Exploring and Evaluating Different Machine Learning Algorithms on a Classification Task

Fabio Rafael Santos Lopes (m20200597@novaims.unl.pt), Felipe Soares Costa (m20201041@novaims.unl.pt), Jorge Miguel Ramos Pereira Silvestre (m20201085@novaims.unl.pt)

Abstract—Classification is an important task on supervised learning that allow us define labels or classes to each element on a dataset. There's a variety of algorithms and techniques that can be applied to create a classification model. On this research several techniques were compared to find the best model that will be used to classify if the income of each individual on a dataset is equal or below the average. The study tested different algorithms, scale methods and resampling approaches. The F1 Score was used as evaluation metric. The results showed that the Ensemble Methods had the best performance overall. The final solution was an under-sampling approach with Tomek Links and GrandientBoosting Classifier.

Keywords : Classification,Supervised Learning, Python, Pandas, Sickit-Learn, F1 Score, Ensemble Methods...

I. INTRODUCTION

Supervised Learning is an important technique on Machine Learning that allow us to learn patterns and predict values of new instances based on previous information. For cases where the aim is to assign each input vector to one of a finite number of discrete categories, are called classification problems. If the values that will be predicted are continuous the task are called: Regression [1]. This research will focus on Classification problems and methods to support and solve this kind of task.

To create a robust classifier several aspects must be observed. It's important to look for potential problems on the data like missing values and outliers. Also, when the distribution of examples is much larger in one of the labels than the other this can cause bias towards the majority class. It's very important to select the most important features to avoid ambiguous or unnecessary information and to improve the training speed.

This research will compare different methods to build a robust Classification model. The data that will be used is part of a fictional problem: classify individuals that will be part of a new mission that will be launched to a new discovered planet called Newland.

To each individual that will be part of the mission a binary tax will be applied based on the individual income:

- 15% of the income if it is equal or below the average
- 30% otherwise

The goal is to classify the individuals on these two categories above using data from the previous mission.

Candidates from the previous mission were classified in three groups:

- Group A: Most of the people (volunteers) were carefully chosen through an extensive selection process.
- Group B: People that were payed to participate in the mission
- Group C: People that payed to participate in the mission

The dataset that will be used to build the model contains 22400 observations. The [Table 1] contains a brief description of each column of the dataset.

Table 1: Description of variables on the training set

Name	Description
Citzen_ID	Unique identifier of the citzen.
Name	Name of the citzen
Birthday	The date of birth
Native Continent	The continent of the citzen on Earth
Marital Status	The marital status of the citzen
Lives With	The household environment of the citzen
Base Area	Neighborhood of the Citzen on Newland
Education Level	The education level of the citzen
Year of education	The number the years of education
Employment Sector	The employment sector of the citzen
Role	The job role of the citzen
Working Hours per Week	The number of working hours per week of the citzen
Money Received	The money payed to the elements of Group B
Ticket Price	The money received by the elements of Group C
Income	The dependent variable (Where 1 is Income higher than the average and 0 Income Lower or equal to the average)

An Exploratory Data Analysis (EDA) process will be used to investigate and understand the data and build the final model. The programming language Python will be used to prepare the data and create the model. Mostly the libraries Pandas and Sickit-Learn [2]. The F1 Score will be the metric that will be used to evaluate the model performance. Different techniques will be compared to achieve the best performance and without overfitting. The final model will be submitted on Kaggle.

II. BACKGROUND

The proportion of the labels on the training set is one important factor that can have a huge influence on the model performance. When the training set has more data classified with one class rather than other, the resultant model can be biased to predict new data with this class. This problem is described as Class Unbalance problem. However, in some cases the Class Unbalance is intrinsic to the problem. On a fraud detection model, for example, the fraud instances will be much greater than the honest ones [3].

Different solutions can be used to deal with class Imbalance problem. For [Chawla et al. 2004], the solutions can be applied at the data level, which involves resample the data by adding or removing instances, or at the algorithm level by adjusting cost functions and other parameters.

[Kubuz. 2020] compared a set of resampling methods on unbalance class problem and conclude that combine Random Forests with undersampling methods returned the best results. [Moniz et al. 2017] study the application of resampling techniques on time series analysis and conclude that applying resampling combined with standard regression tools can lead to much better results.

This study will focus on the resampling algorithms provide by the Imbalanced Learn Library: Tomek Links, Condensed Nearest Neighbors, Near Miss, SMOTE and Random Under Sample, Random Oversample and the ADASYN.

II.a SMOTE

The Synthetic Minority Oversampling Technique, or SMOTE is an oversample approach that create examples in the minority class without adding any new information to the model. An example from the minority class is taken, and then the k nearest neighbors of this example are found. A random neighbor between these two examples are selected and then a synthetic example is created at a point selected randomly. Combine SMOTE with and under-sampling technique can lead to better results [6].

SMOTE create examples very similar to the real examples. However, since the examples are created without take the majority class in consideration, ambiguous examples may be created.

II.b Near Miss

The Near Miss is as a set of methods of Undersampling based on the distance of the majority to the minority class. There are three versions of the algorithm:

- NearMiss-1: Majority class examples with minimum average distance to three closest minority class examples.
- NearMiss-2: Majority class examples with minimum average distance to three furthest minority class examples.
- NearMiss-3: Majority class examples with minimum distance to each minority class example.

Basically, the algorithm removes the found instance of the majority class.

II.c Tomek Links

The Tomek Link techniques consists in finding a pair of instances of from different classes on a binary classification problem that has the smallest distance. These pair of instances are called “Tomek Link”. This approach may not necessarily balance the dataset but remove ambiguity. In practice, the Tomek Links procedure is often combined with other methods, such as the Condensed Nearest Neighbor Rule (CNN). The CNN is an undersampling method that uses the 1 nearest neighbor rule to decide if a sample should be removed or not. CNN is a relatively slow procedure and for that reason should be used on small datasets.

II.d ADASYN

ADASYN (Adaptive Synthetic Method Sample) is an oversampling technique that creates synthetic examples from the minority class. ADASYN uses a density distribution to find out the number of synthetic examples that needs to be generated for each instance of the minority class. This distribution is a measurement of weights for each minority class example according with their level of difficulty to learn. The final data set will not only be balanced but will also force the algorithm to focus on these difficult to learn examples.

[He et al. 2010] study the use of the ADASYN for Imbalanced Learning and conclude that the ADASYN approach improves learning with respect to the data distribution in two ways:

- Reducing the bias introduced by imbalance
- Shifting the classification boundary to difficult examples

II.e Random Undersampling and Random Oversampling

The Random Undersampling method randomly removes instances from the majority class on the training set. The process can be repeated until the correct number of instances of this class is achieved. This will potentially remove important information from this class since the instances are randomly removed. The Random Oversampling randomly select examples from the minority class with replacement and duplicate them on the training set. This can potentially overfit the minority class, so it’s important to observe the results on the training and test sets with and without oversampling. Better results may be obtained by combining the two approaches: apply a moderate random oversampling on the minority class to improve bias and a moderate random undersampling on the majority class to reduce bias.

III. Methodology

This study follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) model, which is a standard framework that helps on planning, manage and replicate data driven projects. The CRISP-DM process model aims to make large data mining projects, less costly, more reliable, more repeatable, more manageable, and faster [8]. The method splits the data mining project in six phases:

1. Business Understanding: On this phase the goal was understand the requirements and objectives for the project.
2. Data Understanding: On this phase an Exploratory Data Analysis was conducted to understand each feature and the distribution of the data. Also, look for missing values and potentially other problems on the data.
3. Data Preparation: On this phase all the problems discovered on the previous phase will be fixed and other important steps were conduct such as: Fill missing values, Feature Engineering, Data Scale, Train/Test Split.

4. Modeling: On this phase a set of algorithms were tested and compared. Also, feature selection and hyperparameters tunning are performed based on this model.
5. Evaluation: Finally, the model is evaluate using the confusion matrix and F1 Score.
6. Deployment: Set up the process on a production environment. On this study, this phase is used to produce the file that will be submitted on Kaggle.

The phases 1 through 3 were done once to each dataset (train and test). A set of different methods and algorithms were tested and compared on the phases 4 and 5:

- First, simple models with default hyperparameters, with all the initial features, and without scale were created.
- The best four models were selected to next step, and a set of resampling methods were applied to them. The best resample approach was selected on this phase.
- On the next step a set of scaling methods were tested and applied to the four models. The best scaling approach was selected on this phase.
- Then a feature selection method was applied to find the best features and the two best models were selected.
- Finally, a hyperparameters tunning was applied on the two models and the best model was selected.

The resultant model was used to classify the instances on the test set, and to get the final score on Kaggle.

IV. Results

At the beginning of the study, it was observed that the dataset contained some missing values identified with the question sign (?) [Table 2]. This data was replaced by the most frequent occurrence on the column: the mode.

Column	Total
Base Area	395
Employment Sector	1264
Role	1271

Table 2. Total of missing values by column

The dataset has more data classified with the label “0” rather than “1” [Fig. 1], which means that the dataset is imbalanced. This can potentially be a problem and the model can be biased to better classify the class “0”.

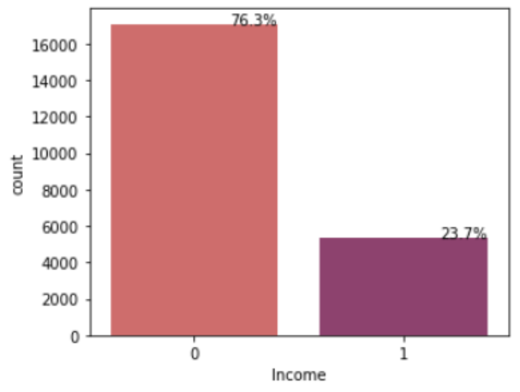


Fig. 1: Class Labels Distribution

As a result of the Feature Engineering process 4 new columns were created [Table 3]. A One-Hot Encoding approach was used to deal with the categorical variables. The method *get_dummies* of the Pandas DataFrame class was used for this purpose.

Name	Description
Age	Calculated using the Birthday column
Group	Using the Ticket Price and Money Received columns
MStatus	Binary column to indicates if the candidate is married or not.
LWith	Indicates if the candidate lives alone, with spouse, children or with other relatives

Table 3. Feature engineering

The Pearson correlation was computed using the method *corr* of the Pandas DataFrame. Since the columns Ticket Price and Group_C have a higher correlation (0.98), the column Group_C was removed from the Dataset [Fig. 2]. The same situation was observed between the columns Education Level and Year of Education, so the Education Class was removed from the dataset. Columns with correlation less than 0.1 were also removed from the dataset.

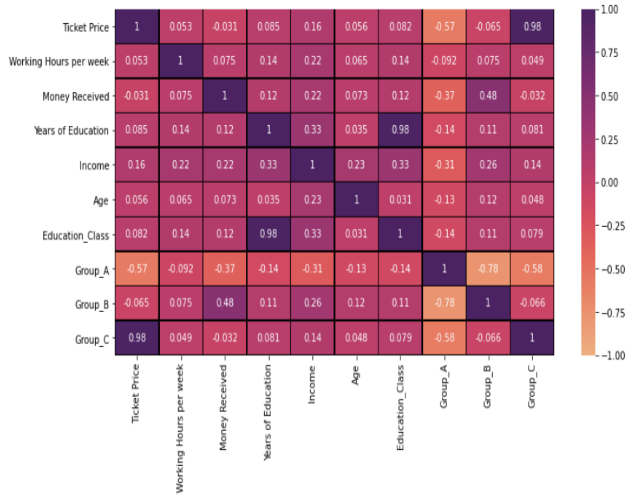


Fig. 2: Pearson Correlation

The Isolation Forest method was used to identify outliers. The parameter *contamination* was set to .05 (5%). To better visualize the results a Principal Component algorithm was used to extract 3 components and then plot them on 3D chart [Fig. 3].

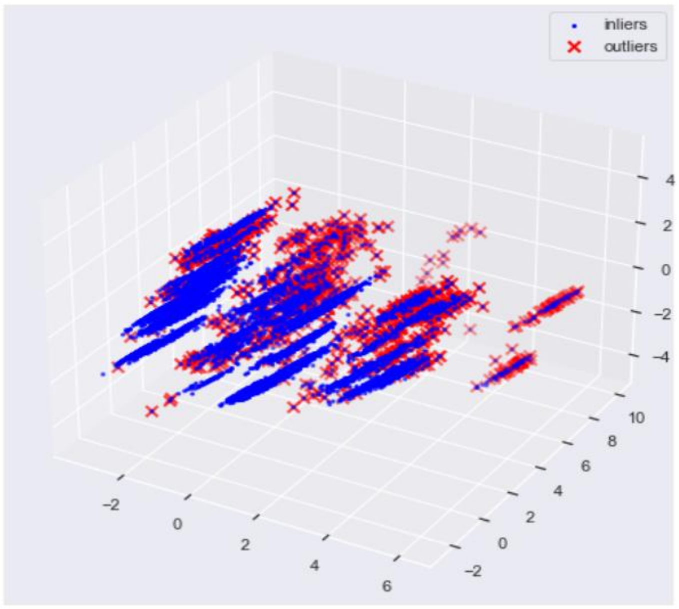
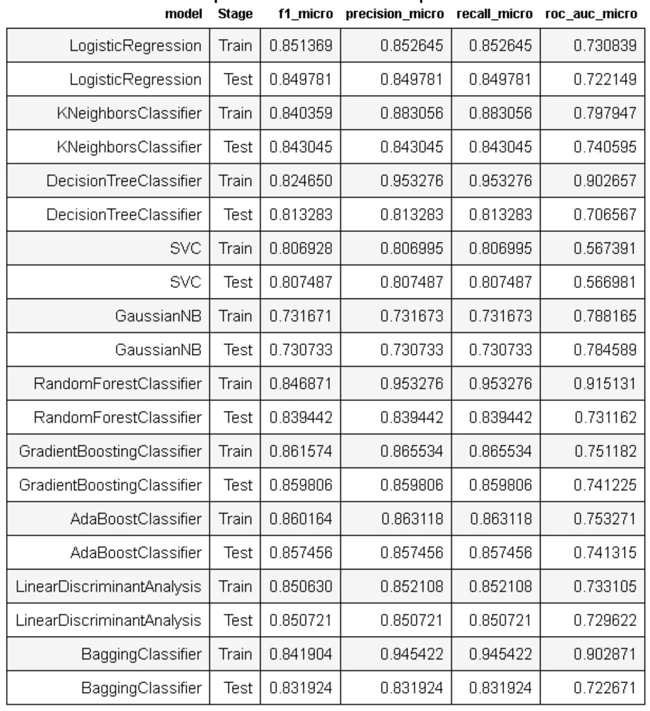


Fig. 3: Outliers Detected by Isolation Forest

On the Modeling step, a set of algorithms were defined with default hyperparameters, without applying any sampling strategy and without scaling. For each of the algorithms, f1_micro, precision, recall and roc_auc score was computed [Table 4]. All the algorithms applied on this phase are Sickit-Learn implementations. Four models were selected to next phase, based on the result of this step: LogisticRegression, GradientBoostingClassifier, AdaBoostingClassifier and RandomForestClassifier. To select these models the f1_micro was observed, and the difference between the train and test sets. The goal was to get the better perfomance avoiding overfitting. Gaussian Nayve Bayes and SVC were the models with the worse perfomance overall: GaussianNB presented a f1_score of 0.72 aproximatelly, wich is the lower score, and SVC despites of presents good scores (0.80~0.81), the roc_auc score was 0.57~0.59. The GradientBoosting and the AdaBoostClassifiers presented the best perfomance.



model	Stage	f1_micro	precision_micro	recall_micro	roc_auc_micro
LogisticRegression	Train	0.851369	0.852645	0.852645	0.730839
LogisticRegression	Test	0.849781	0.849781	0.849781	0.722149
KNeighborsClassifier	Train	0.840359	0.883056	0.883056	0.797947
KNeighborsClassifier	Test	0.843045	0.843045	0.843045	0.740595
DecisionTreeClassifier	Train	0.824650	0.953276	0.953276	0.902657
DecisionTreeClassifier	Test	0.813283	0.813283	0.813283	0.706567
SVC	Train	0.806928	0.806995	0.806995	0.567391
SVC	Test	0.807487	0.807487	0.807487	0.566981
GaussianNB	Train	0.731671	0.731673	0.731673	0.788165
GaussianNB	Test	0.730733	0.730733	0.730733	0.784589
RandomForestClassifier	Train	0.846871	0.953276	0.953276	0.915131
RandomForestClassifier	Test	0.839442	0.839442	0.839442	0.731162
GradientBoostingClassifier	Train	0.861574	0.865534	0.865534	0.751182
GradientBoostingClassifier	Test	0.859806	0.859806	0.859806	0.741225
AdaBoostClassifier	Train	0.860164	0.863118	0.863118	0.753271
AdaBoostClassifier	Test	0.857456	0.857456	0.857456	0.741315
LinearDiscriminantAnalysis	Train	0.850630	0.852108	0.852108	0.733105
LinearDiscriminantAnalysis	Test	0.850721	0.850721	0.850721	0.729622
BaggingClassifier	Train	0.841904	0.945422	0.945422	0.902871
BaggingClassifier	Test	0.831924	0.831924	0.831924	0.722671

Table 4: Perfomance of Simple Models

A set of resampling methods were tested to find the best strategy. The initial dataset had the following class label distribution: 0: 11693, 1: 3203. The algorithms used on this phase are Imbalanced Learn implementations [9]:

- SMOTE: The class distribution after applying the method was: 0:11693, 1:4677. The method didn't significantly affect the model.
- ADASYN: The class distribution with this method was: 1:11799, 0:11694. The worst result was with ADASYN. A lot of overfitting was introduced on the model [Fig 4].
- Random Over Sampler: This method introduced some overfitting as well. The class distribution was: 1:11694, 0:7016
- Random Under Sampling: On this approach the class label distribution was: 1:10673, 0:3202
- Tomek Links: This method results on the best performance. The class distribution after applying the method was: 1:11414, 0:3203. The model keeps the performance and didn't introduce overfitting.
- Tomek Links with Random OverSampling: Using this approach and changing the value of the sample_strategy parameter for both methods lead us to a model that had a good performance and was able to classify better both classes. However, when the model was tested on Kaggle the performance was a lot worse.
- SMOTE and Tomek Links: On this approach the class distribution was: 1:11528, 0:9355. The discrimination capability was a lot better as well, but also the performance on Kaggle was a lot worse.

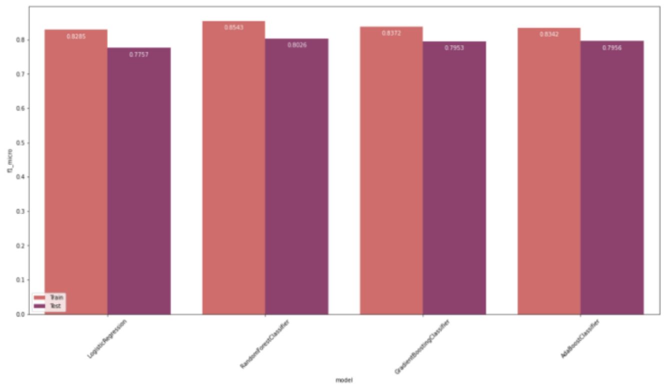
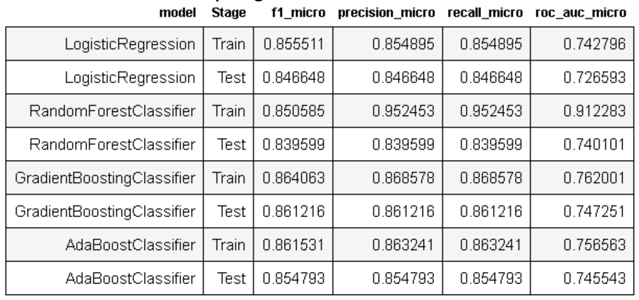


Fig. 4: ADASYN: F1 score Trainning X Test Set

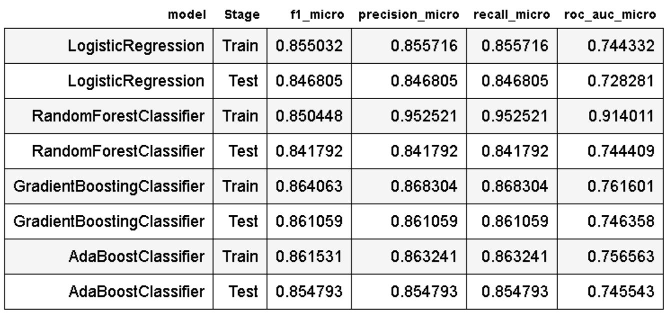
None of the methods significantly increase the performance of the model. Tomek Links was chosen due the better behaviour. The [Table 5] presents the performance of the Tomek Link Oversampling method in all models.



model	Stage	f1_micro	precision_micro	recall_micro	roc_auc_micro
LogisticRegression	Train	0.855511	0.854895	0.854895	0.742796
LogisticRegression	Test	0.846648	0.846648	0.846648	0.726593
RandomForestClassifier	Train	0.850585	0.952453	0.952453	0.912283
RandomForestClassifier	Test	0.839599	0.839599	0.839599	0.740101
GradientBoostingClassifier	Train	0.864063	0.868578	0.868578	0.762001
GradientBoostingClassifier	Test	0.861216	0.861216	0.861216	0.747251
AdaBoostClassifier	Train	0.861531	0.863241	0.863241	0.756563
AdaBoostClassifier	Test	0.854793	0.854793	0.854793	0.745543

Table 5. Oversampling Performance with Tomek Links

The same approach was used to find the best scale method. Robust Scaler, Standard Scaler and MinMax Scaler with ranges 0:1, and -1:1 were the methods analyzed. The results showed that there's not significantly difference between using any of these methods. The Min-Max Scaler with range -1:1, was selected to the next step, due to having a better performance across all the models [Table 6].



model	Stage	f1_micro	precision_micro	recall_micro	roc_auc_micro
LogisticRegression	Train	0.855032	0.855716	0.855716	0.744332
LogisticRegression	Test	0.846805	0.846805	0.846805	0.728281
RandomForestClassifier	Train	0.850448	0.952521	0.952521	0.914011
RandomForestClassifier	Test	0.841792	0.841792	0.841792	0.744409
GradientBoostingClassifier	Train	0.864063	0.868304	0.868304	0.761601
GradientBoostingClassifier	Test	0.861059	0.861059	0.861059	0.746358
AdaBoostClassifier	Train	0.861531	0.863241	0.863241	0.756563
AdaBoostClassifier	Test	0.854793	0.854793	0.854793	0.745543

Table 6. Performance with Min-Max Scaler (-1:1)

The Random Forest Classifier created on the previous steps was used to select the best features using the method SelectFromModel from Sickit-Learn [Fig. 5].

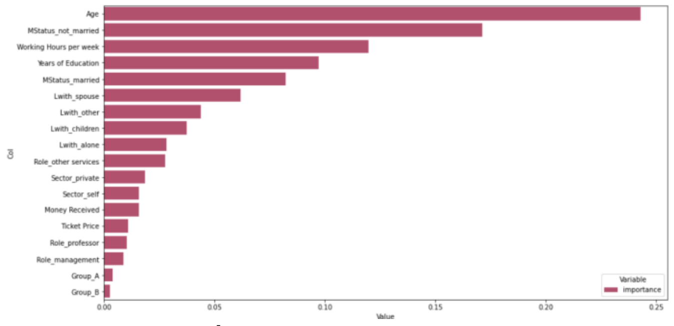


Fig 5. Feature Importance

The selected features were: Years of Education, Working Hours Per Week, Money Received, Age, Lwith_spouse, MStatus_Married, MStatus_not_married. To the next phase the best two models were selected: AdaBoosting and GradientBoosting.

Finally a GridSearchCV was performed on the two last models to find the best hyperparameters. Due the log processing time the tunning process was executed on stages. First the learning rate and n-estimator were tunned. The other parameters receive default values. Then the tree parameter were tunned: max_depth, num_samples_split, min_samples_leaf and max_features. Finnaly the learning_rate was decrease and the n_estimators incrase until find the best model. For Gradient Boosting the best hyperparameters were:

- n_estimators = 900
- learning_rate = 0.02
- max_depth = 11
- min_samples_split = 800
- min_samples_leaf = 30
- min_samples_split = 2000
- subsample = 0.8

The confusion matrix for the Gradient Boosting Classifier [Fig 6] indicates that the model is better in classify the class "0" rather than "1". Hower a slightly improvement on the classification was noticed.

For the AdaBoosting model, the parameters were tunned once since the trainning time is not so long. The best hyperparameters were:

- learning_rate = 0.1
- n_estimators = 1000

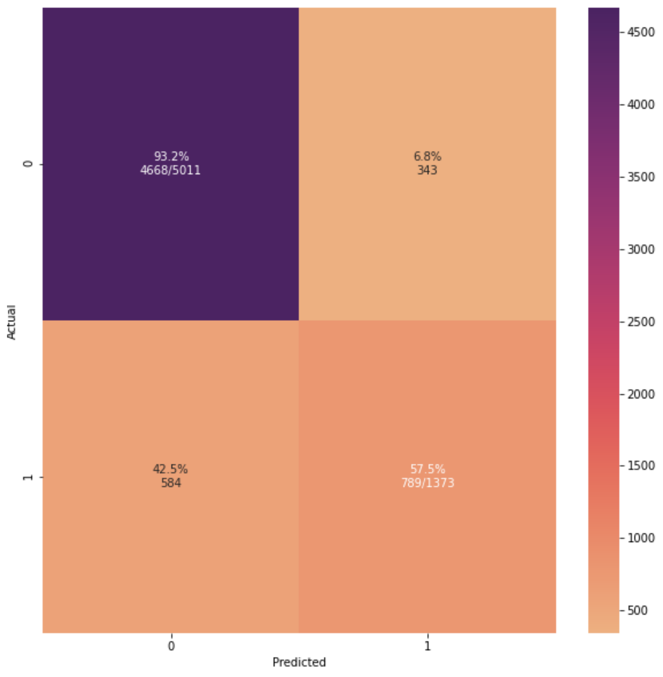


Fig. 6. Confusion Matrix for Gradient Boosting

On the confusion matrix for the AdaBoosting model [Fig. 7] is possible to observe the same situation as on the Gradient Boosting: the model is biased to the class "0".

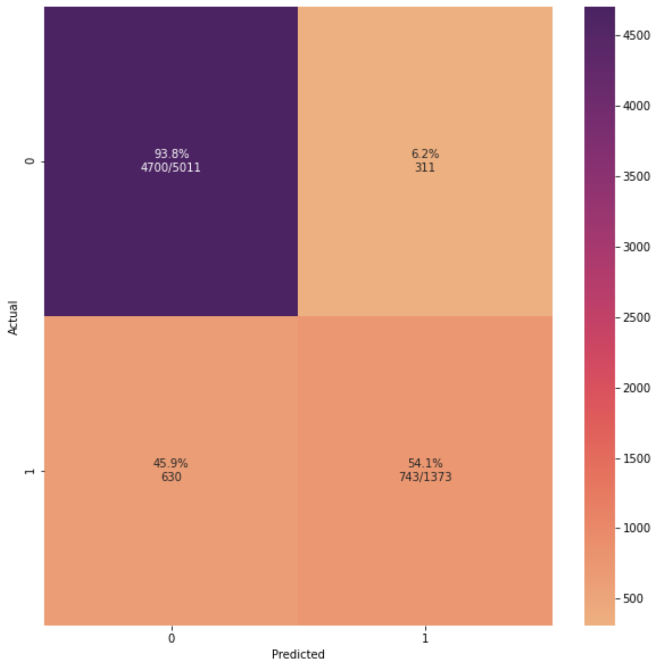
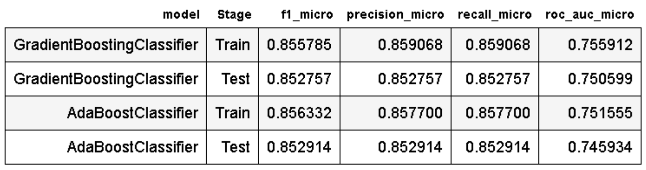


Fig 7. Confusion Matrix for Ada Boosting

[Table 7] sumarize the performance of both models. Both models had similar performances. The selected model was GradientBoosting, due the slightly better classification ability on the negative class. [Fig. 8] shows the ROC-AUC curve for GradientBoosting. The curve bows up to the top left of the plot, wich means that the model has a good performance.



model	Stage	f1_micro	precision_micro	recall_micro	roc_auc_micro
GradientBoostingClassifier	Train	0.855785	0.859068	0.859068	0.755912
GradientBoostingClassifier	Test	0.852757	0.852757	0.852757	0.750599
AdaBoostClassifier	Train	0.856332	0.857700	0.857700	0.751555
AdaBoostClassifier	Test	0.852914	0.852914	0.852914	0.745934

Table 7. Performance of Gradient Boosting and AdaBoosting

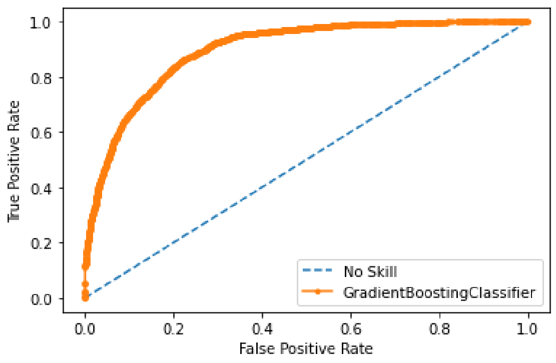
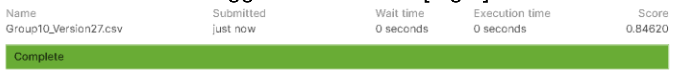


Fig 8. ROC-AUC Curve for Gradient Boosting

The final score on Kaggle was 0.84620 [Fig 9].



Name	Submitted	Wait time	Execution time	Score
Group10_Version27.csv	just now	0 seconds	0 seconds	0.84620

Complete

Fig 8. Final Score for Gradient Boosting

V. Discusion

Class Imbalance is a very common problem on Classification tasks. Different studies have been done on this field. Ensemble methods have been widely studied and used to try to solve this problem.

Liu Z. et al. Studied the highly imbalanced data on classification task and proposed a learning Framework, Self-paced Ensemble that aims to self-paced harmonize the data sing under sampling. The research compared the framework with different other methods. The framework proved to achieve a better performance, wider applicability, and higher computational efficiency,

Galar M. et al., compared and studied the use of Ensemble Methods for class Imbalance Problem, with focus on binary classification. The comparation has shown that simplest approaches that combine bagging or boosting ensembles with random undersampling achieve better results. The study also identifies a positive synergy between bagging methods with sampling techniques.

Feng W. et al., proposed an ensemble margin based algorithm, that handles imbalanced classification. The algorithm combines ensemble methods with undersampling techniques as well, but instead of balancing class randomly, the method creates higher quality balanced datasets for each base classifier. The study compared the method with different other to prove the effectiveness.

Santos M. S. et al, studied the problem of overfitting and overoptimisc approaches on imbalanced datasets. The research study and distinguish the both approaches and showed that overoptimistic is associated with cross-validation, and overfitting with resampling. The research also find that apply resampling in the training sets on each iteration is the correct way to perform validation on imbalanced scenarios.

Pastor-Díez et al., study the combination of Resampling techniques and ensembles methods. SMOTEBoost and SMOTEBagging are examples of methods that combine both techniques to solve the data imbalanced problem.

VI. Conclusion

This study compared different methods to build a machine learning classifier. Different resampling techniques were applied to try to solve the class imbalance problem. Some of the resample methods, gives us a better score and discrimination capability on both training and test sets but ,when the model was applied on the final dataset on Kaggle the score was a lot worse. The Tomek Links was the preferred method. The final dataset was not balanced but the performance, measured by F1-Score was not affected and was almost the same in both trainning sets.

The ensemble methods proved result on better solution with the best performances in all experiments. AdaBoosting and GradientBoosting were the methods with the best results and the methods compared at the end of the study.

The final model was the GradientBoosting Ensemble with a score of 0.86 on the test set and 0.84 on the Kaggle DataSet. Despise of the score the model proved to be not so good on predict the positive class.

On future work the proposal is study the application of resampling on each iteration of cross-validation to solve the overfitting problem. Also try different algorithms that combine Ensembes with Resampling techniques like SMOTEBagging, SMOTEBoost.

References

[1] Ch. M. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006

[2] Scikit-Learn: Machine Learning in Python. http://scikit-learn.org/stable/

[3] Chawla N.V., Japkowicz N., Kołcz A., 2004, Special issue on learning from imbalanced data sets, ACM Sigkdd Explorations Newsletter, 6(1),.1-6.

[4] Mariusz Kubuz, Evaluation of Resampling Methods in the Class Imbalance Problem, Econometrics. Eknometria: Advances in Applied Data Analysis, 2020.

[5] Moniz N., Branco P., Torgo L., Resampling strategies for imbalanced time series forecasting, Spring International Publishing Switzerland, 2017.

[6] Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P., 2002, SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research, 16, 321-357.

[7] He H., Bai Y., Garcia A. E., Li S., ADASYN: Adaptative Synthetic Sampling Approach for Imbalanced Learning, IEEE Xplore, 2010.

[8] Wirth R.,Hipp J., CRISP-DM: Towards a Standard Process for Data Minning, 2000.

[9] Imbalanced Learn, https://imbalanced-learn.readthedocs.io/

[10] Liu Z., Cao W., Gao Z., Bian J., Chen H., Chang Y., Liu T., Self-Paced Ensemble for Highly Imbalanced Massive Data Classification, IEEE 36th International Conference on Data Engineering (ICDE), 2020.

[11] Galar M., Fernández A., Barrenechea E., Bustince H., Herrera F., A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches., IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2011.

[12] Feng W., Huang W., Ren J., Class Imbalance Ensemble Learning Based on the Margin Theory, Applied Sciences, 2018.

[13] Santos S. M., Soares J. P., Abreu P. H., Araujo H., Santos J., Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2018.

[14] Díez-Pastor J. F., Rodríguez J. J:, García-Osorio C., Kucheva L. I., Random Balance: Ensembles of variable prior classifiers for imbalance data, Knowledge-Based Systems, ScienceDirect.