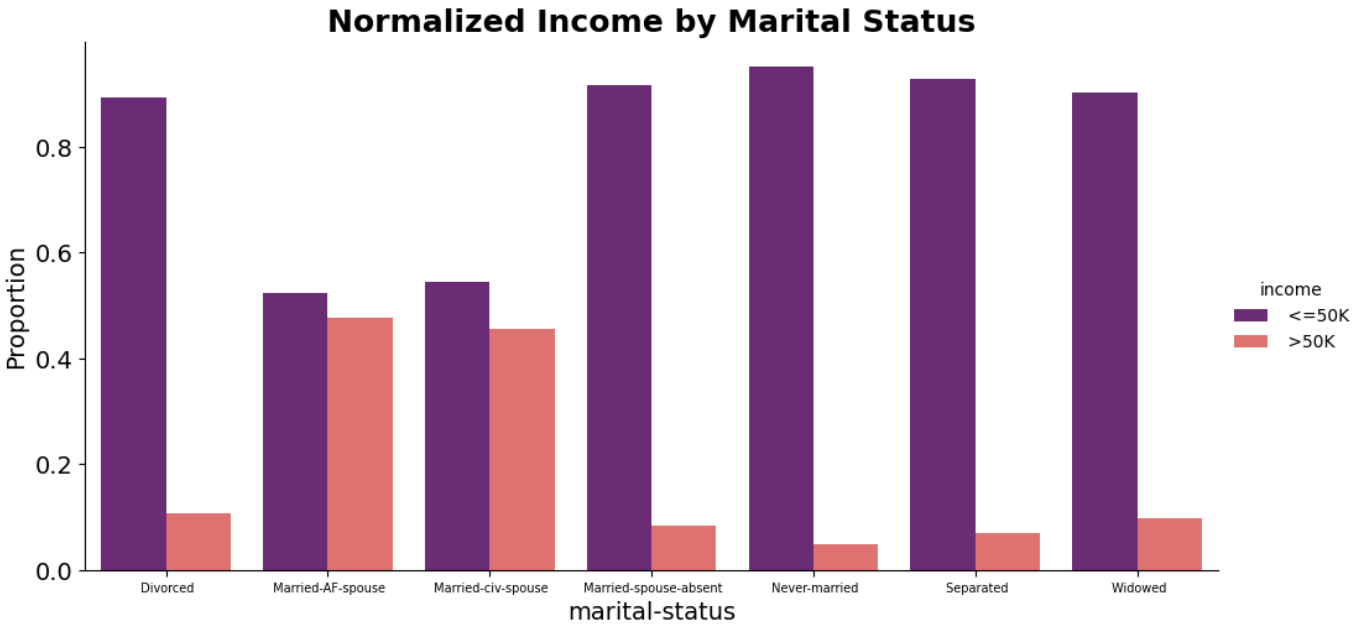
[8] Learning Curve to identify Overfitting and Underfitting in Machine Learning. https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5. Accessed:2023-04-07.

[9] Long short-term memory (lstm) in keras. https://pythonalgos.com/long-short-term-memory-lstm-in-keras/. Accessed: 2023-04-07.

[10] Mahalanobis distance: Simple definition, examples. https://www.statisticshowto.com/mahalanobis-distance/. Accessed: 2023-04-07.
[11] MLP-UCI-Adult-dataset. https://github.com/mrazizi/MLP-UCI-Adult-dataset/blob/master/mlp_uci_adult_dataset.ipynb. Accessed:2023-04-07.
[12] sklearn.ensemble.RandomForestClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: 2023-04-07.
[13] sklearn.metrics.classification_report. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html. Accessed: 2023-04-07.
[14] sklearn.metrics.confusion_matrix. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. Accessed: 2023-04-07.
[15] sklearn.metrics.roc_curve. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html. Accessed:2023-04-07.
[16] sklearn.model_selection.GridSearchCV. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed:2023-04-07.
[17] sklearn.neural_network.MLPClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Accessed: 2023-03-27.
[18] What is random forest? https://www.ibm.com/topics/random-forest. Accessed: 2023-04-07.
[19] What is the interquartile range rule? https://www.thoughtco.com/what-is-the-interquartile-range-rule-3126244. Accessed: 2023-04-07.
[20] Winsorize: Definition, examples in easy steps. https://www.statisticshowto.com/winsorize/. Accessed: 2023-04-07.
[21] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 161–168, 2004.
[22] Felix Gers. *Long Short Term Memory in Recurrent Neural Networks*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.
[23] Arthur S. Goldberger. *A Course in Econometrics*. Harvard University Press, Cambridge, MA, USA, 1991.
[24] Investopedia. Multicollinearity: Meaning, Examples, and FAQs. https://www.investopedia.com/terms/m/multicollinearity.asp. Accessed:2023-04-07.
[25] Wei-Chun Kao, Kai-Min Chung, Chia-Liang Sun, and Chih-Jen Lin. Decomposition methods for linear support vector machines. *Neural Computation*, 16(8):1623–1640, 2004.
[26] K. M. Kim and C. J. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 19(10):1888–1898, 2008.
[27] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Palo Alto, California USA, 1996. AAAI Press.
[28] Machine Learning Mastery. Why One-Hot Encode Data in Machine Learning? https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Accessed:2023-04-07.
[29] Cory Maklin. Synthetic Minority Over-sampling Technique (SMOTE). https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c. Accessed:2023-04-07.
[30] Minitab. Interpret all statistics and graphs for Correlation. https://support.minitab.com/en-us/minitab/21/help-and-how-to/statistics/basic-statistics/how-to/correlation/interpret-the-results/all-statistics-and-graphs/. Accessed:2023-04-07.
[31] scikit-learn. sklearn.preprocessing.LabelEncoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed:2023-04-07.
[32] Scribbr. Chi-Square Test of Independence: Formula, Guide and Examples. https://www.scribbr.com/statistics/chi-square-test-of-independence/. Accessed:2023-04-07.
[33] TOWARDS DATA SCIENCE. SMOTE. https://towardsdatascience.com/smote-fdce2f605729. Accessed:2023-04-07.
[34] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 114–121, 2004.

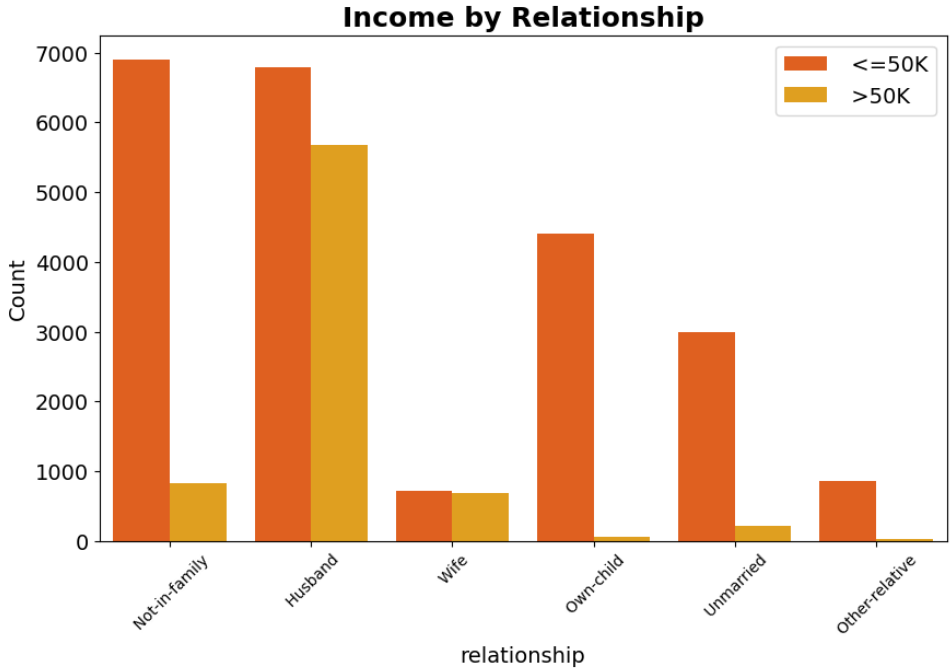


(a) Income Distribution by Marital Status (Not-Normalized)

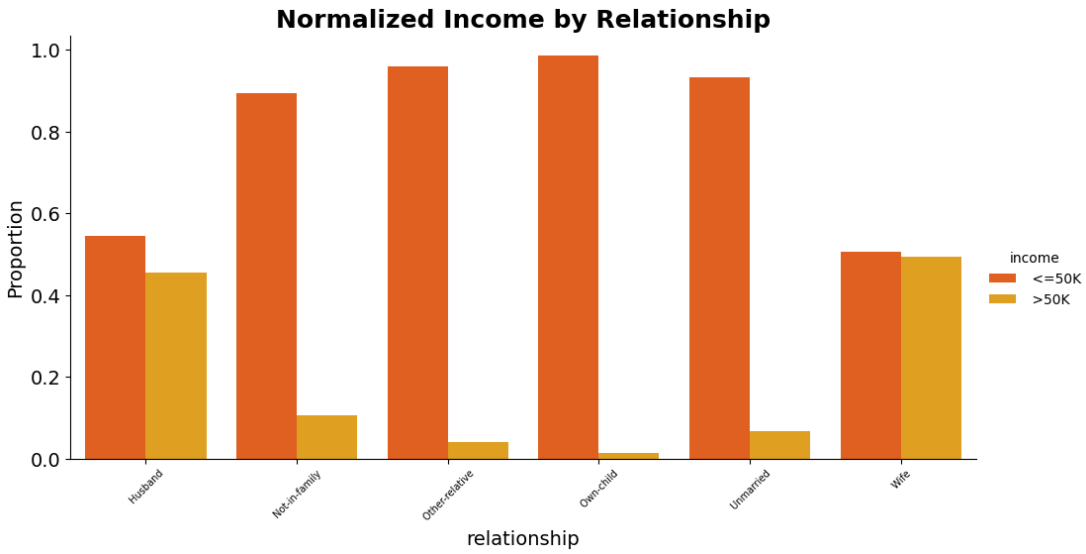


(b) Income Distribution by Marital Status (Normalized)

Fig. 39. Income Distribution by Marital Status



(a) Income Distribution by Relationship (Not-Normalized)



(b) Income Distribution by Relationship (Normalized)

Fig. 40. Income Distribution by Relationship.

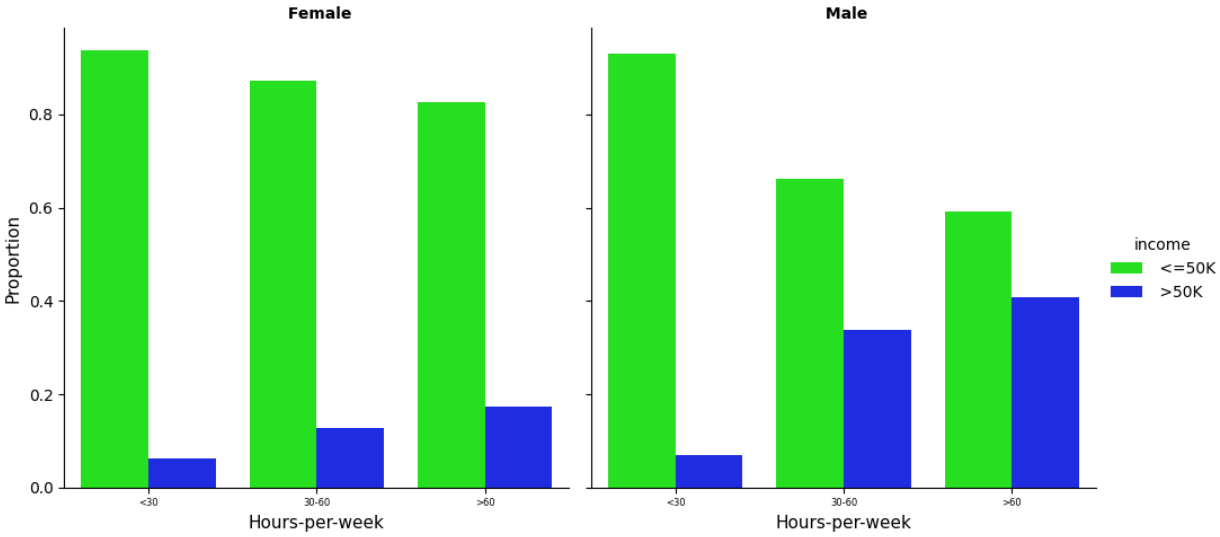


Fig. 41. Income Distribution by Hours per Week and Gender.

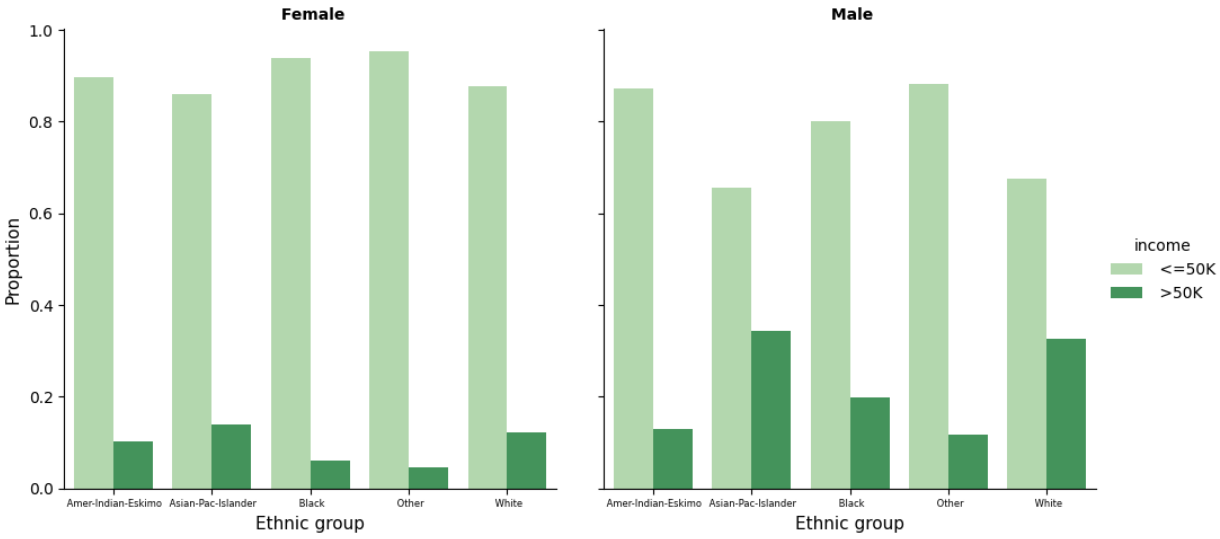


Fig. 42. Income Distribution by Ethnic Group and Gender.

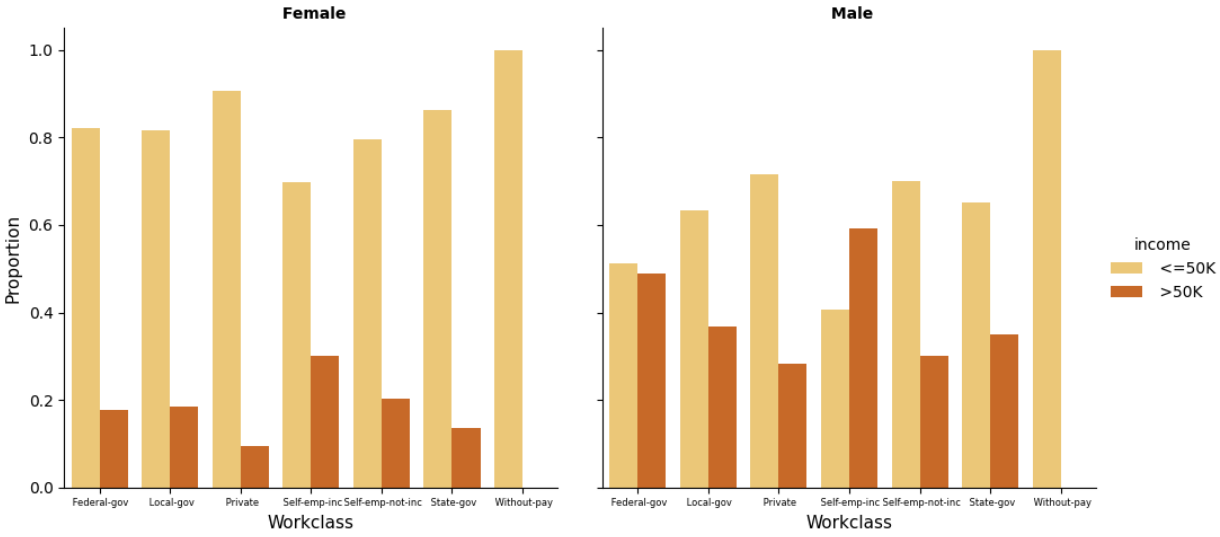


Fig. 43. Income Distribution by Workclass and Gender.

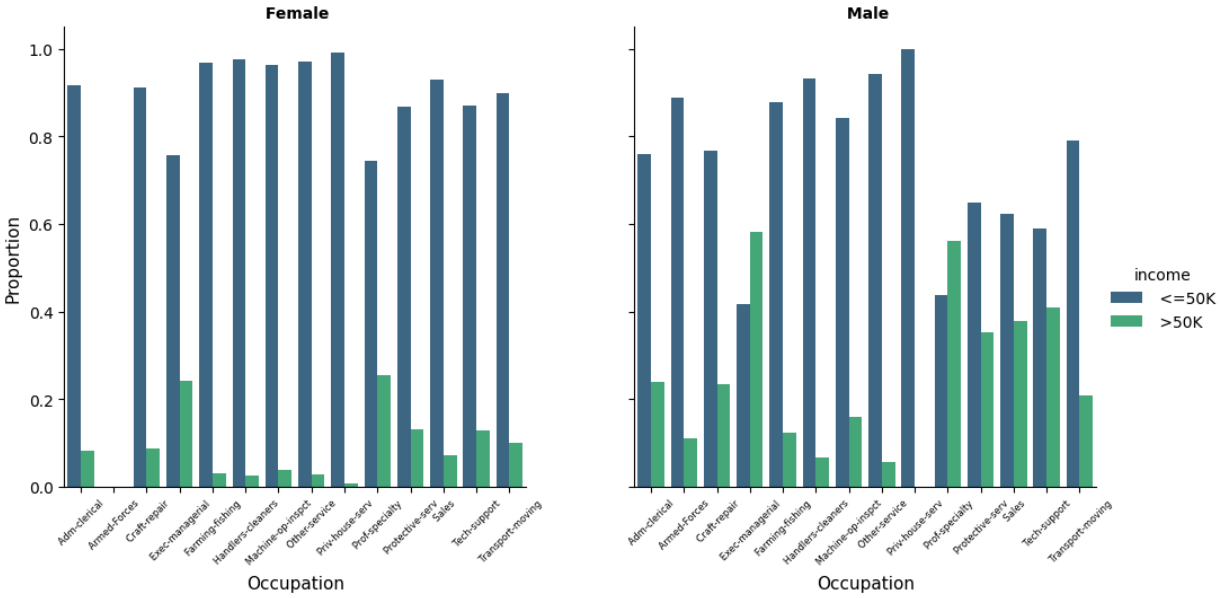
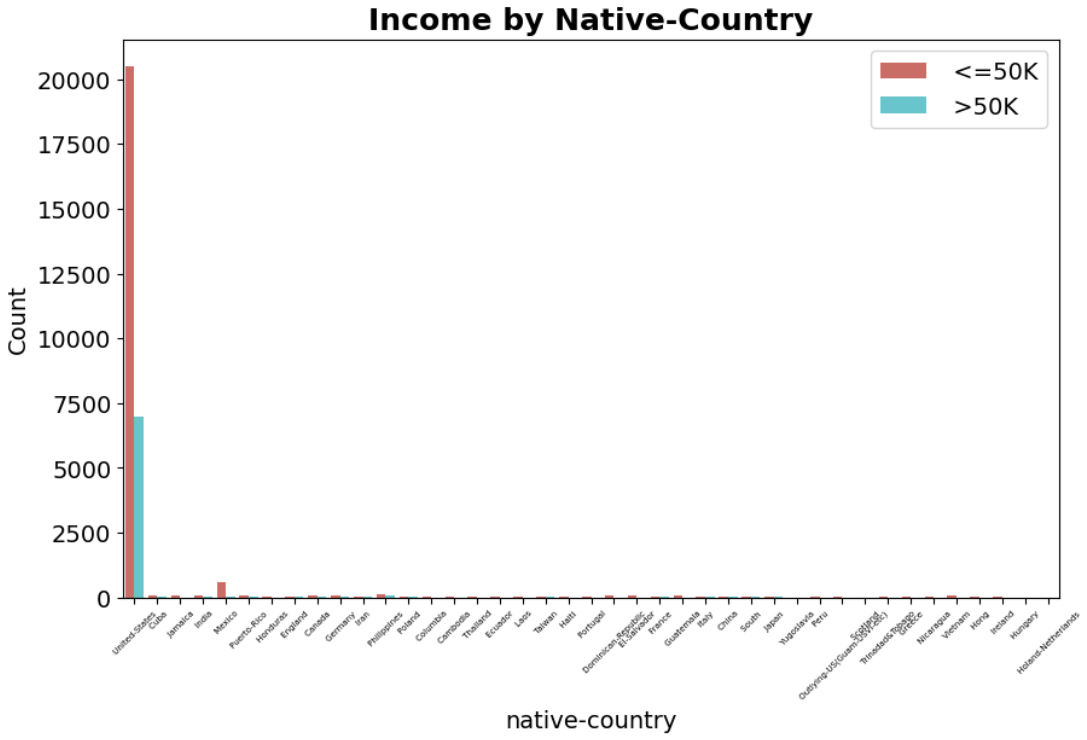
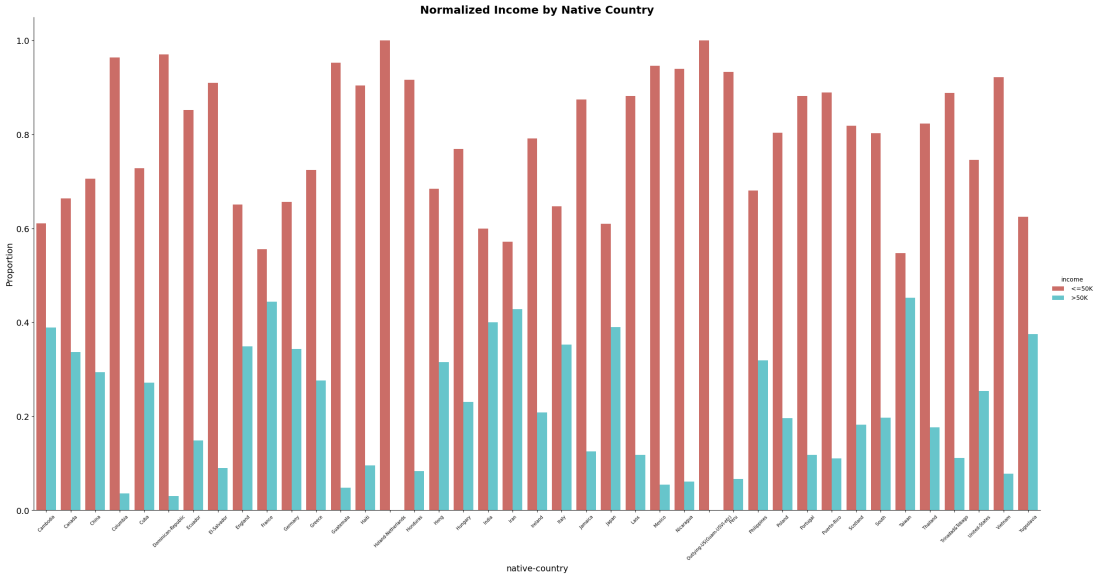


Fig. 44. Income Distribution by Occupation and Gender.



(a) Income Distribution by Native Country (Not-Normalized)



(b) Income Distribution by Native Country (Normalized)

Fig. 45. Income Distribution by Native Country.

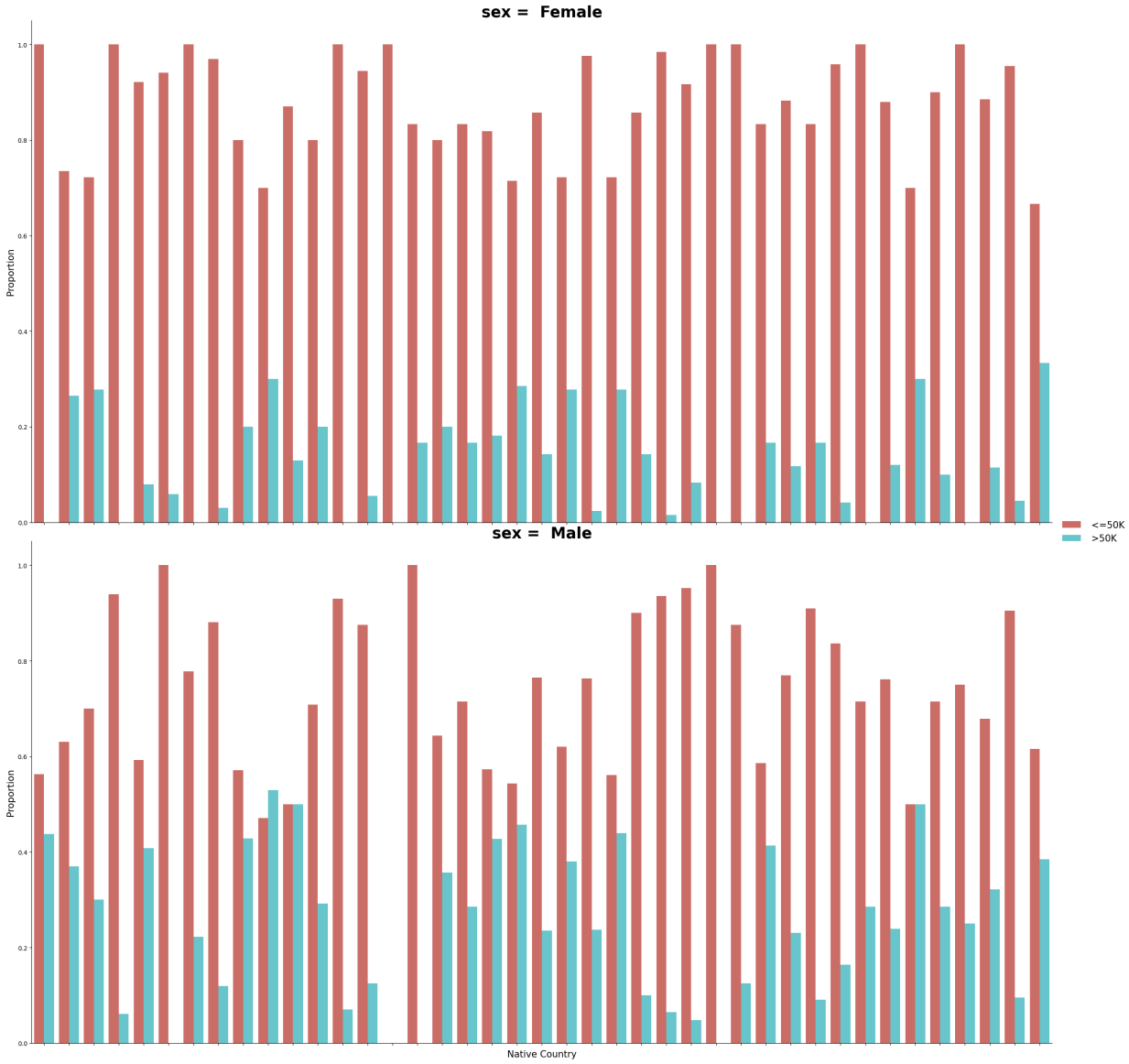


Fig. 46. Income Distribution by Native Country and Gender.

A. *Hyperparameter tuning using GridSearchCV (1st try)*

In our first try, we searched for the following param_grid:

- 'n_estimators': [10, 50, 100]
- 'max_depth': [None, 10, 20]
- 'min_samples_split': [2, 5, 10]
- 'max_features': ['auto', 'sqrt', 'log2']

And we got, for the plots of the mean score per hyperparameter:

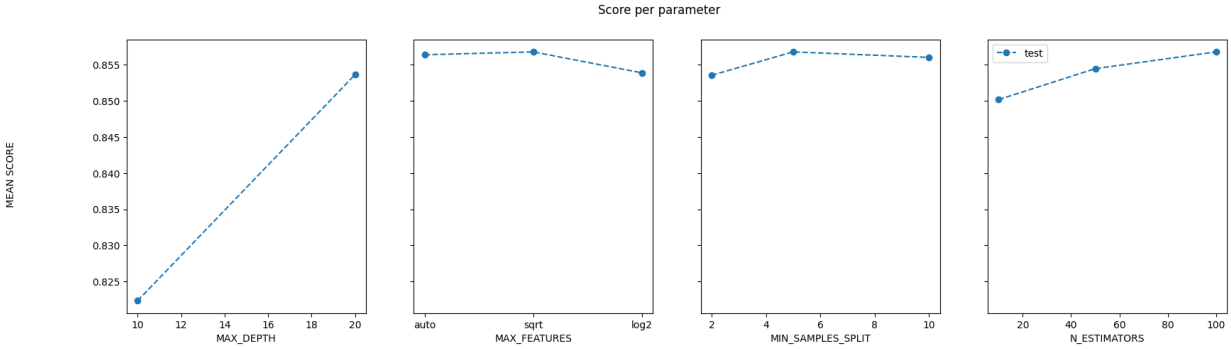


Fig. 47. Mean Score per hyperparameter (1st try).

We can see from the figure 47 of the model score as a function of the hyperparameters that the there's still room for improvement for the max_and n_estimators hyperparameters. We will now search for different values of these hyperparameters using a grid search again. We will inscrease the values for the max_depth hyperparameter first.

B. Hyperparameter tuning using GridSearchCV (2nd try)

So, now, we searched for the following param_grid:

- 'n_estimators': [10, 50, 100]
- 'max_depth': [20,30,40]
- 'min_samples_split': [2, 5, 10]
- 'max_features': ['auto', 'sqrt', 'log2']

And we got, for the plots of the mean score per hyperparameter:

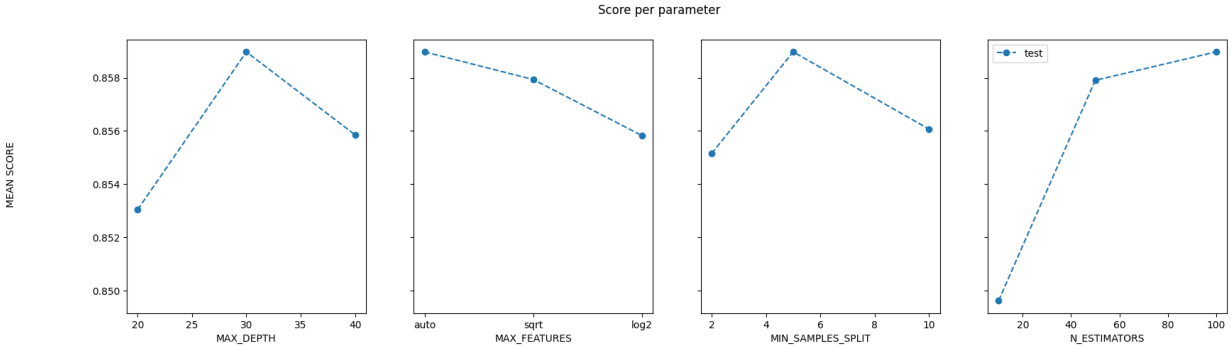


Fig. 48. Mean Score per hyperparameter (2nd try).

Again, we could see from the plot of figure 48 that there's still room for improvement for the n_estimators hyperparameter. We will increase the range for the max_depth hyperparameter.

C. Hyperparameter tuning using GridSearchCV (3rd try)

So, now, we searched for the following param_grid:

- 'n_estimators': [10, 50, 100]
- 'max_depth': [None, 10, 20,30,40]
- 'min_samples_split': [2, 5, 10]
- 'max_features': ['auto', 'sqrt', 'log2']

And we got, for the plots of the mean score per hyperparameter:

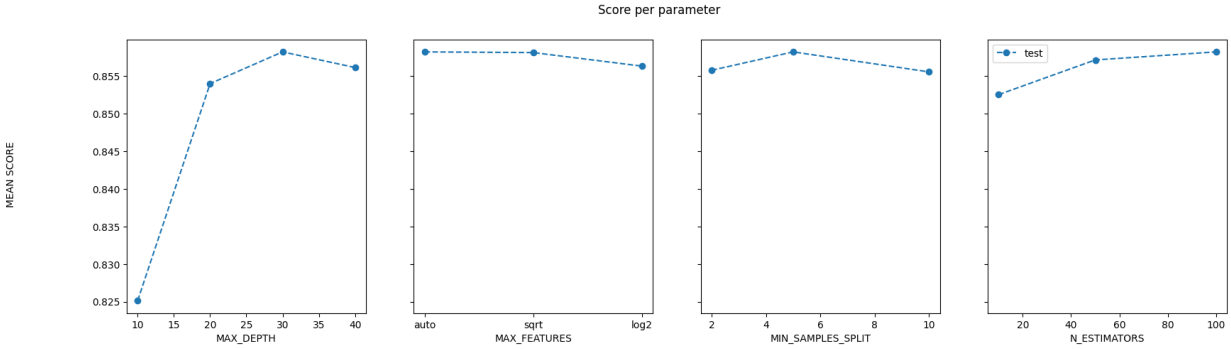


Fig. 49. Mean Score per hyperparameter (3rd try).

D. GridSeachCV for dataset without outliers

For the error rate in function of the number of trees we get the following plot:

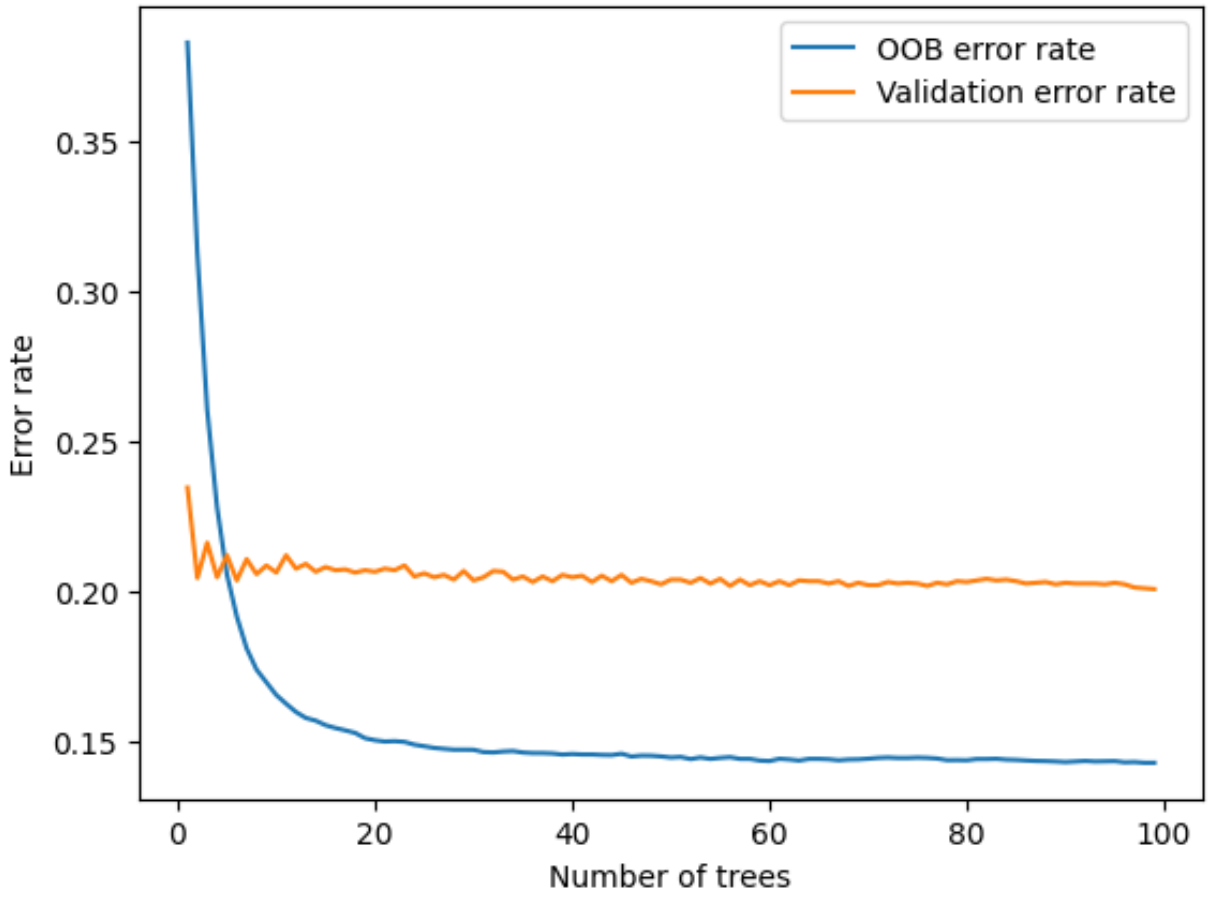


Fig. 50. Error rate Vs. Number of trees (4th try).

Looking at the error rate as a function of the number of trees of figure 50, we can see that the error rate decreases as the number of trees increases. However, the error rate does not decrease monotonically. Instead, the error rate decreases rapidly at first and then levels off. This is because the random forest is able to reduce the error rate by adding more trees, but only up to a certain point. After a certain number of trees, the error rate does not decrease any further. The validation error decreases a little but it is always higher than the training error. This is a sign of overfitting. (We can see it is pretty similar to the plot we got when we add the dataset with the outliers).

We tried again some hyperparameter tuning and we got the exact same response as for the dataset with outliers. Here are the plots of the mean score Vs. the hyperparameter (the hyperparameters tested were the same as the ones used when tuning for the dataset with outliers, in the same order):

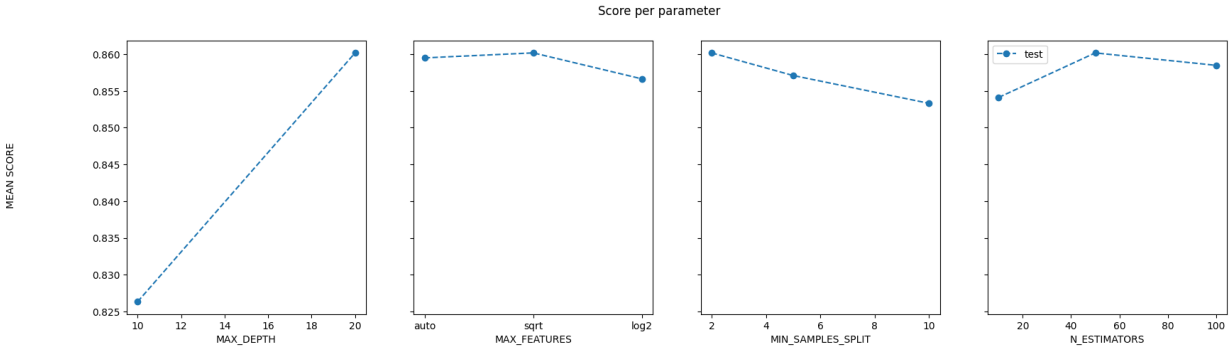


Fig. 51. Mean Score Vs. Hyperparameter without outliers (1st try).

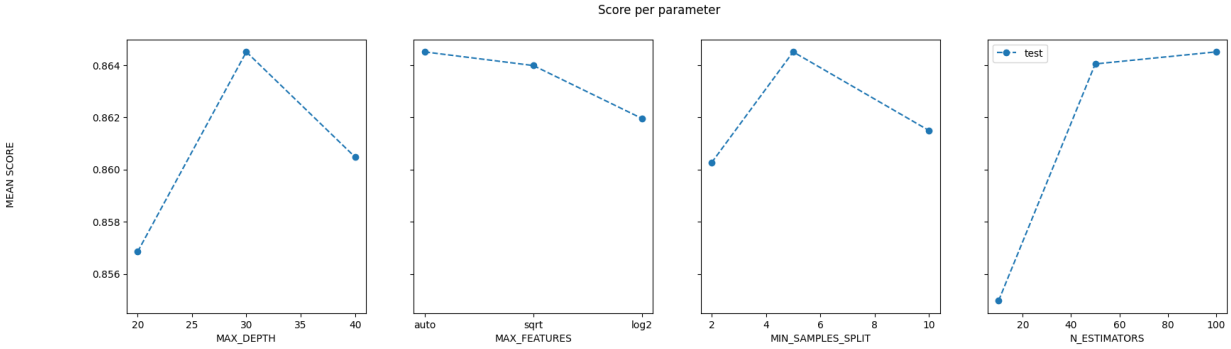


Fig. 52. Mean Score Vs. Hyperparameter without outliers (2nd try).

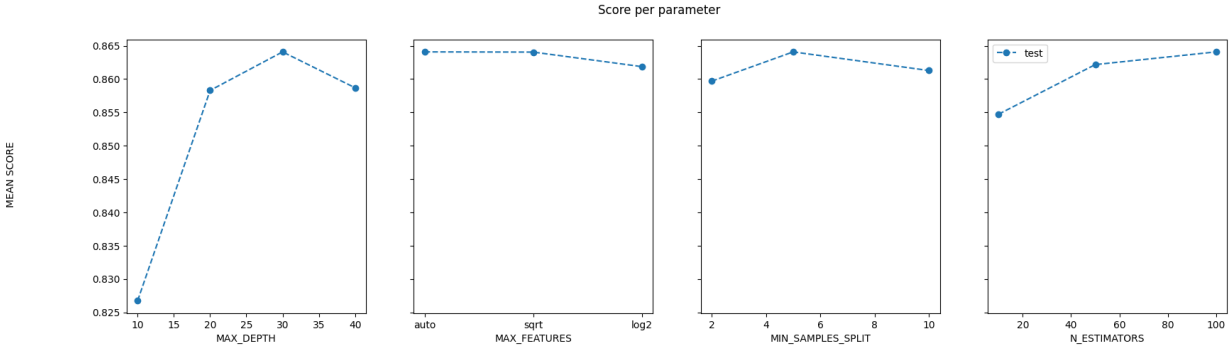


Fig. 53. Mean Score Vs. Hyperparameter without outliers (3rd try).