ML Project Report: Group MAA_202021_37

Francisco Hermenegildo (m20200737@novaims.unl.pt), Gil Gonçalves (m20201066@novaims.unl.pt),
Ikram Bouziri (m20200753@novaims.unl.pt), Onyealisi McBenny (m20201440@novaims.unl.pt),
Zhenghao Li (m20201007@novaims.unl.pt)

Abstract: This project provides insights on the various steps required to create a classification model that predicts with high accuracy a binary target variable in a dataset.
It ranges from the early Pre-Processing of the data, where we optimize it through removing outliers, transforming and feature engineering some of the variables, to the tweaking of the chosen model parameters to achieve the best possible score.
Throughout the entire project, we will be making use of Machine Learning techniques learned during practical classes and some learned on the internet while searching for improvements and fixes to the numerous problems that came up along the way.

Keywords: Machine Learning, Predictive Models, Random Forest, Gradient Boosting Classifier, Newland.

I. Introduction

A mission called "Newland" was launched in 2046, on a planet that was discovered in our galaxy with conditions similar to those found on planet Earth. This mission aims to inhabit the new planet; therefore 3 populated spaceships were sent, each with a capacity of about 40,000 people. Although most of the selected people are volunteers, some people received money to participate and others paid to participate in the mission.

In 2048, the Newland government decided to implement a binary tax rate, where people who receive less than average are taxed 15% of their income, and the ones with their income above-average are taxed 30%.

The goal of this project is to create a model that predicts each individual's income class, where individuals who receive less than average are assigned an income class of 0, and the ones that receive above are assigned a 1.

II. Background

Synthetic Minority Oversampling Technique (SMOTE) and GridSearchCV

The search for optimizing the parameters of the models led us to find GridSearchCV online. It's a function that receives three inputs: Parameters, Model, and the number of iterations (CV).
It iterates over the selected parameters options and returns the optimal choice.

SMOTE was an option that we did not cover during classes, but it fills the same purpose as Over and Under-Sampling, which we did cover. SMOTE works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

III. Project

3.1 Initial dataset analysis

All the records of the Newland mission are stored in two datasets, Train and Test, where the first (with 22400 rows), is intended to be used, as the name suggests, to develop and train our model and the latter (with 10100 rows), to test its prediction accuracy. During the entire project, only the Train dataset was used.

Before any transformation in the pre-processing stages occurred, an initial analysis of the dataset was needed to obtain a good overview of all its characteristics. This dataset is composed of 15 columns, being 4 metric features, 9 non-metric features and with 1 binary target variable.

Metric Features				**Target**
Years of Education	Working Hours per week	Money Received	Ticket Price	Income

Table 1 .Metric Features

Being the most correlated with income:

Variable	Pearson Correlation
Years of Education	0.3
Working Hours per week	0.2
Money Received	0.2

Table 2. Pearson Correlation

Non-Metric Features

Citizen ID	Birthday	Name	Native Continent	Marital Status	Education Level	Employment Sector	Role	Lives With	Base Area

Table 3. Non-Metric Features

3.2 Data Pre-Processing

Data Pre-Processing is an important step to prepare the data for further analysis. In this step, we perform data cleaning, feature selection, data transformation and feature engineering to prepare the information in a way that eases the model creation.

3.2.1 Data Cleaning

Missing Values

After an initial overview of the data, it was immediately identified the presence of incoherent data, symbolized with "?" , being interpreted as missing data and consequently replaced by Nan. Since this lack of values was only present in categorical features, all of them were replaced by the correspondent mode.

Outlier Removal

After the identification and treatment of the missing values, it was necessary to obtain a more in-depth knowledge of the distribution of all variables of the data. In this way, it was possible to identify in a first glimpse for possible outliers.

	CITIZEN_ID	Years of Education	Working Hours per week	Money Received	Ticket Price	Income
count	22400.000000	22400.000000	22400.000000	22400.000000	22400.000000	22400.000000
mean	23685.500000	13.173884	40.483795	1324.915357	109.145313	0.237098
std	6466.467351	2.512451	12.370921	9227.771813	500.208904	0.425313
min	12486.000000	2.000000	1.000000	0.000000	0.000000	0.000000
25%	18085.750000	12.000000	40.000000	0.000000	0.000000	0.000000
50%	23685.500000	13.000000	40.000000	0.000000	0.000000	0.000000
75%	29285.250000	15.000000	45.000000	0.000000	0.000000	0.000000
max	34885.000000	21.000000	99.000000	122999.000000	5358.000000	1.000000

Figure 1 - Overview of numeric features

The goal of Outlier Removal is to reduce noise, thus improving the training of the dataset in the chosen model. The variables "Working Hours per Week" and "Years of Education" were the only variables that, from the group's perspective, contained possible univariate outliers, making it necessary to visualize the respective boxplot.

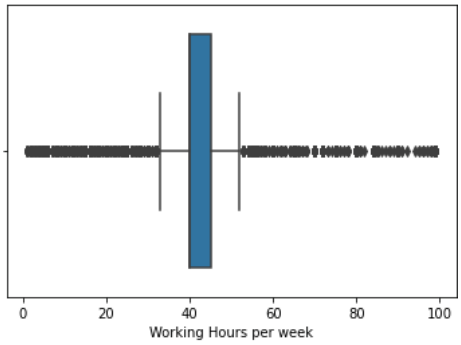

Figure 2 - Boxplot of "Working Hours per Week"

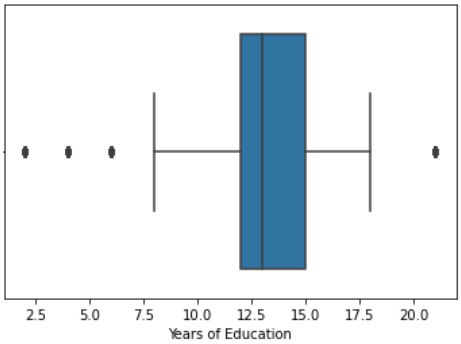

Figure 3 - Boxplot of "Years of Education"

The first approach to remove the outliers from both variables was the IQR method, although, in the end, it proved not to be ideal since it removed too much data (approximately 20%). Instead, we decided to pursue a manual approach while keeping below 4% of the total data.

Several combinations were used, below are some of the combinations we tried to apply:

- Removing entries with less than 3 “Years of Education”,
- Removing only the entries with above 70 “Working Hours per week”,
- Removing only the entries with below 10 “Working Hours per week”,
- Combining the removal of the three above

Despite all these combinations, the one that ensured the best results was the removal of entries with more than 70 and less than 10 “Working Hours per Week”.

3.2.2 Data Transformation

In this step, a decision was made to cluster some of the values of the categorical variables in accordance with their proportion to the Income Class 1. This decision was taken to reduce the number of categorical values in mind.

After the Outlier Removal step, Income Class 1 is 23.8% of the total entries. So we kept the same rationale for every variable and named the new clusters accordingly.

The variables we transformed are the following:

1. **Role**

```
data_1.groupby(['Role'])['Income'].value_counts(normalize=True)
```

Out[6]:

Role	Income	
Administratives	0	0.860528
	1	0.139472
Agriculture and Fishing	0	0.887658
	1	0.112342
Army	0	0.800000
	1	0.200000
Cleaners & Handlers	0	0.936334
	1	0.063666
Household Services	0	1.000000
IT	0	0.700647
	1	0.299353
Machine Operators & Inspectors	0	0.882824
	1	0.117176
Management	0	0.516940
	1	0.483060
Other services	0	0.961416
	1	0.038584
Professor	0	0.653077
	1	0.346923
Repair & constructions	0	0.780762
	1	0.219238
Sales	0	0.738761
	1	0.261239
Security	0	0.700229
	1	0.299771
Transports	0	0.812439
	1	0.187561

Name: Income, dtype: float64

Figure 4 - Income proportion of Role

According to what is above, we aggregated in 5 buckets, each one with the following ranges:

Range	Bucket	Role
0-7%	VeryLowPaidJobs	Other Services / Household Services / Cleaners & Handlers
10-14%	LowPaidJobs	Administratives / Agriculture and Fishing / Machine Operators & Inspectors
20-26%	MediocrePaidJobs	Army / Repair & Constructions / Sales / Transports
29-35%	AboveAVGPaidJobs	Security / IT / Professor
48%	BestPaidJob	Management

Table 4. Clustering of Role

We ended up not applying this to our final version.

2. Native Continent

```
In [12]: data_1.groupby(['Native Continent'])['Income'].value_counts(normalize=True)

Out[12]: Native Continent  Income
         Africa            0         0.882326
                           1         0.117674
         America           0         0.881517
                           1         0.118483
         Asia              0         0.743323
                           1         0.256677
         Europe            0         0.745939
                           1         0.254061
         Oceania           0         0.926966
                           1         0.073034
         Name: Income, dtype: float64
```

Figure 5 - Income proportion of Native Continent

The rationale of this variable is by far the easiest:

- Both Europe and Asia have the same proportion of Income Classes.
- Africa and America have the same proportion of Income Classes.
- Oceania, although closest to Africa and America, was allocated in its own cluster.

We aggregated the values in 3 different clusters:

Bucket	Continents
EuroAsia	Europe / Asia
AfricaAmerica	Africa / America
Oceania	Oceania

Table 5. Clustering of Native Continent

3. Lives With

```
In [8]: data_1.groupby(['Lives with'])['Income'].value_counts(normalize=True)
```

```
Out[8]: Lives with       Income
        Alone            0         0.931808
                         1         0.068192
        Children         0         0.986168
                         1         0.013832
        Husband          0         0.513327
                         1         0.486673
        Other Family     0         0.895046
                         1         0.104954
        Other relatives  0         0.967066
                         1         0.032934
        Wife             0         0.557807
                         1         0.442193
        Name: Income, dtype: float64
```

Figure 6 - Income proportion of "Lives With"

Following the same rationale as before, we aggregated in 4 buckets, each one with the following ranges:

Range	Bucket	Variables
1-3%	ChildOtherRelative	Children / Other relatives
7%	Alone	Alone
10%	OtherFamily	Other Family
44-49%	HusbWife	Husband / Wife

Table 6. Clustering of Lives With

4. Marital Status

```
In [10]: data_1.groupby(['Marital Status'])['Income'].value_counts(normalize=True)
```

```
Out[10]: Marital Status               Income
         Divorced                     0         0.893738
                                      1         0.106262
         Married                      0         0.558832
                                      1         0.441168
         Married - Spouse Missing     0         0.895683
                                      1         0.104317
         Married - Spouse in the Army 1         0.538462
                                      0         0.461538
         Separated                    0         0.937042
                                      1         0.062958
         Single                       0         0.951841
                                      1         0.048159
         Widow                        0         0.909516
                                      1         0.090484
         Name: Income, dtype: float64
```

Figure 7 - Income proportion of "Marital Status"

Following the same rationale as before, we aggregated in 3 buckets, each one with the following ranges:

Range	Bucket	Marital status
4-7%	SingleSep	Single / Separated
9-11%	WidDivMSM	Widow / Divorced / Married - Spouse Missing
44-46%	MarriedSIA	Married / Married - Spouse in the Army'

Table 7. Clustering of Marital Status

5. Base Area

Following the same rationale as before, we aggregated the values in 6 different buckets, each one with the following ranges:

Range	Bucket	Role
0-5%	VeryLowInc	Aberuthven / Bellmoral / Cherrytown / Fanfoss /King's Watch / Woodpine / Ironforge /Mensfield
7-13%	LowInc	Auchenshuggle / Carlisle / Drumchapel / Laewaes / Lanercost / Pran / Tranmere / Willesden
18-20%	BelowAVGInc	Wigston / Marnmouth / Conriston
24-26%	AverageInc	Aerilon / Bellenau / Fool's March / Lewes / Northbury / Sharpton
28 - 33%	AboveAVGInc	Alverton / Aroonshire / Butterpond / Kirkwall / Middlesbrough / MillerVille / Orilon / Redwick Bush / Watford
39 – 45%	WayAboveAVGInc	Eelry / Kald / Knife's Edge / Laenteglos / Sharnwick

Table 8. Clustering of Role

6. Employment Sector

In this step, we replace the values 'Never Worked' and 'Unemployed' with 'None'.

3.2.3. Feature Engineering

We tried to obtain the most out of the dataset by engineering to our best knowledge all the features present in the dataset:

- We created an 'Age' column from the 'Birthday', being the current year 2048.
- We created a 'Gender' column from the 'Name', where Mr. is a Man and everything else Woman.
- We created a 'Group' column from the 'Money Received' and 'Ticket Price', where the citizens with 'Money Received' > 0 were assigned a 'B' and those with 'Ticket Price' > 0 were assigned a 'C'. All the others were assigned an 'A'.

3.2.4. Data Removal

There were some redundant variables we decided to drop:

- 'Education Level' - Being a categorical representation of 'Years of Education' and highly correlated with each other.
- 'Birthday' - As it didn't provide us with any improvement in terms of accuracy of prediction. Even after transforming it to age.
- 'Name' - No real use for this variable even after we got the gender out of it.

3.2.5 One Hot Encoding

Since machine learning algorithms require only numerical input, we used the One Hot Encoding technique to convert categorical data into numerical data in order to be able to implement the models.

After using the one hot encoding, we created a data frame with the one-hot encoded categorical features that we concatenated with the numerical variables.

	x0_AfricaAmerica	x0_EuroAsia	x0_Oceania	x1_AboveAVGInc	x1_AverageInc
0	0.0	1.0	0.0	0.0	1.0
1	0.0	1.0	0.0	0.0	1.0
2	0.0	1.0	0.0	0.0	1.0
3	0.0	1.0	0.0	0.0	1.0
4	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...
21739	0.0	1.0	0.0	0.0	1.0
21740	0.0	1.0	0.0	0.0	1.0
21741	0.0	1.0	0.0	0.0	1.0
21742	0.0	1.0	0.0	0.0	1.0
21743	0.0	1.0	0.0	0.0	1.0

Figure 8 - Categorical variables after OHC

3.2.6 Feature Selection

The first OHC to our data resulted in more than 90 features and made clear it was necessary to select the most useful ones for our model to better predict the target variable, as we knew the noise associated with having several variables involved in the training of the model was going to be a problem later.

Three different feature selection techniques were used, RFE, Ridge and Lasso regression, all giving mixed and contradictory results:

Technique	Variables to keep
RFE	**All except:** Age, Money Received, Ticket Price, CITIZEN ID, and x5_Administratives
LASSO	Age, Money Received, Ticket Price
RIDGE	**All except:** Age, Money Received, Ticket Price, Citizen_ID, x5_Administratives

Table 9. Feature Selection Techniques and respective results

Since, in the end, we verified that we couldn't rely 100% on these techniques to select the best features, therefore we pursued the bucketing strategy of the categorical variables.

The OHC-dataset ready for the split was constituted by 41 variables:

- 'Citizen_id' as Index
- All the Clustered Variables (ClustContinent. ClustBase, ClustLiveWith and ClustMarital)
- The previously categorical variables (Employment Sector and Role)
- The already numerical variables (Working Hours per week, Ticket Price, Money Received and Years of Education)

3.3 Train Test Split

Having completed the data pre-processing steps, it was necessary to split our data into train, validation and test, making it possible to develop, optimize and apply all the different models. Since we were working with an imbalanced dataset, a stratified split was necessary, making it possible to maintain the same distribution of target labels across all subsets.

Subset	Split
Train	60 %
Validation	20 %
Test	20 %

Table 10. Train, test, validation split percentages

3.4 Models

The scope of this project is based on the implementation of a classification algorithm to predict if a certain person receives above average (1) or not (0). We worked with all the possible classifiers in order to see, in a first approach, what kind of score we were facing in a very early stage. The models are the following:

- LogisticRegression (LR)
- RandomForestClassifier (RF)
- GaussianNB (NB)
- GradientBoostingClassifier (GBC)
- DecisionTreeClassifier
- Support Vector Machine (Svm.SVC)
- LinearDiscriminantAnalysis (LDA)
- QuadraticDiscriminantAnalysis (QDA)
- KNeighborsClassifier (KNN)
- AdaBoostClassifier

The models with the highest score are Random Forest, Gradient Boosting and AdaBoost Classifiers.

3.4.1 Parameters of Models

In order to get the best scores out of each of the already best-scored models, we searched the internet for answers and came across GridSearchCV. A tool designed to help tune the parameters of each model, thus achieving the best score possible.

Random Forest
Because it was one of the highest-scoring models, we decided to search for the best parameters to apply to it by using GridSearchCV. Our conclusion was to change 2 parameters:

- max_depth = 19
- n_estimators = 700

which resulted in a very slight increase in the accuracy of 1's.

Gradient Boosting

Throughout all the tests, most often proved to be the highest-scoring model. Providing the best score with default parameters and after changing them. We also only changed 2 parameters:

- min_samples_split = 90
- n_estimators = 600

this increased both the accuracy of 0's and 1's.

AdaBoost

This model also revealed high scores with default parameters and increased marginally the accuracy after changes were done to them:

- n_estimators = 400
- learning_rate = 0.6

this increased both the accuracy of 0's and 1's.

3.5 Synthetic manipulation of data

3.5.1 Imbalanced Data

Since the beginning of the project, it was stated the presence of a class-imbalanced dataset resulting in a significantly lower F1-score for the minority label. In an attempt to balance the data and ultimately improve the classifier accuracy, techniques such as Under Sampling, Over Sampling and SMOTE were applied, however, all of them seemed to worsen the accuracy.

3.5.2 Data Scaling

Scaling techniques such as Min-Max Scaler and Standard Scaler were also implemented in the training data in order see if any kind of improvement of accuracy occurred, however, as expected, there was no effect since GBC is a tree-based classifier, and thus not being sensitive to the scale of the features.

IV. Results

The storytelling of the Project

This project was the first contact with creating a Machine Learning predictive model for all of us, therefore, it was a process of going back and forth between applying and removing changes and implementing solutions to achieve a consistent result between our local score and Kaggle submission score.

The starting step was to brainstorm what could be done to clean the data and improve the amount of information extractable from it. The conclusions were: feature engineering and pre-processing the data.

One of the first challenges we faced at an early stage of the project was deciding which features were the most important to keep, as the pre-processing of the data was still in a primitive state, and having yet to cluster the variables, we obtained roughly 80 variables after One Hot Encoding the data. This raised many questions on how we should reduce this number, as we knew this was going to be a problem later on with all the noise associated with having several variables involved in the training of the model.

That was the moment when the decision to cluster most categorical variables was made.

We started by aggregating the data in 2 clusters on each categorical variable as it seemed to be a good approach towards reducing the number of total variables. This fix brought better results to our post-OHC data frame, where we ended up having 30 variables to work with, but reduced in what is our opinion, the individuality of each citizen by generalizing them into almost identical entries.

Thus, our second approach towards the clustering of the categorical variables led us to:

- Drop ‘Education Level’, as it was a mere categorical representation of ‘Years of Education’
- Cluster features ‘Role’, ‘Native Continent’, ‘Base Area’, ‘Marital Status’ and ‘Lives with’ all in respect to the proportion of 1’s and 0’s of Income Class.

We conducted a test for each change in the combination of variables by implementing one at a time and getting the scores to compare them.
These tests were conducted using GBC as the benchmark model for our comparisons.

This resulted in dropping the feature engineered created variables ‘Age’, ‘Gender’ and ‘Group’. We also noticed that the clustering of ‘Role’ decreased the accuracy of the model, leaving us no choice but to revert to the original status.

This closes the Pre-Processing of the data and results in a final dataset consisting of:

- ‘Citizen_id’ as the index
- ‘ClustContinent’, ‘ClustMarital’, ‘ClustBase’, ‘ClustLivesWith’, ‘Employment Sector’ and ‘Role’ as our categorical variables.
- ‘Years of Education’, ‘Working Hours per week’, ‘Ticket Price’ and ‘Money Received’ as our numerical variables.

This results in a 10 variable dataset pre-OHC and 41 post-OHC, which is the dataset that all the following actions were applied.

This part of the process started earlier with the identification of all the models that can be used towards the identification of the Income Class.

Classifier	Class 0 accuracy	Class 1 accuracy	Total accuracy
Gradient Boosting	0.92	0.68	0.87
Ada Boost	0.92	0.68	0.87
Random Forest	0.91	0.68	0.86
KNN	0.91	0.69	0.86
Logistic Regression	0.91	0.66	0.86
Decision Tree	0.91	0.66	0.85
LDA	0.9	0.63	0.85
Gaussian NB	0.89	0.66	0.83
SVM.SVC	0.89	0.43	0.81
QDA	0.66	0.54	0.61

Table 11. All models tested and respective scores

After a test run of all the models and understanding which ones could provide a good and balanced prediction ability, we stuck with just 3 from the initial 10. They are Random Forest, Gradient Boosting and AdaBoost Classifiers.

Since all the scores were very similar, we started investigating online about how to improve scores regarding each one of the models. That led us to use GridSearchCV to assess which would be the best settings of the parameters of each model.

After long hours of running GridSearchCV with different combinations of parameter values and some manual changes in the mix, we reached the parameters that enabled the best score of each model. They are:

Classifier	Optimized Parameters
GBC	n_estimators = 600, min_samples_split = 90
Random Forest	n_estimators = 700, max_depth = 19
Ada Boost	n_estimators = 400, learning_rate = 0.6

Table 12. Classifiers and their optimized parameters

The final scores after the new parameter values:

Classifier	Class 0 accuracy	Class 1 accuracy	Total accuracy
Gradient Boosting	0.92	0.72	0.88
Ada Boost	0.92	0.69	0.87
Random Forest	0.92	0.68	0.87

Table 13. Classifiers and respective scores after parameter optimization

The most accurate model was GradientBoosting according to both local (0.87928) and Kaggle submission (0.86831) scores.

There were also some changes to the train, test, validation split that were conducted manually in order to improve the score, namely changes to random_state and split percentages. We stuck with random_state= 40 in both test and validation splits.

The remaining option to increase the accuracy of the model, was to introduce a way of reducing the imbalance of the dataset, which led us to implement the following Synthetic Manipulators of Data:

- Over-Sampling
- Under-Sampling
- SMOTE

After trying out each one of the options above, and although the accuracy of the prediction of 1's improved, we consistently got a lower score on Kaggle, which led us to believe we were overfitting the model. This was discomforting to realize and meant there were no further improvements that we knew of that could be applied.

Results after SMOTE:

Classifier	Class 0 accuracy	Class 1 accuracy	Total accuracy
Gradient Boosting	0.87	0.88	0.87
Random Forest	0.86	0.87	0.87
Ada Boost	0.85	0.86	0.86

Table 14. Classifiers and respective scores with SMOTE

The final score is the best compromise we reached between local and Kaggle scores. Although we tried several options to find improvements, we realized there wasn't much room for progress from the initial scores.

V. Discussion

Since this is an academic project with very little complexity there is little relation with any project of target binary variable that we have come across, therefore, most conclusions we took are unique to this specific dataset/problem.

VI. Conclusion

This project was an important bridge between the theoretical and practical components of Machine Learning, making us understand some of the most frequent problems by facing them:

- Categorical variables can be encoded in different ways, and sometimes is difficult to decided which is the best approach.
- Applying a Machine Learning algorithm is a process of "trial and error" and there is a lot of time and research involved in exploring new ways of improving a Machine Learning algorithm.
- On some occasions, theoretically, applying some changes should bring improvements to the accuracy of the model but that doesn't mean so in practical terms as we have verified with 'Age' and 'Gender' variables and also oversampling for example.

VII. References

All our references are from online searching, so we will leave some links regarding articles that inspired us on our project.

Hyperparameters tuning: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae
Machine Learning competition experience with some advice: https://link.medium.com/Ih1295Wskcb
SMOTE and how to use it: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/