Project Report: Newland, The City of the Future

António Pinto (m20200659@novaims.unl.pt), Davide Farinati (m20201080@novaims.unl.pt),
Mohamed Elbawab (m20201102@novaims.unl.pt), Tomás de Sá (m20200630@novaims.unl.pt)

Abstract - This paper gives an insight of how can we predict the taxes that the residents should pay on a found planet based on their information. The biggest challenge of the whole process is to handle the dataset reasonably size. Models from sklearn such as Decision Trees, Random Forest, Gradient Boosting or Neural Networks were used in order to create a solid predictive model. Gradient Boosting revealed to be the best model with 87% accuracy on train and 86.6% on validation. There were analyzed different approaches for each model to guarantee that all the options were explored but the number of categorical values presented in the dataset may have been limiting due to time cost and model acceptability to treat them.

Keywords : Machine Learning, Predictive Models, Python, Project, Newland

II. INTRODUCTION

With the population growth, the resources provided by the earth are not enough anymore. The year is 2048 and we need to start searching for solutions outside the globe. Thousands of people were sent to a new planet, Newland, to live there. To make it financially sustainable, the residents need to start paying taxes based on their income. But how can we define who pays what? The government needs to decide to which class each resident belongs to.

It is necessary to create a predictive model that analyses the residents income and decides the tax amount to be paid for each resident. This model will be tested in 32500 people living on earth while being developed so that, after finished, can be applied in the Newland's residents dataset.

II. BACKGROUND

***Normalization -** Power Transformer Scaler*

Power Transformer Scaler is a feature transformation technique used when building linear models. This method changes the distribution of the variable making it more normal. Usually, with other power transforms, it is necessary to first study the original distribution and only after make a decision and this is the difference from this to other methods: the Power Transformer automates the decision making by introducing a parameter called lambda. By using the Box-Cox and Yeo-Johnson power transforms, it finds the lambda value that best fits. [1]

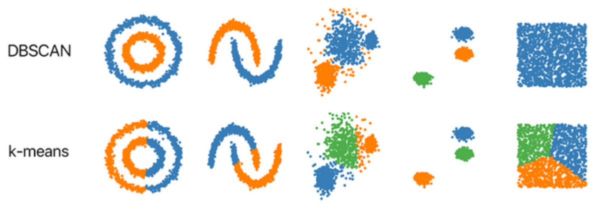
***Outliers -** Median Absolute Deviation*

MAD (Median Absolute Deviation) is a method to check for outliers. It is a robust measure, which means it is resistant to outliers (against the mean, for instance) and it is a very good alternative to the standard deviation if your data is not normal [2]. MAD is a variation of the mean absolute deviation and is defined as:

$$MAD = \text{median}(|Y_i - \text{median}(Y_i|)$$

***Outliers -** DBSCAN*

DBSCAN is a very simple but popular density-based algorithm for discovering clusters in large spatial databases with noise. By using distance and a minimum number of points per cluster, it classifies a point as an outlier or not. What makes this algorithm so mediatic is that it doesn't need to know the expected number of clusters in advance but, on the other hand, since it only uses the distance, can be vulnerable to certain types of false positives. [3]



Outliers - *Isolation Forest*

Isolation Forest follows the same principle of the Random Forest. It identifies anomalies instead of profiling normal data points, as other outlier detection methods. As the Random Forest, this method is built on the basis of decision trees (partitions are created by first randomly selecting a feature and then selecting a random split between the minimum and maximum value of the selected feature). The outliers are less frequent than regular observations and that's why, by using random partitioning, they should be identified closer to the root of the tree with fewer splits necessary. After choosing an observation from the random partitioning, it is necessary to set an anomaly score. The smaller the path length (number of edges an observation must pass in the tree going from the root to the terminal node) of the observation, the higher the score and greater the probability of being an anomaly. [4]

Missing Values - *XGBoost*

XGBoost is an implementation of gradient boosted decision trees algorithm designed for speed and performance. In this case, for missing values treatment, XGBoost uses an algorithm feature called Sparse Aware which automatically handles the missing values. [5]

Missing Values - *One-Hot Encoding*

One-Hot encoding can be applied to the integer representation and it is where the integer encoded variable is removed and a new binary variable is added for each unique integer value. Having some missing values and some not that much common ones, it is possible to create a column called other (for instance) and set to 1 on those specified cases. [6]

Feature Selection - Boruta

Firstly developed in R, Boruta is a smart algorithm designed to automatically perform feature selection on a dataset. It is also statistically grounded and works extremely well even without any specific input by the user based on two ideas:

1) Shadow features: the features, instead of competing among themselves, compete with a randomized version of them (called shadow features). The shadow data frame is then attached to the original one and the importance of each original feature is taken. When the importance of a feature is higher than the set threshold, it is a 'hit'. A feature is useful only if it is capable of doing better than the best randomized feature.
2) Binomial distribution: to decide which feature is more useful, we choose a feature (doesn't matter which one is) and the maximum level of uncertainty associated to that feature is 50% (as tossing a coin). During the experiment of getting the uncertainty level of the feature by trying n times, it follows a binomial distribution.

   The output values are divided into 3 areas:

   I. Refusal area (red area): the features that end up here are considered noisy and dropped;
   II. Irresolution area (blue area): Boruta is indecisive about the features in this area;
   III. Acceptance area (green area): these features are considered as predictive and kept.

   The areas are defined by selecting the two most extreme portions of the distribution - distribution tails. [7]

Feature Selection - *Voting Methodology*

A voting methodology is used when we intend to aggregate the information from all the feature selection algorithms and where the selected features and the ones that were considered relevant by its algorithms at least 4 times.

Over Sampling - *SMOTE*

SMOTE (Synthetic Minority Over-sampling Technique) oversamples the underlying dataset with new synthetic points. It is parameterized with the number of nearest neighbours that it considers and the number of new points you wish to create. This technique will randomly select a minority point, then select any of its neighbours, specify a lambda value [0, 1] and then generate and palce a new point on the vector between two points (located lambda percent of the way from the original point).

Dimensionality Reduction - *PCA*

PCA (Principal Component Analysis) is one of the most widely used dimension reduction techniques in order to transform a large dataset into a smaller one. How? It identifies the correlations and patterns preserving the most valuable information. By selecting the principal components (the ones that represent the others), it overcomes the feature redundancy making the data visualization easier to handle, decreasing the model complexity and increasing the computational efficiency. [8]

III. METHODOLOGY

Various techniques are used in Machine learning for the detection of the value of income for the new observations for the Newland project. Some of the models and feature selection techniques are discussed within this section of the paper. An experimental phase section describing the code is also presented at the end of the methodology section.

3.1 Neural networks

The Neural networks are one of the best-known AI techniques. They are biologically inspired, and they mimic the structure of the human brain. The neural network consists of many simple neurons that are connected together. Each of them would produce a sequence of real valued activations. One of the main problems of neural networks is that the final model is a sort of black box. Thus, the readability of the model is very limited. [9]

3.2 Decision Trees

Decision trees are machine learning technique that can address both regression and classification problems. A decision tree helps to build a classification model that has the form of a tree structure. The internal nodes of the tree contain the independent variables, while the leaves correspond to the possible target outputs. Each internal node has several branches, corresponding to the possible values that the variable can assume. One of the limitations of decision trees is that the division of input space is based on hard splits which would make be compensated when used in an ensemble. [10]

3.3 Random Forest

Random forest belongs to the family of ensemble methods. The idea introduced with random forest is to build different decision trees, each one considering a randomly selected subset of the independent variables of the problem. This point is a crucial because it leads to decision trees with different structures that can model different aspects of a given problem. [11]

3.4 K-Nearest Neighbor

K-Nearest Neighbors is a technique that can be used for both classification and regression problems. The KNN algorithm calculates the distance between each pair of points in the training set. Then, a new data point p is classified using a majority vote. From the training set, it considers the K points that are closer to p and assigns p to the class to which the majority of the K neighbors belong. The algorithm is basic to implement, and the performance only depends on the choice of the K parameter. [12]

3.5 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees [13]. The gradient boosting approach can be used to optimize any loss function that is at least convex and differentiable. [14]

3.6 K-fold cross validation

The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn. [15]

3.7 Boruta

Boruta is a feature selection algorithm which is statistically grounded and works extremely well. In Boruta, features do not compete among themselves. Instead, they compete with a randomized version of them. Boruta divides the features into three areas, area of refusal, area of irresolution and area of acceptance. This technique is a highly recommended one in automated featured selection.[16]

3.8 Isolation Forest

At the basis of the Isolation Forest algorithm there is the tendency of anomalous instances in a dataset to be easier to separate from the rest of the sample (isolate), compared to normal points. In order to isolate a data point the algorithm recursively generates partitions on the sample by randomly selecting an attribute and then randomly selecting a split value for the attribute, between the minimum and maximum values allowed for that attribute. When all the trees are grown, outliers are identified as points easier to isolate, therefore with a smaller path length in the tree, being closer to the root.

3.9 *Experimental Phase*

This study code is developed using python and *Jupyter notebooks*. The libraries included can be shown in the first part of the code attached. The libraries include "*pandas, numpy, Sklearn, seaborn*". The dataset used in this project is provided by the government of Newland. Several groups of data scientists were invited to create a predictive model based on a dataset of 22400 observations.

At the beginning of the code, data is imported. Data exploration step is usually the next step in most Machine learning projects. Data exploration helps in identifying the given data and variables. Then we start to proceed with the data handling and feature extraction. We changed the provided "date of births" into "*ages*". We added '*Money received/ Ticket* Price', '*Gender_M'*. Money received/ Ticket price is calculated from the money received divided by the ticket price. We also added 3 variables for 'Group A', 'Group B' and 'Group C' ,however they were not needed as 'Money received/ Ticket Price' is a representation of the groups. So Group A, B and C were not added to the final code. We have also checked the 'Years of Education' and the 'Education level', the relevance between both variables was clear. So we dropped the 'Education Level'.

As for the data incoherence, we have checked the 'Working hours per week'. We noticed some values that don't make sense, indeed it doesn't make sense for a person to work for more than 65 hours per week. So we dropped these values which were less than 4 percent.

Then data was divided into binary ,metric and non-metric. We will apply the *Box Cox*, *Yeo-jhonson* and *quantile power transformations* to the metric variables, for power transformation purposes.

Then data is split for train, validation and test. As for the univariate outliers, three methods are used. Z_score method, the median absolute deviation method and the iqr method are the ones used [10]. As per the different techniques and sensitivity factors, we are going to consider outliers that are satisfied from at least 2 of the previously mentioned methods. As per the notebook, 4.42% of values are classified as outliers for 2 of the methods. Hence, these values are dropped.

As for the multivariate outliers, first we started by scaling. Four scaling techniques were experimented. The four techniques are Standard, Robust, MinMax(-1,1) and MinMax (0,1). After the experiment, the MinMax (0,1) resulted in the best performance. MinMax (0,1) is the one used.Then , we proceeded with 2 techniques for detecting multivariate outliers. We have used DBscan and isolation forest. Finally, Removing both the univariate and multivariate outliers contributed to a decrease in the performance of our model, for this reason we decided not to remove any outliers from the dataset.

For the next step "Imputing Missing values ", we start by replacing '?' values with nan to detect the amount of missing values. As per the outputs shown in the code, we will work on filling the data of the columns "Base Area", "Employment Sector" and "Role".

For the categorical values, one hot encoding and label encoding are tested. One hot encoding was the one used, as per higher performance results. One hot encoding is applied to 'Native Continent', 'Marital Status' and Lives with.

For filling the missing values, Several models were tested. The models tested are (K neighbor, Random forest, XGBoost, Gaussian). We are going to use XGBoost to impute missing values for Base Area and Employment Sector, while for the Role we will keep the missing values as '?' since we didn't find a good enough method to impute them. As per the score, the missing values are imputed using XGBoost. Then one hot encoding is also applied to the columns which we imputed the missing values.

In the feature engineering step, new features were added to aid in the detection of the output. The new added features include "ratio_yearsedu_age, ratio_workingh_age, ratio_workingh_yearsedu, work_experience".

The previous steps will be implemented again on the validation dataset for getting higher accuracy. Finally, the steps will be implemented also on the test data set.

Correlations are used to show the relationship between two or more items. We will see the correlation between items to drop the highly correlated ones, as they would show the same relationship. 'Marital Status_Married' item as per the high correlation with 'Lives with partner'.

Model selection and assessment is one of the most important steps in a machine learning project. For the model selection, several functions were developed. The developed functions are "avg_score, show_results, metrics, eval_models, evaluate_model". The functions are interrelated, so any function could be calling another function. "avg_score" is used for calculating the average score and the average f1 score. "show_results" is a function that receives an empty dataframe and different models, and then will call the function avg_score. "metrics" is the function used for printing the different outputs. "eval_model" is the function used for generating the ROC curve, ROC and AUROC values. The most important metric in a classification project is the area under the ROC curve (AUROC), which summarizes in one single value the precision. The higher the AUROC, the better the model is at predicting the correct class [18]." evaluate_model" is used in evaluating different models using the repeated stratified K folds [15].

Different Models are then tested with the applicable data. Non Metric data wouldn't be suitable with the same models as with metric ones. Metric data is used with all of the models. However, categorical data is used with decision tree, random forest and gradient boosting. The tested models are: Logistic regression, Gaussian naïve bayes, Random Forest, Gradient boosting, support vector machine, Neural network, K-nearest neighbor, Decision tree. All the models will be tested. The model with the best AUROC will be chosen. The next section of the paper ,section IV, will discuss the results thoroughly. After the model is chosen, features need to be chosen. Different feature selection techniques are used including: Boruta [17], Random forest, Ridge and Lasso regression. In order to aggregate the information from all the feature selection algorithm, a voting methodology is used, where the features selected are the ones that were considered as relevant by the algorithms at least 4 times by the different mentioned techniques.

IV. RESULTS

The results obtained in the final version of our project are the final product of a combination of different approaches and techniques combined with trials that shaped our way to find the best model in terms of predicting the income of unseen data in the validation dataset.

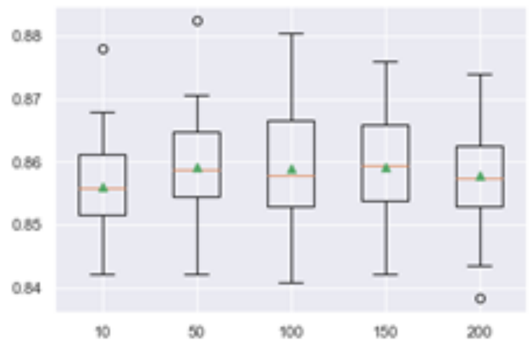
The first assessment we've done, and the most important one to contribute for a good accuracy, is to select the model we are using for the final prediction. With the constraint of using only models from sklearn, we started by selecting the following models: Logistic Regression, Gaussian Naïve Baise, Random Forest, Gradient boosting, Support Vector Machine Classifier, Neural Networks, K-Nearest-Neighbors and Decision Tree Classifier.

Random Forest seemed promising, with an initial accuracy of 98,4% on the train dataset and 84,9% on validation, thus we decided to explore some parameters to take the most out of the model.

Since our intention is to compare each model and select the best one, finding the best parameter is very important to reveal the potential that otherwise could be unseen due to overfitting purposes or any other reasons that may affect the performance of the algorithm.

In this case we are clearly dealing with an overfitting situation and to overcome this issue we searched for the best parameter in terms of number of trees and samples, minimum samples split, max depth, the criterion for splitting and class weight. Find below an example of the output from the cell that gave us an insight of the minimum number of samples required to split an internal node with its accuracy and standard deviation.

```
>10 0.856 (0.008)
>50 0.859 (0.009)
>100 0.859 (0.009)
>150 0.859 (0.008)
>200 0.858 (0.008)
```



By observing the results, 150 was chosen to be the best number for the minimum samples split.

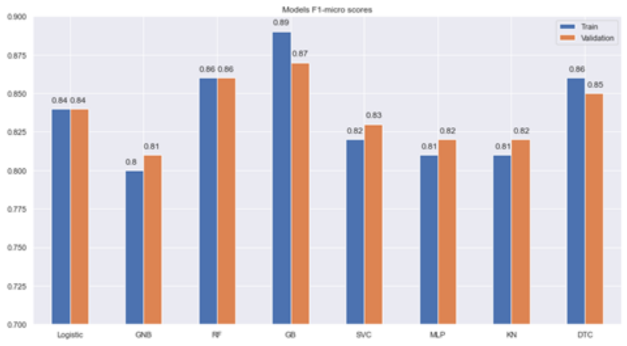
After finalizing the tuning for the rest of the parameters, our model had an improvement in classifying the target variable, as it stopped overfitting to the train dataset, with a final accuracy of 86,1% for the train and 85,7% for validation.

Decision Tree Classifier, a simpler variant of the model addressed above, also had some promising results, with the scores on train and validation of 85,5% and 85,4% respectively, after tuning.

Gradient boosting was revealed to be the best model of all the models used, even before any tuning with an accuracy of 87% on train and 86,6 on validation. For this algorithm the parameters set chosen to be tuned for this model assessment section were the number of estimators, fraction of samples (subsamples), learning rate, tree depth and minimum samples split. The end results for the Gradient Boosting Classifier, where the max depth was set to 4, 200 estimators are used and the rest of the parameter as default, were the following:

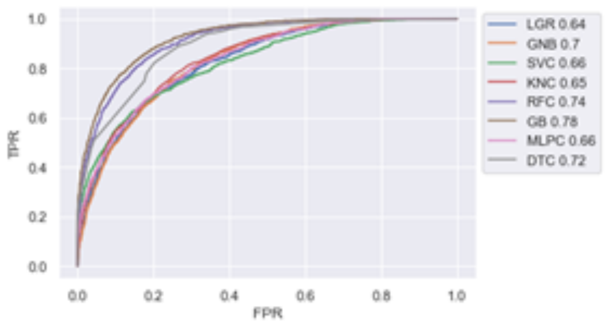
```
         Time         Train     Validation  f1score train  f1score validation
6.188+/-0.12  0.893+/-0.0  0.869+/-0.01    0.893+/-0.0          0.869+/-0.01
```

The rest of the models, regardless of the tuning, didn't reveal any potential so we decided not to address them in the report. Nevertheless, the results of these algorithms can be observed in the notebook.



From the bar plots above, that display the F1-micro score of each model in both the train and validation dataset, it is evident, as we previously stated, that Gradient boosting is the best model and was elected to be the one used on our final prediction.

We also plotted a ROC curve for each model and compared the AUC (area under curve). As expected, the Gradient Boosting Model has the highest AUC which reinforces the idea that this model is the most accurate one at classifying the income of the citizens.

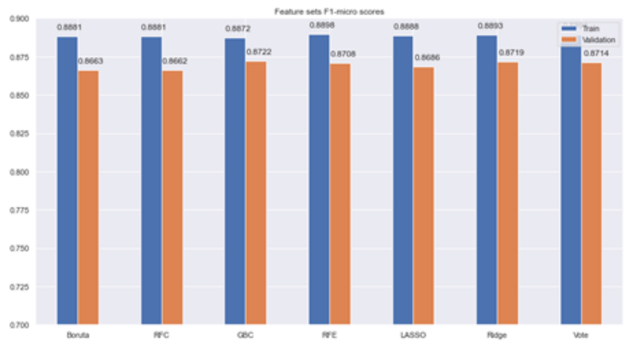


At this stage, after having our best model elected, we decided to go back and test if the model performed better with or without the outliers. The results were conclusive, and we decided to keep them as including them revealed to have an improvement on the model's performance.

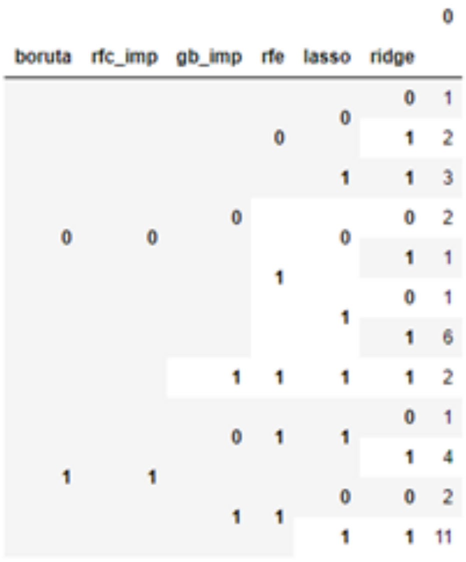
Even though all the desired transformations were applied to the data, there is still some filtration that could improve our results. In Machine Learning, the feature selection process is used with the intent of reducing the number of variables, preserving only the ones that are useful to the model to predict the target variable. This step is particularly important, since feature selection is used to reduce the complexity of the model and therefore makes it easier to interpret it, reducing the training time, enhancing generalization by reducing overfitting and reducing variable redundancy.

With resemblance to the model assessment part, also the feature selection section included several algorithms that were tested with the Gradient Boosting Classifier in order to find the best set of features.

As we could observe below, there aren't big discrepancies between feature sets.



We decided to use the one based on voting from all the models, since there weren't any algorithms that perished over the rest and this one is considered to be the most robust one as it takes information from all the others. The strategy is simple, if a feature is elected by more than 3 models, it will be kept.



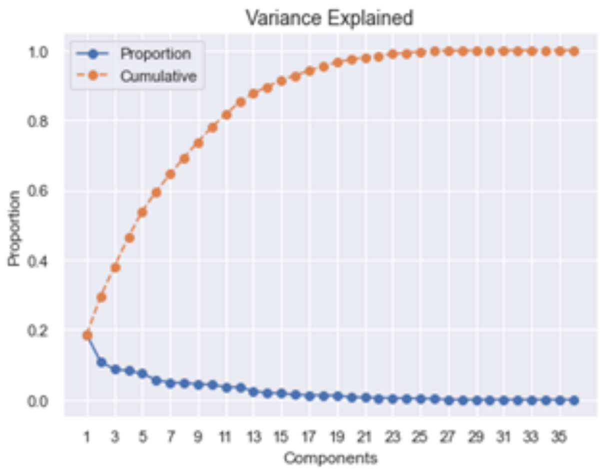
						0
boruta	rfc_imp	gb_imp	rfe	lasso	ridge	
0	0	0	0	0	0	1
					1	2
				1	1	3
			1	0	0	2
					1	1
				1	0	1
					1	6
		1	1	1	1	2
1	1	0	1	1	0	1
					1	4
		1	1	0	0	2
				1	1	11

In the end, we were able to improve the performance of the train to 88,9% and the validation to 87,1%.

Even though, we are aiming for the f1-micro score and this one concerns about the overall performance of the model and not how good it is to predict each class, giving them similar weights (like the normal f1-score), we decided to explore over and under sampling techniques to make our dataset more balanced in terms of the dependant variable.

As expected, since these techniques introduce some biases, our models got worse. For the validation dataset, the smote got an accuracy of 85,3% and the random under sampler of 84,1%.

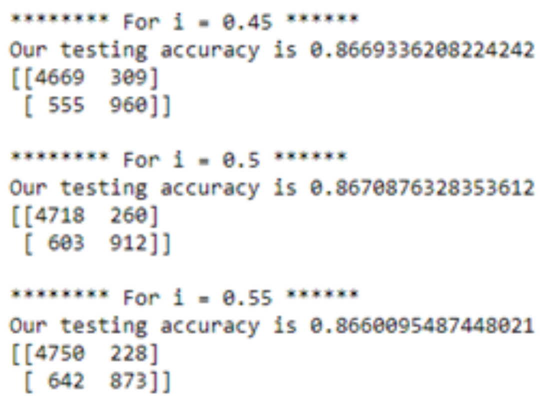
We also tested applying principal component analysis, that is the technique of reducing the number of variables by having a smaller number of orthogonal vectors that explain to some degree the variance of the features. This could be useful to deal with problems such as overfitting, however it decreased the performance of our model, so it wasn't applied to the data.



From the variance explained plot, we could identify 13 components to be the ‘elbow’, explaining around 90% of the variance of our dataset, so we tested the PCA with 13 components but in the end we got a result of 82,4% on validation so we discarded this approach.

Lastly, in our last attempt, before applying a grid search to obtain the best parameters for our model, we also experimented stacked models, with the hope that combining different models could improve the overall results of the prediction. For the automatic approach, using the stacking classifier, our model decreased the performance on validation to 86,8%.

As for the manual stacking, we adapted the code to enable the use of different datasets to different models, for instance, models like random forest can use non metric features while support vector classifiers do not perform well with those. Also, while aggregating the results from all the models, logistic regression uses 0.5 as the threshold probability where the class placed above this value is elected, we also ran some experiments with different thresholds for the probabilities, however 0.5 was still the best value and the final results of these stacking models couldn’t outperform the ones obtained previously.



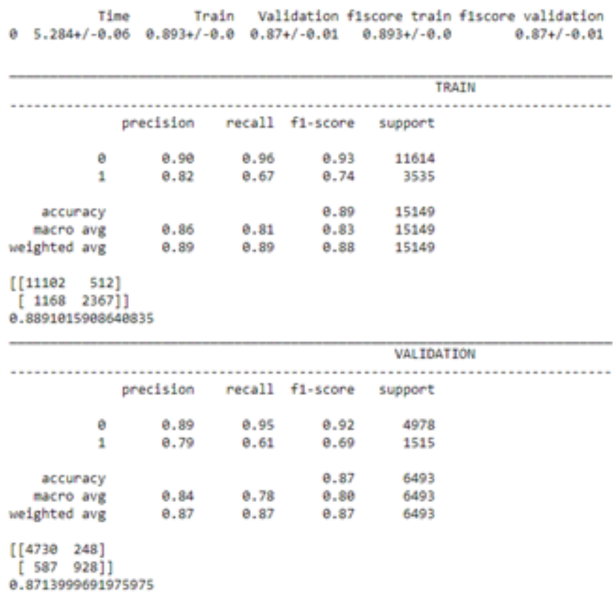
```
******** For i = 0.45 ******
Our testing accuracy is 0.8669336208224242
[[4669  309]
 [ 555  960]]

******** For i = 0.5 ******
Our testing accuracy is 0.8670876328353612
[[4718  260]
 [ 603  912]]

******** For i = 0.55 ******
Our testing accuracy is 0.8660095487448021
[[4750  228]
 [ 642  873]]
```

As mentioned above, in order to obtain the best model possible, we ran a GridSearchCV with different sets of parameters. With a total of 224000 fits, the algorithm found what it should be the best set of parameters, however, regardless of the excessive amount of time it took to obtain those, our model still performed worse than the previous configuration, setting only the number of estimators to 200 and max depth to 4.

The results of our final model where the following:



```
          Time         Train    Validation f1score train f1score validation
0  5.284+/-0.06  0.893+/-0.0  0.87+/-0.01   0.893+/-0.0          0.87+/-0.01

_______________________________________________________________________________
                                                   TRAIN
-------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.96      0.93     11614
           1       0.82      0.67      0.74      3535

    accuracy                           0.89     15149
   macro avg       0.86      0.81      0.83     15149
weighted avg       0.89      0.89      0.88     15149

[[11102   512]
 [ 1168  2367]]
0.8891015908640835
_______________________________________________________________________________
                                                   VALIDATION
-------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      4978
           1       0.79      0.61      0.69      1515

    accuracy                           0.87      6493
   macro avg       0.84      0.78      0.80      6493
weighted avg       0.87      0.87      0.87      6493

[[4730  248]
 [ 587  928]]
0.8713999691975975
```

This configuration also had the best results on Kaggle, with a micro f1-score of 86,03% on 30% of the test dataset. Score that we expect to increase once the evaluation is performed on the entire train dataset.

V. DISCUSSION

The results shown before were achieved by analyzing different approaches, both in handling the data and experimenting with the models. Despite this we feel like the results could still improve.

Indeed the dataset had a reasonable size, although splitting between training and validation and using cross-validation led to have not many data points for the training process. More data is always useful and every model can improve by adding more information.

The second thing that we thought could have improved this experiment was more variables. We had many categorical variables, that are time expensive to treat and not all the models can handle them, even after encoding them. More numerical variables would be useful in improving the models.

VI. CONCLUSION

For this project we focus a big part of time on handling the data. The dataset we were provided was far from raw data, although we felt that putting more effort in handling the data would help us later. Since the first steps we tried to implement different approaches and assess the best one only after passing through the models.

After selecting the best way to handle data, so treating categorical variables, outliers and which scaling to use, we started experimenting with the models. We tried many different models, and for each one we looked for the best parameters. Then we found that Gradient Boosting was the best performing one, and we proceeded to feature selection.

For this step, instead of using just one method, we combined different methods, such as Boruta, feature importance in classifiers, and others, trying to obtain a more generalized selection.

Before proceeding to submit the results, we first tried to combine the tuned models using stacking, although the results were not an improvement and we kept using Gradient Boosting.

VII. REFERENCES

[1] https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/

[2] https://www.statisticshowto.com/median-absolute-deviation/

[3] https://data-blog.gbif.org/post/outlier-detection-using-dbscan/

[4] https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[5] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[6] https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[7] https://www.kaggle.com/residentmario/oversampling-with-smote-and-adasyn

[8] https://towardsdatascience.com/all-you-need-to-know-about-pca-technique-in-machine-learning-443b0c2be9a1

[9] Schmidhuber, J., 2015. Deep Learning in neural networks: an overview. Neural Network.1 (6), 85–117.

[10] Bishop, C. M. (2011). Pattern Recognition and Machine Learning. Springer. Chapter 14 : Combinig models

[11] Zhang, C., Ma, Y., 2012. Ensemble machine learning: methods and applications. In: Ensemble Machine Learning: Methods and Applications.

[12] Mitchell, T. (1997) Machine Learning, McGraw-Hill; 1st edition (October 1, 1997) Chapter:8

[13] https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

[14] Binder, H., Gefeller, O., Schmid, M., & Mayr, A. (2014). The Evolution of Boosting Algorithms. Methods of Information in Medicine, 53(06), 419-427. doi:10.3414/me13-01-0122

[15] https://machinelearningmastery.com/k-fold-cross-validation/

[16] https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

[17] https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-1-4ece5098b755

[18] Castelli, M., Vannesch, L., Rubio Largo, _A., 2019. Supervised learning: classification. In: Guenther, R., Steel, D. (Eds.), Encyclopedia of Bioinformatics and Computational Biology. Elsevier, Oxford, pp. 342–349.