MACHINE LEARNING
Group Project

Class prediction

for Newland citizens

Master Degree Program in Data Science and Advanced Analytics

2020/2021

Ana Luísa Mestre | Lorenzo Pigozzi | Mariana Domingues | Rebeca Pinheiro

Contents

1. Abstract

In this project, we will analyze the Newland dataset to complete a prediction for the target variable, Income. The classification will be used for the tax policy that it is going to be applied in the new planet discovered, called Newland.

In the preprocessing phase, we will explore some techniques, such as One-Hot-Encoder, KNN-Imputer, Mutual Information Classifier, and others, in order to prepare and explore the data.

In the core part of the project, we are going to focus on the creation of different models, comparing different solutions and using different approaches to get the best performance for each possibility considered.

Step by step, the final model will come to life and take shape: The Hist-Gradient Boosting.

We will explore this new algorithm and predict the target that in the future could decide the tax-system of Newland.

Keywords: Machine Learning, Classification, MIC, Hist-Gradient Boosting, F1 score

2. Introduction

In 2044, due to the worrying and accelerated climate changes on planet Earth, all forms of life are in danger. Fortunately, in 2039, a planet with similar life conditions to those of planet Earth was discovered and a mission (Newland) to dwell it started being developed. Nine years later, in order to increase the financial sustainability of the planet, it was decided that residents should start paying taxes; thus, a binary tax rate was applied: for people with an income below or equal to the average the rate would be 15% of their income, and for the remaining people 30%.

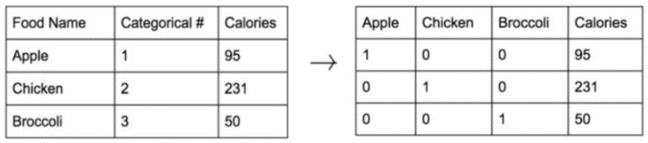
In this project, we are going to create a predictive model based on a dataset of 22400 observations (the train dataset) to predict if the income of each person is below or equal (value 0) or above (value 1) the average. Then, this model will be applied to a new dataset of 10100 observations (the test dataset), of which 30% of them will be tested in Kaggle, to check its performance on unseen data.

3. Background

3.1. One Hot Encoder

One of the major machine learning problems is that several machine learning algorithms are not able to work with categorical variables. To solve that, categorical variables can be represented as binary vectors by one hot encoding method. This method allows a better representation of categorical data by converting them into numbers.

The one hot encoding method takes the categorical data column and split it into multiple columns replacing the data for 1s and 0s, where 1 represents the affirmative value and 0 the negative. Consider the dataset below, where it is possible to notice that the categories that were rows are now splitted into columns and the numerical data stayed in the same position. It was added a binary column for each categorical variable.



Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

$\rightarrow$

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Fig. 1 – One Hot Encoder

3.2. Mutual Information Classifier (MIC)

Many machine learning problems consist in predicting future values given a database. However, this can imply a big number of correlated features, of which some are irrelevant to the analysis and the prediction.

Knowing that, features must be selected to provide the right information and to avoid problems like overfitting, reducing the model's complexity. One possible way to do that is by Mutual Information, which consists in first joining similar features and then selecting groups of them instead of individual ones. The mutual information of two random variables X and Y consists in reducing the uncertainty between them. It can also be written as:

$$MI(X,Y) = \int \mu_{X,Y}(x,y) \log \frac{\mu_{X,Y}(x,y)}{\mu_X(x)\mu_Y(y)} dxdy$$

where µX(x) and µY(y) are the probability of density functions of X and Y; and µX,Y(x, y) is the joint distribution of the variables X and Y. The Mutual Information Classifier can provide a solution in an objective way, trough the balance among error and reject types. The equation from mutual information classifier derivates from the maximization of mutual information over all pattern as described below:

$$y^{+} = arg \max_{y} NI(T = t, Y = y)$$

where T is the target, Y the decision output and y+ represents an optimal classifier. It is important to refer that mutual information classifier describe the relative entropy between two random variables and provide a better perspective for a processing classification problem.

3.3. Histogram Gradient Boosting

One of the principal machine learning tasks is to build a strong model able to predict future variables. There are different ways to do that and ensemble existent models is one of them. However, in practice, doing this consists in combining a lot of relatively weak models to build one strong predictive model. The boosting method works in a different strategy way of ensemble, which is adding new models for ensemble in sequence. For a particular iteration, a new base-learner model is trained according to the error of the ensemble made until then.

The principal purposes of Histogram Gradient Boosting are reducing time to find the best split, solving missing value problems eliminating imputer need and improving the accuracy.

Quoting the paper "A memory efficient gradient boosting tree system on adaptive compact distributions" of Yingshi Chen (link in the references):

"Considering a dataset with N samples and M features, K bin histogram can reduce the tree- split algorithm complexity from O(NM) to O(KM), being K = 256 (or 64,512..)."

This improve the accuracy because once upon one element is put in some bin, for all related operation it will be used the distribution parameter as new distribution to represent original features, and the statistic value of a unique bin is a regulation to all bin's values, doing a smooth filter and reducing the noise.

4. Methodology

4.1. Data and Variables Description

For this project's elaboration, two datasets are provided: one to train the models (the train dataset) and the other one to predict the values of the target, Income, in two classes, 0 and 1, as already explained. The output predictions are submitted on Kaggle in order to have an idea of the performance based on the Micro F1 Score.

The train dataset has 22400 observations and 15 variables, of which 14 are independent variables and 1 is the dependent variable, that are described in the following table.

VARIABLE	DESCRIPTION	TYPE
Citizen_ID	Unique identifier of the citizen	int64
Name	Name of the citizen (First name and surname)	object
Birthday	The date of Birth	object
Native Continent	The continent where the citizen belongs in the planet Earth	object
Marital Status	The marital status of the citizen	object
Lives with	The household environment of the citizen	object
Base Area	The neighborhood of the citizen in Newland	object
Education Level	The education level of the citizen	object
Years of Education	The number of years of education of the citizen	int64
Employment Sector	The employment sector of the citizen	object
Role	The job role of the citizen	object
Working Hours per week	The number of working hours per week of the citizen	int64
Money Received	The money payed to the elements of Group B	int64
Ticket Price	The money received by the elements of Group C	int64
Income	The dependent variable (Where 1 is Income higher than the average and 0 Income Lower or equal to the average)	int64

Table 1 – Variable Description and Type

4.2. Imports

Our first step is to import the needed libraries: some general ones, such as Pandas, Numpy, Seaborn, Matplotlib; and Sklearn, in order to apply the algorithms that we are going to use in this project, as Logistic Regression, KNN, Neural Networks, Decision Tree, Random Forest, Gradient Boosting, Support Vector Machine, Hist Gradient Boosting, Ada Boost, Bagging, Stacking and others. After that, we import the train dataset and store it in the object newland.

4.3. Data Exploration

Secondly, we start exploring and trying to understand the data we have, identifying possible problems and errors and getting major insights. In this step, we use the methods head(), info() and describe(include='all') to have a general view of the data, including the values of the variables, their data types, if there are missing values and, also, check the descriptive statistics of the numeric and non-numeric variables.

4.3.1. Descriptive Statistics

To get insights from our data, we decide to create two tables with descriptive measures of our features: one with the categorical variables and the other one with the numerical variables, as the measures presented are different depending on the attribute types.

VARIABLE	MIN	P25	P75	MAX	MEAN	STD	NaNs
Years of Education	2	12	15	21	13.173884	2.512451	0
Working Hours per week	1	40	45	99	40.483795	12.370921	0
Money Received	0	0	0	122999	1324.915357	9227.771813	0
Ticket Price	0	0	0	5358	109.145313	500.208904	0
Income	0	0	0	1	0.237098	0.425313	0

Table 2 – Descriptive Statistics for the Numerical Variables

Looking at the information presented above, we can get some conclusions about our numeric variables. Some of them look like they contain outliers, due to the difference between the minimum value and the percentile 25 or between the maximum value and the percentile 75. For example, for Money Received, the percentile 75 is 0 while the maximum value is 122999, meaning that we probably have upper candidates to outliers. Another example is the variable Ticket Price, where the percentile 75 is also 0 and the maximum value is 5358.

In the further step, analyzing better the variables, we can realize that those who seemed to be outliers are actually providing important information to build the predictive model because they are highly correlated.

VARIABLE	NaNs	UNIQUE CATEGORIES	MODE	MODE FREQUENCY (%)
Name	0	16074	Mr. James King	0.018
Birthday	0	11257	August 15,2025	0.040
Native Continent	0	5	Europe	85.317
Marital Status	0	7	Married	45.621
Lives with	0	6	Wife	40.232
Base Area	0	40	Northbury	89.616
Education Level	0	16	Professional School	32.286
Employment Sector	0	9	Private Sector - Services	69.638
Role	0	15	Professor	12.719

Table 3 – Descriptive Statistics for the Categorical Variables

In most of the categorical variables, we can see that the frequency of a certain category is very high. For example, for the variable Base Area, the category Northbury is referenced in 89.616% of the observations.

4.4. Data Preparation

After the exploration and understanding of our data, we move to the data preparation. In this step, we are going to transform our data, create new variables with higher predictive power, check the coherences, fill the missing values, and remove some outliers.

4.4.1. Creating new variables

In this step, we use the variable Name to create a new one, Gender: if the feature Name begins with “Mr.”, then the variable Gender has value 0; if it begins with “Mrs.” or “Miss”, then the variable Gender has value 1. We also use the variable Birthday to create Age: we store the year of birth of each observation in a column named year_of_birth, which contains the last four digits of the variable Birthday and then we subtract these four digits to our ‘present’ year (2048) and store that value in the variable Age. At the end, we remove the auxiliar column year_of_birth and the original columns Name and Birthday.

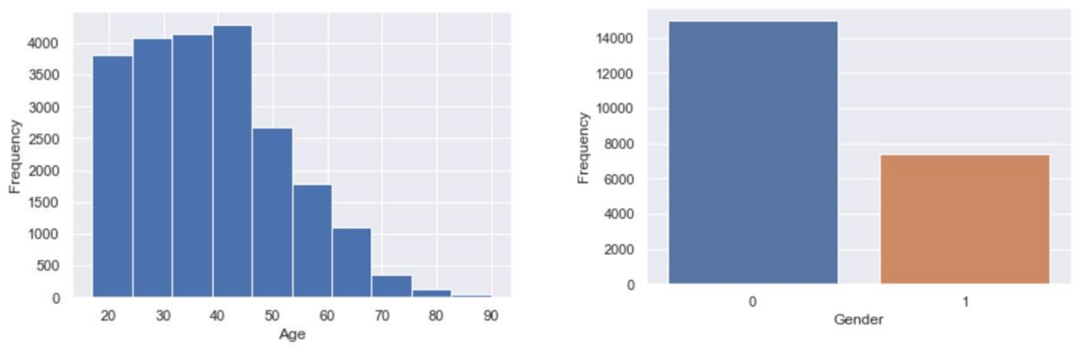


Fig. 2 – Distributions of the new variables, Age (left) and Gender (right)

We check the minimum value of the variable Age and we get 17, which is coherent to the indications of our dataset in the project presentation’s document.

4.4.2. Encoding Categorical Variables

Some variables seem to have a few values with a really low frequency. Exploring deeply this aspect, we end up with the choice of re-distribute some of these features, which present a really unbalanced distribution.

To solve this problem, we apply the following changes to the values of these variables:

- In Education Level, we order the values from the easiest (Primary School, level 1) to the most complex degree of education (PhD, level 16);
- In Native Continent, we replace by Other the three continents that have lower frequencies (Asia, America and Oceania) and keep the remaining ones with higher frequencies (Africa and Europe);
- In Marital Status, we replace the four names that have lower frequencies with Other and keep the ones with higher frequencies (Married, Single, Divorced);
- In Lives with, we join the observations that live in couple replacing Wife and Husband with Wife / Husband, who are the people who live with their spouses;

- In Base Area, we replace everything, except Northbury, with Other, since the observations that didn't have Northbury as category were very few, as we already saw before (in the descriptive statistics section);
- In Employment Sector, we replace the five names with lower frequencies with Other and keep the remaining ones with higher frequencies (Private Sector – Services, Public Sector – Others, Self-Employed (Individual)).

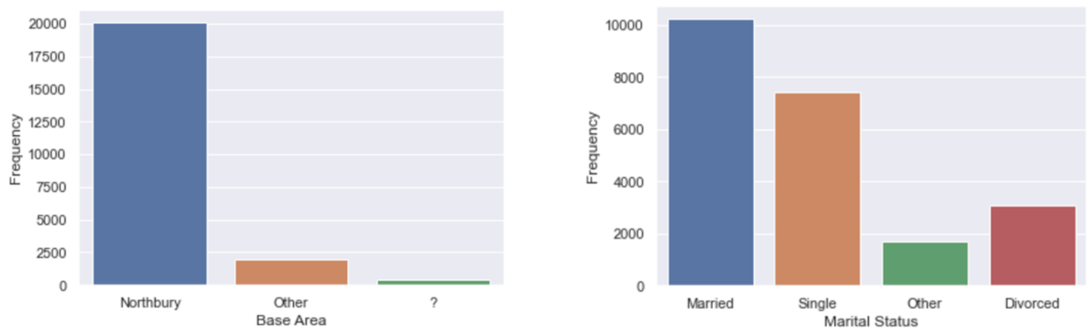
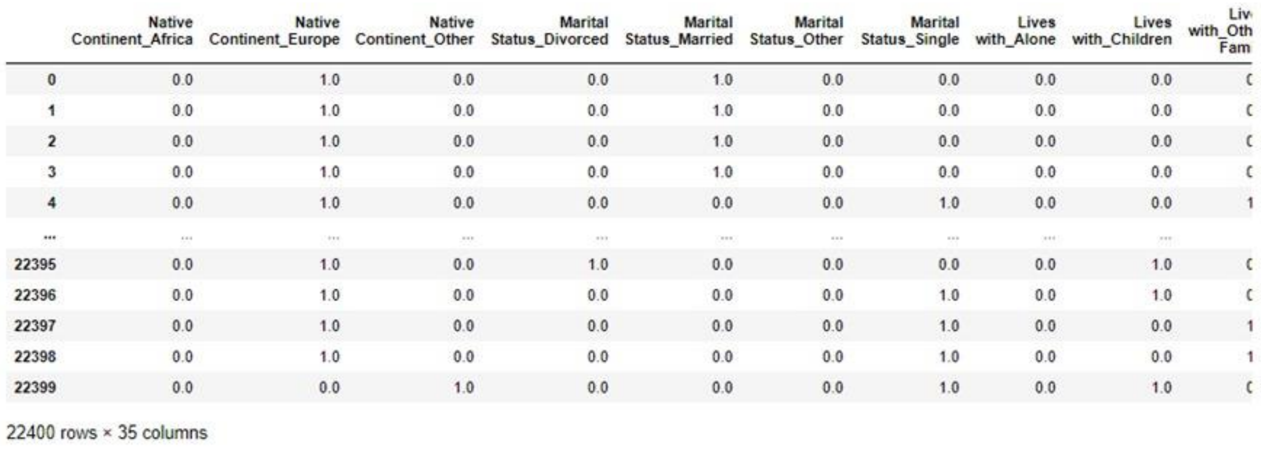


Fig. 3 – Distributions of some of the variables encoded, Base Area (left) and Marital Status (right)

Now we only have Native Continent, Marital Status, Lives with, Base Area, Employment Sector and Role in non-numeric type. So, we apply One Hot Encoder to these variables.

Applying this type of encoder, we end up getting 40 independent variables; for each one of the previous categorical variables, now we have as many features as the number of values for that variable. The plot below gives a better idea of this transformation.



	Native Continent_Africa	Native Continent_Europe	Native Continent_Other	Marital Status_Divorced	Marital Status_Married	Marital Status_Other	Marital Status_Single	Lives with_Alone	Lives with_Children	Liv with_Oth Fam
0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	(
1	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	(
2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	(
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	(
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1
...	...	...	...	...	...	...	...	...	...	
22395	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	(
22396	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	(
22397	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1
22398	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1
22399	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	(

22400 rows × 35 columns

Fig. 4 – Categorical variables encoded

4.4.3. Filling Missing Values

Regarding the missing values, exploring more deeply the train dataset, we conclude that we do not directly have NaNs, but we have question marks ('?').

The variables that have question marks are: Base Area (395 values); Employment Sector (1264 values) and Role (1271 values).

Due to the one-hot-encoder applied, the algorithm treats the question mark as a value, thus in the new dataset we get these three variables: Base Area_?, Employment Sector_?, Role_?.

The interpretation of an observation with a positive value for one of these 3 variables is the missing information for the original categorical variable.

To solve this problem, for the observations with Base Area_? == 1, we set the NaN for the variables:

- Base Area_Northbury
- Base Area_Other

For the observations with Employment Sector_? == 1, we set the NaN for the variables:

- Employment Sector_Other
- Employment Sector_Private Sector – Services
- Employment Sector_Public Sector – Others
- Employment Sector_Self-Employed (Individual)

For the observations with Role_? == 1, we set the NaN for the variables:

- Role_Administratives
- Role_Agriculture and Fishing
- Role_Army
- Role_Cleaners & Handlers
- Role_Household Services
- Role_IT
- Role_Machine Operators & Inspectors
- Role_Management
- Role_Other services
- Role_Professor
- Role_Repair & constructions
- Role_Sales
- Role_Security
- Role_Transports

After that, we remove the variables with the question mark: Base Area_?, Employment Sector_? and Role_?.

To maintain the value and the variability, and to not waste valuable data of our dataset, we decide to apply the KNN Imputer to fill the missing values in the three encoded variables (Employment Sector, Base Area and Role), which is a more accurate and efficient approach than filling them with the mode, in this case.

Thus, we use 5 neighbors and the default values for the parameters weights (=uniform) and metric (=nan_euclidean). Also, we decide to use all the remaining variables to create the classifier. We move on filling the 395 missing values related to the Base Area, the 1264 missing values related to Employment Sector and the 1271 missing values related to Role.

4.4.4. Outliers

Regarding the outliers, we use the IQR method with the values 0.99 and 0.01 for the quantiles, in the variables Years of Education and Working Hours per week because, in our opinion, removing values in these ones affects too much the variability of the dataset (contrary to what happened in the variables Ticket Price, Money Received, Age and Education Level; Education Level and Years of Education are highly correlated, so we think that we should not delete outliers in both).

The variables Ticket Price and Money Received, as we analyzed before, have strange distributions; more than 75 % of the observations have value equal to 0.

The reason is that we do not have a lot of observations that paid the ticket and neither that are paid to take part to the mission.

Considering that, to do not lose the information that these variables provide, we are not going to treat them as outliers, so we are not going to remove the observations with positive values for Money Received and Ticket Price.

In this sense, we end up removing 2.24% of our dataset, which is an acceptable percentage according to most of the studies regarding data treatment.

Before moving to the next big step of the project (Modelling) we also define as index of the dataset the column CITIZEN_ID and then we select our variables to enable the machine learning algorithm to train faster, reduce the complexity of our model and improve our score.

4.4.5. Feature Selection

For the feature selection, we start using the Pearson correlation matrix, from which we remove the variables Years of Education and Marital Status_Married. The first one because it is very correlated with the variable Education Level and also because if a person takes, for example, 6 years to graduate, it doesn't mean that he/she has so much knowledge; the second one because it is very correlated with the variable Lives with_Wife / Husband and, between these two, the one that is more correlated with the target is Lives with_Wife / Husband.

Then, we use the Mutual Information Classifier technique from the Sklearn for the encoded categorical features and here we choose the variables that have more than 0.001 of mutual information with the target. We end up selecting 21 variables, of which 20 are the independent variables and 1 is the dependent variable. Finally, we add the original numeric variables, ending with a total of 27 variables.

4.4.6. Split of the Dataset

To create a predictive model, we also need to split our data into at least two different datasets - the train and the validation. In this sense, first, we split the train dataset into the input (X, independent variables) and the output (Y, dependent variable) and then we split them into train-set (75% of the observations) and validation-set (25% of the observations), using the train_test_split().

4.4.7. Normalization

During the modelling phase (the next step), applying different models we notice that the consideration of different scalers influences the result.

Thus, we decide to train our models using three different approaches regarding the normalization: the robust scaler (X_train_robust, X_validation_robust), the standard scaler (X_train_std, X_validation_std) and without normalization – raw data – (X_train, X_validation).

To have an idea of the kind of explorations and considerations that we are doing for the scaler, we use a plot that shows the different Micro F1 Scores values for the KNN model with different scalers:

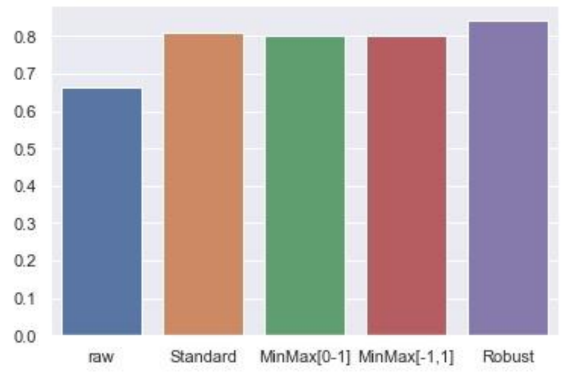


Fig. 5 – Micro F1 Scores for normalizations with KNN

In the next steps, we are going to apply different predictive models and explore which is the best one for this dataset.

5. Results and Interpretations

5.1. Modelling

Creating our models, we use the standard and the robust scalers because they are the two best ones in most of the models. Then, we also decide to give a chance to train our models without normalization to see if we have some improvements.

The first algorithm that we implement to create our first model is the Logistic Regression with the default parameters. To get the best parameters for the model, we apply the Grid Search. Then, we fit again the model with these parameters to the training data. With the model just created, we predict the target values for the validation dataset.

To improve our score, we apply the Bagging Classifier with the model of the algorithm mentioned above with the parameters discovered with the Grid Search.

We repeat these steps for each normalization approach (Robust, Standard and Raw) and, based on the results of all the algorithms that we implement, we choose the model with the highest score.

Regarding the Naive Bayes and the Support Vector Machine, we decide to do not apply the Grid Search or the Bagging Classifier because they are simpler models and the scores in the validation dataset with the Naive Bayes are not so good. However, as we did before, we select the highest of the three scores.

Implementing the Neural Networks, we find out that the Grid Search for this algorithm is really computational expensive, so we discard this option and, instead, we choose only one value for each parameter. The remaining steps, such as Bagging Classifier, are equal to the ones described above.

In the Decision Trees, in addition to the Grid Search and the Bagging Classifier, we also apply the Ada Boost Classifier to the model with the best parameters found by the Grid Search.

Then we implemented the K-Nearest Neighbors (with the default parameters), the Random Forest (with the parameters random_state = 15 and n_jobs = -1) and the Gradient Boosting (with the default parameters). The remaining steps, such as the Grid Search and the Bagging Classifier, are equal to the ones described above.

We also try to combine weaker learners in the Stacking Classifier, such as KNN, Logistic and Support Vector Machine. We see the normalization approach for the best score of each of these three models and we choose the standardized data when applying this model.

In the following table are represented the best scores found for each model of each implemented algorithm:

Best Model	Normalization Approach	Validation Score
Logistic Regression	Robust/Standard	84.785
Naive Bayes	Raw	82.301
KNN with Bagging	Raw	86.612
Neural Networks with Bagging	Standard	85.352
Decision Trees with Bagging	Raw	86.393
Random Forest	Robust/Raw	86.265
Gradient Boosting with Bagging	Raw	86.868
Support Vector Machine	Standard	84.694
Stacking	Standard	84.968

Table 4 – The best models

So, in the next steps, we are going to focus on the best model, the Gradient Boosting, and explore it in more detail, in order to improve it and get a better score.

5.2. Gradient Boosting

5.2.1. The Best Features

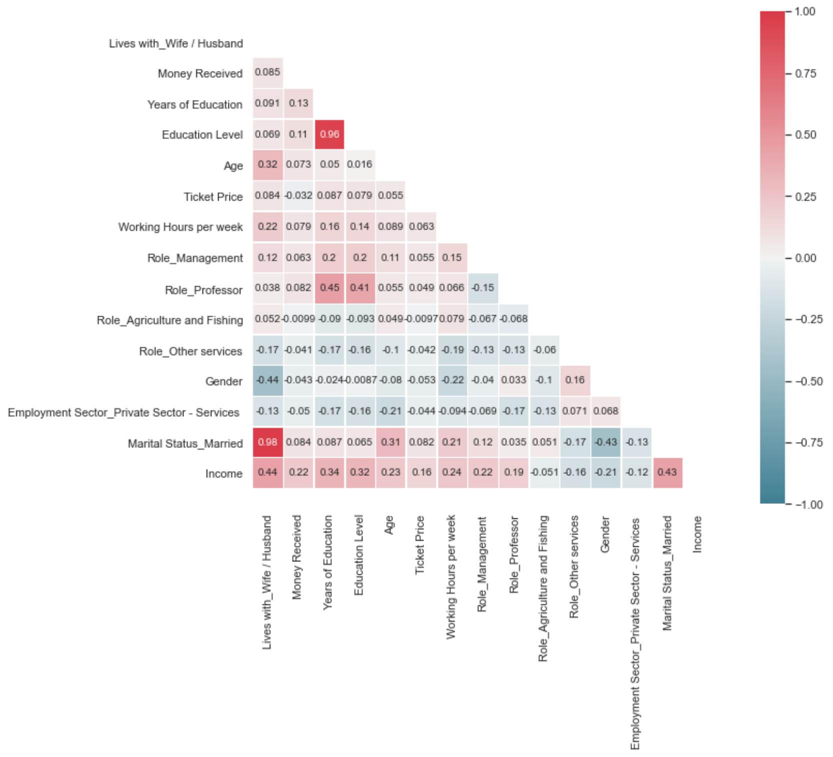
Considering these assumptions and the results obtained, we focus on Gradient Boosting, that is the best model found so far. The result does not surprise us too much, due to the popularity and good performances that this model is acquiring in the last years, becoming one of the most used algorithms for machine-learning predictions.

Focusing on it, we try to improve as much as possible the model.

Firstly, we focus on the best feature selection for this specific model, in order to give it the best features for the creation of the trees. Thus, reconsidering the previous choice that we did using the MIC, we explore deeply the relations between the features.

Running the algorithm with all the original features (categorical already encoded) and then also applying the Recursive Features Elimination (RFE), we notice that the variables Years of Education and Employment Sector – Self-Employed are also considered important by this algorithm. So, we re-insert these variables in our first selection (based on MIC), and then we check again the feature importance method that the Gradient Boosting provides.

We also use the Pearson correlation matrix to get a better idea of the correlations of all the variables with the target and the correlations among the independent variables themselves.



	Lives with_Wife / Husband	Money Received	Years of Education	Education Level	Age	Ticket Price	Working Hours per week	Role_Management	Role_Professor	Role_Agriculture and Fishing	Role_Other services	Gender	Employment Sector_Private Sector - Services	Marital Status_Married	Income
Lives with_Wife / Husband															
Money Received	0.085														
Years of Education	0.091	0.13													
Education Level	0.069	0.11	0.96												
Age	0.32	0.073	0.05	0.016											
Ticket Price	0.084	-0.032	0.087	0.079	0.055										
Working Hours per week	0.22	0.079	0.16	0.14	0.089	0.063									
Role_Management	0.12	0.063	0.2	0.2	0.11	0.055	0.15								
Role_Professor	0.038	0.082	0.45	0.41	0.055	0.049	0.066	-0.15							
Role_Agriculture and Fishing	0.052	-0.0099	-0.09	-0.093	0.049	-0.0097	0.079	-0.067	-0.068						
Role_Other services	-0.17	-0.041	-0.17	-0.16	-0.1	-0.042	-0.19	-0.13	-0.13	-0.06					
Gender	-0.44	-0.043	-0.024	-0.0087	-0.08	-0.053	-0.22	-0.04	0.033	-0.1	0.16				
Employment Sector_Private Sector - Services	-0.13	-0.05	-0.17	-0.16	-0.21	-0.044	-0.094	-0.069	-0.17	-0.13	0.071	0.068			
Marital Status_Married	0.98	0.084	0.087	0.065	0.31	0.082	0.21	0.12	0.035	0.051	-0.17	-0.43	-0.13		
Income	0.44	0.22	0.34	0.32	0.23	0.16	0.24	0.22	0.19	-0.051	-0.16	-0.21	-0.12	0.43	

Fig. 6 – Correlation Matrix

Looking at the correlation matrix above, we can easily see that there are 2 couples of variables that are really high-correlated among each other: Years of Education – Education Level and Lives with_WIife / Husband – Marital Status_Married.

To select between these variables, we consider again the feature importance and the correlation that these variables have with the target (last row of the correlation matrix). We notice that Years of Education and Lives with_Wife / Husband seem to be both more important for the algorithm itself and for the relation with the target, so we drop the other 2 variables.

Combining all the results and the explorations for the feature selection based only on the improvement of the chosen model, we get this final list of variables.

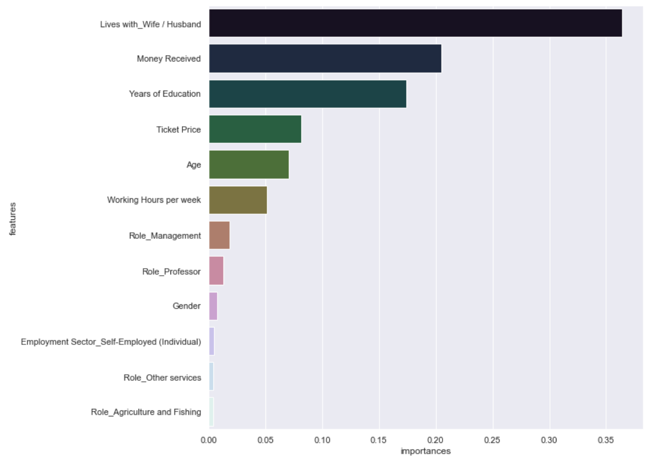


Fig. 7 - Feature importance for the Gradient Boosting model

5.2.2. Hyper-Parameters Exploration

In the second step for the model improvement process we try to explore as much as possible all the different hyper-parameters provided by the Gradient Boosting algorithm, using the knowledge acquired about our dataset and the structure and functionality of the algorithm itself.

For instance, we try to explore deeply the parameter for the number of estimators (n_estimators) to consider. To get a better idea of this parameter, we use a plot.

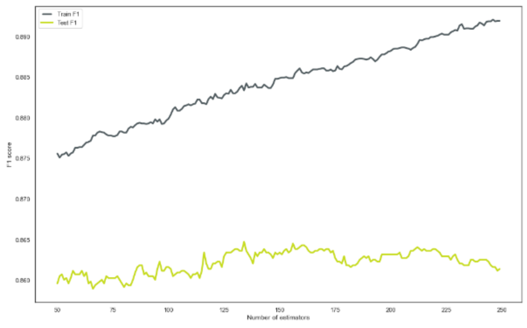


Fig. 8 – Variation of the F1 Score with the parameter n_estimators in the Gradient Boosting model

As we can expect, the bigger the number of estimators, the higher the score for the training set; however, it also leads to overfitting. Therefore, considering the score for the test set, we choose 135 estimators, which is the highest value for the test score.

We use the same approach to test the learning rate parameter, and we choose learning_rate = 0.3.

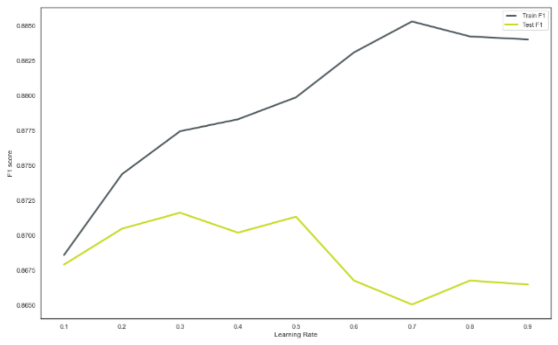


Fig. 9 – Variation of the F1 Score with the parameter learning_rate in the Gradient Boosting model

5.3. Hist-Gradient Boosting

5.3.1. From Gradient Boosting to Hist-Gradient Boosting

Exploring the results that we get from the Gradient Boosting model, we notice an inconsistency for the samples classified as above the average for the target (Income = 1).   Comparing the result with previous models and exploring a bit more, it looks like these observations are in minority respect the distribution of the variable Income for the train set.

To verify this supposition, we check the confusion matrix and, also, some distributions of the variables for the validation set. For instance, considering the real value of the target and the predicted value, the distribution for the Age variable looks like this:

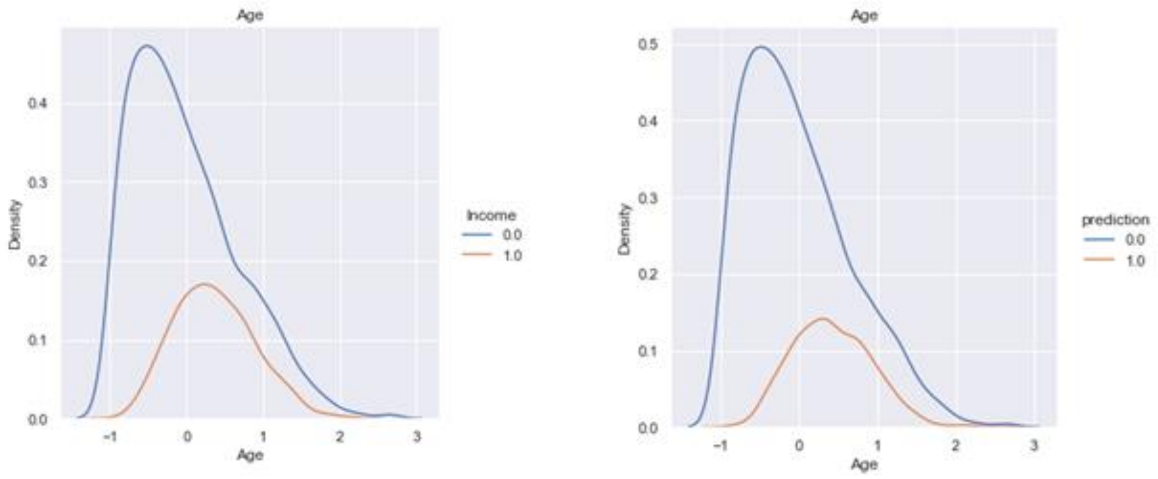


Fig. 10 – Distribution of the variable Age in the test (left) and in the validation (right) datasets.

**Note: data already scaled using Robust-Scaler*

As we can conclude from the plot, that it's just an example of all the other distribution that we checked, the two distributions look very similar, and that's good news for the performance of our prediction-model. We can also notice a little underestimation of the values 1, comparing with the real values of the validation set that we have in the left plot.

Exploring deeply our result and all the process done to obtain it, we also try different approaches for the imputation of the missing values, that we did using KNN imputer.

The hypothesis that we consider now is that using the KNN imputer we are introducing in the dataset some bias. Probably the KNN Imputer for our dataset is working but not in the most efficient way.

In this sense, trying to adjust it, and considering the other possibilities that exist to deal with the missing values, we come across the Hist-Gradient Boosting algorithm.

In Scikit-Learn, this algorithm is presented as new and experimental. Researching on it, we discovered that it gives us the possibility to 'leave' to it the problem of the missing values. Indeed, a very interesting characteristic of this algorithm is the native support that provides for the missing values, whenever it pops up with a NaN for the variables that it selects randomly for the creation of the trees. Quoting Scikit-Learn:

"During training, the tree grower learns at each split point whether samples with missing values should go to the left or right child, based on the potential gain. When predicting, samples with missing values are assigned to the left or right child consequently"

The power of the Histogram Gradient Boosting is the subdivision in bins of the samples. With this particular version of Gradient Boosting, the algorithm gains in speed and, also, often in terms of performances. Considering the samples grouped in bins allows the reduction of the noise in the data: indeed, a single value for a single feature is less important if it is considered in group with others, and so during the learning process the algorithm gains in generalization.

Considering that the Hist-Gradient Boosting is an implementation of the Gradient Boosting, we decide to switch to this algorithm, without introducing problems or inconsistency with all the other choices done so far.

To get the best performance for this algorithm, we choose to maintain all the same preprocess of the data, except for the KNN Imputer, which perhaps is not performing very well, based on our hypothesis.

Thus, we leave the missing values inside the dataset.

Running the Hist-Gradient Boosting, ceteris paribus (all the other preprocess decisions are still the same except for the KNN imputer), we get a better result, both for the train and for the validation predictions.

5.3.2. Hyper - Parameters Exploration

- Max Bins

One of the parameters that we can define appositely for this new algorithm is the number of bins (max_bins) to group the samples. This parameter seems to be important, so, as done before, we explore the different possibilities that we have using a plot.

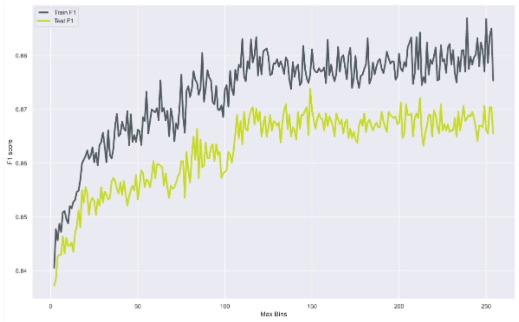


Fig. 11 – Variation of the F1 Score with the parameter max_bins in the Hist-Gradient Boosting model

The plot above shows that, for a small number of bins, the scores for both the train and the validation sets are lower. With the increment of the number of bins considered, the scores keep increasing, until more or less 150. From this point ahead, the increment of bins seems to do not impact anymore the score.

- Loss Function

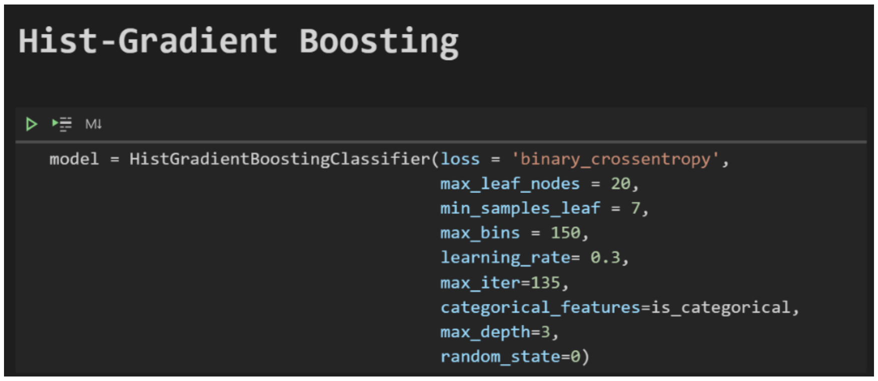
Furthermore, we explore the possibility that the algorithm provides for the Loss Function. Since we are dealing with a classification problem, it's available a loss function, "binary_crossentropy" (also known as logistic loss), that is specifically used for binary classifications.

- Categorical Features

Finally, exploring the algorithm in Scikit-Learn, we notice that it is available a parameter called categorical_features, which allows us to specify to the algorithm which are the categorical features, in order to use the values for the encoded categorical features in a different way from the numerical ones. In other words, the algorithm can assign an observation in a branch or in another of the tree considering that the variable is not continuous, and so for instance the distance between 0 and 1 is not metric.

In order to provide this information to the algorithm, we set the parameter equal an integer array (a categorical mask) that defines the encoded categorical variables and the numerical variables.

The final algorithm created with all the parameters tuned is the following:



Hist-Gradient Boosting

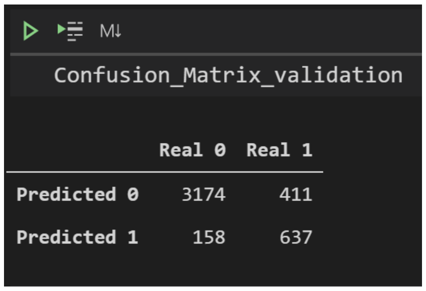
```
model = HistGradientBoostingClassifier(loss = 'binary_crossentropy',
                                       max_leaf_nodes = 20,
                                       min_samples_leaf = 7,
                                       max_bins = 150,
                                       learning_rate= 0.3,
                                       max_iter=135,
                                       categorical_features=is_categorical,
                                       max_depth=3,
                                       random_state=0)
```

6. Discussion

To sum up, the result that we get with the Hist-Gradient Boosting is better than the one obtained with the Gradient Boosting. The algorithms are similar; however, probably we think that for this dataset the assumption made for the missing values is correct: probably the KNN Imputer is inserting not only the missing values, but also some bias.

An insight that this project may have is to compute some operations in order to improve the estimation of the samples with predicted Income = 1, i.e. above the average. Indeed, exploring the result with the validation set we can easily notice that these observations are more difficult to predict for the algorithm.

We can visualize this assumption using the confusion matrix of the validation set, that is useful to have an idea of the prediction error for the class 0 and for the class 1.



```
Confusion_Matrix_validation
```

	Real 0	Real 1
Predicted 0	3174	411
Predicted 1	158	637

To improve the prediction for the class 1, it can be considered to tune other parameters deeply in order to help the algorithm to interpret what defines the classes based on the inputs. Probably, it gives too much importance to some aspects that characterize the class 0.

7. Conclusions

In this project, we were able to test different techniques and try different approaches in the dataset in order to predict the target variable Income, which was our main task. We applied some of the techniques studied in the classes and others studied by us.

We splitted our data into train and validation to train our model and check its performance, and then we used three different approaches regarding the normalization.

To train our predictive model we used several algorithms. We also applied the Grid Search, in order to get the best parameters, and some ensemble methods to get improvements in our previously created models.

In the end of our first analysis, we concluded that the Gradient Boosting had the best score between all the other models, so we decided to work harder on that model.

Our work and research led us to the exploration of some new algorithms that are not considered the most common ones. One of the more significant is the final model chosen: the Hist-Gradient Boosting.

As already explained, this algorithm is presented by Scikit Learn as en experimental one. However, in our project it was a valid competitor of other models.

In the future, it will be improved for sure; it is a work in progress, for instance the native support for the categorical features introduced with the parameter "categorical_features" (tested in this project) is available only in the last version of Scikit-Learn 0.24.0.

Finished our task, we hope to be the chosen group of data scientists that will be invited to join the Newland's government.

The final model created can predict if the income is above or below the average, in order to know which task to apply to this new planet; maybe if there are a great amount of people with income above the average, perhaps it is possible for the government to increase the tax.

In the future, this model can be used as an origin point for other applications of studies, such as health insurance, for example.

We are open to collaborate with teams that may want to explore this model or simply that want our help, having as request the importance of the moral values and the rights and the needs of the people involved in the study.

8. References

- Yingshi Chen, “LiteMORT: a memory efficient gradient boosting tree system on adaptive compact distributions”, https://arxiv.org/ftp/arxiv/papers/2001/2001.09419.pdf
- Scikit-Learn, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html
- David Moore. “Towards Data Science”, https://towardsdatascience.com
- Machine Learning mastery, https://machinelearningmastery.com
- Github/Scikit learn, “ENH add categorical support for HistGradientBoosting”, https://github.com/scikit-learn/scikit-learn/commit/b4453f126f34447967f52996039d11b0d2fa0090
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/naive_bayes.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/tree.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/svm.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
- Scikit-Learn, https://sckit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html
- Scikit-Learn, https://academic.oup.com/bioinformatics/article/17/6/520/272365
- Jason Brownlee. (2020, Jun). “Why One-Hot Encode Data in Machine Learning?”, https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
- Michael DelSole. (2018, Apr). “What is One Hot Encoding and How to Do It”, https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179
- Bao-Gang Hu. (2012, Mar). “What are the Differences between Bayesian Classifiers and Mutual-Information Classifiers”, https://arxiv.org/pdf/1105.0051.pdf
- ESANN'2007 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium). (2007, Apr). “Feature clustering and mutual information for the selection of variables in spectral data.”, http://apiacoa.org/publications/2007/krierrossietal2007feature-clustering.pdf