UTRECHT UNIVERSITY

Fraud detection methods in transaction data

by

Jan Androsiuk

A THESIS FOR THE DEGREE OF

Msc in Applied Data Science

Utrecht, Netherlands

July, 2022

SUPERVISORS

Dr. I.R. (Ioana) Karnstedt-Hulpus

Dr. M.W. (Mel) Chekol

MSc. Vahid Joghan Shahrivari

ABSTRACT

Although the number of transaction fraud events grows slower than the number of transactions in total, it is still a problem for many institutions. Detecting fraudulent transactions is challenging for multiple reasons, including a general lack of labels, class imbalance, and hidden and evolving fraud patterns. Even more difficulties emerge while modeling public transaction datasets, namely feature anonymization, missing information, and data aggregation. This work suggests a pipeline of modeling fraudulent transactions, which accounts for most of those concerns based on other researchers' experience. From the modeling approaches, one can distinguish those based on transaction features and those using graph anomaly detection methods. This research combines both methods and presents cross-validation results over two datasets. Performance scores did not indicate the superior predictive power of any presented approach. Nevertheless, the addition of graph features in the case of the second dataset significantly improved validation scores and therefore indicated the direction for further research.

Key Words: [fraud detection, Random Forest, graph theory, transaction data].

Contents

1 Introduction

Credit card fraud and payment fraud are a part of the larger group of financial crimes and can affect all levels of society (Gottschalk 2010). ComplyAdvantage company provides Anti-Money Laundering (AML) technology and tracks statistics in various financial crime branches. In the recent 2022 report, the authors showed that despite a slight decrease, fraud is still one of the top three out of fourteen branches that financial institutions and governments are screening against. From the perspective of big institutions, recent trends focus on fraud and loan defaults regarding COVID-19 relief programs (over $84 billion loss in the USA). On the other hand, the UK has reported over a 70% increase in authorized push payment (APP) fraud cases in the first half of 2021 (ComplyAdvantage 2022). On the bright side, European Central Bank concluded in their latest report (ECB 2021) that throughout 2015-2019 fraudulent card transactions using SEPA increased at a slower pace than card payments in general. Nevertheless, in 2019 the fraudulent ones amounted to €1.87 billion, so there is still scope for improvement. Those statistics emphasize the need to accurately identify fraudulent transactions when prevention measures fail (Bolton and Hand 2002). Some researchers Pourhabibi et al. 2020 underline that detecting fraud is especially critical in emerging areas such as the FinTech industry. It is because such companies are particularly vulnerable to fraud in the early stage of development when security measures are not prioritized.

Modeling fraud on labeled transaction data sets often includes many challenges, such as evolving fraud patterns, capturing fraudulent transactions in real-time, skewed features' distributions, and class imbalance (Abdallah et al. 2016, Awoyemi et al. 2017). Moreover, the data is constrained by security and privacy concerns. Laws such as EU General Data Protection Regulation (GDPR) or California Consumer Privacy Act (CCPA) prevent third parties from using it. Therefore, fraud detection data sets are usually synthetic and often insufficiently rich (Ryman-Tubb et al. 2018). On the other hand, available real-world data sets contain anonymized and often aggregated features, making it impossible to perform inference analysis. Moreover, companies providing such data often delete sensitive observations that could expose real customers and/or merchants. This fact poses an additional task of choosing and implementing the missing data imputation technique. Those factors make fraud detection a particularly challenging task.

1.1 Research aim

This research aims to suggest the modeling pipeline to account for each of the mentioned difficulties (3.2). That includes feature engineering, missing data imputation, dimensionality reduction, and validating Random Forest model (3.5) results. Every mentioned step is based on other researchers' experience in fraud detection. The analysis is performed on publicly available and labeled transaction datasets. The other aim of this research is to evaluate whether representing transaction data as graphs improves the model performance. For this purpose, the graph scheme is proposed, and its node-level and community-level features are chosen and extracted. Finally, those features are merged with the preprocessed original data and the performance is compared. The assumed hypothesis states that adding graph features significantly improves the model score. The analysis for this project is programmed in Python and R languages, and the detailed source code can be found in the GitHub repository[1].

[1] `github.com/JanAndrosiuk/fraud-detection-transaction-data`

2 Literature Review

2.1 Challenges of modeling fraud

There are many industry surveys in the field of fraud detection. One of the most cited is the paper published by Abdallah et al. 2016. The authors reviewed five areas of industry - credit card, telecommunication, healthcare insurance, automobile insurance and online auction. The period captured by their research (1994-2014) reflects the tendency of the credit card and money laundering operations accounting for more than half of the observed fraud. The core of the work focuses on presenting other researchers' methods of dealing with the challenges of modeling fraud in different branches. Regarding the credit card fraud area, the authors mention concept drift, class imbalance, a large amount of data, and the support for real-time detection. As to the concept drift, one of the indicated methods is to train ensemble classifiers by streaming sequential chunks of credit data. This method assures that model addresses fraud pattern evolution. On the other hand, the class imbalance can be handled by random under-sampling majority class, random over-sampling minority class or Synthetic Minority Over Sampling Technique (SMOTE). The latter generates synthetic observations from minority class based on the proximity to actual observations. However, it is also common to perform cost-sensitive learning by tuning class weights to punish the classifier more in case of, e.g. false positive predictions. The Random Forest classifies in the scikit-learn package Pedregosa et al. 2011 provides such a feature. The authors also mention several methods of dealing with large numerosity and dimensionality of data. These include feature selection, aggregation, and extracting eigenvectors from Principal Component Analysis (PCA). However, the PCA in its raw form is not suitable for high cardinality categorical variables (Niitsuma and Okada 2005), while more suitable variations of this method are more complex to implement. As to real-time detection, the general idea behind it is the need for a rapid detection model which also minimizes memory usage. The authors point out multiple algorithms that increase the speed of data processing and/or filtering out relevant data bits. These include Bootstrapped Optimistic Algorithm for Tree Construction (BOAT) and self-organization map (SOM).

2.2 Anomaly detection in transaction graphs

Transaction data can also be transformed into a graph. The graph structure can be created in many ways (Belle et al. 2019). The most common straightforward method is keeping the natural bipartite nature of credit card transactions by assigning cardholders and merchants as nodes with edges as transactions. However, in some cases, merchant nodes are not the origin of the fraud, so it might be more beneficial to make the one-mode projection. This operation casts a bipartite graph to its monopartite equivalent (single population of nodes). In the example of cardholders and merchants, the graph's edge weights represent the sum of connections between each pair of cardholders and merchants. Another approach is to represent transactions as nodes. This way enables the creation of embedding vectors for each transaction easily.

After graph creation, it is possible to detect fraud using the Graph-Based Anomaly Detection (GBAD) methods. Pourhabibi et al. 2020 did a thorough literature review of documented GBAD approaches between 2007 and 2018, covering data mining and machine learning techniques. It appears that GBAD methods can differ on many different levels. Those include:

- Learning method - the choice is correlated with the availability of labels. One can choose between supervised, unsupervised and semi-supervised techniques.
- Nature of the graph - monopartite vs bipartite, homogeneous vs heterogeneous (referring to number of unique connection types), directed vs undirected, static vs dynamic, attributed vs plain (regarding both nodes and edges).
- Occurrence of anomaly - whether anomaly is detected on the level of nodes, edges, sub-graphs, or events.
- Detection method - most approaches include community-based (distinguishing groups of nodes based on their interconnections), probabilistic-based (constructing a model of normal behaviour and tracking outliers), and structural-based (detecting anomalies based on topological features of nodes and/or edges of the graph).
- Structural representation - either feature engineering or graph representation learning. In the case of the first, the authors point out the importance of choosing significant features for the analysis.

One of the conclusions from the paper is that supervised methods are rarely examined - accounting for roughly 5% of collected publications. That happens due to the general lack of labels in real-world data sets. Research trends in other levels seem to favor monopartite, homogeneous, static and attributed combinations. Meanwhile, the division between directed and undirected approaches is almost equal. As to detection methods - community-based approaches are usually selected. The authors also advise using evaluation metrics adequate to the problem of fraud detection using supervised methods. In case of high-class imbalance, it is better to use precision-recall (PR) curves instead of Receiver Operating Characteristic (ROC). In the latter case, a significant change in the number of false positives (FPs, so-called false alarms) will not change the score as much as in the case of PR. Alternatively, F-measure is also suggested as it balances precision and recall scores.

2.3 Approaches

Huang et al. 2018 underline that methods of financial fraud detection usually focus either on network topological features or entity features separately. It is emphasized that although GBAD methods may detect that certain entities are fraudulent, they do not consider why such activities happen based on available data. It is underlined that data pieces in the transaction set are internally linked. Therefore they violate the usual assumption of supervised and unsupervised methods, which is the independent and identical distribution (i.i.d) and auto-correlation. It is also pointed out that detection methods should address different case scenarios of fraud. Those include:

- Outlier points - those address exceptionally low or high attribute values of nodes or edges.
- Merge - a scenario when multiple nodes are connected with one fraudulent node and between themselves. By dispersed transactions of a similar type, the fraud from the central node is covered.
- Ring - while the merge scenario describes the dispersed activity, the import/export chain explains the ring most accurately. In such a chain, all involved nodes are tightly connected and form a directed circle of transactions.

The fact which is necessary to realize is that although outliers can be detected solely from the entity data, the merge and the ring cases require information about the local structures of the network. This remark is the second reason, but both pieces of information should be included in modeling transaction fraud. The authors propose a self-developed algorithm that takes advantage of both approaches.

On the other hand, Xuan et al. 2018 analyzed the B2C transaction dataset provided by the Chinese e-commerce company. The data covered a period from 11.2016 to 01.2017 and included over 30 million observations with 62 features where 0.27% was labeled fraudulent. To detect the fraud, the authors applied two variations of the Random Forest (RF) classifier and other algorithms such as Support Vector Machine (SVM), Naive Bayes classifier, and Artificial Neural Network (ANN). The first RF variation is described in detail in this paper's section (3.5). As to the second, the splitting rule has been changed to account for centers (means) of classes instead of the common *Gini index* approach (Equation 11). The class imbalance was tackled by random under-sampling of the majority class, and the ratio between the training and validation set was set to 7:3. Despite the lack of promising results, the authors compared the first RF approach with every other used algorithm. Validation results were acquired on the sample of training data. Those have shown the superiority of the first RF classifier over every other method in *F-Measure*.

3 Data and methods

3.1 Data description

3.1.1 Vesta dataset

For this study, two data sets are analyzed. The first one is provided by the Vesta Corporation, which specializes in guaranteed payment card-not-present (CNP) transactions for the telecommunication industry. The dataset contains information about online credit card transactions. Although most of the information about dataset features was anonymized and/or aggregated, some aspects are still worth noticing. It consists of 334 continuous variables and 57 categorical ones. Categorical features describe product codes, payment card information, concealed home addresses, e-mail domains, device information and types (operation system information, browser information, screen resolutions, and other), and some aggregated features, e.g. sum of matching names between payment cards. In total, categorical features account for 15040 levels, which by default makes the popular one-hot encoding method useless for classifiers which cannot handle very sparse feature matrices. Numerical ones mainly consist of aggregated features such as countings and rankings and specific ones such as physical distances between transactions and time differences between transactions. The target variable is boolean, and the fraudulent accounts for approximately 7.8% of all observations.

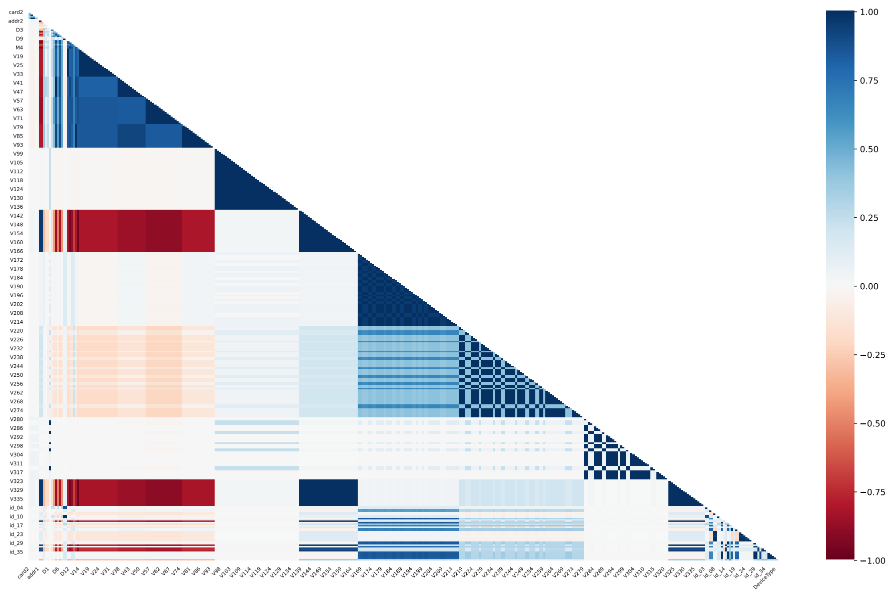
3.1.2 Elliptic dataset

According to the second dataset, it was provided by the Elliptic company, which specializes in money laundering detection by analyzing blockchain transactions. Organizations such as cryptocurrency businesses, financial institutions, and regulators are usually customers of Elliptic and similar companies. The company has been actively collecting bitcoin transaction data since 2013 and claims to have collected over 100 billion data points. The analyzed dataset is the sample of 234355 transactions collected over approximately two years and distributed over 49 time steps. Therefore, the set represents aggregated information about transactions between pairs of entities for each time step. However, in this case, transactions are not solely payments for products but rather transfers between different entities. The dataset contains 166 features assigned to those pairs, which are continuous. Although the vast majority of variables' labels are hidden, the company claims those features relate to mentioned time-step information, transaction fees, the volume of received Bitcoins, counts of incoming and outgoing transactions, and other aggregated data. The target is of boolean type with illicit label comprising around 3.5% of total observations.

3.2 Data preparation

The original dataset from the Vesta Corporation is comprised of variables missing around 22% of information on average. There is also no row with a complete set of observations. That is why some imputation techniques had to be considered. Following the work of Rubin 1976, the Missing at Random (MAR) scenario was assumed to continue the analysis. In reality, it is most probably the missing not at random (MNAR) case due to the hidden anonymization process. Figure (1) there are some variables which most probably directly explain the missing pattern of other features. However, these are not the majority. In most cases, variables lack data simultaneously.

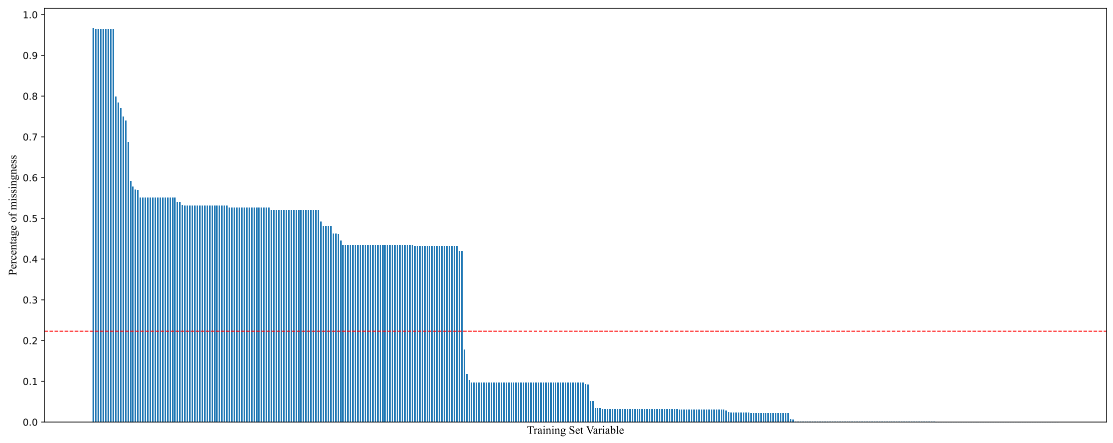
Figure 1: Vesta dataset - Missing pattern correlations



Note: The figure shows correlation patterns between missing values. High positive correlation between variables means that the missing pattern is similar between those two. On the other hand, pairs with low negative correlation may be a sign of causality within them.

Moreover, Figure (2) shows some extreme cases of variables where missing data accounts for 95% of total numerosity. Apart from that, it is worth noticing that there are in total 10821 unique missing value patterns in the training dataset.

Figure 2: Missing values per variable



Note: The figure shows degree of missingness per variable in Vesta train dataset. The red line represents the average missingness of 22.2%.

In such a case, adding variables explaining missing patterns is advised. However, for datasets analyzed in this research, it is impossible. Simple operations such as list-wise deletion or mean imputation (most frequent class imputation in the case of categorical features) should be avoided as those may cause loss of information, an increase in model variance, and distortion of feature distribution. On the other hand, the multiple imputation method via mice and miceforest packages (Buuren et al. 2021, Wilson 2020) proved to be extremely slow for highly dimensional datasets with a large number of observations. Especially in those, where a significant part is missing. That is why it was decided to use multivariate imputer from the well-known scikit-learn package with custom estimators: Random Forest classifier for categorical features and Random Forest regressor for continuous variables.

The Random Forest algorithm is explained in detail in section (3.5) of this paper. According to the source code documentation, the Iterative Imputer trains the model, which predicts values of one feature based on other features. It can either include all of the other variables or just a subset. If some particular count is specified, the algorithm will sample several variables with probabilities proportional to the absolute correlation between the variable mentioned above and other ones. The algorithm can be run multiple times on the same data to ensure convergence with an optional early stopping threshold. The convergence is measured by the change in the infinite norm of a difference between imputed matrices in each time-step (Equation 1).

$$\text{diff} = \|\boldsymbol{X_t} - \boldsymbol{X}_{t-1}\|_\infty \tag{1}$$

Additionally, the algorithm can be set to perform the multiple imputation (Buuren et al. 2021) and generate multiple imputed sets for different random seeds. Then, such datasets can be used as training sets for multiple independent models and the final results can be averaged. Apart from the fact that this approach yields more robust estimations than standard imputation, it may also be used to study the sensitivity of estimations over different random states. However, it is very computationally heavy to perform such imputation in the case of datasets like the one provided by Vesta.

For this research, the parameter of maximum iterations of the Iterative Imputer was set to 10. The motive was to reduce the high computational complexity of the method. With this parameter, the default convergence criterion was not met, but a significant reduction in the difference norm value between iterations could be seen.

Every operation in this research has been run on the laptop with an i7-9750H CPU (12 cores, 2.60GHz) and 32GB RAM (2666MHz, DDR4). With this setup, it took approximately 20 hours to perform the imputation on the training set.

Another problem during the data preparation part was the high cardinality of 43 categorical variables. Those should not be one-hot encoded as it would result in an additional 14442 dimensions in the data set. Therefore, the input matrix would be too sparse for the Random Forest classifier. Using Principal Component Analysis (PCA) is a common practice to reduce the dimensionality of data at known cost (James et al. 2013). The general motive is representing the given data by independent vectors (principal component loadings) directed towards variation. For each mth component, equation (2) must be solved, where n and p respectively stand for the number of observations and features, and ϕ_m is the mth loading vector.

$$\max_{\phi_m} = \left[\frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right) \right] \tag{2}$$

Eigenvalue decomposition or Singular Value Decomposition (SVD) techniques are commonly used for the effective equation solution. Once the loading vector is approximated, analyzed observations may be approximated using formula (3), where z_{im} is the so-called score vector of mth component.

$$x_{ij} \approx \sum_{m=1}^{M} z_{im} \phi_{jm}, \text{ where } z_{im} = \sum_{j=1}^{p} \phi_{jm} x_{ij} \tag{3}$$

It is also worth mentioning that before performing PCA, the data must be normalized to calculate the variation in reference to the center of the coordinate system (and also ease some computations). When PCA is performed on unscaled data, loadings will be skewed towards variables with the highest variances. The number of components is a tunable parameter, which can be either chosen based on marginal explained variance gain, the *scree plot* (both referring to the so-called *elbow* method), or simply by the bottom threshold of variance explained by the PCA model. The last useful feature regarding PCA is the *bi-plot* which can be created based on loadings assigned to features and observations. Such visualization helps analyze correlations in data, but it is suitable only for pairs of components.

Therefore, it becomes useless for higher number dimensions.

Correspondence Analysis (CA) follows similar intuition to the PCA but is more suitable for nominal variables. The general idea is to preserve the χ^2 distances between pairs of observation rows and feature vectors. Following Abdi and Valentin 2007, similarly to PCA, it is crucial to find the matrix which will be subject to the SVD method. This time, the data matrix $\boldsymbol{X}$ is defined as a one-hot encoded matrix of exclusively categorical variables, and $\boldsymbol{Z} = N^{-1}\boldsymbol{X}$ is its version normalized by the sum of all observations of $\boldsymbol{X}$. Then, weights (so-called masses) for rows and columns are defined based on their respective sums (4).

$$\boldsymbol{r} = \boldsymbol{Z}\boldsymbol{1}, \quad \boldsymbol{c} = \boldsymbol{1}^T\boldsymbol{Z} \tag{4}$$

In the above equation, $\boldsymbol{1}$ is the column vector of ones. After this step, metnioned weights are square-rooted, inversed, and represented as diagonal matrices (5).

$$\boldsymbol{D_r} = \operatorname{diag}\left(\frac{\boldsymbol{1}}{\boldsymbol{r}}\right), \quad \boldsymbol{D_c} = \operatorname{diag}\left(\frac{\boldsymbol{1}}{\boldsymbol{c}}\right) \tag{5}$$

Then finally, the subject of SVD is defined by the equation (6).

$$\boldsymbol{D_r}\left(\boldsymbol{Z} - \boldsymbol{r}\boldsymbol{c}^T\right)\boldsymbol{D_c} \tag{6}$$

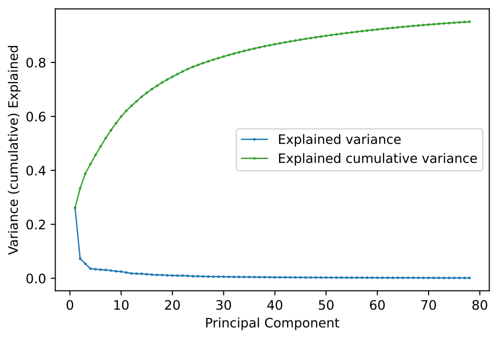
In a nutshell, such representation of data describes deviations of data points from the origin of the vector space, which are weighted by masses of observations and variables. From this point, the process continuous in line with the PCA. As the CA is a method meant only for two nominal variables, Multiple Correspondence Analysis (MCA) is its generalization to a more extensive set of categorical features.

Entity Embedding (Guo and Berkhahn 2016) serves as an alternative to the methods based on SVD. A detailed explanation of this method is beyond the scope of this paper. It was not evaluated due to the complexity of integration with the cross-validation method. Nevertheless, the core concept is to train the Artificial Neural Network (ANN) model with transformed categorical feature matrices as the input and the target vector as the output. The core of this approach lies within mentioned matrices, as their dimensionality accounts for the number of unique records of each feature and the horizontal dimension is the tunable parameter. There is no direct way to calibrate the latter, apart from checking the performance score of the fit. Nevertheless, once the ANN is trained, its weight matrices may be exported to another model. The advantage of this method is that one may arbitrarily choose the output column dimension but at the cost of interpretability.

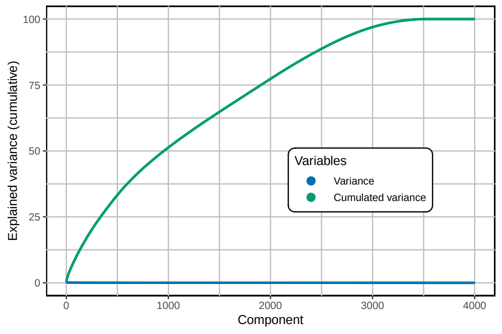
Regarding the Vesta dataset, it was chosen to perform the PCA on numerical features and MCA on categorical ones. After that, the results were concatenated and used as a training set for the Random Forest model. Both PCA and MCA scree plots (Figure (3)) show relatively flat and logarithmic-shaped cumulative curves. That indicates that none of the components explains the variance significantly more than any other. Therefore, the standard *elbow* method of choosing a set of the most significant components is inapplicable. That is why it was decided that for the PCA, the number of the components will be chosen based on the threshold percentage of accumulated variance arbitrarily set to 95%. However, in the case of MCA, such a method would yield more than 3000 components. That is why it seemed more rational to let the Random Forest handle Vesta set categorical variables automatically. This feature is briefly explained in section (3.5).

Figure 3: Variance and cumulated variance of PCA and MCA components for the Vesta dataset

(a) PCA Scree plot



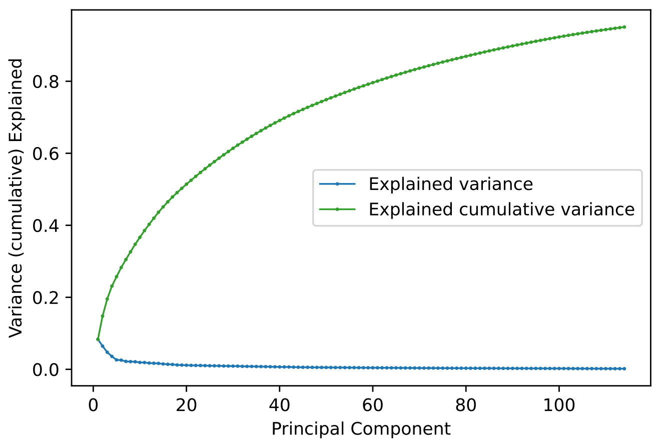
(b) MCA Scree plot



Note: X axis refers to an index of particular component (beginning with the one explaining the highest fraction of variance), while Y axis describes explained variance. Figures differ in aesthetics as the PCA was created in Python matplotlib package, while MCA - with R ggplot package. Nevertheless, both Y axes refer to the same scale (0-100% of explained variance).

As to the Elliptic dataset, the target variable was re-engineered to inform about the fraudulent transaction (edge) rather than the illicit node. For this purpose, it was assumed that if at least one node in a transaction pair was labeled as illicit, the whole transaction is considered fraudulent. As the dataset did not contain any categorical features, only the PCA was performed again for dimension reduction. Figure (4) shows the pattern of explained variance distribution for the Elliptic training set. The conclusion remains the same as in the Vesta set, which is why it was again decided to use the threshold variance. That accounted for 114 variables.

Figure 4: Elliptic dataset - PCA Scree plot

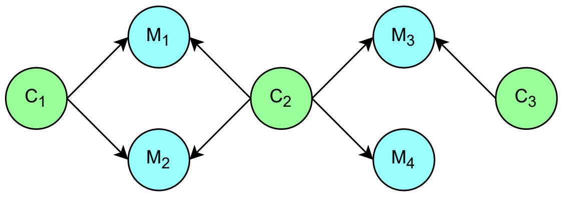


Note: X axis refers to index of particular component (starting from the one explaining the highest fraction of variance), and Y axis - to the explained variance.

3.3 Graph scheme

As was mentioned in the section 1, the part of this research aims to examine the contribution of graph features inclusion on the performance of the fraud detection model. The crucial part of this task is to define a graph scheme. Following Belle et al. 2019, one of the common approaches is to create two distinct populations of nodes - cardholders and merchants. That yields a bipartite representation of the data. In such a scheme, edges have equal weight but may include different attribute values (Figure 5).

Figure 5: Bipartite graph scheme for online transactions

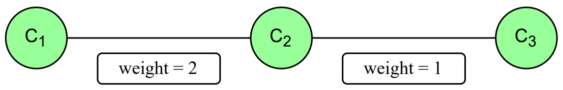


Note: Nodes signed as C represent cardholders, while M represent merchants. In case of the Vesta dataset those refer to accounts and devices, respectively.

For the Vesta dataset analysis, a bipartite graph scheme included two populations of nodes - cardholders and devices. Cardholders were identified by the string concatenation of features referring to payment cards. On the other hand, entries of the device-info feature were assigned as device nodes.

The problem with such representation comes when calculating some particular graph metrics. Eigen-vector centrality is one such example, as it is heavily based on the directed nature of the graph and is not robust to nodes outside of *Strongly Connected Components (SCCs).* In the case of the Vesta dataset, the number of SCCs equals the nodes' count. That is because merchants cannot point back to cardholders. Due to this fact, in some cases, it might be worth performing *one-mode projection* (projecting from bipartite to the monopartite scheme). This operation aggregates bipartite edges by summing them. Furthermore, it assigns those sums to weight attributes of new edges in the monopartite scheme (Figure 6).

Figure 6: One-mode projection to monopartite graph scheme



Note: One-mode projection from bipartite graph from the Figure (5). C stands for cardholder nodes while weight is the new edge attribute.

This method yields an undirected graph that describes a number of joint intermediaries connect cardholder nodes. Such a scheme is more robust to multiple graph operations than a bipartite one. Mathematically, one-mode projection can be obtained by the dot product of an adjacency matrix $\boldsymbol{A}$ with its transposition and zeroed out diagonal (Equation 7). In the equation below, $\boldsymbol{D}$ refers to the matrix $\boldsymbol{Y}$ with preserved diagonal elements and zeroed-out rest.

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{D} \tag{7}$$

In the case of the Vesta dataset, the one-mode projection was used on the bipartite scheme to aggregate information about connections between cardholders. As to the Elliptic dataset, the monopartite projection was imposed by default, because the dataset in its raw form represented only one population of account nodes.

3.4 Graph features

This research focuses on detecting fraudulent transactions. This fact is particularly interesting while analyzing bipartite projection on the Vesta set, where every record contains information about the origin node (account) and the destination node (device). It means that statistics regarding both populations enrich the final dataset. However, transaction data in the Elliptic set was extended only by statistics from a single population of nodes. The graph features were extracted using the Neo4j graph database management system queries via Python scripts. Subsections below describe the choice of graph metrics.

3.4.1 Centrality metrics

The first metric is *degree centrality*, which informs about number of neighbors of a particular node (Freeman 1978). In the case of directed graphs, degree centrality can be further distinguished between in-degree and out-degree. The metric for a particular node can be easily computed as a sum of either row or column of the adjacency matrix.

On the other hand, *betweenness* centrality of the node informs about the ratio of shortest paths that go through that node to all shortest paths that occur in the graph. As calculating every shortest path in the graph is highly computationally expensive, Neo4j uses Brandes' approximate algorithm (Brandes and Pich 2007). In this approach, start nodes are sampled with a probability relative to their degree. The time complexity for each start node is $O(nm)$, where n and m refer to the number of nodes and edges, respectively.

The *PageRank* centrality is an iterative algorithm based on assumptions that state: i) particular node's importance increases with its in-links count, ii) that it's also influenced by an importance of those in-links source nodes, and iii) that each node's importance is divided equally among destination nodes (Brin and Page 1998). The iterative process is defined in Equation (8), where $\boldsymbol{q}$ is the PageRank centrality vector in a time-step t, $\boldsymbol{P}$ is the transition matrix (the adjacency matrix $\boldsymbol{A}$ which values are divided over the node degree matrix $\boldsymbol{D}$), n stands for the node numerosity, and β is the tunable parameter.

$$\boldsymbol{q}^{(t)} = \beta \boldsymbol{P}^T \boldsymbol{q}^{(t-1)} + \frac{1-\beta}{n}\mathbf{1}, \quad \boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{D}^{-1} \tag{8}$$

In simple words, creating matrix $\boldsymbol{P}$ refers to assumptions i) and iii), while the iterative process of passing scores between nodes refers to the assumption ii). On the other hand, parameter β accounts for the problem of *sink-nodes* and *periodic traps* which may cause the score to get zeroed and/or not lead to a stationary state. The parameter balances the inheritance of neighbors and the node itself. The convergence of the above formula can be proved by transformation to *ergodic Markov chain* model, which is guaranteed to converge.

Hyperlink-Induced Topic Search (HITS) expands on the idea taken from PageRank by accounting for both in-degree metric (*authorities*) and the out-degree (*hubs*) (Kleinberg 1999). The iterative process is very similar to the PageRank, apart from the fact it is normalized in the Euclidean space sense (meaning that squares of scores sum up to one) rather than in the probabilistic sense (scores sum up to one). Update rules can be seen in the Equations (9), where $\boldsymbol{a}$ and $\boldsymbol{h}$ refer to scores of authorities and hubs, respectively. α and β are tunable parameters to adjust relative weights between those two scores.

$$\begin{gathered} \boldsymbol{a}^{(t)} = \alpha \boldsymbol{A}\boldsymbol{h}^{(t-1)}, \quad \boldsymbol{h}^{(t)} = \beta \boldsymbol{A}\boldsymbol{a}^{(t-1)} \\ \|\boldsymbol{a}^{(t)}\| = 1, \ \|\boldsymbol{h}^{(t)}\| = 1 \end{gathered} \tag{9}$$

As can be seen, those two scores are complementary to each other, meaning that one is dependent on the other. Moreover, as mentioned before, both scores are normalized after each iteration t. The convergence of both scores when the number of iterations approaches infinity can be proved by *Perron Frobenius theorem.*

3.4.2 Component and community metrics

The *Weakly Connected Component (WCC)*, which was firstly introduced by Galler and Fisher 1964, represents a subgraph which other subgraphs cannot reach. SCCs (3.3) and WCCs yield different results for directed graphs, but they are the same for directed ones. In the SCC, every node is reachable by others, while in the WCC, no cycle is required. It was not specified in the Neo4j documentation which algorithm is used to detect WCCs. However, common implementations use either *Breadth-First Search (BFS)* (Moore 1959) or *Depth-First Search (DFS)* (Tarjan 1972), which are both popular algorithms to perform graph traversing. The difference lie in data-structures used for implementation - *priority queue* for BFS, and *stack* for DFS. For this research, only the size of the WCC to which a particular node belongs is analyzed.

Louvain method (Blondel et al. 2008) is a more sophisticated method of exploring relevant subgraphs. The method follows the idea that mentioned subgraphs should have strong internal and weak external connections. The algorithm does so, by maximizing *modularity* score for the whole graph (Equation 10).

$$Q = \frac{1}{2m} \sum_{ij} \left(\boldsymbol{A}_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j), \tag{10}$$

where:

- $\boldsymbol{A}_{ij}$: Adjacency matrix cell representing edge weights between nodes i, j
- k_i and k_j: node degrees
- C_i and C_j: indices of communities that nodes i and j belong to
- δ: *Kronecker delta function* (1 if $C_i = C_j$, else 0)

- m: sum of edges in the graph

The difference inside parentheses informs about the deviation of edge weights from its expected value. The score itself belongs to $[-0.5, 1]$, with scores higher than 0.3 meaning in practice that the graph has a significant community structure. The algorithm is two-phased. Initially, every node is in its community, and the process only assigns nodes to their neighbors' community if that improves the modularity score. This phase repeats until there is no possible improvement for each node. In the second part, the algorithm aggregates communities by representing them as nodes and assigning new edge weights as sums of previous edge weights between communities. This phase is also performed until reaching a stationary state. Such transformed network is then re-iterated with both steps until the mentioned criterion is reached. Some implementations also include the *resolution* parameter, which controls the final sizes of communities. It is sometime advantageous because the Louvain method is an *approximation algorithm* and therefore might yield inconsistent results. Tuning the resolution parameter between iterations of the algorithm helps with this fact. Following the WCCs detection, only the sizes of the communities were appended to the training set.

3.5 Prediction model

Regarding the actual classification of fraudulent transactions, it was decided to use the Random Forest (RF) classifier. The choice was dictated by the promising performance of other researchers within the field of fraud detection (Xuan et al. 2018, Sahin and Duman 2010, Seyedhossein and Hashemi 2010). First of all, it is worth explaining how the simple Decision Tree (DT) algorithm works (James et al. 2013). The idea is to divide the feature space into high-dimensional rectangles called *boxes* which yield information about the expected class label. Those boxes are created by iteratively splitting the feature space, and the splitting point in each iteration is determined by the *purity* of created splits. The purity itself is based on the proportion of class labels in each box. The metrics which are commonly used for this purpose are: *Gini index* (11) and *Entropy* (12). In following equations, K refers to number of class labels and $\hat{p}_{mk}$ - to proportions of those classes in mth box.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{11}$$

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}) \tag{12}$$

For the two-class problem, $G \in [0, 0.5]$, and $D \in [0, -log(0.5)]$ (as $\lim_{p \to 0} p \log(p) = 0$) and in both cases the goal is to minimize the score for each box. However, it would be computationally expensive to evaluate which point is splitting the space optimally for each variable and then reiterate through each possible sequence of variables. Therefore, decision trees use *greedy* approach by default. In each step the algorithm chooses the best combination of feature and observation (within this feature) to perform the split. Although such an approach is easy to implement, it does not handle well situations where a less optimal intermediate step leads to the a better overall result. The Random Forest builds on top of ideas of Decision Trees by performing the *bagging* operation, which is a method of averaging results from multiple DTs trained on different samples of the original set. In such a way, the algorithm deals with the problem of high variance which occurred in DTs. Moreover, on top of sampling observations, RF also samples features. Consequently, the algorithm can overcome the problem of correlation between variables which skews the results. As correlated variables occur commonly, it is advised to use a low number of variables in each sampled set, which is usually the square root of total numerosity. It is worth mentioning that tree-based methods can automatically deal with nominal variables. However, there is a seldom-mentioned problem regarding nominal variables with high cardinality. It is logical that in such case, the algorithm will tend to split the feature space based on individuals rather than groups of observations leading to overfitting. That was one of the reasons to perform PCA and/or MCA on both Vesta and Elliptic sets. The other one was to make sure that the final dataset is universal for future implementations (using other estimators), which may not be able to handle categorical variables automatically.

4 Results

For the purpose of this research, several models have been validated. For the Vesta set, following approaches have been evaluated.

Vesta baseline - the model was trained on the dataset with missing data imputed by the Iterative Imputer. Numerical variables were replaced with principal components using PCA to reduce dimensionality and simultaneously account for 95% of existing variance. Categorical variables were label-encoded, and no further transformations were applied. Finally, the RF classifier with default scikit-learn parameters was used to generate predictions in cross-validation. To account for the class imbalance, the weight parameter inside classifier settings was set to *balanced* to randomly over-sample minority class.

Vesta bipartite - this approach extends the baseline by the addition of bipartite graph features. Section (3.3) explains bipartite scheme in detail. Selected graph metrics are thoroughly described in the section (3.4) of this paper. Ultimately, each transaction record in the dataset was enriched by metrics describing the origin and source node.

Vesta mono - the approach serves as an alternative to the bipartite scheme. The idea behind it is to emphasize the relationship between cardholders at the cost of information aggregation. Details of the one-mode projection are described in section (3.4). The selection and addition of mentioned metrics follows the bipartite approach.

As to the Elliptic set, only two approaches were validated due to the monopartite nature of the data.

Elliptic baseline - continuous data was transformed with PCA to capture 95% of existing variance. Settings of the classifier remained the same as in the Vesta dataset analysis, because the random search method did not show any improvement.

Elliptic graph - the baseline Elliptic approach enriched with graph features. The choice of metrics and their implementation follows the Vesta set analysis procedure.

Table (1) shows the results for all of the validated approaches. Those were acquired using the same random seed to make them comparable.

Table 1: Cross validation results for multiple models

Approach	Accuracy	Precision	Recall	F1-score
Vesta baseline	95.25%	92.04%	43.14%	58.73%
Vesta bipartite	95.28%	92.28%	43.45%	59.06%
Vesta mono	95.27%	92.16%	43.44%	59.03%
Elliptic baseline	98.00%	90.54%	47.53%	62.32%
Elliptic graph	98.18%	92.02%	52.09%	66.51%

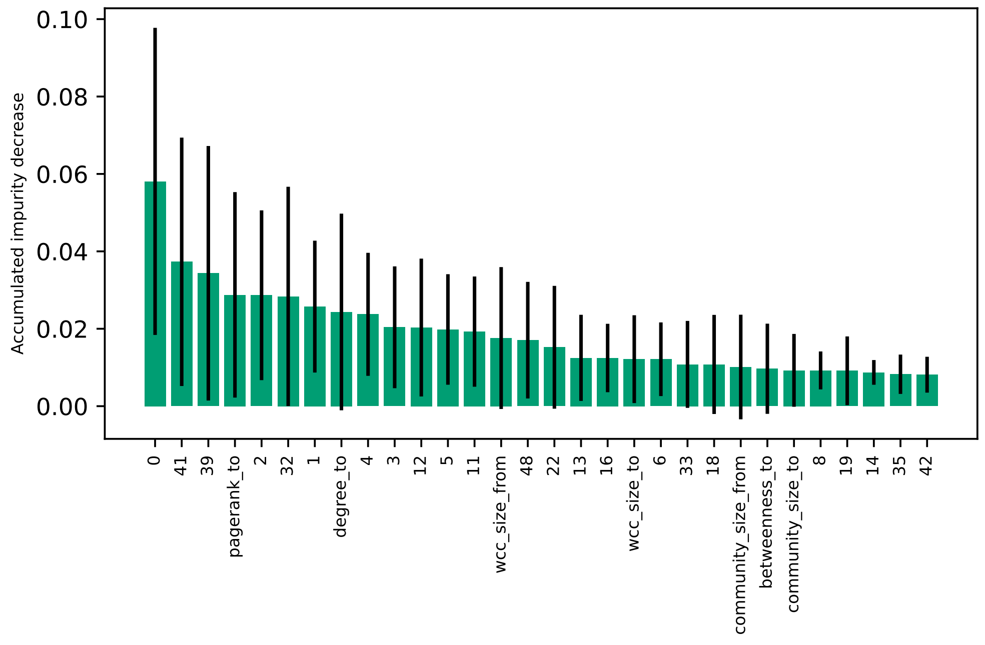
Note: The table shows cross validation results for multiple models. Specification of each approach is described in the section (4) of this paper. Random seed was set using the *Random* python package to a value of 2022 in order to make results comparable within the same dataset.

Firstly, the above results indicate that any proposed graph representation of the

Vesta dataset significantly improved the fraud detection model performance. However, it might be the case that de-anonymization of the data would lead to better results (section 5). Every selected metric confirms those results.

However, the Elliptic dataset analysis results indicate the opposite conclusion. The improvement can be especially seen in the Recall score. This metric is especially crucial as it captures information about transactions which were falsely labeled as non-fraudulent. Although that score remains relatively low in this approach, its improvement can serve as a promising direction for further research. To investigate that more, feature importance order was investigated (Figure 7).

Figure 7: Elliptic graph approach - feature importance order



Note: The figure represents importance scores of analyzed variables of Elliptic dataset in descending order. The Y axis informs about the accumulated impurity decrease, which is the total decrease in node impurity weighted by the proportion of samples reaching that node and averaged over all trees of the ensemble method. Black error lines on each bar represent standard deviation of the score over all trees. Suffixes *from* and *to* refer to origin and destination nodes, respectively. On the other hand, numbers in X axis represent indices of principal components.

Feature importance scores indicate that graph metrics were indeed significant for the analysis. Especially PageRank and degree scores of destination nodes significantly reduced the node impurity. However, based on the relatively significant standard deviation of those scores, it is suggested to average them on the larger number of iterations in further research.

5 Conclusion and Discussion

The research aimed at suggesting the pipeline of modeling transaction data to detect the underlying fraud. Presented framework handles the challenges of missing data imputation using the Iterative Imputer, transformations of categorical data (PCA, MCA), creating graph scheme, extracting graph features, and finally detecting fraudulent transactions using the class-balanced RF classifier. Every method and processing step were described in sections before.

The Vesta dataset proved to be relatively more challenging than the Elliptic, which is partially why final performance scores favoured the latter. It has also been evaluated that further aggregation using one-mode projection did not yield any visible improvement. Although using card data to represent cardholder signatures seemed promising, that cannot be said about the device information data. It might be the case that getting more specific, user-oriented data such as anonymized IP addresses and more specific device information could help form a more densely connected network. Entities of such a graph would then represent less aggregated features and could potentially distinguish fraudulent nodes.

On the bright side, Elliptic dataset analysis confirmed the hypothesis that projecting transaction data as a graph and including its features may significantly improve the overall predictive ability. Those observations were confirmed by the feature importance investigation, which showed that some of the included graph features are at the forefront of the importance order. Nevertheless, their impact should be investigated more by averaging multiple iterations.

5.1 Further research

There are multiple aspects of this research that could have been investigated, if not for computational time limitations. The first such thing is hyperparameter tuning. Given the used hardware (section 3.2), it took approximately 15 hours to evaluate 15 combinations of hyperparameters using the RF classifier. Such time made the thorough tuning process difficult. Moreover, it would be beneficial to compare presented approaches with state-of-the-art graph representation learning methods such as Graph Neural Networks (GNNs). It would be particularly interesting in the case of the Elliptic dataset, which modeling yielded promising results. Moreover, entity embedding and multiple imputation methods should be implemented within the cross-validation process and compared with the MCA approach. Apart from that, every combination of approaches should be tested against different random seeds to check its robustness. As to the classifier implementation itself, it would be worth evaluating implementations based on graphical processing units which offer higher parallelism of computations.

References

Abdallah, Aisha, Mohd Aizaini Maarof, and Anazida Zainal (2016). “Fraud detection system: A survey”. In: *Journal of Network and Computer Applications* 68, pp. 90–113.

Abdi, Hervé and Dominique Valentin (2007). “Multiple correspondence analysis”. In: *Encyclopedia of measurement and statistics* 2.4, pp. 651–657.

Awoyemi, John O, Adebayo O Adetunmbi, and Samuel A Oluwadare (2017). “Credit card fraud detection using machine learning techniques: A comparative analysis”. In: *2017 international conference on computing networking and informatics (ICCNI)*. IEEE, pp. 1–9.

Belle, Rafaël Van, Sandra Mitrović, and Jochen De Weerdt (2019). “Representation learning in graphs for credit card fraud detection”. In: *Workshop on Mining Data for Financial Applications*. Springer, pp. 32–46. DOI: `10.1214/ss/1042727940`.

Blondel, Vincent D, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre (2008). “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10, P10008.

Bolton, Richard J and David J Hand (2002). “Statistical fraud detection: A review”. In: *Statistical science* 17.3, pp. 235–255. DOI: `10.1214/ss/1042727940`.

Brandes, Ulrik and Christian Pich (2007). “Centrality estimation in large networks”. In: *International Journal of Bifurcation and Chaos* 17.07, pp. 2303–2318.

Brin, Sergey and Lawrence Page (1998). “The anatomy of a large-scale hypertextual web search engine”. In: *Computer networks and ISDN systems* 30.1-7, pp. 107–117.

Buuren, Stef van, Karin Groothuis-Oudshoorn, Gerko Vink, and Rianne Schouten (2021). *mice: Multivariate Imputation by Chained Equations*. R package version 3.14.0. URL: `https://github.com/amices/mice`.

ComplyAdvantage (2022). *The State of Financial Crime 2022.* URL: `https://complyadvantage.com/wp-content/uploads/2022/01/ComplyAdvantage-State-of-Financial-Crime-2022.pdf`.

ECB (2021). "Seventh report on card fraud". In: DOI: `10.2866/793033`. URL: `ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202110~cac4c418e8.en.html`.

Freeman, Linton C. (1978). "Centrality in social networks conceptual clarification". In: *Social Networks* 1.3, pp. 215–239. ISSN: 0378-8733. DOI: `doi.org/10.1016/0378-8733(78)90021-7`. URL: `sciencedirect.com/science/article/pii/0378873378900217`.

Galler, Bernard A and Michael J Fisher (1964). "An improved equivalence algorithm". In: *Communications of the ACM* 7.5, pp. 301–303.

Gottschalk, Petter (2010). "Categories of financial crime". In: *Journal of financial crime* 17.4, pp. 441–458.

Guo, Cheng and Felix Berkhahn (2016). "Entity embeddings of categorical variables". In: *arXiv preprint arXiv:1604.06737*.

Huang, Dongxu, Dejun Mu, Libin Yang, and Xiaoyan Cai (2018). "CoDetect: Financial fraud detection with anomaly feature detection". In: *IEEE Access* 6, pp. 19161–19174.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An introduction to statistical learning.* Vol. 112. Springer.

Kleinberg, Jon M (1999). "Authoritative sources in a hyperlinked environment". In: *Journal of the ACM (JACM)* 46.5, pp. 604–632.

Moore, Edward F (1959). "The shortest path through a maze". In: *Proc. Int. Symp. Switching Theory, 1959*, pp. 285–292.

Niitsuma, Hirotaka and Takashi Okada (2005). "Covariance and PCA for categorical variables". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, pp. 523–528.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pourhabibi, Tahereh, Kok-Leong Ong, Booi H Kam, and Yee Ling Boo (2020). “Fraud detection: A systematic literature review of graph-based anomaly detection approaches”. In: *Decision Support Systems* 133, p. 113303.

Rubin, Donald B (1976). “Inference and missing data”. In: *Biometrika* 63.3, pp. 581–592.

Ryman-Tubb, Nick F, Paul Krause, and Wolfgang Garn (2018). “How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark”. In: *Engineering Applications of Artificial Intelligence* 76, pp. 130–157.

Sahin, Yusuf and Ekrem Duman (2010). “Detecting credit card fraud by decision trees and support vector machines”. In: *World Congress on Engineering 2012. July 4-6, 2012. London, UK.* Vol. 2188. International Association of Engineers, pp. 442–447.

Seyedhossein, Leila and Mahmoud Reza Hashemi (2010). “Mining information from credit card time series for timelier fraud detection”. In: *2010 5th International Symposium on Telecommunications.* IEEE, pp. 619–624.

Tarjan, Robert (1972). “Depth-first search and linear graph algorithms”. In: *SIAM journal on computing* 1.2, pp. 146–160.

Wilson, Samuel (2020). *miceforest: Fast, Memory Efficient Imputation with lightgbm.* Python package version 5.0.0. URL: `https://pypi.org/project/miceforest/`.

Xuan, Shiyang, Guanjun Liu, Zhenchuan Li, Lutao Zheng, Shuo Wang, and Changjun Jiang (2018). “Random forest for credit card fraud detection”. In: *2018 IEEE 15th international conference on networking, sensing and control (ICNSC).* IEEE, pp. 1–6.