Newland Machine Learning Project

Ana Amaro (m20200598@novaims.unl.pt),
Andreia Tabuleiros (m20200581@novaims.unl.pt),
Gonçalo Almeida (m20200594@novaims.unl.pt),
Manuel Borges (m20200596@novaims.unl.pt),
Omar Jarir (m20201378@novaims.unl.pt)

Abstract

This paper was written in the pursuit of a model to allow the government to *a priori* classify citizens according to the tax group they will belong to (either 15% or 30%), in order to bring economic sustainability to the new civilization. While trying to reach this goal, the data handled was processed to extract new variables, and to correct inconsistences and missing inputs. In addition, and using cross-validation to assure the consistency of the results obtained, different models, standardization, oversampling, and feature selection techniques were attempted, as well as a series of parameter tunning to try to extract the best F1 score (using the micro average) of each model tested. In the end, the model that gave the best score was the Gradient Boosting. This model achieved a F1 score in the Train set of 0.892, and 0.8713 in 30% of the Test set. Furthermore, according to our predictions, 2111 of the new citizens will pay the higher tax, while the remaining 7989 will pay the lower rate. Moreover, the model shows a higher precision than recall, which in this context may allow the government to create more conservative budgets. Lastly, further studies may be appropriated, as the mission evolves, considering the representativeness of this Train set may be put at risk, depending on the future composition of the citizens entering the mission.

Keywords: Mission Newland, Machine Learning, Gradient Boosting, Classifier Model, F1 Score

I. Introduction

Since 2044, the conditions on the planet Earth have become unfeasible to human life, due to accelerated climate change. In order to solve this, a mission named “Newland” was developed with the goal of povoating a recently found planet, with conditions similar to Earth. As the new society forms, the implementation of taxes will be crucial, in order to make it financially sustainable. The desired form of doing this is a binary tax, where people that have a higher income than average pay 30% of their income, while the remaining pay only 15%. With this arises the necessity to create a model that based on demographic conditions of each person is able to predict to which group of people each citizen belongs to, as new future citizens are expected to arrive soon. It is based on this need for classification of citizens, that this paper is written, attempting to find the model that yields the best results.

II. Background

The dataset provided by the government to train the classification model had an unbalanced target, with only around 0.31 citizens belonging to the 30% class of taxation for each citizen belonging to the 15% class. Consequentially, we decided to use two methods -not learned in practical classes- to attempt to circumvent this issue: Stratified K-fold and Oversampling techniques.

Regarding the first, we used the implementation from Scikit-Learn. This method allows to assure that for each split made during the cross-validation process, the percentage of citizens belonging to each income label is kept [1] -meaning that in each split, we have 0.31 citizens belonging to the 30% class for each one in the other class. This approach helps us assure that in every fold we will be training and validating our model against a representative target.

The Oversampling techniques used were implemented by Imbalanced-Learn, and these methods allow for the model to train in a less unbalanced dataset. One of the techniques, Random Oversampling, uses random sampling with replacement of the minority class -in practice this means that it will duplicate entries from the minority class-, until it reaches a requested target ratio of citizens from the minority class to each one of the majority class. Although this technique may risk creating overfitting [2], depending on the model and on the percentage used, may produce good results.

The other technique is called Synthetic Minority Oversampling Technique-Nominal Continuous (SMOTE-NC), and this technique attempts not only to fix the problem of overfitting that may happen with Random Oversampling, but also the problem that arises when using the method Synthetic Minority Oversampling Technique (SMOTE). When dealing with continuous data, both SMOTE and SMOTE-NC, instead of sampling with replacement, rely on the method of K-Nearest Neighbours (by default the number of neighbours is 5, in the implementation used) to attempt to create new similar entries to the ones of the minority class, until the requested ratio of entries from the minority class to the majority class is reached. The problem with this approach, and that SMOTE is not able to face, is that as the K-Nearest Neighbours uses the Euclidean distance to create new entries, it cannot rely on this approach to fill the categorical data, as it cannot simply extract meaning from the distances between these type of variables -even if encoded. SMOTE-NC, instead, uses the continuous data to find the K-Nearest Neighbours and based on those, creates new, but close, entries to the neighbours in what regards the continuous fields, while for the categorical data, instead of attempting to create similar entries, it just looks for the most frequent value of each categorical feature in the neighbours and uses it to fill the new entries' categorical data.[3]

Moreover, when performing feature selection, two of the techniques used were also not mentioned in practical classes: normalized mutual information and principal component analysis. The first one, measures the quantity of information shared by two random variables, in terms of entropy, and as it is normalized it allows for comparability between variables, having the values comprehended between 0 and 1, where 0 represents no information shared (no joint entropy), and 1 represents the variables that are completely redundant (high joint entropy) [4].

The other is a technique used for dimensionality reduction, and although it involves some deep statistical concepts, putting it in a simple way, it tries to summarize the variance of correlated variables, creating new ones that are linear combinations of the previous, and that are orthogonal among themselves. Furthermore, the new variables created are ordered in such a way that the first one will be the one with a higher eigenvalue -the one explaining more variance [5]. If keeping all the new variables (called principal components), we would keep the same variance explained and the same number of variables as originally, but this strategy is usually used with a threshold of variance set, in order to only keep as many components as needed to achieve that threshold. Overall, and although this technique enables to construct variables that are completely independent, while keeping as much variance as we want to, it also has as major drawback, the fact that the new variables created lose interpretability in the light of the initial context.

Lastly, when dealing with selecting the right model to use, we attempted a model not learned in class: Voting Classifier. This model, working as an ensemble, attempts different models to fit the data, and based on the prediction of those models, uses a voting system to classify a specific entry. Basically, by using different models to predict the outcome, it is able to take "different perspectives" into consideration, which ends up creating robustness.

The vote system can be based on "soft" or "hard/majority" voting, in the particular implementation used. In the soft case, and considering a binary classification, it considers the probability predicted by each model for each outcome, then performs the average and only if the average for the positive outcome is higher than 0.5, an entry is classified as a positive case. In the case of the hard/majority vote, the model, instead of looking at the probabilities, looks at the prediction of each model, and selects the most frequent outcome as the probable outcome, and therefore the one it will use. To illustrate the two voting scenarios, let us look at an example: if we have three models (A, B and C), and the probability of the outcome being 1 for a specific entry is 0.55, 0.51 and 0.35 according to A, B and C, respectively, using the soft vote, our average probability would be (0.51+0.55+0.35)/3, which is lower than 0.5, and therefore the Voting Classifier would predict this entry as a 0. On the other hand, using the hard/majority vote, both models A and B would predict an outcome of 1, while only C would predict a 0, which would lead our Voting model to, considering the majority, classify this entry as a 1.

III. Methodology

The data used to analyse was provided by the government, and it was divided in two different files. The first one, called Train (that will be used for train and validation), contained not only the independent variables (demographic information about the citizens), but also the target value, while the second one, Test (that will only be used as test set), simply contained the independent variables. Furthermore, there was the possibility to submit our predicted target for the Test set and get the F1 score (with micro average), corresponding to 30% of the data. The programming language used was Python and resourced to the libraries: Matplotlib and Seaborn for data visualization, Pandas, Numpy, Imblearn and Scikit-Learn for processing data, being the last one also used for modelling. Lastly, and to ensure the reproducibility of the results, whenever necessary, the Random State was defined to 42.

Regarding the proceedings used, the first step was to go through the variables and try to do some pre-processing: Firstly, the variable Name was used to extract the Gender of the citizen, and the Birthday date was converted into Age to be easier to work with. Furthermore, the variable Education Level was decomposed in Education Level and Postgraduation – firstly because Postgraduation is not an official level of education, but also because it allows us to isolate the effect of the Postgraduation-, while the Marital Status was also used to create two new variables: Spouse Missing and Spouse in the Army – in order to make the variable Marital Status cleaner. Lastly, we also corrected some inconsistences with the data: converting the entries with question marks to None (as we considered those as missing values), and in the cases where people reported to be "Unemployed" or "Never Worked" and had Working Hours per Week different from 0, we replaced them by 0. Each one of those changes were done directly on the full Train set (and afterwards on the Test set), as none of them created data leakage: we only did feature engineering and fixed inconsistencies, that in a predictive phase can be done to every entry we receive before predicting.

Then, using the Stratified K-fold method, with the number of folds set to 10, being this an empirical good value[1], we proceeded to the outlier remotion in the train set of each fold -using for Age (between 17 and 90), Working Hours per Week (between 0 and 80) and Years of Education (between 2 and 21) what we considered as natural thresholds, while for the Ticket Price (up to 3000) and Money Received (up to 40000), the thresholds were based on a visual analysis of the distribution of values-, and with the imputation of the None values, where the method used was to replace the None values, both for the train and validation/test set, of each variable with the mode in the train set (to avoid data leakage). Note that using purely statistical based methods in the outlier remotion, like the Intra Quartile Range Method, proved to be too restrictive (too much information lost), whereas with the approach taken, we were able to keep 99.335% of the original data, while still removing the most extreme cases, what ultimately allowed us to train our models in an effective way.

The categorical variables were all encoded using the One Hot Encoder method (with a fit and transform in the joint of the train and validation/test set, as this doesn't lead to data leakage, but allows to have all labels taken into consideration, even if fully filled with 0's), even in the case of the Education Level, as we considered that there was too much ambiguity regarding the order of some levels (e.g. High School and Profession School) to consider it an ordinal variable. Note also that we set the One Hot Encoder to leave one of the labels out, in order to avoid multicollinearity problems.

We attempted to use three techniques of data standardization -Standard Scaler, Min-Max Scaler and Robust Scaler-, even for models that were agnostic to the scale of variables, and therefore did not need the process of standardization [6]. Note that the fit was purely done in each train set, to avoid leakage, but applied to both the train and test/validation set.

Concerning the feature selection component, we followed three different approaches. The first one, and the one that was used in combination with the other approaches was a small statistical selection using two statistical correlation tests: Pearson correlation among numeric features and Normalized Mutual Information among categorical features, that ultimately did not capture any variable in any of the folds or final training process. Furthermore, we also attempted to use the Principal Component Analysis -that we defined to keep the dimensions that together explained 80% of the variance from our numeric variables-, and feature selection based on the best parameters provided by a Ridge Regression or a Decision Tree, where we used the method SelectFromModel, by Scikit-Learn, that by default, keeps all variables that have a higher importance than the average importance of variables. Note that in all feature selection approaches, the fit, or the evaluation of the correlation, was always only made to the train set, but applied to both the train and the test/validation.

The models that were tested were the following: Random Forest, Gradient Boosting, Decision Tree, Multi-Layer Perceptron, K-Nearest Neighbours, Logistic Regression, Naïve Bayes, Banging using the K-Nearest Neighbours and the Decision Tree, being that all models were tested with the default parameters. One important remark is that for the K-Nearest Neighbours (and Banging using the K-Nearest Neighbours) we only fitted to the model the numeric data, as this model cannot measure in a meaningful way distances between categorical features using the same distance metric as the one used to measure the numeric data proximity -even if the categorical data is encoded, it would be theoretically incorrect.

From the models mentioned above the ones that given their performance we will attempt to improve are: Gradient Boosting, which, independently of the standardization method used and utilizing only the correlations for feature selection, achieved an average F1 score of 0.864 in the validation set and 0.868 in the train; Neural Networks that by using the Min-Max Scaler and with the Ridge Regression's feature selection method, obtained an average F1 score of 0.852 in the validation set and 0.86 in the train; Random Forest, with the method Standard Scaler and recurring only to correlations for feature selection, reached an average F1 score of 0.847 in the validation set and 0.981 in the train. Note that the Logistic Regression, in fact, performed slightly better than the Random Forest -with an average F1 score of 0.85 in the validation set and 0.853 in the train-, but considering how the other models have a significantly higher degree of customization than the Logistic Regression, we decided to leave this one behind and focus on the other three.

Before proceeding to the phase of parameter tunning, we tried a different approach first: to use the model Voting Classifier (both with "hard" and "soft" voting) with all the possibilities of standardization and feature selection mentioned above, but using only the four models (Neural Networks, Gradient Boosting, Random Forest, and Logistic Regression) that achieved the best scores in the previous phase, to see if this ensemble is able to combine them in a way it improves our classification efforts. Nevertheless, the results obtained were still worse than the ones obtained from our best previous model, Gradient Boosting, so we proceeded to the phase of parameter tunning with the three models selected above and decided to re-attempt this approach later on the road, once we had all the three models mentioned above tunned.

Regarding the tunning phase, the first step we took was to stop using a Standardization method with the Gradient Boosting, as it was not making any difference, which goes in line with the theory, considering this model does not rely on the scaling of the variables. We expected the same behaviour from the Random Forest. Nevertheless, and although the scores for the different standardization methods (and no standardization method) were almost similar, the Standard Scaler method got better results, so we will keep using it on the Random Forest model, even acknowledging that theoretically, this should be insignificant. In addition, in all the three models we will explore further, we stopped using the correlation methods for feature selection, as no variable was being removed by this.

Then, we started the process of parameters tuning, and in this process, we decided not to use any Grid Search mechanism. The reason for this is that, inside each fold in the cross-validation process, we need to make some changes to our data (e.g., removing outliers from the train set and filling None values), and since the Grid Search automatically do the K-fold when fitting the model, there is no way we can make the necessary changes and still avoid data leakage: for instance, even if we split our data into test /validation and train set before making the changes, after fitting the Grid Search to the data, it will make splits where the test set will necessarily in some folds contain elements of the previous train set (the one that was before fitting the Grid Search, and was processed with direct fits for standardization methods and feature selection, in addition to have had the None values filled based on its own mode), meaning some data leakage will necessarily occur. Considering this, and despite the process being more demanding in "human supervision", in order to preserve the reliability of results, we will do the process of parameters tunning manually. Regarding this process, below we will present some of the attempts we have undertaken, after an initial phase, where more extreme values (and different parameters) were attempted to point us to the right direction.

Regarding the Gradient Boosting, and considering it was not overfitting in the attempt with the default parameters we decided to focus on increasing the overall fit, while trying to keep this tendency of not overfitting. In order to do this, we decided to attempt to increase the number of estimators to values significantly higher than the default -1000, 1150 and 1250 estimators, while the default is 100- which basically means that our model will use more Decision Trees in order to build the full model, and considering the model use of sequential estimators, if done in excess can increase the overfitting. In order to prevent this, we attempted to balance the risk of overfitting by setting the minimum samples to constitute a leaf (to values higher than the default, 1, but given the model didn't show a strong tendency to overfitting with this particular dataset, not high enough that could risk underfitting) to the value of 2 and 3, which will allow each estimator to only keep splitting if it creates a leaf node with the designated number of "citizens" in that leaf node. Furthermore, we also attempted to provide to each estimator different values for the limit of features it can use -being those values the full, the square root and 35% of the available number of features-, with the goal of seeing if limiting the number of features, which can also reduce overfitting, works better than just letting each estimator use up to all features available.

For the Random Forest and considering it was overfitting with the default parameters, we focused on fixing this. Firstly, we selected high values for the number of estimators -500, 1250 and 1750-, which contrary to the Gradient Boosting case, given the estimator are independent from each other, does not increase necessarily the chances of overfitting, and will most likely help the model to improve. Moreover, we endeavoured to increase the number of minimum samples to constitute a leaf -to the values of 3 and 5- in an effort to correct the overfitting this model was expressing. Lastly, and for the same logic as with the Gradient Boosting, we attempted to let each estimator either use as many features as available or subjecting it to a limiting condition -square root or 35% of the number of features.

The Multi-Layer Perceptron was a little harder to tune. The parameters we tried to improve were: Number of hidden layers -with the goal of seeing if an increase in the model complexity could increase the overall fit-, where we attempted not only the default value of only one layer (with 100 neurones), but also two layers (both with 75 neurons) and three layers (with the middle layer with 100, and the other two with 75 neurons); Activation function and Solver, by testing both the "relu" and "tanh", and "adam" and "sgd", respectively; Number of iterations, where we increased from the default 200 to 350, to improve the chances of convergence; Learning Rate of initialization, in order to not increase the overfitting, given we tried to increase the complexity of the model, we not only attempted the default learning rate of initialization (0.001), but also 0.0001; The learning rate, in the cases we used the solver "sgd" and the default learning rate of initialization, we also attempted to use not only the default ("constant"), but also the "invscaling" learning rate, to again, try to prevent overfitting, as this allows the model to decrease the learning rate progressively.

For the best performing parameters set of each model, we tried techniques of oversampling –both Random Oversampling and the SMOTE-NC method, with different percentages of oversampling (0.35, 0.4, 0.45)–, given our target was unbalanced. Note that we decided not to use under sampling techniques. Although we have a dataset sufficiently large to do so, this process might take some diversity from our sample, and we prefer to avoid this, especially considering that similar results can be achieved with Oversampling techniques.

Before declaring a winner model, we once more attempted to use the model Voting Classifier (using again all combinations of feature selection and standardization originally presented, as we are not able to select for each model the best combination individually), but this time with the best performing parameters of each of the 3 models we attempted to tune at a deeper level.

With the validation phase completed, the Gradient Boosting kept getting the best performance, so we decided to fit this model in the full dataset in the Train file –after the same pre-processing steps of variables already discussed above, with no standardization, oversampling, and feature selection methods-, in order to get a model capable of predicting the target variable for the dataset contained in the Test file.

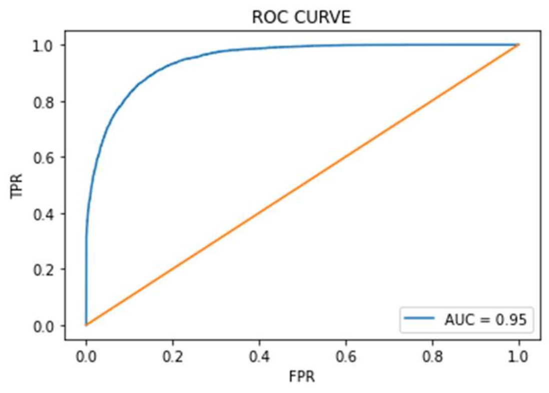
IV. Results

As mentioned, the model which performed better while doing cross-validation was the Gradient Boosting. With the number of estimators being 1150, the minimum number of samples to create a leaf being 2, and the maximum number of variables to be used by each weak learner as 35% of all the variables available, the model was able to achieve an average mean F1 score of 0.871 in the validation set and 0.892 in the train set. Furthermore, no oversampling technique was used, given the results presented below when trying to use it.

	Time	Micro $F1$ - Train	Micro $F1$ - Test
GB with 0.35 SMOTE-NC	14.907 ± 1.35	0.889 ± 0.0	0.869 ± 0.01
GB with 0.4 SMOTE-NC	15.677 ± 1.65	0.885 ± 0.0	0.866 ± 0.01
GB with 0.45 SMOTE-NC	17.275 ± 1.23	0.884 ± 0.0	0.864 ± 0.01
GB with 0.35 Random Oversampling	14.056 ± 0.85	0.888 ± 0.0	0.868 ± 0.01
GB with 0.4 Random Oversampling	14.738 ± 0.87	0.884 ± 0.0	0.867 ± 0.01
GB with 0.45 Random Oversampling	15.172 ± 0.9	0.88 ± 0.0	0.865 ± 0.01
Gradient Boosting with no Oversampling	14.549 ± 0.64	0.892 ± 0.0	0.871 ± 0.01

Table 1 – Time and Micro F1 score for train and test sets, using Gradient Boosting with the identifiable oversampling techniques, in addition to the following parameters: max_features set to 0.35, min_samples_leaf equal to 2 and n_estimators set to 1150.

The recall of our model in the full dataset from the Train file, regarding the Income above average, was 0.698, while the precision was 0.82, but despite the apparent higher focus of the model concerning precision, the score in the Area Under the Curve of the Receiver Operating Characteristic, where a balance between those two indicators is important, was significantly high.



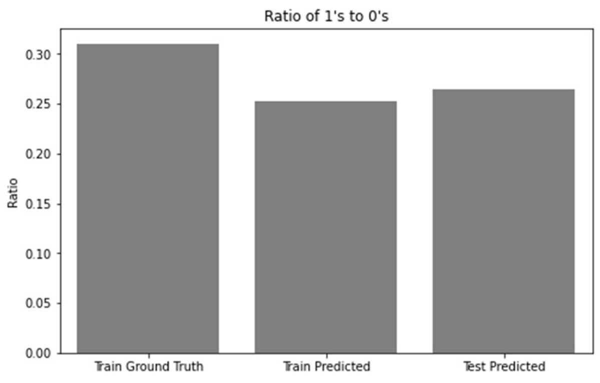
Graph 1 – ROC Curve for the Train dataset, in blue, with the ROC Curve for a random model, in orange, for reference.

On the same note, concerning the recall and precision, the distribution of True Positives, False Positives, True Negatives and False Negatives in the full Train set is summarized on the confusion matrix below.

		Predicted Values		
		0	1	
Actual Values	0	16 178	805	16 983
	1	1 593	3 675	5 268
		17 771	4 480	

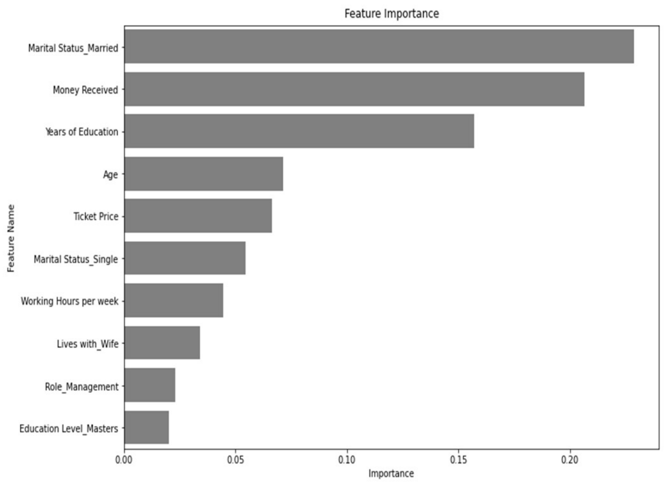
Table 2 – Confusion Matrix for the Train set.

Moving on to the final results, when using the whole dataset from the Train file, the F1 score achieved in the train phase was 0.892, and the result on Kaggle, where it is only possible to verify 30% of the data, was approximately 0.8713. Based on the predictions of our model the number of citizens from Test dataset that will have to pay 30% in taxes is 2111, while the 15% tax will be paid by 7989 citizens. This presumes a ratio of 0.264 citizens earning an income higher than average for each below or equal to the average, which is in line with the ratio predicted by the model for the Train set (0.252), despite the ground truth ratio in the Train set being slightly higher than 0.31.



Graph 2 – Value of the ratio of ones to zeros corresponding to the Ground Truth and Prediction for the Train set and the Prediction for the Test set.

Lastly, given the ability of the Gradient Boosting model to rank the importance of the features while fitting the data, we decided to present the ten most important features for our model, as shown below. Note this feature importance is based on the impurity decrease ability of each feature, when applied to the model in use, and not an absolute ranking.



Graph 3 – Ten most important features, according to the final Gradient Boosting model, when classifying a citizen as belonging to the above average Income group and to the below or equal average Income group.

V. Discussion

The first thing worth mentioning about the results obtained is the little overfitting that the model obtains, with a difference in F1 score between train and test of around 0.02. Additionally, the results obtained in the validation and train set in the process of cross validation are very close to the ones being obtained in the full Train set, and in 30% of the Test set, which gives a reasonable degree of certainty that in the remaining 70% of the Test set, the model will keep the same performance.

Furthermore, and considering the model used, it is not surprising how feature selection and standardization were not useful in improving its performance, as it does not rely on the scale of the variables and has an intrinsic mechanism to prioritize the features while fitting the data. Nevertheless, one interesting thing about the results achieved was how the oversampling techniques, not only did not help to improve our model, but, in fact, even decreased its predictive capacity.

Regarding the recall and precision, our model, despite achieving a good F1 performance and AUC value (that balances recall and precision), clearly favours the precision over the recall. In the context of the problem, this seems to be a good approach, as the government will probably make budgets based on the predictions of the model discussed in this paper (before the citizens arrive, as after arrival they will know the ground truth: if a citizen is receiving an income above or below the average and will not need the model). Bearing this in mind, it is fair to consider the false positives more costly than the false negatives: it is better for the budget to be constructed in conservative terms, expecting more people only paying 15% than the ground truth, than risking predicting in excess the number of citizens paying 30%, and then do not have the necessary revenues to face the costs.

Another interesting remark about our findings concerns the most important variables. Even in the ten most important, it is clear that there are variables significantly more important than others, namely if a citizen is married, how much money has received to be on the mission and the years of education (all with an importance higher than 0.15), while variables like the number of hours worked per week do not seem to be as relevant when predicting if a citizen earns more, less or the average income, which may seem somehow counterintuitive. One important remark about the feature importance is that it may suggest a potential improvement for the process of how the government selects the citizens of Type A to be part of the mission.

On a different note, a challenge that may arise in the future, with the model suggested in this paper has to do with the data used to train the model. In the dataset provided the number of citizens with a 15% tax was significantly higher than the number with a tax of 30%, which statistically speaking may not be accurate. If we have a normal distribution of incomes, we should have the median equal to the average, which would mean that 50% of the citizens would have to pay a 30% tax, and the remaining would have to pay only 15%. Being these percentages considerably different from the ones our dataset provided, it raises the question if either the income distribution of the full population from the planet Earth that will embrace this mission does not follow a normal distribution, or the sample we got is not statistically representative. This ultimately may imply that in the future the model will need to be retrained to adjust to contexts where the distribution of incomes differs from the one until now.

Furthermore, and still regarding the balance of the dataset, it would be interesting to explore how the different types of passengers vary across the different phases of the mission, as that may also have an impact in the model, which can call for improvements, or even the use of a different model altogether. To illustrate better our concern: In the future, and as the civilization of this mission grows and the level of life on Earth keeps deteriorating, it is probable that more people will be willing to pay high sums to enter the mission, as well as the government may need to reinforce some position and may need to selectively pick and pay people to be there, which will imply much more citizens of Type B and C, than we have now.

Regarding the variables used in the study, we felt its quantity was too restrictive, in the sense, that given the mission, and how the majority of the participants has to go through a very specific process, some further data could be of use (e.g., field of study), and that could improve the performance of the model. Further studies about this mission may ask for more dense register of characteristics of the citizens in order to have more features to work with.

One last remark, considering the quick evolution that the field of Machine Learning experiences, in future reproductions of this work, some models may have suffered improvements in the implementation, or better predictive models may have appeared, so any further update on this paper should be preceded of a rigorous analysis to the Machine Learning standards at the time.

VI. Conclusion

Considering the challenge of finding the best classifier to predict if a citizen will pay a tax of 30% or 15% due to having or not an Income above the average, the best model found was based on the Gradient Boosting. With this model, we estimate that from the new citizens, 2111 will have to pay the higher tax rate, meaning that the majority of the citizens will pay a lower tax. We back our premise with a F1 score of 0.892 in the data we fitted the model and with 0.8713 in the available public score on Kaggle.

Despite the good results achieved, there are still some points open for further research to build on. Further studies should try to find out if the data used in this paper is representative of the population and/or if the representativeness will depend on phases of the mission. Moreover, there is the possibility to try to enrich the model here presented with more variables if they become available, with a very limited risk of contaminating the model, as the Gradient Boosting prioritizes variables, which means that in the case the variables added do not have useful information, they most likely will not impact as much this model as others might be impacted.

VII. References

[1] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[2] Zhuoyuan Zheng and Yunpeng Cai, Ye Li. Oversampling Method for Imbalanced Classification. In *Computing and Informatics*, volume 34, 2015.

[3] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. In *Journal of Artificial Intelligence Research*, volume 16, 2002.

[4] Pablo A. Estévez, Michel Tesmer, Claudio A. Perez and Jacek M. Zurada. Nomalized Mutual Information Feature Selection. In *IEEE Transactions on Neural Networks*, volume 20, 2009.

[5] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[6] Dimitrii Borkin, Andrea Némethová, German Michal'Conok and Konstantin Maiorov. Impact of Data Normalization on Classification Model Accuracy. In *Research Papers Faculty of Materials Science and Technology in Trnava*, volume 27, 2019. 218100100