Explainable Feature Engineering for Multi-class Money Laundering Classification

Petre-Cornel GRIGORESCU, Antoaneta AMZA
Bucharest University of Economic Studies, Romania
grigorescupetre19@stud.ase.ro, amzaantoaneta10@stud.ase.ro

This paper provides insight into typical money laundering typologies used in the financial crime domain and provides a concrete set of methods through the use of which fraudulent transactions may be classified using traditional machine learning algorithms and proving the efficacy of tree-based models in not only predictive power, but also explainability and ease of interpretation of results.
Keywords: *Anti-money laundering, Machine learning, Tree-based models, Explainability*
DOI: 10.24818/issn14531305/29.1.2025.06

1 Introduction

Detecting money laundering (AML) activities is critical for financial institutions, given the significant financial and reputational risks involved. Fraudulent transactions can generate considerable losses for banks, affecting both their financial stability and public confidence. In addition, banks operate within a strict regulatory framework that requires active monitoring and reporting of suspicious activity. At the European level, Directive (EU) 2015/849 requires banks to implement effective mechanisms for fraud detection and prevention. In this context, advanced analytical techniques play a crucial role by enabling the analysis of large volumes of data and the identification of abnormal transactions. These techniques are also useful for supervisors, who can assess the robustness and effectiveness of the models used by banks, thus ensuring that risks are managed according to the required regulations and standards. Through SupTech (Supervisory Technology), supervisors can leverage advanced analytics and AI-driven tools to enhance their monitoring capabilities, allowing for a more data-driven, proactive approach to oversight.

In line with current research trends in AML, our work explores the use of machine learning techniques, such as Random Forest, Logistic Regression and K-Nearest Neighbors for classifying fraudulent transactions. The main contribution of this study lies in going a step further down the chain and classifying those suspicious transactions into a number of fraudulent typologies, in our particular case 17, which would significantly aid Know-Your-Customer investigators by pin-pointing the characteristics of a transaction that lead to its classification into a particular typology and further validating the submitted suspicious activity reports (SARs).

This approach not only allows for greater accuracy in detection, but also provides a clearer interpretation of the results, an essential aspect in evaluating the decisions made by the model. Our paper thus contributes to the improvement of the AML detection framework by combining the latest machine learning techniques with elements of interpretability and selection of relevant features.

2 Literature Review

Quantitative methods have become increasingly central to anti-money laundering (AML) research, driven by growing complexity of financial transactions and the advances in data processing. Extensive academic research has examined methods for combating money laundering, from traditional statistical approaches to cutting-edge machine learning and data mining techniques [5] . As [6] noted following a systematic review of reputable academic sources, machine learning is currently the predominant method in financial fraud detection. In this vein, [10] utilized machine learning algorithms to identify suspicious transactions. [2] showcased the use of

supervised learning techniques in online fraud detection, highlighting their adaptability across financial crime contexts. Similarly, [3] , after analyzing various databases, algorithms, and pre-processing techniques used in fraud detection, concluded that machine learning algorithms offer an effective and increasingly accurate solution for fraud identification.

The increasing complexity of money laundering schemes and the sophistication of tools available to combat them created a critical need for interpretable models that meet regulatory requirements while maintaining detection efficacy. The work of [4] on explainable Artificial Intelligence (XAI) becomes particularly relevant, offering guidance into how complex AI systems can be made more interpretable and trustworthy in the fight against money laundering.

In the field of Anti-Money Laundering (AML) it is important to establish the characteristics that determine an increased risk of fraudulent transactions, and risk typologies are created for managing exposures and identifying suspicious activities. The literature suggests that geographic risk, customer risk, product risk and transaction risk are the main categories analyzed by financial institutions to prevent money laundering and terrorist financing [7] . In this context, high-risk jurisdictions represent countries or regions known for deficiencies in the implementation of AML controls, which are monitored by international organizations such as the Financial Action Task Force (FATF), for which increased due diligence measures are involved on the part of banks that have transactions in these areas.

Building on this, feature engineering represents an essential step in the process of developing machine learning models, having a direct impact on the performance and accuracy of the algorithms used. By creating, selecting, and transforming relevant features from raw data, feature engineering allows models to better capture the patterns and relationships in the dataset. In the context of financial fraud detection and anti-money laundering, where data is complex and often irregular, feature engineering helps highlight behavior indicative of money laundering by turning transactions into useful features for algorithms. Characteristics such as the frequency of transactions, relationships between accounts or unusual volumes of money transferred are examples of extracted attributes that can help identify illicit behavior.

3 Data

Due to legal and privacy constraints that limit access to real transaction data, we decided to use a synthetic dataset approach commonly applied by researchers in fraudulent activity detection. The selected state-of-the-art dataset developed by [1] is relatively new at the time of this paper's writing and incorporates advanced financial crime data features and typologies.

Table 1. Descriptive statistics of the SAML-D dataset

	Amount	**Is_laundering**
count	9.504852e+06	9.504852e+06
mean	8.762968e+03	1.038733e-03
std	2.561495e+04	3.221263e-02
min	3.730000e+00	0.000000e+00
25%	2.143688e+03	0.000000e+00
50%	6.113720e+03	0.000000e+00
75%	1.045846e+04	0.000000e+00
max	1.261850e+07	1.000000e+00

This dataset includes over 9.5 million transactions, with approximately 0.1% labeled as potentially fraudulent, reflecting a typical imbalance in fraud detection scenarios.

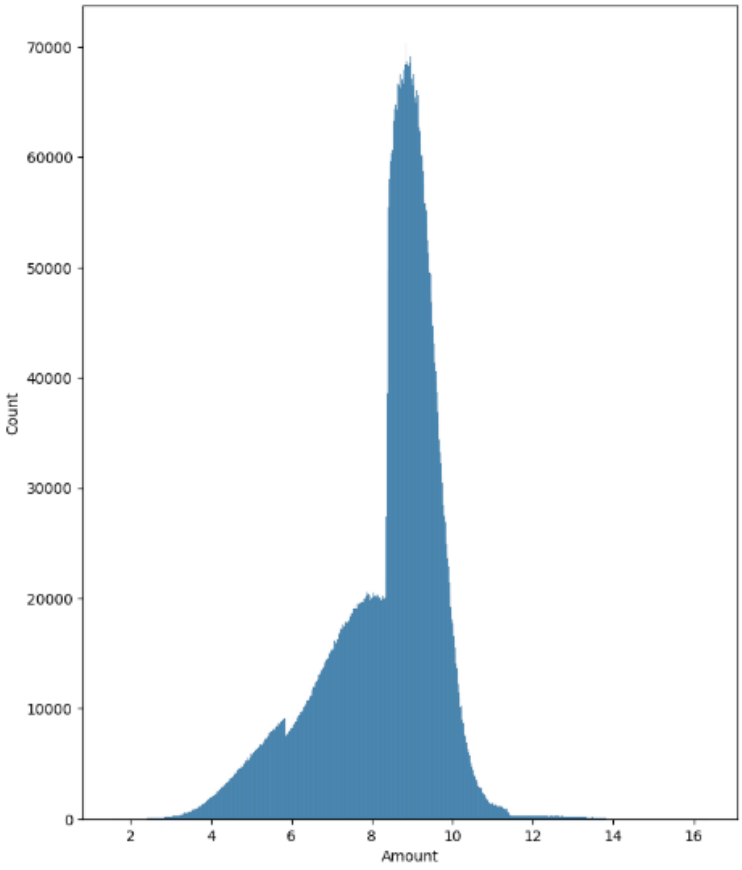

Fig. 1. Distribution of log10 transacted amounts

The dataset consists of 12 features related to transaction specifics, including the transaction timestamp, counterparty details (accounts and countries involved), transaction amount, transferred and received currencies, payment type, laundering flag and 28 transaction categories, divided into 11 categorized as normal and 17 classified as laundering money. The descriptive statistics of the base numerical variables, present initially, are presented in Table 1.

The maximum value of amount in any given transaction is 12618498.4, the median 6113.72 and 8762.967 as the mean, most likely influenced by the very large values present in the dataset.

More information on the dataset may be found at https://ieeexplore.ieee.org/document/10356193.

Due to these high values skewing the dataset, a direct histogram/distribution cannot be drawn, however taking the log10 of the amounts transacted results in the right-hand side plot. Distribution statistics are presented in table 2.

Table 2. Descriptive statistics of the log10 transacted amounts

Log-Transformed Statistics	**Value**
Mean	8.3480
Median	8.7184
Standard Deviation	1.4026
Skewness	-1.0103
Kurtosis	1.0373

We can quickly note that statistically there is some skewness to the left present in that the median is higher than the mean, apparent and confirmed by the calculated skewness and of

course, the visually graphed distribution. The kurtosis statistics confirmed that transaction amounts do not follow a normal distribution.
To detect the outliers present in the amounts, we use the traditional inter-quartile range method, where x being a value:
$x < Q1 - 1.5 * IQR$ or $x > 1.5 * IQR + Q3$ would indicate an outlier on the respective side of the distribution. The number of outliers is quite large even for such a large dataset, at 429848 detections or 4.522% of the total dataset. We do not exclude these as they are valuable input for the model to detect random patterns with and better predict suspicious transactions. Lastly, in terms of unique occurrences in each variable, table 3.

Table 3. Unique value counts of the initial variables in the dataset

Variable	**Unique occurrences**
Time	86400
Date	321
Sender_account	292715
Receiver_account	652266
Amount	2314277
Payment_currency	13
Received_currency	13
Sender_bank_location	18
Receiver_bank_location	18
Payment_type	7
Is_laundering	2
Laundering_type	28

We have a span of around a year of data in terms of the Dates present, split across two calendar years. The time, reflecting the 24 hours of the day, is at the maximum possible unique count = 86400 or the total number of seconds in a day, meaning across the whole dataset, each possible second of a single day appears at least once as a transaction. There are many more receiver accounts than senders, meaning the general outlook is that an account is expected to send monetary amounts to multiple accounts on average. There are banks from 18 countries present in the dataset spread across 13 currencies, although the currency and countries present do not vary much in terms of switching them with an altogether different set, assuming independence of the high-risk jurisdictions which we treat in a different variable later. There are multiple possible payment types, such as card, cheque or wires, and lastly of course, the classes we aim to train on.

4 Risk typologies

The risk typologies we aim to capture are 17 in nature and result from the dataset providers' research and collaboration with a European banking group and IBM. The most straight-forward are the behavior changes of an account, which are divided into 2 – *Behavioral Change 1* captures when a main account starts trading with new accounts through unusual patterns. *Behavioral Change 2* on the other hand captures when the main account starts transacting with accounts located in high-risk locations according to the latest list by the EU, USA or international organizations. Specific typologies exist for capturing behaviors such as *Single Large* transactions or anomalous *Cash Withdrawal*s for, typically, private entities, as well as *Over-invoicing* for "businesses" that may be used as money laundering fronts and over-charging certain clients, the money mules, in an attempt to cover the laundering as service charges. The most common techniques for money laundering currently are under the *Smurfing* and *Structuring* typologies – while these two terms are often used interchangeably, these represent different levels of difficulty or sophistication for the same technique, of dividing a large sum of money into smaller, more innocuous amounts.

Structuring is a basic application in which the context around the amount is known, such as the source of the funds. The source of structuring money is often not illegal, such as gambling or award/prize money, and may be used to avoid paying taxes. Smurfing on the other hand utilizes more complex networks of "smurfs" or money mules which are instructed on how to deposit or send amounts of money such that certain systems are not triggered. This may also include cross-border physical movement of the money from its source of funds and go as far as corrupting public or private officials through bribery to conceal or ignore the smurfing activity. Smurfing is often utilized by organized crime networks, while structuring is much lower level, typically involving a single private individual. [9]
Following, many subsequent behaviors can be explained through graphical representations as we will do in the Figure 2.

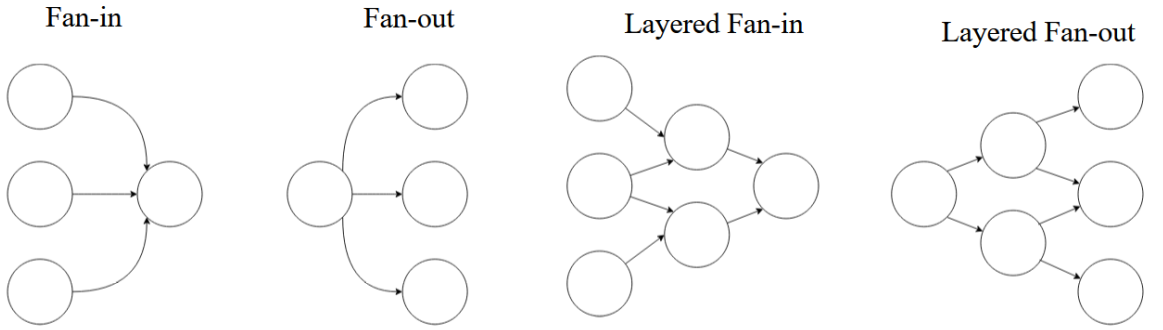

Fig. 2. Fan-in and Fan-out typology graphs

The *Fan-in* and *Fan-out* typologies capture behavior in which a single individual spreads out a sum of money across multiple other accounts (nodes) in an attempt to avoid Single-Large, Structuring or Smurfing detection, or perform the reverse, where a single concentrated node receives money amounts from multiple nodes. This can be done on multiple levels, as can be seen in the *Layered* versions of the *Fan-* typologies.

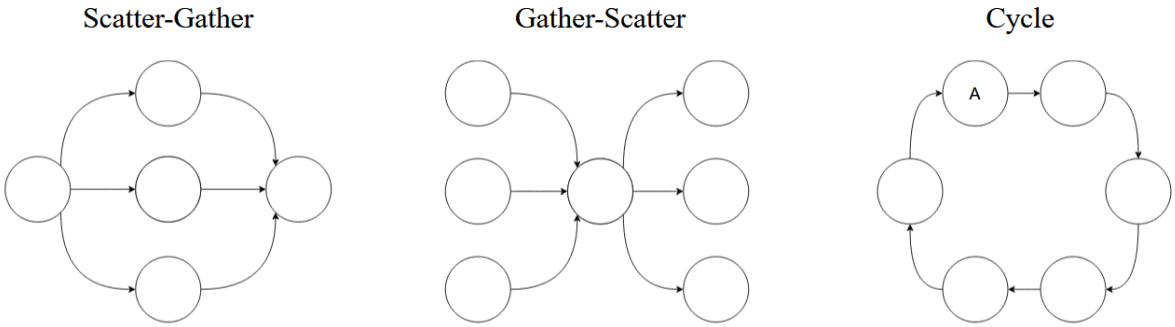

Fig. 3. Scatter-Gather/Gather-Scatter and Cycling typology graphs

The Scatter-Gather and Gather-Scatter typologies refer to more complex combinations of the fan-in and fan-out laundering strategies. The scatter-gather typology starts with a fan-out from a single node to multiple, followed by the same amount of money ending in the same initial node or another, similar to a fan-in. The gather-scatter strategy is the inverse of this, starting with a fan-in to a single concentrated "distributor" and ending in multiple other accounts with a fan-out. The *Cycle* typology is self-explanatory, with an amount of money being send through multiple accounts before returning to the initial one, in an

attempt to create a "trail" that would legitimize the money.

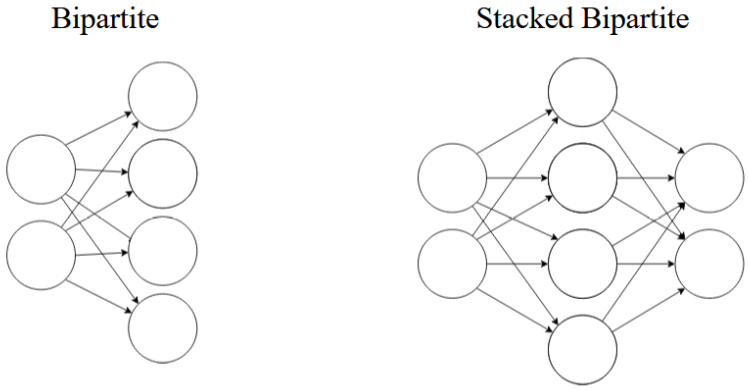

Fig. 4. Bipartite and Stacked Bipartite typology graphs

Lastly, the bipartite and stacked bipartite typologies represent a complication of the *Fan-in/Fan-out* and *Scatter-Gather* typologies, where there is not one, but two "distributor" parties involved in the laundering activity. [1] [8]

4 Methodology

In order to achieve the results in the following section, we utilize the traditional end-to-end machine learning process, starting with data inspection and cleaning, re-casting data types which optimize the memory allocation requirements and expand the limit of the feature space available to the authors. To begin with, we cast date and time variables as datetime and then convert them into numerical features such as year, month, day (of the week, of the month, and of the year) for the dates, and hour, minute, second for the time variable. As pandas follows the 0-index convention, 0 stands for Monday in the day of the week, and 6 for Sunday. With these additions, the dataset goes from 12 variables to 20, of which 2 are identifiers.

Due to the limited capabilities of data type assertions made by the pandas/NumPy packages during the data import stage, all integer-based variables are cast into the int64 data type which allows values in the ± 9,223,372,036,854,775,808 range of integers, an exaggerated amount considering most of the final variables in the dataset are 0s/1s for the encoded categorical variables, or at most in the lower thousands, such as for the year the transaction occurred. As such, we apply the following logic two times in our dataset, one time before binary encoding of categoricals (to reduce encoding time), and once again before the train-test splits to reduce the memory allocation of the encoding, as these variables are once again cast as int64, Figure 5. The diagram illustrates the decision tree logic to re-casting variable data types. The memory allocation, in bytes, is determined by the maximum of the absolute values of each variable. Most variables in the 2nd iteration of re-casting is allocated int8 sizes using NumPy's custom data type, while the remaining optimizable variables are split between int8 and int16. Edge cases which do not fit into either are cast into int32 or int64 at most, typically resulting in no memory savings. The sending and receiving account numbers are excluded from recasting due to being very large integer identifiers, which are removed from the full dataset right before the training of the model. In the 2nd re-casting, all integer variables are re-cast as int8.

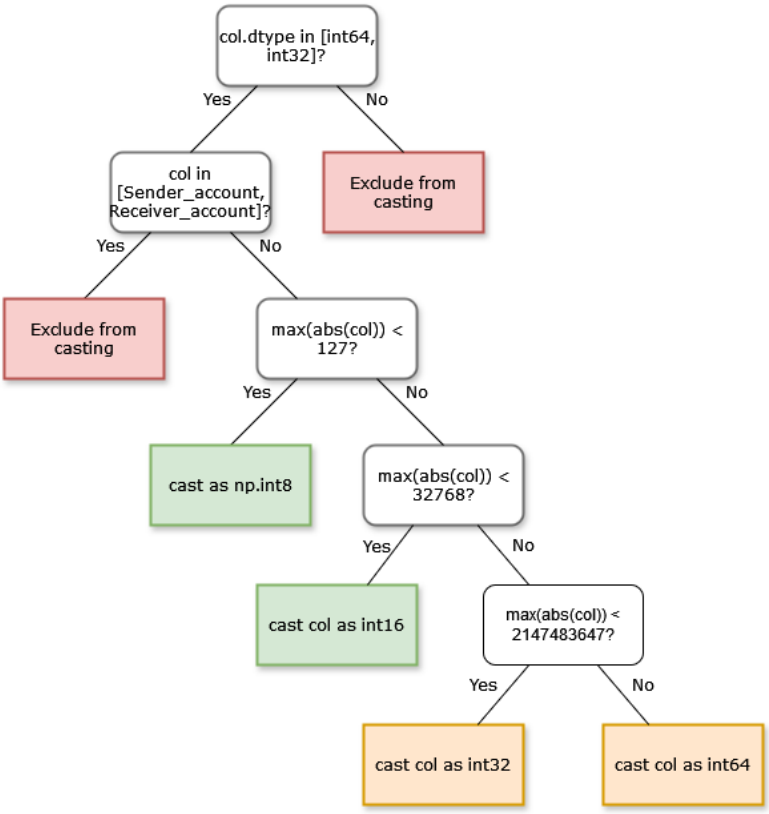

Fig. 5. Variable data type re-casting logic

We engineer 85 additional variables to provide context to the model for suspicious patterns, in line with the typical transaction monitoring risk typologies, such as layering, high risk jurisdiction movements or fanouts. The features we defined are in no way exhaustive or nearly as comprehensive as other possible candidates, but we are also limited by computational constraints with regards to processing some of the possible candidate features. The full features table is present in the appendix to be consulted if needed.

The reasoning behind the features created is straight-forward with regards to their typological relationships. We attempt to cover a wide variety of modus operandi / money laundering typologies by encompassing multiple time frames for metrics such as the number of transactions in a day, a week, a month, the average amount of money transacted per account sent or received, as well as multiple "flags" such as if the transaction is cross-border, happens at a peak hour of activity during the day or not, deviates from the account's typical behavior, and more. Due to the large number of classes to predict, many features were custom-made to better capture the random patterns of a particular typology, but it is true in many cases that an improvement in one typology is also an improvement in another – for instance, the features engineered for the cycling typology provide information for most other classes as well, as the modus operandi of obtaining a sum of money, moving it around multiple accounts, before eventually recovering part or all of the original sum, is common of most money laundering activities. Below are two graphs, one of all transactions (left) and on the right side the top 10% centrality (degree / betweenness) forming the core of the left graph. The yellow edges indicate transactions flagged as fraudulent – the cyclical behavior of these transactions is evident. The graphs and related features were created using the networkx Python library and plots were realized with seaborn (histograms, heatmaps) and matplotlib.pyplot (SHAP).

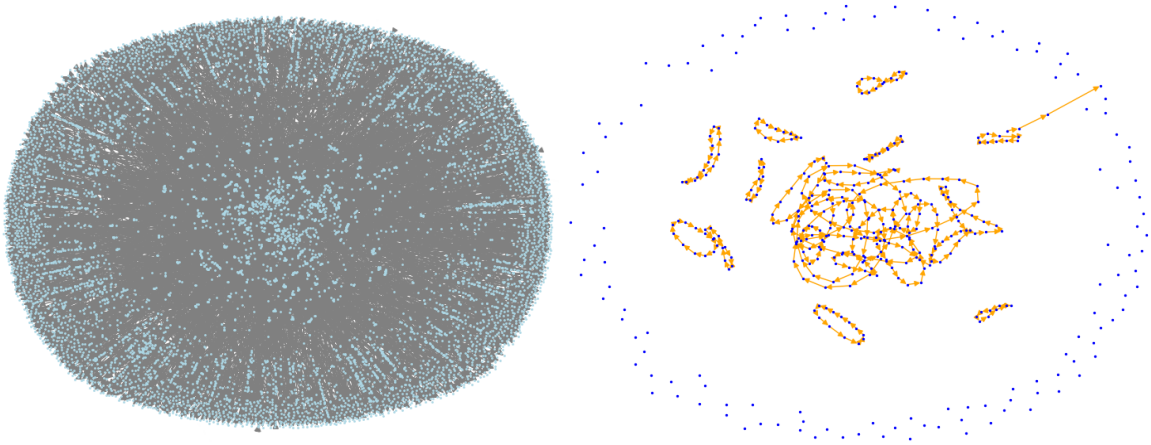

Fig. 6. Network Graphs of the transactions in the SAML-D dataset. All transactions (left) and top 10% centrality (right) pictured

The full feature creation table can be found in the appendix. In the following two figures we showcase for documentation purposes the distribution of correlation coefficients and their matrix.

We build and test logistic regressions, K-Nearest Neighbors models and Random Forests, ultimately choosing and optimizing a Random Forest due to its performance and explainability as opposed to more modern tree-based variations such as LightGBM or XGBoost. In testing the different models, we make use of Intel's Extension for Scikit-learn (sklearn), which optimizes/accelerates sklearn models for Intel CPU-based computing platforms. [11][12] The choice of Random Forest in this case, as opposed to gradient boosting tree-based models, is to emphasize on the explainable part of this paper with the intention of the hypothetical end-user to more easily understand how the model works. Random forest and all models with weak learners at their base function with Condorcet's jury theorem as the intuition behind their performance. That is to say, if a group of voters (decision trees, people, etc.) individually might have a low performance of slightly above a coin flips 50% in getting the "right" choice, when all choices are aggregated through various voting algorithms, the performance of the "jury" is demonstrably higher than any individual voters. Boosting algorithms add an additional layer of complexity which, while possibly significantly improving performance at the class or overall prediction level, lowers the explainability and feasibility of introduction into operations.

Fig. 7. Correlation Heatmap of final feature space

As can be seen in the heatmap above, a number of features in the final feature set have high correlation coefficients. This is normally an issue with models that do not handle collinearity well, such as non-regularized regression models, however tree-based bagging models are good at distributing information across multiple trees effectively. The reason we are keeping these features and not removing either of the correlated ones is because they are part of a subset selected with Recursive Feature Elimination, which, if it has validated these features, means that they provide good predictive power to a Random Forest in spite of the extra information. This is further minimized by 5-fold cross-validation during training, and a full correlation matrix may be consulted in the appendix. There are ultimately three feature spaces that we experiment with in this paper, the *Base* one, which only contains the default variables available with the starting dataset; the *General features* space, which contains the features engineered to improve performance on all 17 classes; and lastly the *Full* feature space, which contains all of the previous features as well as class-specific ones such as for gather-scatter, cycling, smurfing, etc. The appendix contains two separate tables, the first representing the General feature space and the second being the additional class-specific feature space. The general and full feature spaces are scoped through cross-validated Recursive Feature Elimination (RFE) for Random Forest alone, as Logistic Regression and KNN do not have the feature importance and/or coefficients to use for RFE. RFE functions by starting with the full feature space and assigning

feature importance, eliminating the least important few features, and then considering the remaining features, and so on recursively on smaller and smaller subsets of the feature space until a set threshold is met. We use the default 50% threshold for our models, meaning 50% of the full feature space will be ultimately kept. [11]

While the original dataset is quite large at observation-level, with 9.5 million individual transactions, less than 10,000 of these are labeled as fraudulent / is part of money laundering, and this subset is the scope of our classification problem. We train using a 70% training size (6911 obs.) subset and 30% test set (2962 obs.). We additionally use 5-fold cross-validation during Recursive Feature Elimination in order to reduce the variability of model performance caused by typical train-test splits and ensuring the model captures the most it can from the training subset's random patterns.

Lastly, for the explainability part we use calculated Shapley values using the SHAP package. Shapley values are calculated based on game theory, the ultimate output of the calculated values representing the impact of a feature's value upon the model output in terms of positive or negative signed change. That is to say, we represent the feature values in terms of whether they push the final model output higher, or lower, and multiple graphs are generated that showcase this solution. [13][13]

5 Results and discussions

The following table represents a sample of the model runs through multiple scenarios tested, together with the subset of the feature space used and the classification reports across precision, recall and f1-scores. As we can see, random forest outperforms both the KNN and Logistic Regression models by quite a wide margin when using the full feature space. KNN performs similarly on the base feature set as Random Forest, while the multinomial Logistic Regression with regularization reaches a slightly higher performance only when using feature engineering.

Table 4. Model performance metrics

MODEL	FEATURE SPACE	AVG. PRECISION	AVG. RECALL	AVG. F1-SCORE	SUPPORT
LOGISTIC REGRESSION	Base	0.00	0.06	0.00	2962
	Non-typology specific	0.41	0.41	0.40	
	Full	**0.43**	**0.45**	**0.44**	
KNN	Base	0.37	0.36	0.36	
	Non-typology specific	0.65	0.61	0.62	
	Full	**0.75**	**0.72**	**0.73**	
RANDOM FOREST	Base	0.33	0.32	0.31	
	Non-typology specific	0.89	0.87	0.87	
	Full	**0.93**	**0.91**	**0.91**	

The resulting table below showcases the test scores of the RF model with the full feature space created for it. We see that for all risk categories it far outperforms the KNN and Logistic Regression models.

Table 5. Random Forest classification performance metrics, by typology (class)

CODE	**CATEGORY**	**PRECISION**	**RECALL**	**F1-SCORE**
0	Behavioral_Change_1	1.00	0.96	0.98
1	Behavioral_Change_2	1.00	0.98	0.99
2	Bipartite	0.96	0.81	0.88
3	Cash_Withdrawal	1.00	1.00	1.00
4	Cycle	0.96	1.00	0.98
5	Deposit-Send	0.92	0.99	0.95
6	Fan_In	0.94	0.77	0.85
7	Fan_Out	1.00	0.94	0.97
8	Gather-Scatter	0.68	0.75	0.71
9	Layered_Fan_In	0.69	0.86	0.76
10	Layered_Fan_Out	0.93	0.92	0.92
11	Over-Invoicing	1.00	1.00	1.00
12	Scatter-Gather	0.84	0.70	0.77
13	Single_large	1.00	0.97	0.98
14	Smurfing	1.00	1.00	1.00
15	Stacked Bipartite	0.82	0.76	0.79
16	Structuring	0.99	1.00	1.00
-	Accuracy			0.93
-	**Macro Average**	**0.93**	**0.91**	**0.91**
-	Weighted Average	0.94	0.93	0.93

6 Explainability

The explainability component of this paper is done through Shapley Additive Explanations (SHAP) for the Random Forest full feature space model as mentioned in the methodology section. The goal is for an investigator to easily understand what led to a particular transaction being classed into one typology as opposed to any of the other 16, and we can compute this through SHAP feature importance. SHAP does not compute these by variation explained but rather how much a particular feature "moved the needle" by increasing or decreasing the predicted probabilities for the transaction to be classified in a certain class.

Below we have the SHAP feature importance aggregated across all classes. We note that the top three features are all fan-in related, which can possibly be explained by this being one of the most common denominators in all other typologies – as mentioned in the literature review of typologies, fan-ins and fan-outs are the smaller components of more complex money laundering "networks" such as bipartite/stacked bipartite, scatter-gather/gather-scatter and more. The most important SHAP-calculated feature is the month-on-month count of unique transactions where the directed transactions mostly concentrated on a few key receiver accounts. The fan-in/out ratio feature captures the ratio between the fanin_30d and fanout_30d features, while the fan-in intensity ratio feature represents the ratio of fanin_30d over the transactions received in a single day. The 4th most important feature is the standard deviation of amounts per sender account, and the 5th is the sum of money received in a month over the sum of money sent. In many cases of money laundering, this ratio will tend towards one.

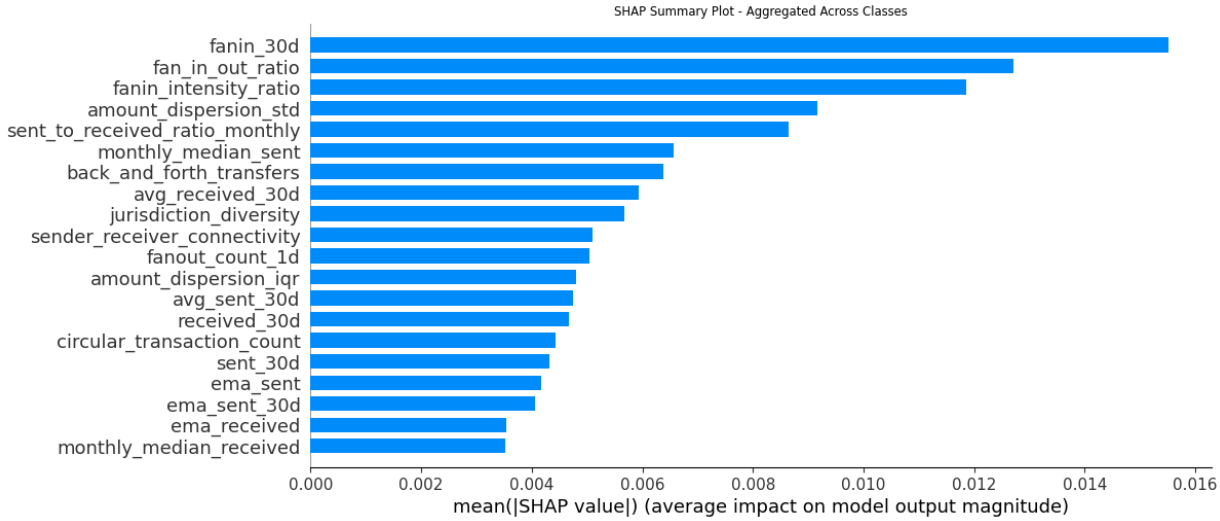

Fig. 8. SHAP Features importance, aggregated across all classes

7 Interpretability – Smurfing

One of the most famous risk typologies in money laundering is "smurfing" or "layering", which is the division of a large sum of money that would typically trigger basic rule-based reporting systems into smaller parts, and depositing these smaller parts of the whole in different, separate transactions, possibly across multiple bank accounts. We see in the below plot that the most important features here are the back and forth transfers count, circular transactions count, the median amount sent in a month, as well as how well-connected ("familiar") the sender and receiver accounts are, followed by a number of other general features including send/received amounts in averages, ratios, standard deviations and more.

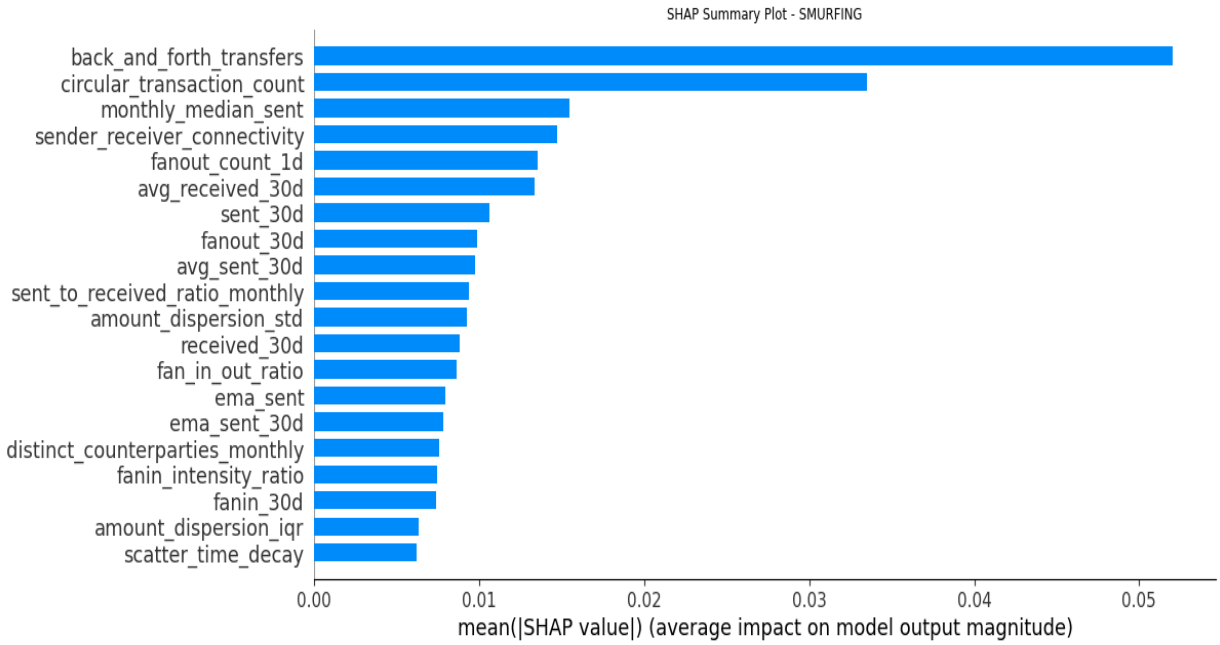

Fig. 9. SHAP-calculated feature importance for the SMURFING typology (class)

Additionally, there are many other features contributing little by little which are not captured in this plot, and this overall context that they provide for the various classes is, as a whole, similarly important as a feature space for these top features to contribute their most.

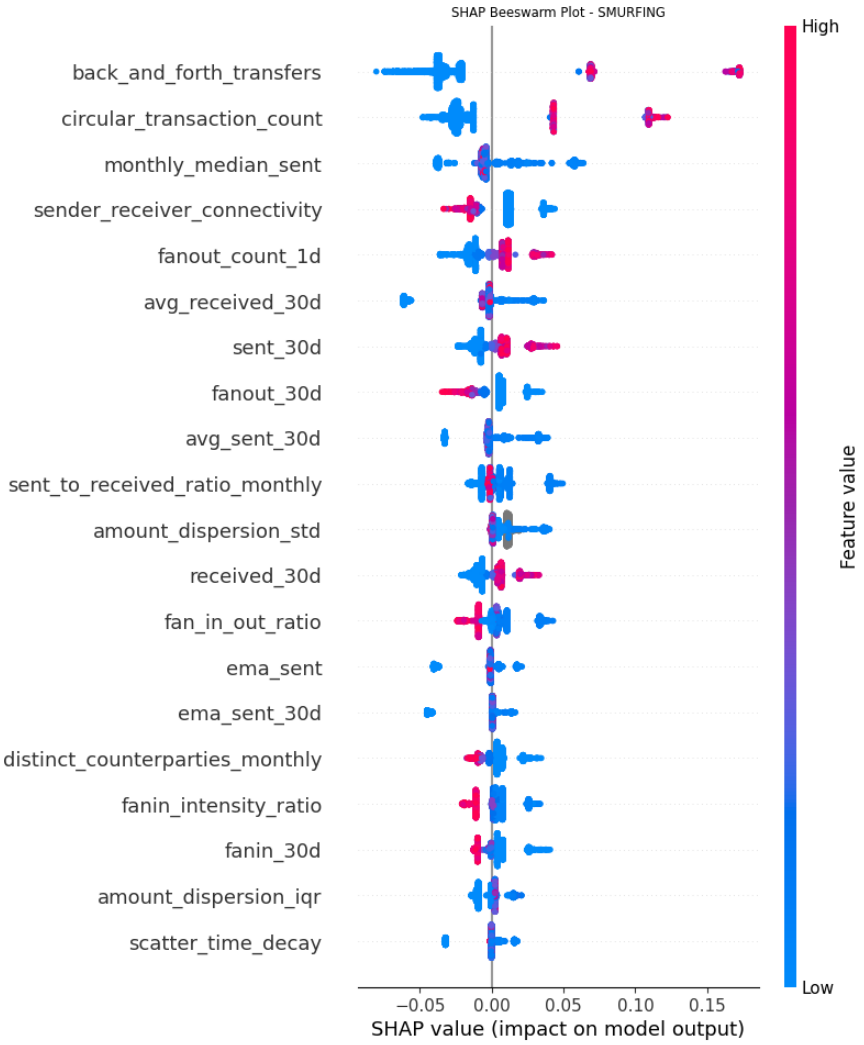

Fig. 10. SHAP Beeswarm Plot for SMURFING

To better understand how any one of these top features contributes in the end to the predicted probabilities, let us look at a SHAP beeswarm plot. The plot below shows how low or high values of the aforementioned features influence the final result of the random forest prediction. For a start, we see that back and forth transfers, calculated as the count of transfers between a sender and receiver account in a single calendar day, push the model's prediction for the smurfing class up if the count is high, and inversely decreases if this count is low. A similar behavior is seen for the count of circular transactions, i.e. those that eventually return to the sender, albeit by possibly taking a longer route. The monthly median amount sent appears to push the prediction up significantly for certain low values, while most high values stay closer to zero on the negative side of the axis. Note that, while many of the features push the prediction "down", this is not to say that the transaction is not fraudulent, but rather that smurfing, in this case, is not a very good fit for the behavior exhibited.

Lastly, let us look at a particular instance which our Random Forest model classified as *Smurfing*. Through the use of SHAP Force plots, this is the ultimate result we are aiming for, that a particular investigator be able to look at specific instances, be they transactions, transfers, etc., and determine at a glance what made the model "think" that the instance was a case of, say, *Smurfing*.

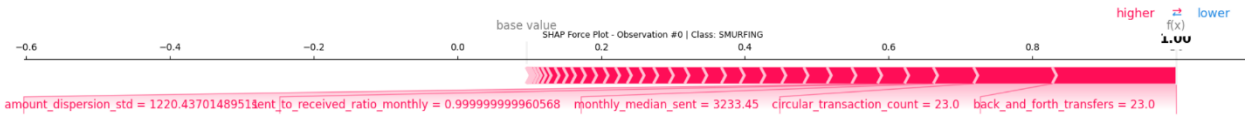

Fig. 11. Smurfing

From right to left we have the *Smurfing* features in descending order of importance. For this particular observation we have the back and forth transfers and circular transaction counts equal at 23, meaning all of the back-and-forth were circular in nature, and all of the send transactions returned as we have the sent_to_received monthly ratio basically equal to one. The monthly median amount sent is 3233.45 and dispersion is 1220.44; we can use these statistics to compare this to the amount transacted which is 1971.78 – fairly low compared to the median amount, and this comparison is likely what triggered the model, together with many other features, to classify this as a certain *Smurfing* classification – and this is in fact the correct prediction.

8 Conclusions

The use of feature engineering paired with explainability metrics such as SHAP values are a powerful tool for organizations to employ in order to safely and feasibly implement machine learning models to utilize the power of the model and improve their own processes. While we only tested a few dozen features in terms of engineering, limitations remain with regards to the predictive power of certain classes that require more in-depth understanding of their characteristics before delving into expanding the feature space to include these, particularly gather-scatter, scatter-gather, stacked bipartite and layering typologies which possibly require custom graph algorithms to better determine the "spill-in" or "spill-out" of transacted amounts from and to particular accounts. An additional proposal for future research is to include other supporting machine learning models' outputs, such as Isolation Forest's anomaly scores, as a feature by itself, as well as making use of the lessons learned with regards to performance gain from each extra feature, to each of the typologies, to understand which direction works best for improving the feature space without harming any of the other typologies' performance.

References

[1] B. Oztas, D. Cetinkaya, F. Adedoyin, M. Budka, H. Dogan, and G. Aksu, "Enhancing Anti-Money Laundering: Development of a Synthetic Transaction Monitoring Dataset," in *Proc. 2023 IEEE Int. Conf. e-Business Eng. (ICEBE)*, 2023. doi: 10.1109/ICEBE59045.2023.00028.

[2] A. Kumar and G. Gupta, "Fraud Detection in Online Transactions Using Supervised Learning Techniques," in *Towards Extensible and Adaptable Methods in Computing*, S. Chakraverty, A. Goel, and S. Misra, Eds. Singapore: Springer, 2018, pp. 305–314. doi: 10.1007/978-981-13-2348-5_23.

[3] P. Kumari and S. Mittal, "Fraud Detection System for Financial System Using Machine Learning Techniques: A Review," in *Proc. 2024 11th Int. Conf. Reliability, Infocom Technol. Optimization (ICRITO)*, 2024, pp. 1–6. doi: 10.1109/ICRITO61523.2024.10522197.

[4] A. Barredo Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, 2020. doi: 10.1016/j.inffus.2019.12.012.

[5] L. S. Goecks, A. L. Korzenowski, P. Gonçalves Terra Neto, D. L. de Souza, and T. Mareth, "Anti-money laundering and financial fraud detection: A systematic literature review," *Intell. Syst. Account. Finance Manage.*, vol. 29, no. 2, pp. 71–85, 2022. doi: 10.1002/isaf.1509.

[6] I. Yuhertiana and A. H. Amin, "Artificial Intelligence Driven Approaches for Financial Fraud Detection: A Systematic Literature Review," *KnE Soc. Sci.*, vol. 9, no. 20, pp. 448–468, 2024. doi:

10.18502/kss.v9i20.16551.
[7] H. Ogbeide, M. E. Thomson, M. S. Gonul, A. C. Pollock, S. Bhowmick, and A. U. Bello, "The anti-money laundering risk assessment: A probabilistic approach," *J. Bus. Res.*, vol. 162, 2023. doi: 10.1016/j.jbusres.2023.113999.
[8] D. Hovstadius, *Anti-Money Laundering with*. Uppsala: Uppsala Universitet, 2024.
[9] ComplyAdvantage, "What is the difference between smurfing & structuring?" *ComplyAdvantage*, Sep. 30, 2024. [Online]. Available: https://complyadvantage.com/insights/structuring-vs-smurfing/. [Accessed: Jan. 30, 2025].
[10] M. Jullum, A. Løland, R. Huseby, G. Ånonsen, and J. Lorentzen, "Detecting money laundering transactions with machine learning," *J. Money Laund. Control*, vol. ahead-of-print, Jan. 4, 2020. doi: 10.1108/JMLC-07-2019-0055.
[11] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[12] Intel, "Intel® Extension for Scikit-learn*," *Intel Developer Tools*, 2024. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/oneapi/scikit-learn.html#gs.jidsao. [Accessed: Jan. 30, 2025].
[13] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *arXiv preprint*, arXiv:1705.07874, 2017. [Online]. Available: https://arxiv.org/abs/1705.07874.

Petre-Cornel GRIGORESCU is a financial crime data analyst and model developer in the banking sector as well as a student of the Applied Statistics & Data Science master's program at the Faculty of Cybernetics, Statistics and Economic Informatics from the Bucharest University of Economic Studies. He holds a keen interest in research around both novel and traditional AI/ML applications in the financial crime detection and prevention domain as well as in the wider tech in finance domain.

Antoaneta AMZA is a financial stability expert and a student of the Applied Statistics & Data Science master's program at the Faculty of Cybernetics, Statistics and Economic Informatics from the Bucharest University of Economic Studies. Her research interests encompass financial market modeling, the stability thereof, as well as the role of central banking in managing risks around disruptive, black-box technologies within the financial and banking sectors.