NAVIGATING IMBALANCED DATASETS

STRATEGIES FOR EFFECTIVE HANDLING AND MODEL ENHANCEMENT IN CREDIT CARD FRAUD DETECTION

Katarzyna Zbroinska

EC Utbildning

Data Science

Examensarbete

2024-02

Abstract

This work focuses on the development of an efficient credit card fraud detection model, based on the Kaggle Credit Card Fraud Detection dataset. A primary objective is to address the significant class imbalance in the dataset, emphasizing the identification of fraudulent transactions while minimizing false positives.

The research explores various strategies, highlighting the effectiveness of resampling techniques, particularly RandomOverSampler and SMOTENC. Performance analysis reveals that, on average, models incorporating SMOTENC outperform others. Hyperparameter tuning further refines the models, with the final model achieving the highest F-beta score using RandomOverSampler.

Acknowledgement

I would like to express my sincere gratitude to my teacher, Antonio Prgomet, for introducing me to the world of machine learning and data science and for his guidance and support throughout the course. His ability to present complicated subjects in an easy and understandable way has been invaluable to me. Antonio's remarkable ability to simplify complex subjects has greatly enriched my learning experience. His expertise and dedication are truly commendable.

I would also like to thank Márk Mészáros for his collaboration on this project. His contributions, ideas, and discussions have been instrumental in shaping the direction of this work.

Finally, I would like to extend my thanks to all the course mates for stimulating ideas, thought-provoking questions, and engaging discussions. Their feedback and insights have been invaluable in helping me to develop my data scientist skills.

Table of Contents

1 Introduction

Credit card fraud is a serious issue that affects both financial institutions and the individuals – credit card users. According to a study performed by the Association of Certified Fraud Examiners, organizations lose an estimated 5% of their annual revenues to fraud (Association of Certified Fraud Examiners, 2022, p. 4), with the financial services industry being among those affected by the greatest number of cases (p. 32)

In recent years, credit cards have experienced a notable rise in popularity, especially during the COVID-19 pandemic, and this trend continues to rise. In the first quarter of 2023 about 1.1 billion Mastercard credit cards were issued worldwide (Statista, 2023). This widespread adoption of credit cards has brought new challenges, with fraudulent transactions emerging as a significant issue, with the financial services industry being among those affected by the greatest number of cases.

1.1 Objectives

The primary objective of this research is to assist financial institutions in combating credit card fraud by developing an efficient fraud detection model based on the Credit Card Fraud Detection dataset from Kaggle (Worldline, the Machine Learning Group of Université Libre de Bruxelles, 2018). This model, based on specific patterns and anomalies, should be capable of flagging suspicious transactions for further investigation by Anti-Money Laundering Departments.

One of the main challenges in fraud detection arises from the significant class imbalance issue where the number of legitimate transactions far exceeds the instances of fraudulent ones. Most of the machine learning algorithms tend to focus on majority class, ignoring minority one or treating it as a noise. Therefore, the primary focus of the model is to identify as many fraudulent transactions as possible however still without compromising precision. While preventing monetary loss to fraudsters is crucial, it is equally important to avoid falsely flagging too many transactions as fraudulent. This is essential to prevent customer dissatisfaction and increased workload for Anti-Money Laundering Department employees.

1.2 Research Questions

1. What are possible ways to overcome the issue of class imbalance in the dataset? Is it possible to build an effective model for detecting credit card fraud based on a highly imbalanced dataset?
2. What metric or metrics can be used to evaluate the performance of such a model?

2 Theoretical Background

This section presents theory which is relevant to understanding the context of this project.

2.1 Evaluation metrics

In the field of machine learning, the selection of appropriate evaluation metrics plays a crucial role in assessing the performance of models, particularly in the context of imbalanced datasets. Traditional metrics, often insufficient in capturing the nuances of imbalanced class distributions, may misrepresent the true performance of a model. In the specific context of a fraud detection model, the main objective is to detect as many fraudulent transactions as possible, making high recall the primary focus. Simultaneously, there is a secondary emphasis on minimizing the number of False Positive transactions to ensure a balanced and effective model.

According to Géron (2019), several key metrics are commonly employed to measure the effectiveness of machine learning classification models:

- **Accuracy** is a most widely used metric in classification cases. Using it with highly imbalanced dataset can be misleading. Achieving a high accuracy score, particularly when the dataset is imbalanced, may not reflect the model's actual performance, as a classifier labelling all transactions as non-fraudulent could still yield an accuracy over 99%.
- **Precision** is a metric that measures the accuracy of positive predictions, emphasizing the ratio of correctly predicted positive observations to the total predicted positives (Géron, 2019, p. 91). For imbalanced datasets, precision is valuable, yet it alone does not account for false negatives, potentially overlooking instances of fraudulent transactions.
- **Recall**, also known as Sensitivity or True Positive Rate, is concerned with capturing the proportion of correctly predicted positive observations in relation to all observations in the actual positive class (Géron, 2019, p. 92). Recall is particularly useful in imbalanced datasets, focusing on identifying positive instances so fraudulent transactions. However, a high recall alone may lead to a significant number of false positives, marking a significant number of legitimate transactions as fraudulent.
- The **F1 Score**, serves as a harmonized measure, balancing the trade-off between precision and recall (Géron, 2019, p. 92), making it particularly useful in scenarios where achieving some balance between the two is essential. However, in an imbalanced dataset where the costs of false positives and false negatives are very different the metric is not very useful as it handles precision and recall as equally important factors.
- A **ROC Curve** is a plot which visualizes the performance of a classification model at various thresholds, emphasizing the trade-off between true positive rate and false positive rate (Géron, 2019, p. 97). It can be a good choice in case of imbalanced dataset. While it can be a good choice in cases of imbalanced datasets, in highly imbalanced scenarios, it may not effectively address issues in the minority class and may be overly optimistic.

The **F-beta score** is a much more suitable metric for highly imbalanced dataset. In the case of fraud detection models the main focus needs to be put on the identification of fraudulent transactions, but precision also needs to be considered. Balancing the trade-off between precision and recall, F-beta score uses a factor beta, allowing customization based on the importance assigned to recall. If β is less than 1, the F-beta score favors precision, and if β is greater than 1, it favors recall (Brownlee, A Gentle Introduction to the Fbeta-Measure for Machine Learning, 2020):

$$F_\beta = \frac{(1+\ \beta^2) * precision * recall}{(\beta^2 * precision) + recall}$$

This factor beta ensures flexibility in aligning the metric with the specific priorities of the fraud detection model, considering both precision and recall as important factors, but allowing customization based on the specific requirements of the application.

2.2 Addressing imbalance

In the domain of credit card fraud detection, the challenge posed by the high imbalance between legitimate transactions and fraudulent ones is substantial. The vast majority of credit card transactions are genuine, leading standard machine learning algorithms to be biased towards the majority class. This bias often results in suboptimal fraud detection performance, as the model may overlook or misclassify instances of fraudulent behavior. This chapter explores the methodologies designed to address issues related to imbalance in the dataset ranging from traditional sampling methods such as under-sampling and over-sampling to more advanced techniques like the Balanced Random Forest and models incorporating class weights as hyperparameters.

2.2.1 Sampling methods

Sampling methods offer a potential solution to this challenge, broadly categorized into undersampling and oversampling (Lemaître, Nogueira, & Aridas, 2017).

Undersampling reduces instances of the majority class, but it introduces the risk of losing potentially vital data and is less suitable for datasets with a significantly low number of minority class instances. Given that the project dataset includes only 284 fraudulent transactions, effective training may be compromised by insufficient instances.

Oversampling, on the other hand balances a dataset by increasing the number of instances of the minority class. This can be achieved through duplicating instances of the minority class or by employing the Synthetic Minority Oversampling Technique (SMOTE), which was introduced by N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer (2002).The drawback with the oversampling method is that even if it can lead to better performance, it may also cause the model to overfit to the minority class.

The **imbalanced-learn library** is a Python package designed to address the challenges posed by imbalanced datasets in machine learning applications, particularly in classification tasks. The library provides a set of tools and techniques for resampling the dataset, including two oversampling techniques which can deal with a dataset with categorical variables:

- **RandomOverSampler()**: A straightforward method that creates new synthetic samples by randomly sampling with replacement and duplicating current available samples of the minority class. (Lemaître, Nogueira, & Aridas, 2017)
- **SMOTE():** Using interpolation, this technique generates new samples adjacent to the original samples with the help of a k-Nearest Neighbors classifier. For datasets containing numerical and categorical variables, the SMOTENC extension is available, treating categorical data differently without interpolation or changes to the original categories. (Lemaître, Nogueira, & Aridas, 2017)

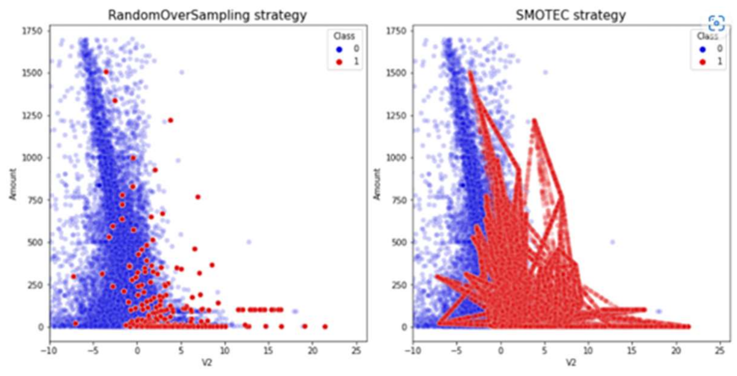


Figure 1. Visual demonstration of the two oversampling strategies.

The influence of the two types of oversampling method on the classification models' performance is included in the project.

2.2.2 Other ways of dealing with imbalance

In addition to traditional sampling methods, various machine learning models come equipped with inbuild capabilities to address class imbalance. One notable example is the **Balanced Random Forest classifier** from the imbalanced-learn library. This classifier employs a unique technique during the bootstrap sampling process, dynamically under-sampling each sample to ensure a balanced representation of classes. (Lemaître, Nogueira, & Aridas, 2017)

Another approach, available in the scikit-learn library, involves models featuring a **class weights hyperparameter**. For instance, classifiers like Logistic Regression or Decision Trees based in scikit-learn allow the adjustment of class weights. By adjusting these weights, the model can penalize misclassifications differently based on the class, offering an alternative way for managing imbalanced datasets. (Pedregosa, F., et al., 2011)

2.3 Feature selection

Feature selection is an important step aimed at identifying the most relevant variables that contribute significantly to the predictive performance of the models, thereby enhancing efficiency and reducing computational overhead in the context of credit card fraud detection. The goal is to streamline the model's learning process, improve interpretability, and potentially boost overall performance.

Following methods were considered for feature selection in the project:

- **Forward Regression:** A feature selection technique that begins with an empty set of features and iteratively adds the most influential feature at each step based on a predefined criterion.
- **Backward Regression**: In contrast to forward regression, backward regression starts with all features and removes the least significant one at each step until the optimal subset is achieved. (Pedregosa, F., et al., 2011)

  Both forward and backward regression methods were considered for feature selection in the project. However, it is essential to note that these methods are time-consuming and demand significant computational power. Given the resource-intensive nature of these techniques, the project also explored alternative approaches like mutual information for feature selection.
- **Mutual Information**: This technique measures the dependency between variables and is less computationally intensive compared to regression-based approaches. It measures the extent to which knowing the value of one variable reduces uncertainty about another variable. It's

similar to correlation in that it measures a relationship between two variables, but it can detect any kind of relationship. (Pedregosa, F., et al., 2011)

2.4 Models

The following machine learning models were tested during the project:

- **Logistic regression** is a linear model that uses the logistic function to predict the probability of an instance belonging to a particular class. (Géron, 2019, p. 142)
- **K-nearest neighbors classifier** – a model which makes predictions for a new entry based on the majority class of its k-nearest neighbors in the feature space. (Pedregosa, F., et al., 2011).
- **Gaussian Naive Bayes (GNB**) is a classification technique based on a probabilistic approach and Gaussian distribution. GNB assumes that each feature or predictor has an independent capacity of predicting the output variable. (Martins, 2023)
- **AdaBoostClassifier** (Adaptive boosting classifier) The core principle of AdaBoost is to fit a sequence of classifiers which are weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction (Pedregosa, F., et al., 2011).
- **Quadratic Discriminant Analysis** is a classification technique, it is similar of Linear Discriminant Analysis (LDA) however it allows each class to have its own covariance matrix. (Pedregosa, F., et al., 2011).
- **Multi-Layer Perceptron classifier** (MLP Classifier) operates by utilizing a layered neural network structure, where interconnected nodes process input data through hidden layers. During training, the algorithm employs backpropagation, iteratively adjusting the weights to minimize prediction errors and enhance its ability to classify and capture patterns in the data. (Géron, 2019, p. 289)
- **Extreme Gradient Boosting classifier** (XGB classifier) is an ensemble learning algorithm. It sequentially builds a series of decision trees, each addressing the errors of the previous ones, and combines their outputs through gradient boosting. This iterative process enhances predictive accuracy while controlling overfitting through regularization techniques. (Chen & Guestrin, 2016)
- **XGBoost Random Forest** (XGB RF) classifier is a combination of the XGBoost algorithm and the Random Forest framework. By combining the strengths of both approaches, it leverages the boosting technique of XGBoost with the ensemble learning and diversity of Random Forests. This hybrid model excels in handling complex relationships within data, providing high predictive accuracy while mitigating overfitting concerns. (Brownlee , How to Develop Random Forest Ensembles With XGBoost, 2021)
- **Random forest classifier** is an ensemble learning algorithm that constructs multiple decision trees during training and outputs the mode of the classes. Each tree is built on a random subset of features and contributes to the final prediction through a majority voting mechanism. (Géron, 2019, p. 197). Variations of the Random Forest model, such as the Balanced Random Forest Classifier, were also explored.
- **Voting classifiers –**building upon the principles of ensemble learning, voting classifiers based on the described models were subjected to testing. This approach involves training multiple models to make predictions, and the final prediction is determined by either a majority vote or a weighted vote of the individual models. This ensemble method is employed to harness the collective predictive power of diverse models, potentially enhancing overall performance in credit card fraud detection (Géron, 2019, pp. 189-192).

2.5 Avoiding data leakage

Another challenge arises when dealing with imbalanced dataset in context of the Grid- or RandomSearchCV usage with a sampler. It is not appropriate to balance the training dataset before the cross validation as during the cross validation a portion of the data will be used for model evaluation and this portion should be imbalanced as it needs to reflect the true distribution of the date. To avoid this data leakage, it is crucial to use the original imbalanced data for the evaluation process. This necessitates the use of the sampler during the cross-validation process, ensuring that only the folds used for training are balanced (Altini, 2015).

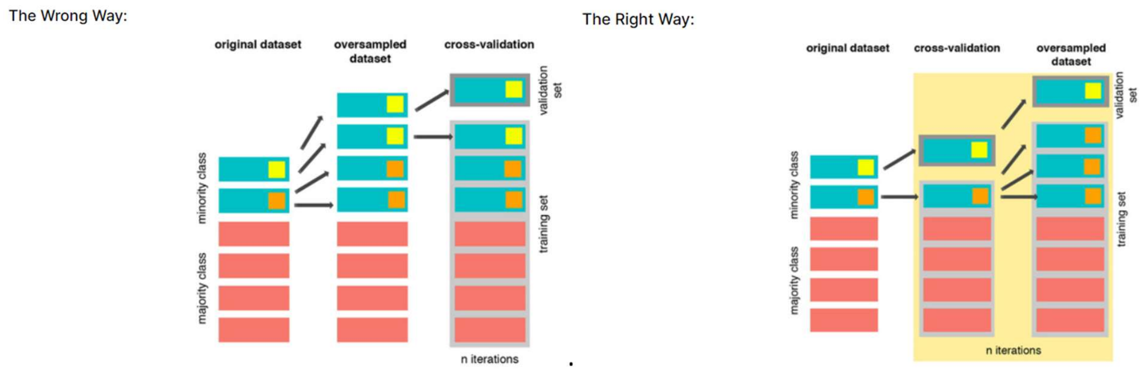


Figure 2. The difference in proper and improper implementation of cross-validation for imbalanced dataset.

As it is presented on the diagrams in Figure 2, in the improper way the original dataset is first oversampled and only then the cross-validation is done. As a result, the same instance as a duplicate (or interpolated in case of SMOTE) instance can be used both for training and for validation. In the appropriate step configuration first the validation sample is excluded and only then the remaining instances are balanced and can be used for the model training (Altini, 2015).

A good and simple solution to avoid data leakage is to use a special *imblearn.pipeline* from the imbalanced – learn library, which implements the proper way of sampling in the cross validation processes.

3 Method

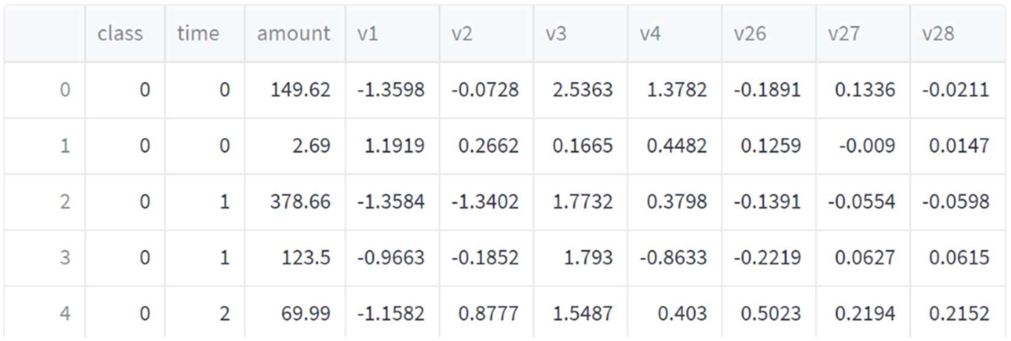
This chapter outlines our step-by-step approach to the project, covering data exploration with Exploratory Data Analysis (EDA), feature selection, model choice, parameter fine-tuning, and a thorough presentation of the final fraud detection model.

3.1 Dataset

The dataset utilized in this study, was collected, and analysed during a research collaboration of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles) (2018)on big data mining and fraud detection. The dataset contains transactional data from a two-day period in September 2013, encompassing a total of 492 fraud cases out of 284,807 transactions.

Several examples of transactions are provided in the table below. It is important to note that not all V features are presented in this excerpt.

Table 1. Examples of credit card transactions from the dataset.



	class	time	amount	v1	v2	v3	v4	v26	v27	v28
0	0	0	149.62	-1.3598	-0.0728	2.5363	1.3782	-0.1891	0.1336	-0.0211
1	0	0	2.69	1.1919	0.2662	0.1665	0.4482	0.1259	-0.009	0.0147
2	0	1	378.66	-1.3584	-1.3402	1.7732	0.3798	-0.1391	-0.0554	-0.0598
3	0	1	123.5	-0.9663	-0.1852	1.793	-0.8633	-0.2219	0.0627	0.0615
4	0	2	69.99	-1.1582	0.8777	1.5487	0.403	0.5023	0.2194	0.2152

The dataset includes:

- Features *V*1, *V*2, …*V*28 which are the principal components obtained with PCA. (Due to confidentiality issues more details couldn't be provided.)
- *Time* contains the seconds elapsed between a transaction and the first transaction in the dataset.
- *Amount* is the transaction amount.
- *Class* is the response/target variable, and it takes value 11 in case of fraud and 0 otherwise.
- All the features have numeric data; however, *Time* and *Amount* needs further analysis.
- *Class* is nominal categoric data.

The dataset contains 1081 **duplicate transactions**, defined as instances with identical values across all columns. The duplicates have been removed to ensure the integrity of the data as duplicated transactions can bias the model's performance metrics.

The primary challenge posed by the dataset is its **significant imbalance**, as fraudulent transactions represent a mere 0.17% of the total transactions. This imbalance necessitates careful consideration during model development and evaluation to ensure robust and accurate results in detecting and addressing fraudulent activities.

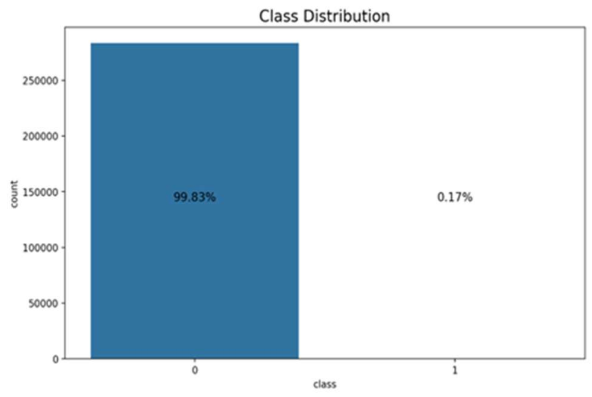
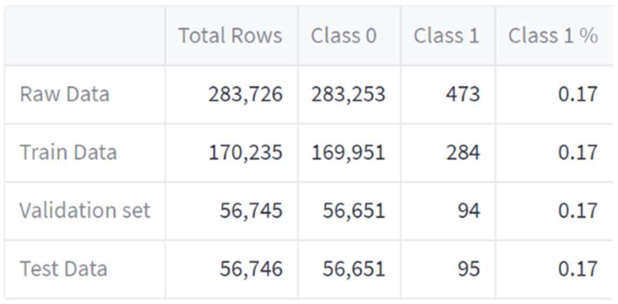


Figure 3. Distribution of the dependent variable.

The dataset has been **split** into three distinct sets: a training set comprising 60% of the data, a validation set encompassing 20%, and a test set also accounting for 20%. The stratification ensures that all the sets maintain a representative proportion of fraudulent transactions, contributing to a robust evaluation of the model.

Table 2. Division of Raw Dataset into Train, Validation, and Test Sets.



	Total Rows	Class 0	Class 1	Class 1 %
Raw Data	283,726	283,253	473	0.17
Train Data	170,235	169,951	284	0.17
Validation set	56,745	56,651	94	0.17
Test Data	56,746	56,651	95	0.17

3.2 Explanatory Data Analysis

Following the dataset split, the subsequent data analysis was exclusively conducted on the training dataset.

3.2.1 Missing values

The dataset has no missing values. After analysing the “0” values the following observations were made:

- Across v1, v2, ... v28 features, there are no instances of '0' values.
- There are 1068 transactions with 0 amount potentially indicating instances where the credit card was registered for future payments without any actual money flow.
- The presence of two transactions with a Time value of 0 suggests that these transactions were recorded at the exact moment the registration started.

Given that all "0" values have reasonable explanations, it is concluded that there are no missing values in the dataset. Nevertheless, due to the uncertainty of potential missing values in future transactions, our custom transformer includes a dedicated step for their handling.

3.2.2 Features description

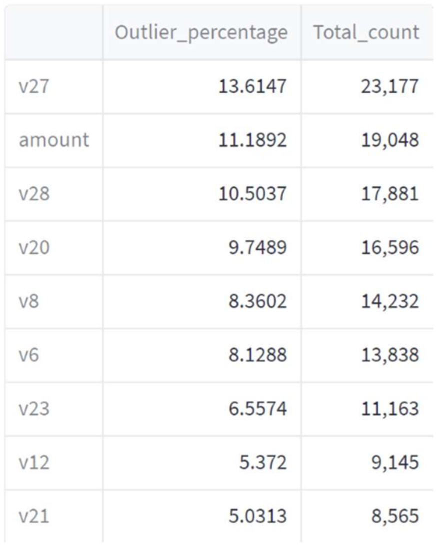
The features' distribution analysis reveals the following observations:

- The *V1, V2 … V28* resulting from PCA, are centered around 0 and standardized. Many of them have outliers.
- The *'Amount'* variable is strongly skewed to the right with numerous extreme outliers.

3.2.3 Outliers

Moving on to outliers, multiple variables show a substantial number of outliers. The table lists variables with over 5% of transactions classified as outliers.

Table 3. Variables with over 5% of transactions classified as outliers.



	Outlier_percentage	Total_count
v27	13.6147	23,177
amount	11.1892	19,048
v28	10.5037	17,881
v20	9.7489	16,596
v8	8.3602	14,232
v6	8.1288	13,838
v23	6.5574	11,163
v12	5.372	9,145
v21	5.0313	8,565

To assess the impact of outlier removal, two scenarios were tested:

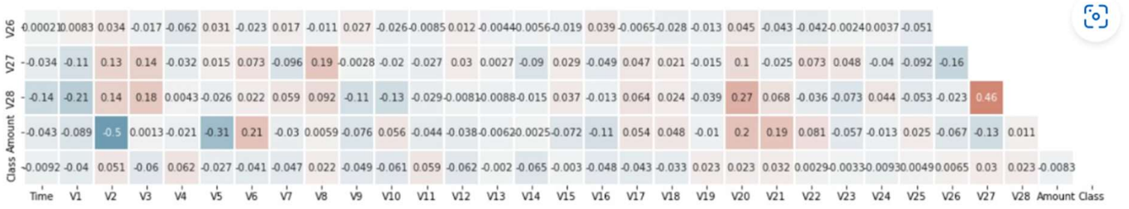
1. Removing 10% of all outliers.
2. Removing 10% of outliers in variables V14, V17, V12, V10 – identified as highly important based on mutual information.

In both scenarios, removing outliers resulted in the elimination of all or nearly all fraud transactions. Consequently, the decision was made to retain all outliers and explore alternative methods to address this issue.

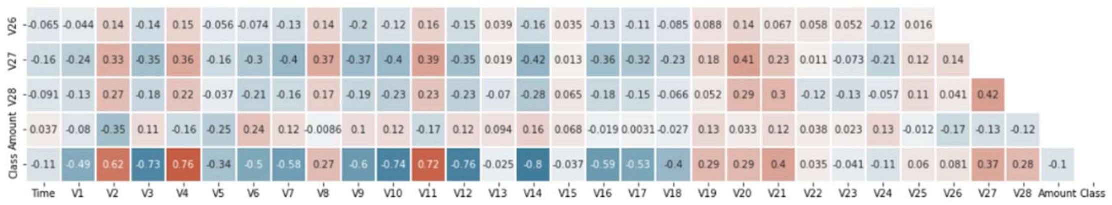
3.2.4 Correlation

In the following stage of the analysis, an examination of relationships between variables was conducted using Spearman correlation. This method was chosen for its ability to assess monotonic relationships, whether linear or not.

The plots below illustrate the correlations between the target variable and other variables for the imbalanced dataset and the balanced ones created using RandomOverSampler and SMOTEC.



Data balanced with RandomOverSampler():



Data balanced with SMOTEC():

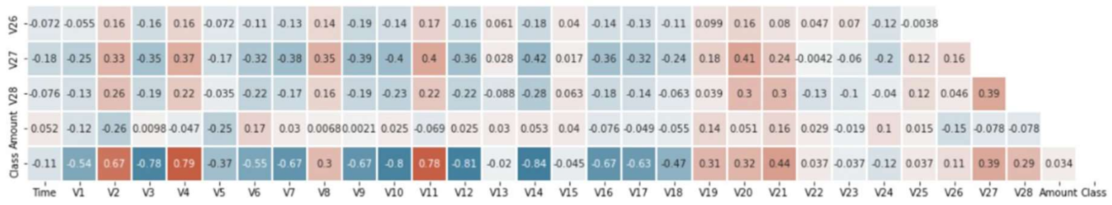


Figure 4. Spearman correlations between the target variable and other variables for the imbalanced dataset and the balanced ones created using oversampling methods.

In the imbalanced dataset, minimal correlations exist between the target variable and others. However, in the datasets balanced with RandomOverSampler or SMOTEC, the relationships show significantly greater strength.

3.2.5 Features distribution analysis

In the analysis of the **Amount** distribution, observations from the plots below led to the conclusion that the variable's distribution is heavily skewed to the right, displaying extreme outliers.

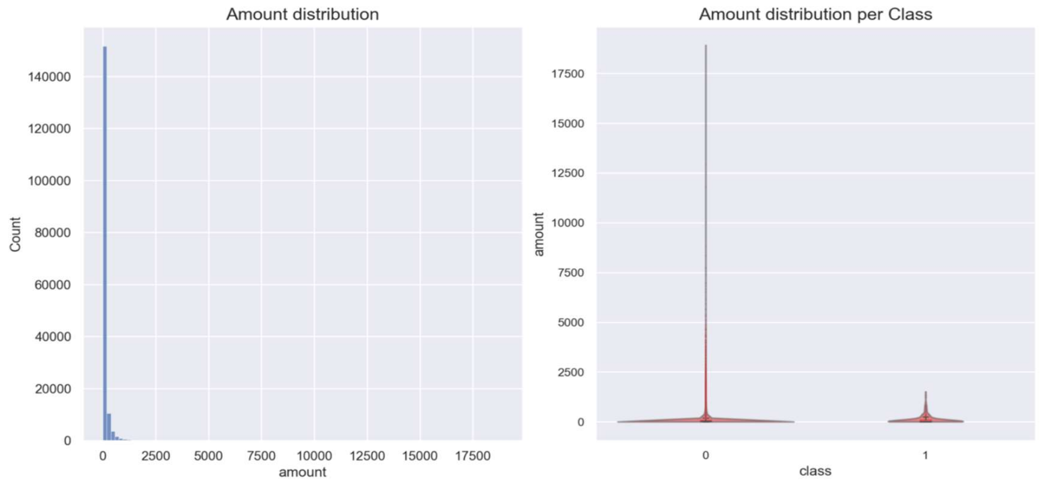


Figure 5. Distribution of the 'Amount' variable.

To mitigate the impact of these outliers and achieve a more symmetrical distribution, a logarithmic transformation was applied.

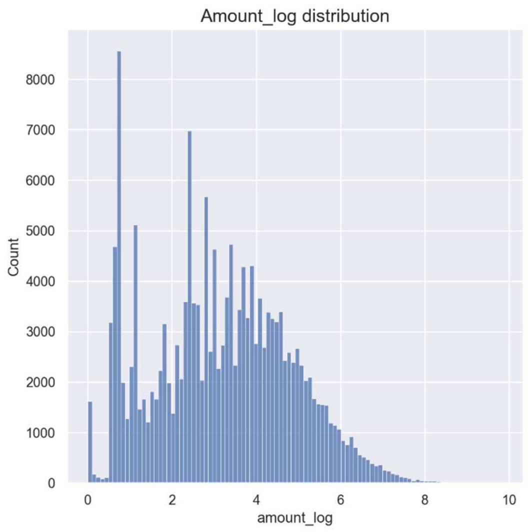


Figure 6. Distribution of 'Amount' variable after logarithmic transformation.

This transformation aids the model in learning patterns and relationships among the features more effectively.

The graph below shows the ***Time*** variable distribution:

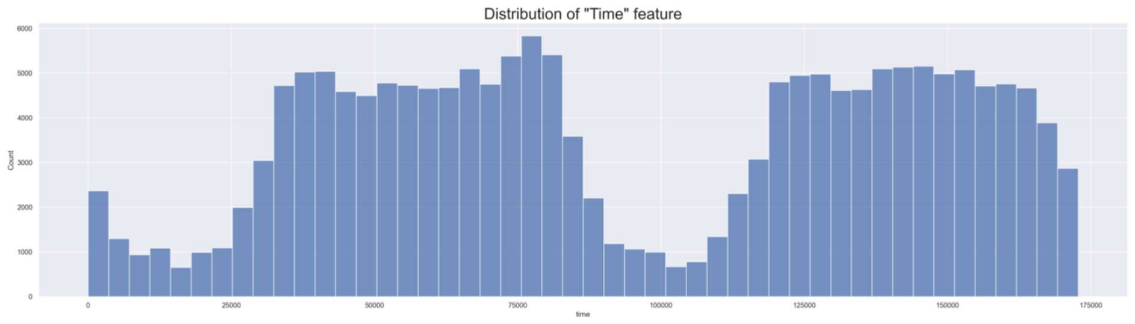


Figure 7. Distribution of 'Time' variable.

It represents time from the start of observation in seconds from 0 to 172792 (covering 48 hours period). It can be stated that the modes are daytimes as the dataset describes two days. The variable was treated as a categorical one and it was transformed, so it represents hours in 24 hours periods. The main reasons for this approach are:

- The values of the Time variable do not have any ordering or directionality. For example, values such as 100 do not inherently imply greater or lesser significance than values like 50.
- There may be certain patterns or relationships in the data that are related to the time of day, rather than the number of seconds since the first transaction. Treating it categorically allows for capturing potential temporal patterns within the dataset.

3.2.6 Feature selection

During the feature selection process, Mutual Information (MI) was explored, with MI being zero for independent variables and higher values indicating greater dependency.

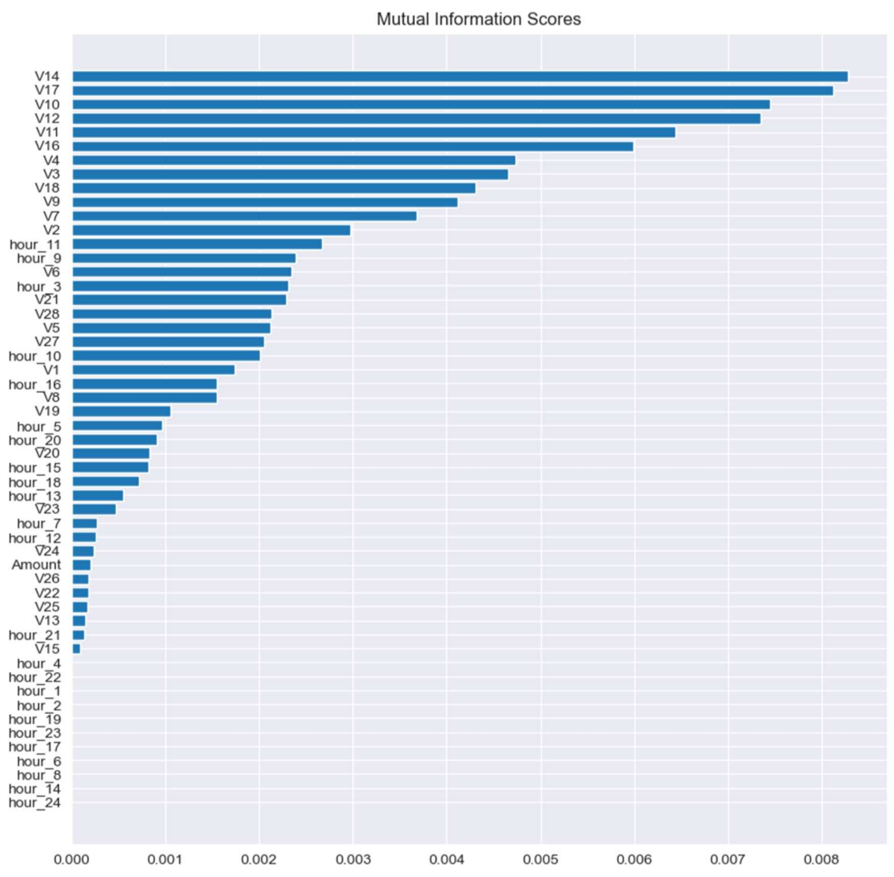


Figure 8. Mutual information scores.

Subsequently, features with the lowest mutual information were removed, and the impact on model performance and training/prediction times was assessed. However, the results indicated that the models did not exhibit improved performance, and there was no significant reduction in training and prediction times. Consequently, the decision was made to continue with all the features.

3.2.7 EDA summary

Following the exploratory data analysis, the decision was made to transform the dataset as follows:

- $Time$ variable will be converted into a categorical variable 'Hour,' representing the hours of a day.
- $Amount$ variable will be logarithmically transformed and scaled with $RobustScaler()$ (due to the extreme outliers identified).
- The v1, v2 ... v28 features as they are the result of PCA (Principal Component Analysis) are already standardized.

An experiment will be conducted to assess the impact of sampling with RandomOverSampler() and SMOTE() on the models' performance.

3.3 Model choosing

Various strategies were tested during the project, starting with the most straightforward approach where classification models were applied to the imbalanced dataset, serving as the baseline.

Another strategy involved classification models with the class_weight hyperparameter. In the scikit-learn library, certain models allow specification of class weights, where default uniform weights for all classes can be adjusted. This enables penalizing a model's mistakes on specific classes.

For resampling strategies, we employed RandomOverSampler and SMOTENC. These techniques, available in the imbalanced-learn library, utilize oversampling to generate data points representing the minority class.

The full list of all the dataset transformation types (balanced/imbalanced) and models combinations tested during the project is presented below:

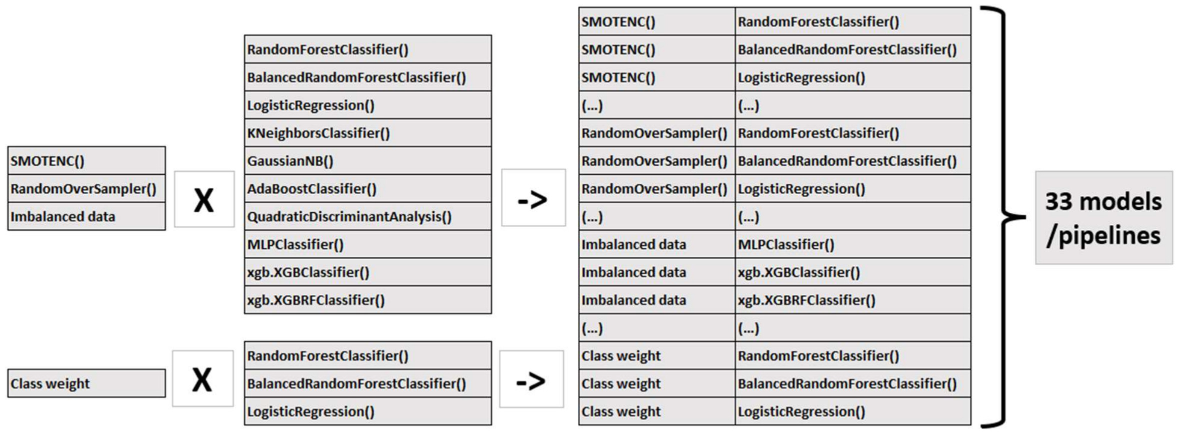

SMOTENC()
RandomOverSampler()
Imbalanced data
X
RandomForestClassifier()
BalancedRandomForestClassifier()
LogisticRegression()
KNeighborsClassifier()
GaussianNB()
AdaBoostClassifier()
QuadraticDiscriminantAnalysis()
MLPClassifier()
xgb.XGBClassifier()
xgb.XGBRFClassifier()
->
Class weight
X
RandomForestClassifier()
BalancedRandomForestClassifier()
LogisticRegression()
->
SMOTENC() RandomForestClassifier()
SMOTENC() BalancedRandomForestClassifier()
SMOTENC() LogisticRegression()
(...) (...)
RandomOverSampler() RandomForestClassifier()
RandomOverSampler() BalancedRandomForestClassifier()
RandomOverSampler() LogisticRegression()
(...) (...)
Imbalanced data MLPClassifier()
Imbalanced data xgb.XGBClassifier()
Imbalanced data xgb.XGBRFClassifier()
(...) (...)
Class weight RandomForestClassifier()
Class weight BalancedRandomForestClassifier()
Class weight LogisticRegression()
33 models /pipelines

Figure 9. List of models/pipelines tested during the project.

For evaluation purposes, we chose the F-beta score with a beta value of 5, placing a higher emphasis on recall over precision. The initial stage involved testing 33 models, each with default hyperparameters, to assess their performance. From this set, the best-performing models were selected for further hyperparameter tuning. The models were subsequently trained on the training data, and their performance was compared using validation results. Given the extensive results table, only the results for the models selected for hyperparameter tuning are presented in Table 4.

Table 4. Performance results of models selected for hyperparameters tuning.

Strategy	model	fbeta	precision	recall	false_positives	false_negatives
smotenc	**knn_clf**	0,773271533	0,322175732	0,819148936	162	17
smotenc	**rf_clf**	0,771322621	0,935064935	0,765957447	5	22
smotenc	**brf_clf**	0,771322621	0,935064935	0,765957447	5	22
smotenc	**xgb_clf**	0,747433265	0,823529412	0,744680851	15	24
smotenc	**mlp_clf**	0,744680851	0,744680851	0,744680851	24	24
smotenc	**lr_clf**	0,742259751	0,518248175	0,755319149	66	23
smotenc	**ada_clf**	0,67357513	0,137614679	0,79787234	470	19
smotenc	**xgbrf_clf**	0,660875161	0,104221636	0,840425532	679	15
ros	**ada_clf**	0,680405626	0,113154173	0,85106383	627	14
class_weight	**brf_clf 0: 0.9/ 1: 0.1**	0,457042408	0,036347115	0,85106383	2121	14

Notably, the SMOTENC sampling strategy demonstrated the highest average model performance, as measured by the F-beta score. Consequently, we decided to continue with the best-performing models on the SMOTENC-balanced dataset. Additionally, for further experimentation with the VotingClassifier, two models with the highest recall were included: ADABoost with RandomOverSampler and balanced RandomForest with class weight parameters of 0.9/0:1. This approach aims to explore whether it is possible to maintain a low number of false negatives achieved by these two models while subsequently reducing false positives in subsequent steps.

3.4 Hyperparameter tuning

Considering the varied training times for different models, we employed GridSearch or Randomized Search with varying n_iter parameters based on each model's characteristics.

The outcomes of the selected models after the hyperparameter tuning are presented in the Table 5.

Table 5. Outcomes of the selected models after hyperparameter tunning.

model/pipeline	fbeta	precision	recall	false_positive	false_negative
ros_ada	0,797417272	0,59375	0,808510638	52	18
smotenc_brf	0,790468365	0,880952381	0,787234043	10	20
voting_3best_fbeta_soft	0,77978636	0,869047619	0,776595745	11	21
smotenc_rf	0,779466119	0,858823529	0,776595745	12	21
smotenc_lr	0,774424146	0,446428571	0,79787234	93	19
smotenc_knn	0,764114462	0,322033898	0,808510638	160	18
smotenc_mlp	0,763795157	0,437869822	0,787234043	95	20
voting_5best_recall_soft	0,756899811	0,261016949	0,819148936	218	17
smotenc_ada	0,72906793	0,39010989	0,755319149	111	23
smotenc_XGB	0,712328767	0,341463415	0,744680851	135	24
brf_class	0,637782733	0,097591888	0,819148936	712	17
smotenc_XGBRF	0,53429232	0,051847051	0,85106383	1463	14

After having our models tunned and trained we explored the performance of several VotingClassifiers. The results of two such classifiers are also included in the table. Their ensemble configurations are detailed in the Appendix A.

3.5 Final model

The model with the best F-beta score is *ros_ada*, which is a pipeline which architecture is shown on the Figure 10.

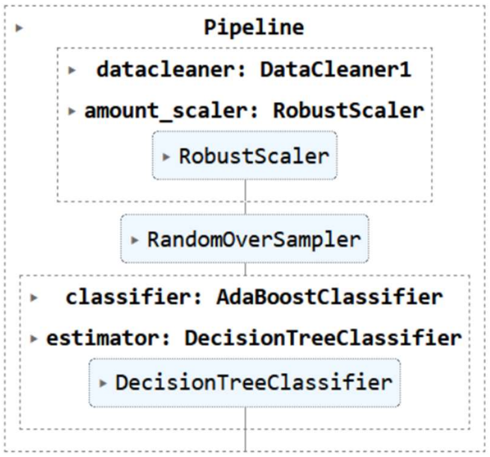

Pipeline
datacleaner: DataCleaner1
amount_scaler: RobustScaler
RobustScaler
RandomOverSampler
classifier: AdaBoostClassifier
estimator: DecisionTreeClassifier
DecisionTreeClassifier

Figure 10. Final model architecture.

The final model includes following elements:

- **Custom Transformer**: a custom written data transformer which imputes any missing data (if there are any), scales the *Amount* variable with RobustScaler and makes a logarithmic transformation on the *Amount*, transforms the *Time* variable into *hours* and dummy encodes them.
- **RandomOverSampler** which resamples the dataset.
- **AdaBoostClassifier** - fitted using the tuned AdaBoostClassifier in conjunction with the tuned DecisionTreeClassifiers

4 Results and Discussion

At this stage, with no specific use for the validation dataset, we merged it with the training dataset. A new model (the entire pipeline) was then initialized and trained on the combined training dataset.

4.1 Results

The results of predictions on the test dataset for the final model trained on the expanded training dataset reveal the best F-beta score achieved in the project.

```
The FBeta score for the final model is: 0.8347689898198905
```

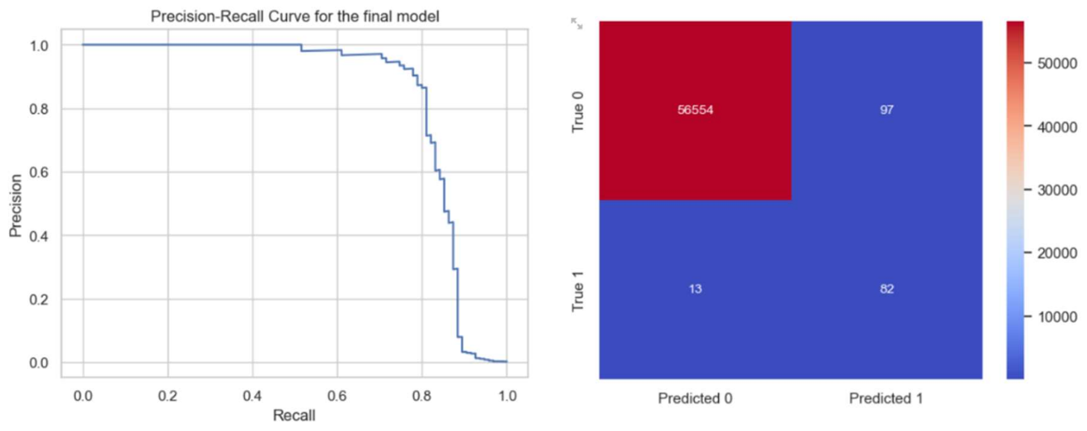


Figure 11. Performance results of the final model.

This suggests that the model does not overfit the data and demonstrates strong generalization to unseen data. Notably, the model missed only 13 fraudulent transactions out of a total of 95, a low number compared to previous results. Simultaneously, it produced 97 False Positive predictions, a relatively low figure considering the high recall score. This indicates that the model maintains a balance, avoiding an excessive number of cases for the AML department to review.

4.2 Discussion

4.2.1 Question 1: Overcoming Class Imbalance

The exploration of various strategies to address class imbalance in the dataset was a fundamental aspect of this research. The exploration of class imbalance mitigation strategies revealed that resampling techniques, particularly RandomOverSampler and SMOTENC, may significantly improve model's performance. The results, as presented in the earlier sections, demonstrated the effectiveness of these approaches in enhancing the model's ability to detect fraudulent transactions in the highly imbalanced dataset.

An in-depth performance analysis of all models before hyperparameter tuning revealed that, on average, the pipelines incorporating the SMOTENC sampler achieved the highest results. However, following hyperparameter tuning, it is noteworthy that the final model with the highest F-beta score utilized RandomOverSampler, emphasizing the nuanced nature of model performance under different configurations.

4.2.2 Question 2: Model Performance Evaluation

The second research question focused on determining appropriate measures for evaluating the performance of the developed model. We selected the F-beta score, with beta set to 5 to emphasize recall, as the primary evaluation metric. The application of the selected evaluation metric, the F-beta score, allowed for a comprehensive assessment of the model's performance. The achieved F-beta score on the final model, trained on an expanded dataset, demonstrated its effectiveness in identifying fraudulent transactions while maintaining a reasonable balance with false positives.

4.3 Conclusion

In conclusion, this research successfully addressed the primary objective of developing an efficient fraud detection model for credit card transactions. The findings highlight the importance of addressing class imbalance through resampling techniques and emphasize the value of the chosen evaluation metric in achieving a well-balanced model. The insights gained from this study contribute to the ongoing efforts to enhance fraud detection in financial transactions.

Appendix A

The ensemble configurations of VotingClassifiers:

```
voting_3best_fbeta_soft = VotingClassifier(
                            estimators=[('ros_ada', ros_ada),
                                        ('smote_brf', smote_brf ),
                                        ('smote_rf', smote_rf )],
                             voting='soft')
```

```
voting_5best_recall_soft = VotingClassifier(
                            estimators=[('ros_ada', ros_ada),
                                        ('brf_class', brf_class),
                                        ('smote_xgbrf', smote_xgbrf),
                                        ('smote_lr', smote_lr),
                                        ('smote_knn', smote_knn)],
                            voting='soft')
```

Table of figures

References

Altini, M. (2015, August 18). *Dealing with imbalanced data: undersampling, oversampling and proper cross-validation*. Retrieved January 20, 2024, from https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation

Association of Certified Fraud Examiners. (2022). Raport Occupational Fraud 2022: A Report to the Nations. 4. Retrieved from https://legacy.acfe.com/report-to-the-nations/2022/

Brownlee , J. (2021, April 27). *How to Develop Random Forest Ensembles With XGBoost*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/random-forest-ensembles-with-xgboost/

Brownlee, J. (2020, January 14). *A Gentle Introduction to the Fbeta-Measure for Machine Learning*. Retrieved January 18, 2024, from Machine Learning Mastery: https://machinelearningmastery.com/fbeta-measure-for-machine-learning/

Chawla, N. V., Bowyer, K. W., & Hall, L. O. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal Of Artificial Intelligence Research, 16*, 321-357. Retrieved January 19, 2024, from https://arxiv.org/abs/1106.1813

Chen, C., Liaw, A., & Breiman, L. (2004). *Using Random Forest to Learn Imbalanced Data.* University of California, Berkeley, Department of Statistics, UC Berkley. Retrieved from https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Retrieved from https://arxiv.org/abs/1603.02754

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* (2nd ed.). O'Reilly Media, Inc.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research, 18*, 1-5. Retrieved January 19, 2024, from User Guide: https://imbalanced-learn.org/stable/user_guide.html#user-guide

Martins, C. (2023, November 02). *Gaussian Naive Bayes Explained With Scikit-Learn*. Retrieved from Build in: https://builtin.com/artificial-intelligence/gaussian-naive-bayes

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Weiss, R. (2011). Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research, 12*, 2825 - 2830. Retrieved from https://scikit-learn.org/stable/index.html

Statista. (2023). *Mastercard credit cards in circulation 2006-2023.* Statista. Retrieved January 18, 2024, from https://www.statista.com/statistics/618137/number-of-mastercard-credit-cards-worldwide-by-region/

Worldline, the Machine Learning Group of Université Libre de Bruxelles. (2018). *Credit Card Fraud Detection*. Retrieved from Kaggle: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud