Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS

Faculty of Chemistry, Biotechnology and Food Science
Associate Professor Aliaksandr Hubin, Assistant Professor Jesper Frausig, Senior Research Scientist Martin Jullum

Using Graph Bayesian Neural Networks for fraud pattern detection and classification from bank transactions data

Osama Abidi
MSc Industrial economics

Contents

1. INTRODUCTION

1.1. Definition

Money laundering is a practice believed to be as old as money itself (Muller, 2007). With the emergence of law enforcement in any society, the development of methods for legitimizing income generated by illegal activities is soon to follow. Money laundering can be defined as any process that aims to conceal the origins of money obtained through illicit means and is an essential enabler for crimes with profit-oriented motives on a large scale. As authorities have increased their efforts in detecting money laundering, criminals have developed increasingly sophisticated ways of laundering money, often utilizing transactions involving multiple entities over various time frames.

1.2. Scale

Since most money laundering goes undetected, there is great uncertainty regarding its scale and regularity. There have been produced various estimates of the amount of money that is laundered on a global level. The United Nations Office on Drugs and Crime estimates that the figure might be as large as 2-5% of global annual GDP (Johansen, 2007). However, given the large uncertainty of the methods used and the great variance of the estimates, there is little feasibility of any consensus on the matter (Reuter, 2013).

1.3. Brief history of anti-money laundering

From the twentieth century and onward, there have been ever-increasing efforts by authorities to squander money laundering. The modern history of anti-money laundering (AML) laws can be traced back to the 1930s. With the surge of organized crime in the interwar period in the US, these laws became an important tool for authorities in their efforts to crack down on the mafia (Johannessen, 2022). Since then, the nature of the crime has evolved. Uncovering money laundering is therefore an evolving problem, as the perpetrators constantly are adapting new strategies and methods as law enforcement renders older methods obsolete. With the establishment of the international organization Financial Action Task Force (FATF) in 1989, countries with inefficiencies in their AML measures were now incentivized to expand and improve upon their efforts to detect money laundering. This marks the beginning of an intensified campaign to stop money laundering that continues to this day.

1.4. Electronic surveillance systems

Given the large volume of transactions that banks facilitate, automating the reporting of suspicious behavior is a task of high priority as it is virtually impossible for a financial institution to manually monitor all the activities it is involved in. Furthermore, since only a small minority of all transactions are part of a money laundering scheme, a system of selecting activities to investigate through random sampling would have to be on a scale that is both large and expensive. Electronic surveillance represents a solution to these problems and is the context in which this thesis is written.

Electronic surveillance systems utilize computers to automate the tedious task of screening transactions and parties for potential money laundering. For most of their history, electronic surveillance systems have relied upon pre-defined thresholds and rules to alert of suspicious activities. These systems would flag transactions if they met a set of criteria relating to things such as the size of the transaction and the

locations of the parties involved (Reuter, 2013). These alerted activities are then further investigated by domain experts.

These criteria-based surveillance systems are for the most part insufficient. They are often too simple to detect the complex patterns and structures of money laundering, and owing to how slow and laborious they are to keep up to date, they are also easy for money launderers to circumvent. In spite of the many billions invested into these systems, they still have been reported of having a false-positive rate of up to 95% (Reuter, 2013). Although both the number and quality of notices have increased in the European context in recent years (Cotoc et al., 2021), there is still a necessity for the development of more sophisticated ways to monitor transactions.

1.5. Supervised machine learning

With the boom of machine learning capability in the last decade, machine learning for AML purposes has become an increasingly viable option. In the context of electronic surveillance, machine learning methods represent a tool that can uncover fraudulent transactions and parties based upon complex patterns that are challenging, if not impossible for humans to understand.

While all subcategories of machine learning are thought to have some application to AML (Sudjianto et al., 2010), this thesis mainly revolves around supervised machine learning methods. This branch of machine learning requires both explanatory variables and response variables, which are used to train models that accurately predict unseen response variables from observed data. Such methods more successfully achieve what criteria-based monitoring systems aim to do, namely creating an accurate mapping between the observed transactional data and their level of suspiciousness as defined by a domain expert.

In the context of AML, supervised machine learning methods are advantageous to criteria-based systems in a variety of ways. With ever-improving methods and model architectures, supervised machine learning models can offer relatively accurate predictions with greater flexibility with regard to input than pre-defined thresholds do. In addition to improving accuracy, these models can also reduce the false positive rate; a task that is of almost the same importance as a high true positive rate (Grint et al., 2017). Many models also offer a large degree of explainability and can offer insight into what variables are decisive as to whether a transaction or party is fraudulent or not.

There have been trained many classifiers for AML purposes throughout the years. In Tang and Yin (2005) Support Vector Machines were applied in an effort to "improve the embarrassments of anti-money laundering (AML) intelligence collection" and found the methods comparatively successful at dealing with large and complex data. Neural Networks and ensemble models composed of decision trees have also been widely explored, with the former having a long history of being applied for fraud detection (Sudjianto et al., 2010).

Of particular interest for this paper, is the work of Jullum et al. (2020). The paper applies the ensemble method XGBoost on the same data set that this work uses. The data set from DNB is both real and large, and the paper is considered the first published AML study of such a magnitude. The data, along with the ways the paper has inspired the data modeling of this work will be discussed further in Section 3. Jullum et al. (2020) introduces an evaluation metric that makes the results of the method comparable

to that of existing methods and demonstrates that their method is preferable to the current criteria-based systems. The paper also suggests that the approach can be extended by including information about how the money is transferred through the network of transacting parties.

Another work utilizing this data set is that of Hubin (2019). By treating the transactional data as a graph, the author used random walks to construct node embeddings for each transacting party. When these embeddings were visualized in a space with reduced dimensionality, they formed distinct clusters. However, there wasn't any notable difference observed between the nodes associated with money laundering and the normal nodes. Additionally, the study attempted to utilize these embeddings as input for a random forest classifier. Unfortunately, this approach yielded a relatively poor predictive performance. Although a great portion of the work was unsupervised, Hubin (2019) recommended further research in this direction using semi-supervised methods.

Another study of interest is the Master's thesis by Johannessen (2022). This work makes use of a DNB data set which is both larger and more descriptive than the one this thesis will use. The author uses Graph Neural Networks (GNNs) to better incorporate the network information in the data for the purposes of node classification. The experiments done are some of many that demonstrate that GNNs are superior in node classification on data that can be represented as graphs.

Supervised machine learning models do, however, also pose a set of challenges and limitations. Supervised machine learning is entirely reliant upon labeled data sets, and while the availability of labeled AML data sets is increasing, it is still relatively limited for the time being. Whereas unsupervised machine learning methods could help expand our understanding of the complex structures and patterns in the data sets without being biased by our insufficient and uncertain labels (Paula et al., 2016), supervised machine learning is biased and wholly dependant on the labels that are available. This makes the methods subject to the same limitations that our knowledge of the issue poses. In other words, supervised machine learning models are often ever as good as the domain experts investigating the fraud themselves, and the many uncertainties associated with the labels they assign will be incorporated into the models.

1.6. Sources of uncertainty when training on labeled AML data sets

Typical AML data sets come with a set of uncertainties that can be detrimental to the performance and utility of machine learning models (Bolton and Hand, 2002).

Class mislabeling The arguably largest of these sources of uncertainty is that many money laundering transactions go undetected. In fact, it is virtually impossible to know how much money laundering there is that goes undetected, owing to the large uncertainty around the scale of the practice (Reuter, 2013). There are likely many money laundering methods and patterns being used that domain experts and surveillance systems simply do not recognize as fraudulent. This means that models will be trained on data sets where many fraudulent transactions and parties are incorrectly labeled as normal. In the worst case, it might be that fraudulent cases that are incorrectly labeled outnumber the ones that are labeled correctly.

Class overlap Even among the transactions that follow patterns and structures that domain experts to a reliable degree can identify as fraudulent, one encounters significant uncertainty from class overlap.

Supervised methods often classify incorrectly when instances of different classes overlap in their feature space. Although overlap is expected, the issue is exacerbated when the alert labels are made using fixed, criteria-based monitoring, where there is a possibility that known patterns of money laundering go undetected if the characteristics of the transactions deviate from predefined thresholds by even the slightest of margins. The large false positive rate of the surveillance systems also leaves a lot of work for domain experts to sort out transactions that should be investigated further. This contributes to the uncertainty, as the scarcity of actual fraud and the inherent ambiguity of the transactions can lead to human error when labeling.

Class imbalance The large class imbalance in AML data sets represents another source of uncertainty. In the United States, only 0.05 - 0.1% of 700,000 wires processed daily were found to be implicated in money laundering (Sudjianto et al., 2010). Machine learning models trained on data sets where the class distribution is highly skewed can become biased in favor of the majority class. As the models strive to maximize accuracy, their predictions tend to heavily favor the majority class. This is because the majority class has a greater contribution to the overall prediction accuracy than the minority class.

1.7. Uncertainty Quantification

These sources of uncertainty make Uncertainty Quantification (UQ) of particular interest when using supervised methods on AML data. While there are many ways traditional supervised methods can quantify uncertainty, this thesis will utilize Bayesian methods as they offer a more comprehensive framework for UQ (Ghahramani, 2015). UQ offers a new dimension of information associated with the results of a model, and can indicate that particular predictions deviate from others in ways that might be significant. While the black box quality of many supervised methods can make it challenging to interpret the practical implication of such deviations, manual investigation into the predictions of interest could yield interesting results. In the context of AML specifically, a non-fraudulent prediction with a relatively large uncertainty might indicate unusual patterns of behavior that could be interesting to investigate further. A fraudulent prediction with large uncertainty is also equally interesting to explore. By looking at the larger network of transacting parties connected to the predictions of interest, one could hope to uncover patterns that could aid in expanding the qualitative understanding of money laundering.

1.8. Present work

This thesis aims to conduct such an investigation into these nodes of interest by synthesizing many of the methods described in the Introduction. Such a methodology begins by training a Bayesian graph neural network on the same data set as Jullum et al. (2020) to learn the relationship between the money laundering status of a transacting party and its features and network properties. One can then get node embeddings by extracting the activation values of each prediction at the last hidden layer in the model. This will hopefully yield informative node embeddings that one could then visualize along with their predicted class and associated uncertainty by applying dimensionality reduction through Principal Component Analysis. If there are nodes that have the same predicted class and magnitude of uncertainty that cluster together, one could try retrieving their associated networks and investigate the patterns.

One can formulate the aim of the thesis into the following sequence of research questions:

1. Can Bayesian Graph Neural Networks and PCA yield node embedding visualizations that show clustering of nodes of similar predictions and uncertainty?

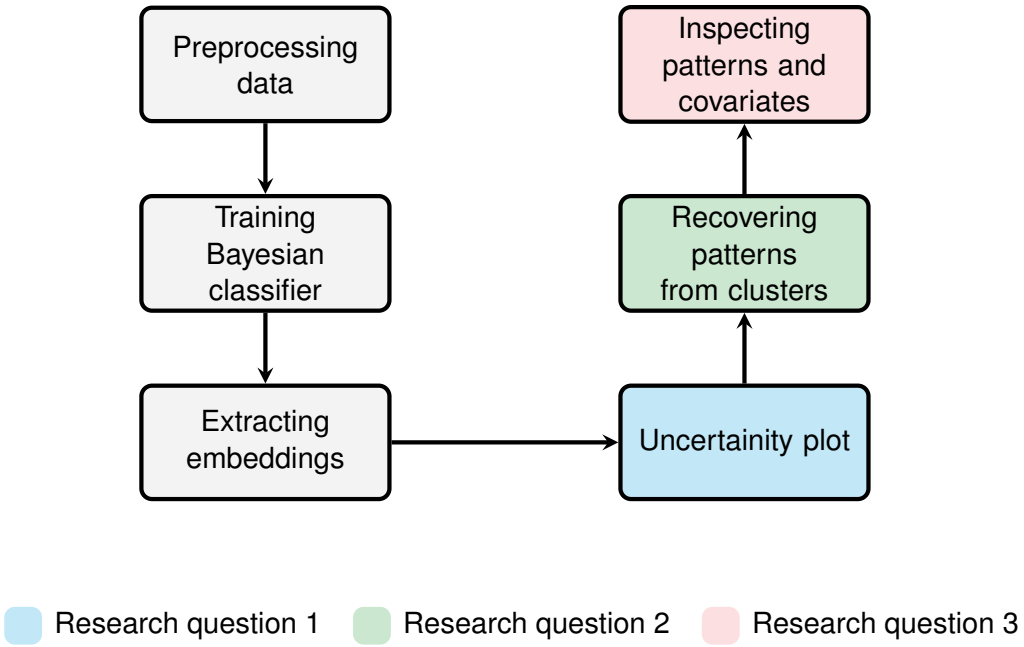


Fig. 1: Flow chart that breaks down the steps to answer the different research questions.

2. Can one recover money laundering patterns in the networks of these clustered nodes?

3. What interpretations can be made from the eventually recovered patterns?

2. THEORY

2.1. Artificial Neural Networks

Artificial neural networks (ANNs) is a subcategory of supervised machine learning that seeks to replicate the ways in which human brains function. ANNs enable computers to learn and adapt through experiences in a manner akin to humans. Through *forward passes* and *backpropagation*, these methods aim to create accurate mappings between explanatory variables and response variables. The contexts in which they are applied are constantly growing in number and range from from picture- and voice recognition to video games and social media algorithms.

ANNs consist of layers of *neurons*, each of which is capable of receiving and transforming information. The neurons in a layer are connected to the neurons in any layer that may precede and/or succeed it. These connections all have a *weight* which dictates how much information is transferred between the neurons. These weights make up the bulk of learnable parameters of a simple ANN, and are trained through backpropagation, which will be explained further later in the section. In addition to a set of weights, each neuron (except for those in the input layer) is accompanied by a *bias*. This term is a constant that is not directly dependent on the input data, and allows for greater learning capacity and generalization.

An example of a simple ANN architecture is visualized in Figure 2. This network consists of an input layer X, a hidden layer H, and an output layer $\hat{y}$. The input layer consists of three nodes as this

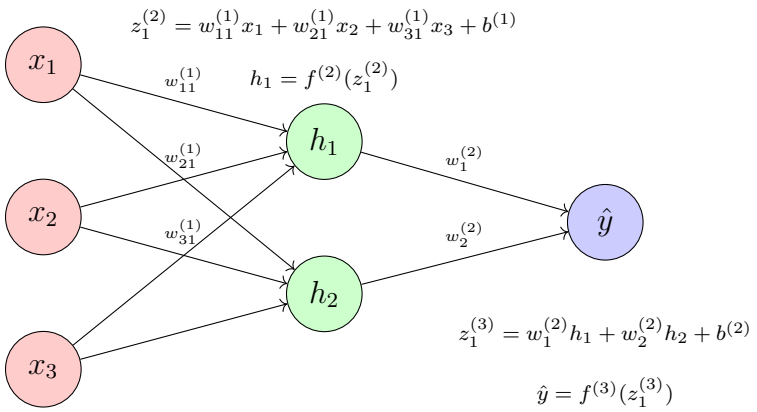


Fig. 2: An example of a simple ANN architecture and the calculations for a forward pass. Notations for the weights $w_{12}^{(1)}$, $w_{22}^{(1)}$ and $w_{32}^{(1)}$, as well as the calculations for h_2 have been omitted for readability.

is the dimension of the input. The hidden layer, whose name comes from its "hiding" between the layers before and after, has two nodes. The number of nodes in any hidden layer is an example of a hyper-parameter, i.e. a parameter whose value is chosen before the model is trained. A greater number of nodes typically corresponds to a greater learning capacity, but at the cost of more memory use and longer training times due to the increase in parameters. In the example, two nodes have been chosen for simplicity. The output layer in our example consists of only one node, as the network is predicting a single binary response variable. Between the layers are the weights $w_{ij}^{(k)}$ and biases $b^{(k)}$, where i and j denote the position of the giving and receiving nodes in their respective layers and k the set of connections.

The process that artificial neural networks go through to generate a prediction is known as a forward pass or forward propagation. In a forward pass, the input data is processed layer-by-layer until a final output is generated. To better conceptualize a forward propagation, consider the calculations shown in Figure 2. The first calculation done is that of the *net input*s in the hidden layer $z_j^{(2)}, j \in \{1, 2\}$, which is a sum of the input variables weighted by $w_{ij}^{(1)}, i \in \{1, 2, 3\}$, plus the bias $b^{(1)}$. An *activation function* $f^{(2)}$ is then applied to the net input. This maps the net input onto the output space defined by the particular activation function. Per the Universal Approximation Theorem, an ANN can approximate any function to an arbitrary degree of accuracy, given a sufficient amount of neurons and an appropriate activation function (Leshno et al., 1993). For the model to be able to learn complex, non-linear relationships, it is essential that the activation function is non-linear. The sigmoid function, tanh and ReLU are all commonly used non-linear activation functions. The same calculations are then done for $z_1^{(3)}$, taking the neuron *activations* h_1 and h_2 as input. The activation function for the output layer $f^{(3)}$ is chosen according to the specific problem being solved. The sigmoid function is commonly used in binary classification problems to yield the probability of a sample belonging to a class, while the softmax function is typically used in multi-class classification problems for the same purpose.

Artificial neural networks use backpropagation to update their weights during training. It is through updating its weights that an ANN learns the relationship between the explanatory variables and the response variable. The final essential component of an ANN is the *loss function*, and it is the centerpiece for the learning process of the network. A loss function quantifies how well the model performs on the data

set. More precisely, it measures how much predictions deviate from the actual responses. While the exact function can vary depending on the type of problem that is being solved, all ANNs have minimizing the value of the loss function as their goal. Backpropagation achieves this by iteratively adjusting the model's weights, propagating backwards through the layers of the network one layer at a time.

The example network is a binary classifier with a sigmoid activation function for $\hat{y}$. A common loss function used for such a model is Binary Cross Entropy (also known as logarithmic loss):

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p(\hat{y}_i)) + (1-y_i)\log(1-p(\hat{y}_i))\right] \tag{1}$$

Where y_i is the true label of the sample i, $p(\hat{y}_i)$ is the probability given by the sigmoid function, and N is the total amount of samples.

Using a simple first-order optimization technique such as gradient descent, one calculates the derivative of the loss with respect to the parameters of the network to update the parameters in the direction that minimizes the loss function (Pierrot et al., 2018). These calculations are first done with regard to the weights and biases associated with the final layer and cascade back to the input layer using the chain rule of derivation. Hence, one says the "error" from the output layer is propagated back through the network (Rumelhart et al., 1986).

When using a large number of layers, there are a number of challenges the backpropagation faces. The gradient that is propagated backward can gradually diminish as it passes through many layers. This problem is called *vanishing gradients*. It is caused by the gradients being products of a large number of factors, and when many of these are less than 1, it gradually vanishes. This problem is especially exacerbated by having activation functions whose derivatives cannot be greater than 1, such as the sigmoid function. Choosing activation functions such as ReLU, and using architectures with skip connections (He et al., 2015) are examples of ways to mitigate this risk.

Another problem that is common for ANNs almost regardless of architecture size is *overfitting*. Overfitting occurs when a model learns to predict well on training data, but does not generalize well to new unseen data. This is often the result of the model learning noise and random fluctuations in the training data, instead of the underlying patterns that are relevant to the problem at hand. Overfitting leads to sub-optimal performance in real-life use cases as the predictions are not as accurate on new data.

To reduce the risk of overfitting it is important to evaluate the performance of a model on data that is not available to it under training. A *train-test split* ensures this by splitting the data set into a training set that is used to train the model, and a test set that is used to evaluate its performance. This helps to ensure that the model performs well on unseen data and is more applicable in real-life situations. During hyperparameter tuning (i.e. training models with different hyperparameters to see which yields the best result), it is common to split the training data yet again into a training data set and a validation data set. In such a case, the training data is used for model training, the validation data serves as a proxy for the test set during hyperparameter tuning while the test set remains the final independent subset of the data that is used to evaluate how well the model generalizes to unseen data.

There are a multitude of other techniques that are used to prevent overfitting such as regularization and drop out. Regularization techniques add a penalty to the loss function which limits the complexity of the model and reduces its ability to overfit. Drop out on the other hand introduces a parameter whose value governs the probability of setting any neuron in the network to zero during training. This forces the network to learn more robust features that are independent of the presence of any particular set of neurons in the network, thereby making more consistent predictions and preventing overfitting.

2.2. Bayesian Statistics

Bayesian statistics is a branch of statistical analysis that emphasizes the role of probability in representing and updating uncertainty (Gelman et al., 2013). The approach is increasingly popular in fields such as medicine, finance and machine learning. Bayes' theorem is central to Bayesian statistics and represents a method for updating prior knowledge based on observed data:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

The Bayesian framework consists of three main components: the *prior* distribution, *likelihood* and the *posterior* distribution. The prior distribution represents the belief about a parameter or hypothesis before the observation of data, and is often based on historical data or expert knowledge. The likelihood is a measure of how likely the observed data is, conditioned on a particular parameter or hypothesis. The posterior distribution is the updated belief about the parameter of hypothesis after having considered the observed data, and is calculated using Bayes' theorem.

Bayesian statistics come with a set of advantages and disadvantages compared to frequentist statistics. The integration of prior knowledge into Bayesian models make them well-suited for use cases with limited data or where expert opinions are important. On the flip side, the choice of prior can introduce subjectivity that might lead to biases and disagreements among experts. Bayesian models also directly quantify the level of uncertainty, which makes them substantially easier to interpret than frequentist p-values. In addition, the Bayesian methods allow for sequential modeling where estimates are refined whenever new data is available. The Bayesian framework can also accommodate complex models such as hierarchical models. A significant drawback, however, is that the models can be computationally intensive, especially when they are complex and the denominator in Bayes' theorem becomes computationally intractable.

An example of a use case of Bayesian statistics is given in an example visualized in Figure 3. Suppose a group of scientists is studying the average height μ of a species of plant. Since the species of plant is scarce and hard to come by, the scientists compensate for the low number of plants they manage to sample by incorporating their expert knowledge into the model. This is done by using their assumptions regarding the height of the plant to construct a prior Gaussian distribution $p(\mu) = N(5, 4)$, which roughly quantifies their belief about μ and the uncertainty associated with that belief. After having observed some samples D of the species, they update their belief with this observed data. By using Bayes' theorem, the scientists get a posterior Gaussian distribution of $p(\mu|D) = N(7, 2)$. The new mean is a compromise between the prior mean and that of the observed data, and the reduced variance reflects the greater certainty around the belief. In contrast to frequentist statistics which focuses on the probability of the observed data given some hypothesis, Bayesian statistics incorporates prior knowledge and updates it, while giving a more comprehensive understanding of the uncertainty of the parameter.

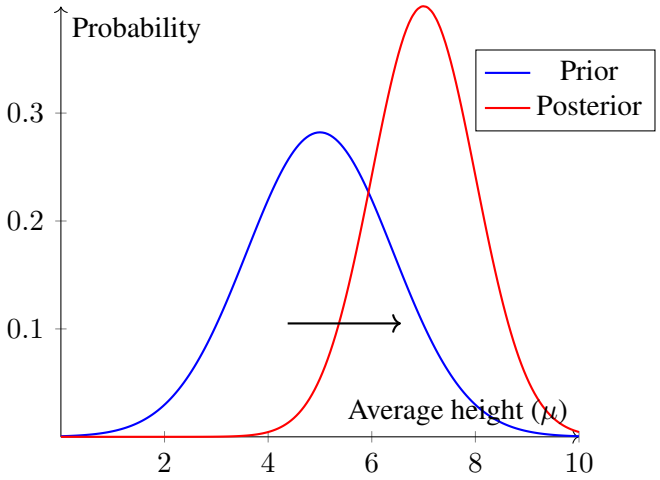


Fig. 3: Example showing how observed data updates a belief in Bayesian statistics.

In addition to doing away with traditional hypothesis testing, Bayesian inference differs from frequentist statistics in other ways. Traditional statisticians often construct confidence intervals that represent the percentage of times that the true parameter value would be expected to fall within the given interval if the same estimation procedure were repeated multiple times with different samples. The Bayesian counterpart, the *credibility interval*, is easier to interpret and is merely an interval where the true value of the parameter is believed to be within up to a percentage point of certainty. While the interval values can be of interest, the width of this credibility interval itself represents a viable measure of the uncertainty around a parameter.

2.3. Bayesian Neural Networks

Bayesian neural networks (BNNs) is a subcategory of machine learning that combines artificial neural networks and concepts from Bayesian statistics (Neal, 1995). Bayesian neural networks introduce uncertainty to their weights and biases, and represent these learnable parameters as probability distributions. This makes the networks more capable of handling uncertainty and noise in the data.

Building on the previously introduced example architecture, one can represent the weights in a typical Bayesian setting as Gaussian distributions, as visualized in Figure 4. Prior Gaussian distributions are defined for the weights $P(\mathbf{W})$, and after training, these distributions have shifted to minimize the loss function. Since the posterior distribution $P(\mathbf{W}|\mathbf{D})$ is a compromise between the prior and the data, the prior distribution effectively acts as a regularizer for the model.

Updating these parameters in accordance with the observed data is often done through *variational inference*. Since BNNs consist of a large number of parameters that often have distributions that make analytical and exact Bayesian updating infeasible, approximation becomes necessary. The process of variational inference for BNNs unfolds as follows:

1. One begins by choosing a family of approximating distributions $Q(\mathbf{W})$ that is flexible enough to approximate $P(\mathbf{W}|\mathbf{D})$, but not so complex that it does not allow for time-efficient computations.

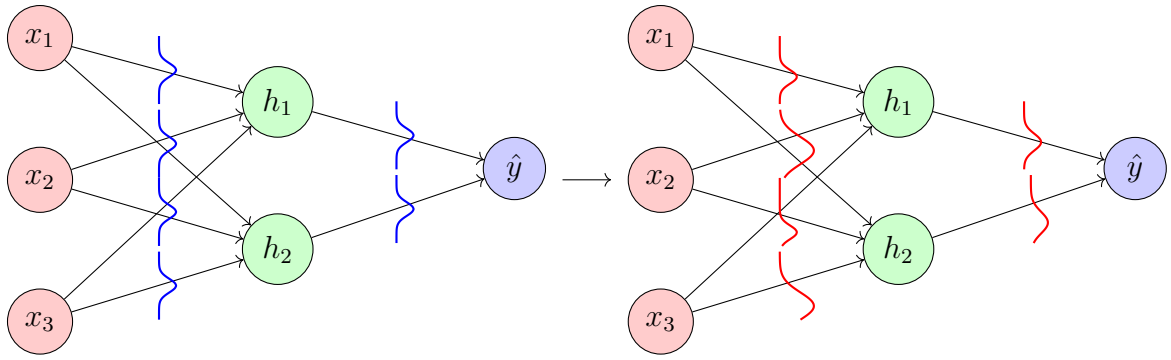


Fig. 4: Example showing the prior and posterior of learnable parameters in a Bayesian Neural network. The weights of x_2 are omitted for readability

This distribution is then parameterized by the variational parameters ϕ. This is denoted as $Q(\mathbf{W};\phi)$, meaning that ϕ are the parameters that define the shape of the approximating probability distribution of $\mathbf{W}$.

2. One then optimizes the loss function with respect to the variational parameters ϕ. The goal is to minimize the divergence between $Q(\mathbf{W};\phi)$ and $P(\mathbf{W}|\mathbf{D})$, as defined by the Kullback-Leibler divergence (KL). The KL divergence is a common way to measure the difference between two probability distributions and is in this case denoted as $KL(Q(\mathbf{W};\phi)||P(\mathbf{W}|\mathbf{D}))$. Since $P(\mathbf{W}|\mathbf{D})$ is the unknown distribution that VI is approximating, one has to reformulate the loss function. A series of mathematical derivations yields the objective function used in variational inference, namely the Evidence Lower Bound *ELBO*.

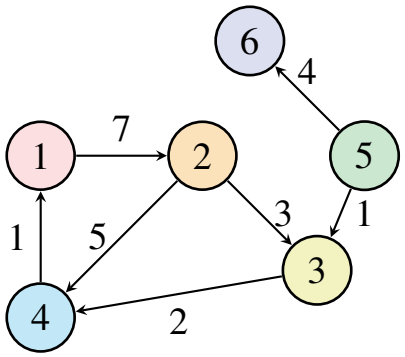
$$ELBO(\phi) = E_Q[logP(\mathbf{D}|\mathbf{W})] - KL(Q(\mathbf{W};\phi)||P(\mathbf{W})) \tag{3}$$

One seeks to maximize ELBO, as this is equivalent to minimizing $KL(Q(\mathbf{W};\phi)||P(\mathbf{W}|\mathbf{D}))$. The first term in ELBO is the data-driven term that represents how well $Q(\mathbf{W};\phi)$ fits the training data. The second term on the other hand measures the distance between $Q(\mathbf{W};\phi)$ and the prior $P(\mathbf{W})$, and acts as a regularizing term that mitigates the complexity of the approximating distribution. The mathematical derivations are outlined in the paper by Blei et al. (2017).

2.4. Graphs

A graph is a mathematical structure that represents pairwise relationships between objects. A graph consists of a set of nodes (also known as vertices) and edges that represent the relationships between them. Graphs can be used to model both simple and complex structures and their applications span a number of fields including data science, physics, and biology. In the context of AML, a graph can model large transaction networks that consider individuals and companies as nodes and transactions as edges.

Graphs can be instantiated in a multitude of ways; the adjacency matrix is the most relevant for the present work. The adjacency matrix $\mathbf{A}$ of a simple graph with node set $U = u_1, \ldots, u_n$ is a square $n \times n$ matrix. The element $\mathbf{A}_{ij}$ is equal to one when there is an edge from node u_i to node u_j, and zero when there is no edge between them. If the graph is weighted, i.e. every edge is associated with a quantity, $\mathbf{A}_{ij}$



	1	2	3	4	5	6
1	0	7	0	0	0	0
2	0	0	3	5	0	0
3	0	0	0	2	0	0
4	1	0	0	0	0	0
5	0	0	1	0	0	4
6	0	0	0	0	0	0

Fig. 5: Example of a directed and weighted graph.

will equal the weight of the edge w_{ij}. If the graph is directed, i.e. the edges have a specific direction, the matrix is not necessarily symmetric, as the existence of an edge at $\mathbf{A}_{ij}$ does not entail the existence of an edge at $\mathbf{A}_{ji}$. That is the case, however, when the graph is undirected. Undirected graphs are therefore always symmetric. Figure 5 visualizes an example of a graph being represented by an adjacency matrix.

2.5. Graph Neural Networks

Graph neural networks (GNNs) are a type of artifical neural networks that are designed to handle data that is represented as graphs. Prior to their introduction, data that naturally would be represented as graphs had to be altered to be used by traditional artifical neural networks, losing valuable information in the process. GNNs are able to make predictions that also incorporate the structural information of the graphs, making them well-suited for node classification, edge predictions and graph classification.

Message passing is central to the ability of GNNs to utilize structural information. Although different implementations of GNNs might differ in the specifics, the general process aims to aggregate information from neighboring nodes in order to update the representation of each node in the graph. Every individual node in the graph has a vector representation called a state vector. In the case of AML, this node could be a transacting party that could be represented by a state vector containing features such as sex, age and financial history. The message passing algorithm facilitates the learning of rich and context-aware representations of each node through an iterative process usually consisting of three steps:

1. **Message function** The message function calculates a message for every edge in the graph, and takes the state vectors of all the nodes into consideration. The goal of the function is to transform and combine the information from neighbouring nodes to then be aggregated in the next step of message passing. The message function can be as similar as a linear transformation and as complex as a neural network.

    - Linear transformation: The message is calculated as the product of the state vectors of neighbouring nodes and a weight matrix, i.e. $M(u, v) = \mathbf{W} * h_v$ where u and v are neighbouring nodes, $\mathbf{W}$ the weight matrix, and h_v the state vector of node v.
    - ANNs: A fully connected layer can be used to calculate the message and allows for more complex interactions between state vectors. In this case, $M(u, v) = ANN(h_u, h_v)$, where

the function takes the vector states of the two neighbouring nodes as input. This is one of the methods that were experimented with in Hamilton et al. (2017) in which the GNN approach GraphSAGE was introduced.

2. **Aggregating** After computing the messages for all edges, the next step is to aggregate the messages for each node. The aggregation function combines the incoming messages for each node. The options for aggregation function are many and include:

    - Averaging: The aggregated message is calculated as the mean of all incoming messages for a node, i.e. $A(v) = \frac{1}{|N_v|} * \sum_{u \in N_v} M(u, v)$, where N_v is the neighborhood of node v. This approach is used in Graph Convolutional Networks (GCN) introduced by Kipf and Welling (2016).
    - Pooling: The aggregated message is calculated as the element-wise maximum of all incoming messages for a node, $A(v) = max(M(u, v))$.
    - Attention mechanisms: Attention mechanisms can be use to weight the importance of different incoming messages, allowing the model to focus on specific parts of the graph. The aggregated message is calculated as $A(v) = \sum_{u \in N_v} \alpha(u, v) * M(u, v)$, where $\alpha(u, v)$ is the attention weight for the message from node u to v. This concept is the basis of Graph Attention Networks (GATs) proposed by Veličković et al. (2018).

3. **Updating** When the aggregated messages have been calculated for each node, the update function produces an updated representation (embedding) for each node by combining the original state vector of the node with the aggregated messages. This updated embedding capture both the original node features and the contextual information from its neighborhood. Different options for updating function includes:

    - Concatenation followed by fully connected layer: The updated node representation is calculated as $H(v) = ANN([h_v, A(v)])$, where $[h_v, A(v)]$ is the concatenation of the state vector and the aggregated message.
    - Element-wise operations: Operations such as addition or multiplication can be used to combine the original state vector with the aggregated messages. i.e. $H(v) = h_v + A(v)$ or $H(v) = h_v * A(v)$.

It should be noted that not all papers break the message-passing operation into these steps. In many papers, two if not all of the three steps are condensed into the aggregation step.

The training of GNNs is associated with a number of challenges, the most pressing of which is arguably oversmoothing. Oversmoothing is a phenomenon where node embeddings in a graph become increasingly similar to each other as the depth of a GNN increases. This is due to the node representations getting increasingly similar to those of their neighbors every iteration of message passing. This can lead to the loss of discriminating information and reduced performance. Reducing the number of layers and thereby reducing the iterations of message passing is the most common approach to mitigating the problem. Residual connections are also commonly used as a tool to allow for greater network depth (Kipf and Welling, 2016).

2.6. Graph Bayesian Neural Networks

As the name suggests, Graph Bayesian Neural Networks (GBNNs), are a subgroup of ANNs that combine the structural properties of GNNs with the Bayesian framework for handling uncertainty. Various motivations exist for combining these frameworks, but a prevalent one involves managing the uncertainty inherent in underlying graph structures by treating the graph as a stochastic variable (Shi et al., 2021). In this work, however, the interest is to be able to incorporate graph structure and uncertainty handling for the sake of more information-dense embeddings and more accurate predictions.

The implementation used in this thesis is the technique introduced in the 2020 paper "Bayesian Graph Neural Networks with Adaptive Connection Sampling" by Hasanzadeh et al. (2020). The paper introduces a comprehensive framework that allows for adaptive connection sampling in graph neural networks that generalizes current stochastic regularization techniques such as DropOut and DropEdge. The technique that they name Graph DropConnect (GDC) allows for the sampling rates to be learned jointly with other parameter models, rather than having to be pre-defined or tuned as hyperparameters. In addition to being robust against overfitting and over-smoothing, GDC is also mathematically equivalent to an approximation for Bayesian GNNs. In the current context, the model allows for training that utilizes the graph structure of the AML data while also providing approximate Bayesian uncertainty quantification.

The model implements 'Drop connection' through a binary mask on which there is imposed a Beta-Bernoulli prior. The element at any position of the binary mask matrix is sampled from a Bernoulli distribution, whose success rate is characterized by a beta distribution, simplified as a Kumaraswamy distribution. The distribution for the success rate is learned jointly with the weight parameters. Since the drop mask is binary, traditional reparameterization is not applicable as the expectation in the first term of the KL divergence cannot be computed. The solution of the paper, which is also applied in this current work, involves directly optimizing the discrete variables through Augment-REINFORCE-Merge (Yin and Zhou, 2019).

3. AML DATA SET

This section will give an overview of the data that is used. Not only does the present work utilize the same data set from DNB that Jullum et al. (2020) uses, but it also builds upon the modeled features that were introduced in the paper. Parts of this section will therefore be dedicated to describing the modeling steps they undertook as well as introducing some of the terminology that this thesis borrows from.

The data set is naturally shaped by DNB's procedures of monitoring, investigating and reporting transactions. Per Jullum et al. (2020), the process at DNB and most Norwegian financial institutions consists of three stages: the alert, case, and reporting stages. All transactions go through the alert stage, where criteria-based monitoring is utilized to alert of suspicious transactions. The transactions that do not get alerted are denoted as belonging to category A in the paper and represent the majority of transactions. In the case stage, a simplified manual process then excludes any alerts that are deemed legitimate, these make up group B. The rest of the alerts are then grouped into cases centering around a main suspect party and a set of possibly involved parties. In the reporting stage, domain experts subsequently do a more thorough investigation that concludes whether a case is to be reported to authorities or not. These transactions are denoted as C and D, respectively. Whether the authorities take juridic action against the reported parties

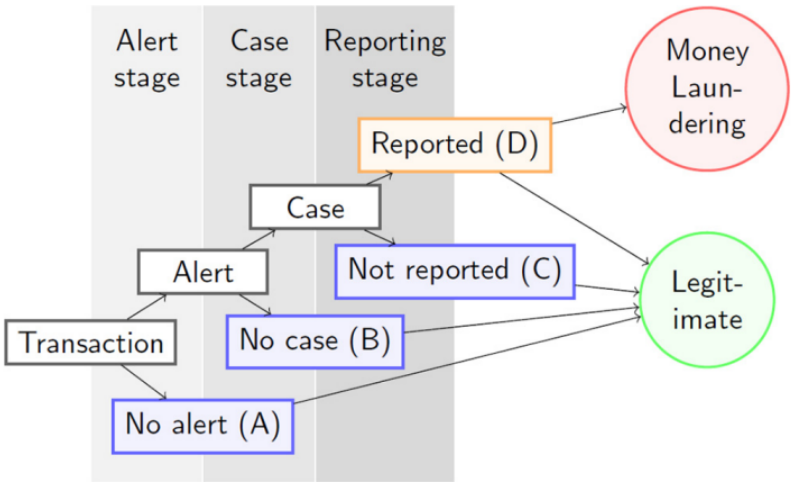


Fig. 6: DNB Monitoring procedure. Reprinted from Jullum et al. (2020)

is not important for DNB, as the role of the financial institution is merely to report suspicious behavior. All stages take information about the transactions and involved parties into consideration, but the case and reporting stages in particular emphasize the use of the background and transactional history of the parties in the decision-making process.

The data set provided by DNB consists of transactions between 1. April 2014 and 31. December 2016, and include transactions belonging to all of the groups A-D. This data set is also accompanied by background variables of the involved parties, as well as their transaction history two months back in time. Although this work differs from that of Jullum et al. (2020) in that it aims to do node classification on the transacting parties instead of classifying the transactions themselves, the objective of both of the classifiers is ultimately the same as that of the bank, namely to distinguish group D entities from the rest. Their work does investigate various combinations of groups A, B, C, and D as explanatory and response variables. However, they find that the changes in performance among these different setups are "far from significant". The present work does not emphasize possible performance differences between configurations and solely focuses on an A-C vs D binary classification setup. The data set includes 31789 A-C parties and 506 D parties, a 1.57%-98.43% class balance.

Jullum et al. (2020) does an extensive job of processing this data set, and the modeled features form the basis of the node features used in this work. Most noticeably, the authors take steps to reduce the dependency between modeled transactions by only including one transaction related to a case, and setting the minimum time lag between transactions coming to or from a party to two days. The modeled features which this thesis has access to are grouped into three categories. The first 19 features consist of background information regarding features such as sex, nationality and customer relationship. The second group of 538 features describes the transactional history of parties for the last two months, summarizing the maximum and total amount transacted across different currencies and transaction types. Whereas the last 18 explanatory features summarize the outcome of alerts and cases for every party and are called responses.

These features are structured into different two-month chunks. In a chunk, the transaction history features

and base features are modeled from the behavior of the party over a two-month period, while the corresponding responses in the chunk are from the subsequent two-month period. The decision to have the intervals consist of two months in particular was made in cooperation with AML domain experts. Also, while both a bank account and an individual or a company can be considered a transacting party, the data is structured around individuals and companies which are identified by their "PartyIDs". These parties represent the nodes in the network.

To also take into account the network information of the data, the present work expands upon this modeled data set by modeling edges that connect the nodes. This makes use of the transaction data set to create simple edges that denote the direction of the transactions. Since the modeled transactional history already summarizes the currencies, amount of transactions and size of transactions of the nodes, the role of the edges is merely to denote which parties that are transacting with each other and the direction of these transactions. There are therefore no features or weights associated with the edges.

The data was housed on a remote server at the Norwegian Computing Center. The primary tool for interfacing with this data was PyCharm, an Integrated Development Environment (IDE) specifically designed for Python development. Secure access to the remote server was achieved through the use of PyCharm's integrated Secure Shell (SSH) protocols, which ensured encrypted and secure data communication.

4. METHOD

4.1. Data Preprocessing

As this is an applied thesis, data preprocessing makes up a considerable share of the work. The preprocessing takes the already modelled features described in Section 3 and the transactions data set from the same time frame as input. One then develops data that can be used for later steps in the process, hereunder the training of the machine learning model as well as the recovery of the networks. The modeled features are in the rest of the subsection referred to as features, and the transactions data set as edges. The following seeks to give a simplified overview of the process and begins by breaking it into its constituent parts, before describing how these make up the overall process.

- **Cleaning** The data cleaning step aims to rid the data of any inconsistencies and errors so that they do not affect other processes downstream the data processing pipeline and lead to unreliable results. The missing data was visualized used the missingno Python library. Any entry in edges or features that were missing a "PartyID" were removed, while any column that showed a considerable share of missing values (more than 20%) was also removed. There were also steps to make sure that any eventual duplicate of entities in features and edges was removed, only retaining the last instance.

- **intersect** This function is implemented to make sure that only PartyIDs that are present in *both* the edges and features are included. This ensures that the rest of the data processing can work as intended and that there are no nodes included whose edges are missing and vice versa.

- **Encode** Every PartyID is encoded in this step. The function works by creating a mapping from the PartyIDs to the indexes of features. This function both returns the mapping, as well as an encoded edge data set upon which the mapping has been applied. This is a necessary prerequisite step for creating an adjacency matrix. It also serves the purpose of anonymizing the data.

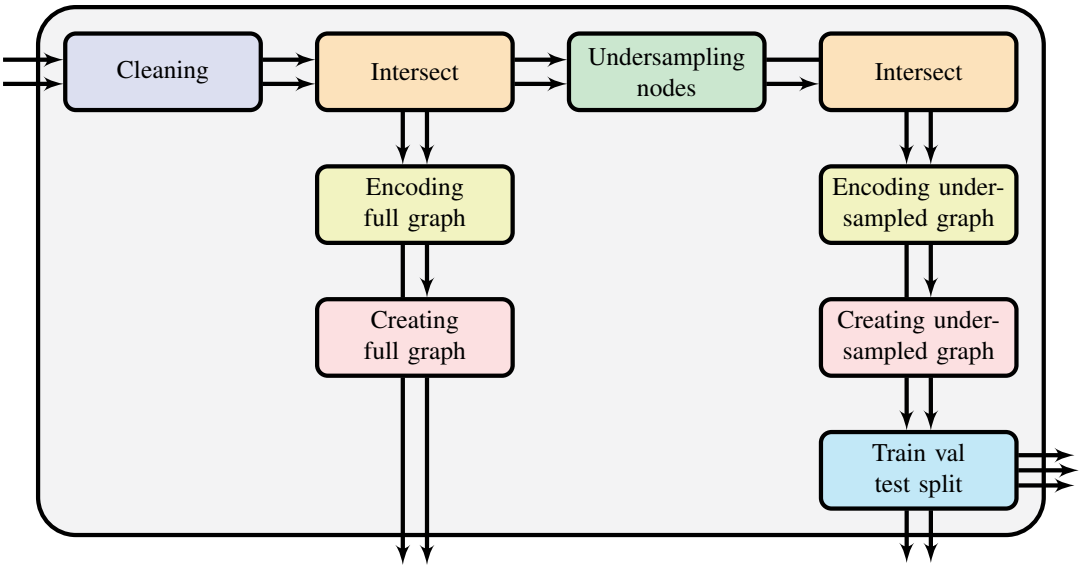


Fig. 7: Data preprocessing flow chart. Beginning at the upper left corner, the lower arrow represents the edges data set, and the upper arrow the features data set.

- **Create graph** This function takes the encoded edges and return a sparse adjacency matrix that takes direction into consideration.
- **Undersampling** For the model to be able to learn on the data set, the class imbalanced has to be reduced. This is done by undersampling the majority class (non-fraudulent nodes). The function allows the user to specify the ratio between majority and minority class nodes. A 50-50 split given by a ratio of 1 was shown to yield the best results, this is discussed more in the Results Section.
- **Train-validation-test split** There were multiple different ways of splitting considered, however, random splitting stratified with respect to targets showed the most promise and is the one used for the final results. The function returns the indices of these subsets of the data set.

As visualized in Figure 7, the data preprocessing naturally begins by cleaning the data, after which the intersection function is applied. At this point the flow chart diverges into two. Following the shortest path, the entire data set is used to create an adjacency matrix. This adjacency matrix, along with the PartyID-index mapping and full and un-normalized features data set may then be used in the recovering networks part of the process. This is so that one have the option of using the full graph before data is lost by undersampling when we are recovering the networks. The longer path begins by undersampling, after which the edges are encoded and the adjecency matrix is made. The PartyID-index mapping is also preserved for the purposes of recovering the networks. The undersampled features data set is then used to create a train-validation-test split, after which all required inputs for running the model have been made.

4.2. Training Bayesian GNN

For the present work, there were a number of modifications made to the model introduced by Hasanzadeh et al. (2020). First and foremost the hidden layer activation functions were all changed from ReLU to the hyperbolic tangent function (tanh). This ensured that there were fewer extreme activation values, as well

as fewer zero activations. This made the PCA much more robust and ensured greater separability of the node embeddings.

In addition, the prediction function was changed from being based upon majority voting to average voting, meaning that the softmax values of the sampled forward passes (which collapse to a sigmoid probability in this binary case) are averaged before assigning a class to the node by some threshold. As opposed to assigning a class to every sampled forward pass and then assigning the most frequent class to the node. The chosen approach is akin to taking the expected value of the predictive posterior distribution and is more in line with the Bayesian framework, in addition to yielding better results.

$$\hat{p}_i = \frac{1}{N}\sum_{j=1}^{N} p(\hat{y}_{ij}), \qquad \text{Class}_i = \begin{cases} 1 & \text{if } \hat{p}_i \geq T^* \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The equation above shows the implemented way of assigning classes, where $p(\hat{y}_{ij})$ is the j-th sampled probability for prediction i, and N is the number of samples from the predictive distribution. T^* is the class threshold.

4.3. Embeddings extraction and processing

The embeddings are extracted by using the *EmbeddingExtractor* class as shown in Listing 1 in the Appendix. The class takes a model and layers of interest as input and registers a forward hook that saves the embeddings at any specified layer. More specifically, it saves the pre-activation value of each neuron in every specified layer during every forward pass. One can then apply the activation function to these values, or choose to have the nodes represented by pre-activation values; both have been attempted for the uncertainty plot in this work. Since a Bayesian model is used as input, the embeddings at any layer are characterized by a distribution rather than a fixed value. We, therefore, sample the embeddings from the forward passes and use the mean of their posterior distributions as point-estimate representations of the embeddings. These are then subject to dimension reduction through Principal Component Analysis in order to be visualized in two-dimensional space (Jolliffe, 2002).

The distribution of the activation values of the output layer is of particular interest. It is this posterior predictive distribution, consisting of the values from the soft-max activation function that is used to quantify the uncertainty associated with the node predictions. A credibility interval is defined over the sampled distributions, and measuring the distance form the upper bound and the lower bound gives a single measure that sufficiently encapsulates the uncertainty. The mean of the distribution, on the other hand, constitutes the point estimate for the soft-max probability of the prediction. In addition to being used for class assignments, the probability itself is another quantity of interest that is used in the visualizations.

4.4. Uncertainity plots

The uncertainity plots are implemented using the Python library Plotly. This library is superior to others in this context as it allows the user to dynamically interact with the plots through drag selection and tooltips. The plots are very information dense and incorporate all of the quantities discussed in the previous subsection. The axes represent the two PCA components, The shape and size of the scatters correspond to their assigned class and probability respectively, while the widths of the credibility intervals (the uncertainty)

of the predictions are encoded as a color spectrum. The Plotly implementation displays the index of a node when hovered over, which makes it so that clusters and nodes of interest can be noted and analyzed further. Using the visualizations, the work also focuses on making clusters that are sensible and attempts to use algorithmic clustering to do this.

4.5. Network retrieval

The networks of the nodes of interest are retrieved through the *get_n_layer* function, shown in Listing 2 in the Appendix. This function takes the adjacency matrix and the indexes of interest, as well as the number of neighboring layers to include as arguments. It then returns the indices of the input nodes and their neighboring nodes. This set of nodes can be expanded upon further by another layer of neighboring nodes, depending on the number of layers specified in the argument. In this implementation, analysis has been done on sets including only the original nodes and the first layer of neighboring nodes. The original nodes have a thick black outline in the visualizations, while the neighboring nodes lack this. The labels are denoted by shape, a square node has been reported to the National Authority for Investigation and Prosecution of Economic and Environmental Crime, while a round node has not.

4.6. Visualizing covariate distributions

The embedding of every node encapsulates a combination of the information of its network and of its features. To explore the features of the clusters a grid of plots showing the feature distributions of different clusters have been used. In order to compare different clusters, the covariate distributions of two clusters are displayed at the same time. A total of 15 features are visualised and they are divided into the following groups:

- The left column of the grid consists of "top 5" plots. These are barplot visualizations and are the visualizations that have required the most feature processing. One can divide them into two types. The first two plots are modeled from the transaction features and show the most common currencies and transaction codes as measured by transaction volume. These are visualized on a log scale as this makes the smaller bars readable. The last three plots are processed from one-hot encoded columns and display the five most frequent nationalities, sectors (sektorkode) and industries (neringskode) in the clusters. To take different cluster sizes into account, these values are divided by the size of the clusters. The relative sizes of the bars are therefore what is emphasized and the reason that the x-ticks labels are not visualized. The plots display the union of the top 5 of both clusters, and may therefore include at most 10 unique categories.

- A plurality of the visualizations are violin plots that illustrate the distribution of features chosen based on maximizing the dissimilarity between the two clusters. The dissimilarity is measured through The Mann-Whitney U test under the null hypothesis that the distributions of both populations are identical. The test works on non-parametric distributions and gives a measurement of dissimilarity inverse to the p-value it returns. To reduce redundancy, features that were already to be plotted were excluded from the test. One hot encoded features were also removed to make the violin plot more sensible. Transactional features were processed to remove the distinction between transactions in and out of the account, thereby reducing redundant information. The 10 most dissimilar features were plotted and have their p-values denoted in parenthesis in the plot.

5. RESULTS SECTION

In this section the results of the methods will be presented, including the results from training the model, the uncertainty plots as well as the retrieved networks and covariate distributions.

5.1. Model training

A lot of experimenting was done to train a suitable classifier, both in regard to the preprocessing of the data and the model specifications. In all cases, the training consisted of 1700 epochs, and a validation data set was used to determine at which epoch the parameters were best tuned for generalized performance. Training curves plotting the training AUC and validation AUC against the epochs were saved for the models. The training curves oscillate significantly, however, this is believed to be due to the relatively few samples used when calculating the AUCs for every epoch. This has been treated as an acceptable trade-off for model training speed, especially considering the amount of models that were experimented with. This oscillation did not decrease significantly by reducing the learning rate or increasing the Gaussian prior regularization either. The training curves are shown in the Appendix.

The selected model proved to have difficulties learning on imbalanced data sets. This limited the total number of nodes that could be used for training, as undersampling was used to reduce the class imbalance. The final model was trained on data consisting of 389 D nodes and 389 nodes randomly sampled from the remaining classes A-C, a 50%-50% class balance. The difficulty associated with learning on more skewed data sets might have come from the relatively few D nodes available, even more so than in Jullum et al. (2020), as the presence of edges imposed a limitation as to which nodes could be included, as any PartyID was required to be present both in the edges data set and the features data set. The selected model appeared to struggle with class imbalances, where the prevalence of a majority class resulted in biased predictions and an AUC score that did not exceed 0.5 significantly.

Model specification	**Class balance**	**Training AUC**	**Validation AUC**	**Test AUC**
Only post-undersampling nodes	50%-50%	0.631	0.724	0.590
Only post-undersampling nodes	71%-29%	0.515	0.621	0.493

Table 1: Evaluation of different models. Final model is written in italic.

Although the model appeared to be slightly overfitted to the validation set, inspecting the performances at other epochs still showed this epoch (1418) to have a relatively good performance overall. Since AUC is measured using all probability thresholds, a search for the threshold that yielded the best performance was undertaken. This was done by measuring the recall, precision and F1 of every data set for every threshold in an interval, as well as subjectively deciding how good the cluster separability is. Recall and precision are the ratios of correctly predicted positive observations to all actual class observations and total predicted positives, respectively. Some might be more familiar with these measurements as Positive Predictive Value and True Positive Rate respectively. The F1 score is a balance between the recall and precision, and its. The results are displayed in Table 2. 0.45 was determined to be the threshold that yielded the best results across the data sets, as it had a high F1 score while maintaining a more sensible precision. It also had a better cluster separability on the uncertainty plot than the rest.

Threshold	Data set	Recall	Precision	F1 Score
0.25	Training	1.00	0.50	0.67
	Validation	1.00	0.50	0.66
	Test	1.00	0.51	0.67
0.4	Training	0.99	0.51	0.67
	Validation	1.00	0.52	0.68
	Test	1.00	0.52	0.68
0.425	Training	0.97	0.53	0.69
	Validation	0.91	0.50	0.65
	Test	1.00	0.54	0.70
0.44	Training	0.96	0.55	0.70
	Validation	0.83	0.49	0.62
	Test	0.97	0.55	0.70
0.445	Training	0.95	0.56	0.70
	Validation	0.83	0.51	0.63
	Test	0.97	0.56	0.71
0.45	Training	0.94	0.57	0.70
	Validation	0.83	0.53	0.64
	Test	0.94	0.55	0.70
0.475	Training	0.70	0.61	0.65
	Validation	0.59	0.53	0.56
	Test	0.59	0.50	0.54
0.5	Training	0.42	0.74	0.54
	Validation	0.31	0.60	0.41
	Test	0.41	0.64	0.50
0.525	Training	0.25	0.84	0.38
	Validation	0.17	0.83	0.29
	Test	0.18	0.67	0.28
0.6	Training	0.03	1.00	0.05
	Validation	0.03	1.00	0.07
	Test	0.00	0.00	0.00

5.2. Uncertainty plot

A series of uncertainty plots were made using the predictions from the trained models and the extracted node embeddings. Uncertainty plot was first made with a distinction between training, validation and test nodes to inspect if the patterns are consistent. Something which they were confirmed to be. A greater uncertainty plot that does not take the distinction into regard is used when observing clusters and patterns for further inspection. These plots are available in the following Github repository. The colors indicate the normalized widths of the arbitrarily chosen 98% credibility intervals of the embeddings, where yellow

is uncertain and purple is certain. The principal components 1 and 2 explain 46.6% and 13.7% of the variance in the embeddings and are shown on the x-axis and y-axis respectively. The plot is shown in Figure 8.

At first glance, certain patterns are clearly noticeable. The upper left region contains the largest cluster, and it contains all predictions and uncertainties. A priori, one can deduct that the decision boundary likely is located close to this cluster, and that it will be difficult to do analysis of these clusters, especially in a real-life setting where the true labels are unknown. One can also see a clear cluster of mostly certain positive predictions in the bottom left corner. East of the clusters, on the other hand, a set of nodes form a pattern resembling an archipelago, consisting mostly of certain positive nodes, and some certain negatives. In the middle and upper regions, one can notice some islands of uncertain positive predictions that form small amalgamations or are separate on their own.

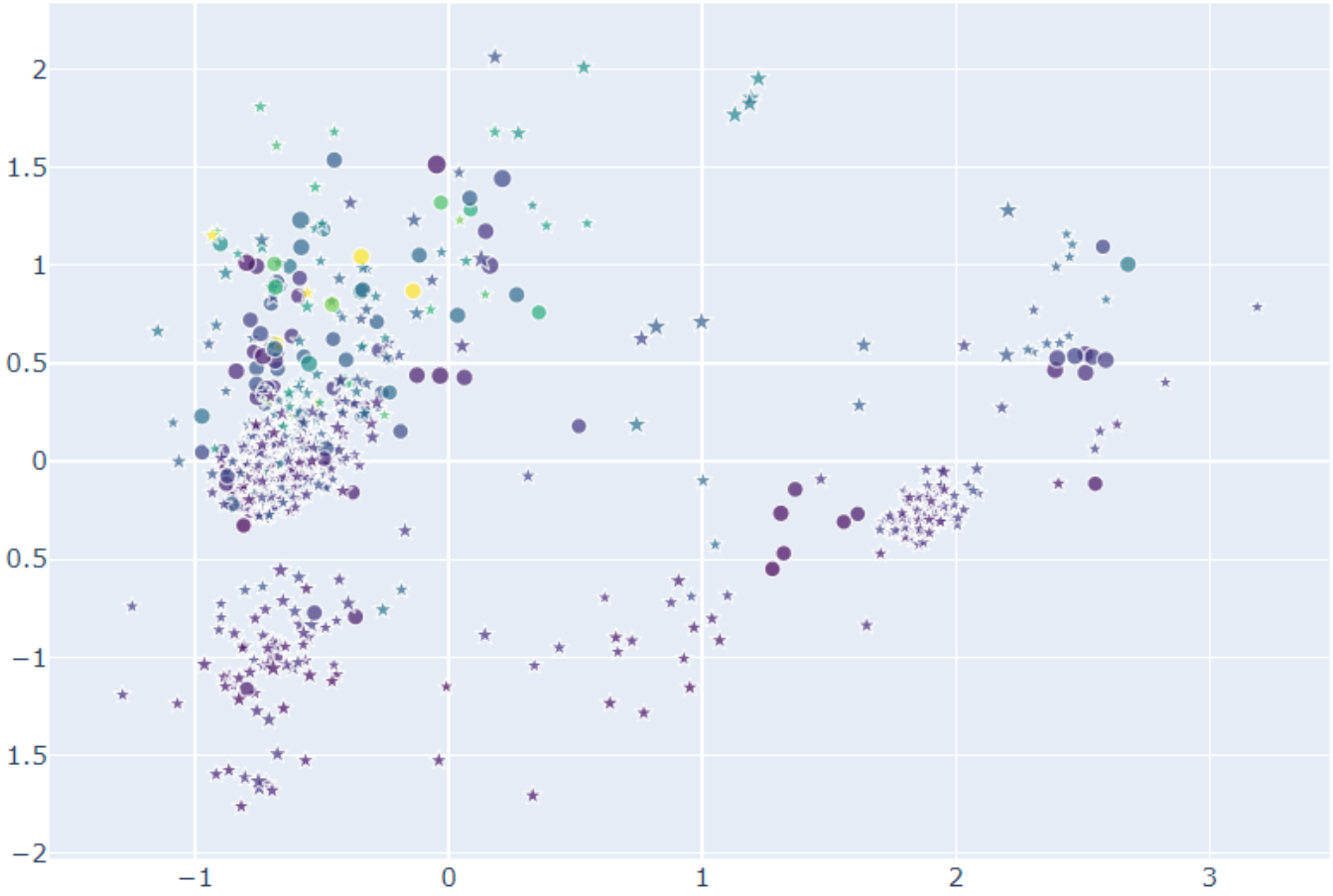


Fig. 8: Uncertainty plot for plot embeddings from BGNN with threshold=0.45.

Apart from this, discerning logical clusters using the naked eye is quite subjective. For instance, one might argue that the upper left mixed cluster could be partitioned into two; one certain and dense mixed cluster at the bottom and a more spacious uncertain one at the top. Partitioning the archipelago-shaped region on the right is especially subject to individual judgment. One could classify it as an entire cluster of its own, or into two positive clusters and a mixed one, or other configurations depending on how one weights different factors. It was therefore interesting to do some exploratory analysis into these clusters

using a simple k-means clustering algorithm to see if one could get sensible clusters. It might be that if the algorithm yields a clustering that performs well in regards to the easily discernible clusters, it can give more objective and consistently reasonable clustering on the regions that are hard to discern. One can measure sensible clustering in the context of using them to manually inspect networks by using the four following criteria:

- Subjective judgment of separability
- Subjective judgment of the number of clusters with regard to the resources spent manually investigating them
- Network connectivity
- Predictive performance on cluster

While the first two are self-explanatory and already the basis of discerning clusters when done with the naked eye, the last two are easier to objectively quantify. The network connectivity of a cluster can be measured in multiple ways, but this analysis will simply use the average degree connectivity of the nodes in each cluster. This is done by creating a subgraph consisting of the nodes in the cluster, and then measuring how many degrees each node has, and averaging it for the cluster. The last criterion is very important and can be calculated using F1. Although the overall F1 will not be affected by the clustering, it is desirable and not improbable at all that good clustering will lead to visually separable clusters having good F1 scores, while "pushing away" the poor F1 scores to more challenging clusters like that in the upper left region. Since predictive performance on the clusters is not available in an application setting, it is interesting to see how the F1 correlates to the other criteria that are available regardless.

It is also interesting to see if one can incorporate the non-spatial dimensions of the plot into the k-means clustering, and how that affects our criteria. This will be investigated by including the predicted probability and the uncertainty as dimensions of the nodes. Both of these measurements are normalized and then weighted. Different weightings are experimented with. For reference with regards to the weighting, Principal Component 1 has the shortest domain span of approximately 4. 4 would be therefore used as a weight if it is desired to consider a non-spatial dimension as approximate in importance to the spatial dimensions. To make such an analysis more feasible, the network connectivity and cluster predictive performance are aggregated on a clustering level (i.e. boiled down to one measurement each per clustering). The average degree connectivity was simply averaged yet again for each cluster, and the same macro averaging was done to the F1 scores of each cluster. The results are visualized in Figure 9, where the F1 measurement is to the left and the degree connectivity to the right. The rows represent different weightings with regard to the non-spatial dimensions. The weights are denoted in the plot titles in that order.

The plots indicate that the macro average F1 actually increases slightly as the number of clusters increases. It also shows that the average degree connectivity drops with the number of clusters, which is unsurprising as discerning clusters severs off edges while the number of nodes remains the same. There is an indication, however, that the inclusion of non-spatial dimensions in the clustering contributes to retaining the edges, and by extension the networks. This effect appears to be strongest when the probabilities are weighted a little less than the spatial dimension, while the uncertainty is weighted a third of that. Visual inspection of the clusters also indicates that the inclusion of non-spatial dimensions contributes to more intuitive clusterings. An example is the uncertain positive predictions in the middle that often are clustered

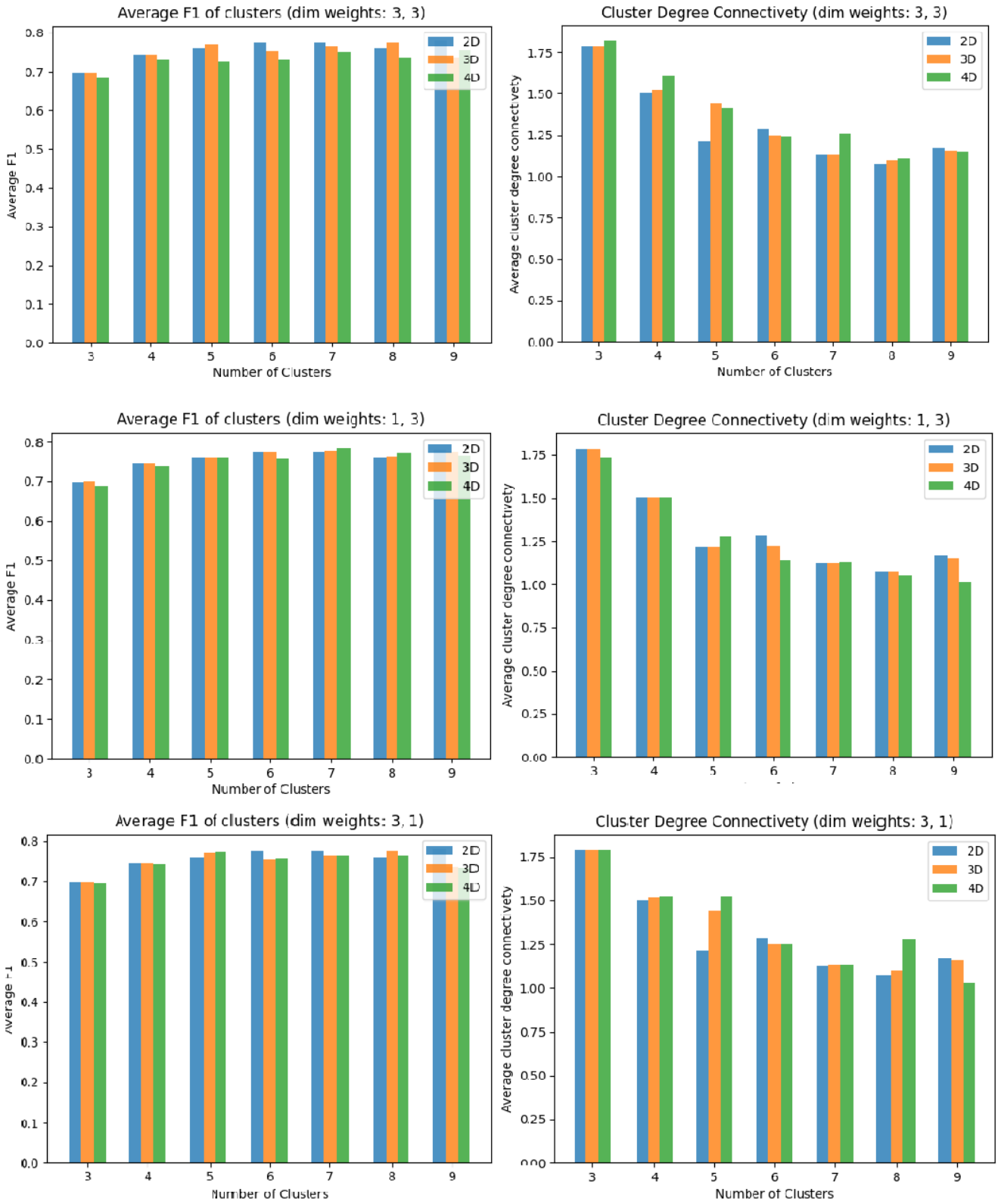


Fig. 9: Cluster measurements. The rows denote different weightings of the F1 and the degree connectivity.

Fig. 10

together, instead of being divided and shared by neighboring clusters. There is a correlation between F1 scores and network connectivity for each cluster of 0.156 (p-value 0.0038). This was, however, calculated by also including up to 17 clusters while excluding clusters 3-6 where the measures correlate negatively as a result of the number of clusters increasing. The correlation is not too surprising, however, considering that the model has a very high recall and low precision, while the negative class has 19% fewer edges due to the undersampling severing many of the connections in class. So while the degree of connectivity does serve the purpose of making inspecting the clusters more interesting, it would require more experimentation to determine whether it can generalize as a tool to select clusters that are predicted well.

One might also attempt inspecting another measure. Yet again, it is not possible to increase the predictive performance of the model, rather one is hoping that good clustering will yield clusters of high and low performance, which hopefully will lead to more interesting analysis. A possible way to measure this is by the spread of the F1 of the clusters. One could also see if the spread of network connectivity correlates with this. Both these spreads are measured by standard deviation and visualized in Figure 10 and 11. Interestingly, the inclusion of non-spatial dimension shows a significant increase in the spread of F1, while the opposite is true for the spread in degree connectivity. In fact, the correlation between the spread of F1 and degree connectivity is in this case -0.49 (p-value: 0.0148). It must, however, be noted that this is in a limited context to a limited data set where the different dimension choices are a significant confounding variable. However, this could suggest the possibility that one might choose to cluster with little spread in network connectivity for good separability of clusters in regards to predictive performance if labels are not available.

Taking the measures into consideration in combination with visually evaluating the clustering in the overview in Figure 12, I would argue that the 9 clusters that are also based on probability and uncertainty are the most sensible clusters. It partitions the archipelago structure to the right into 3 sensible clusters, two of which have good F1 scores, and the other great connectivity. The bottom left cluster is identified

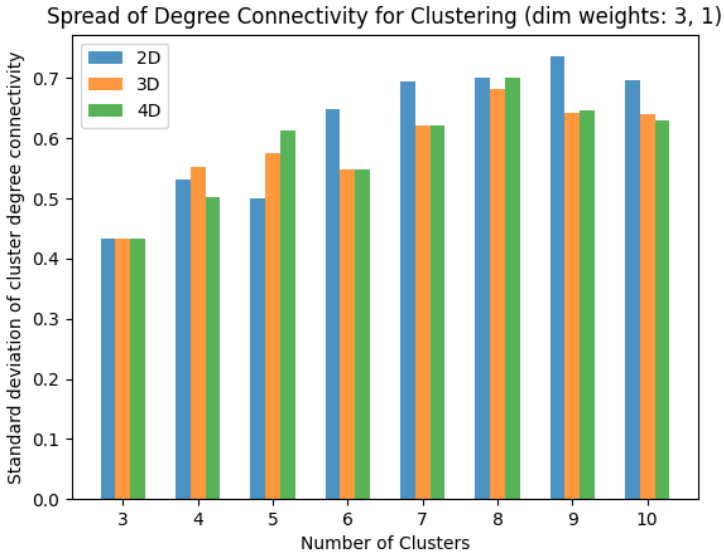


Fig. 11

as well, and so are the uncertain positive predictions in the upper middle part of the plot. It is at this value of k that these patterns that I described a priori stabilize. Increasing the value of k from here on leads to more numerous and less intuitive partitions of the upper left region, and increasing it beyond that the other sensible clusters become subject to the same. Although there is not much optimism around the upper left region, it has been divided into clusters that at least seem probable. It might be argued that the encroachment of the bottom-most of these clusters onto two nodes in the bottom left cluster, seem somewhat unnatural, but given that they are of another predicted class to the nodes around, it is possible that this is sensible. Nonetheless, this collection of clusters will be treated as one large cluster for the rest of the analysis, as the general inseparability of the region likely means analyzing them individually is time-consuming and of little interest.

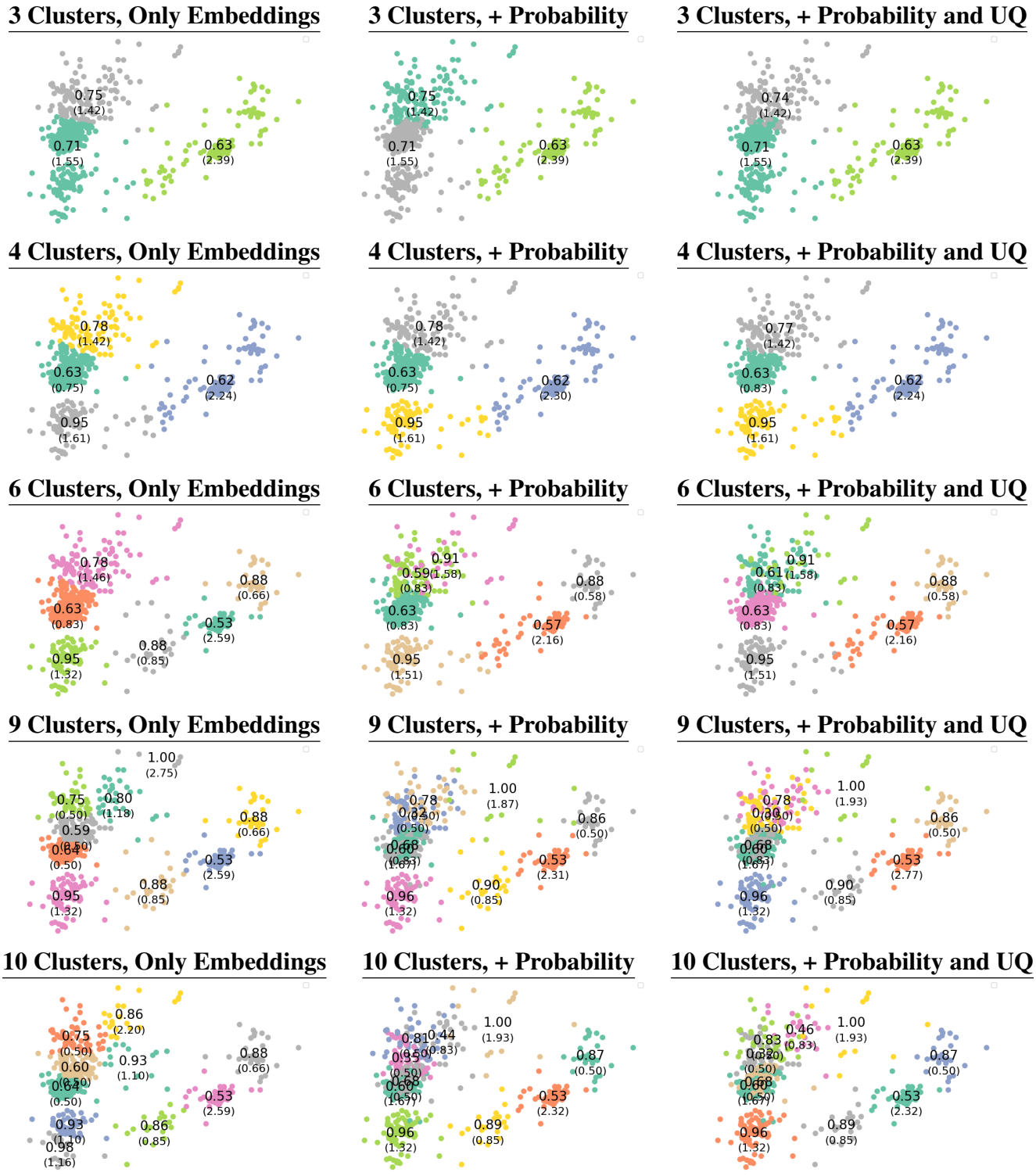


Fig. 12: Clustering overview. Probability and certainty are weighted by 3 and 1 respectively.

5.3. Recovered networks and covariate distributions

In the following, exploratory analysis will be done using the chosen clustering visualized in Figure 13. The clusters will be referred to by their color, with the exception of the pink, yellow, turquoise and (leftmost) gray clusters, which together form a cluster referred to as the upper left cluster.

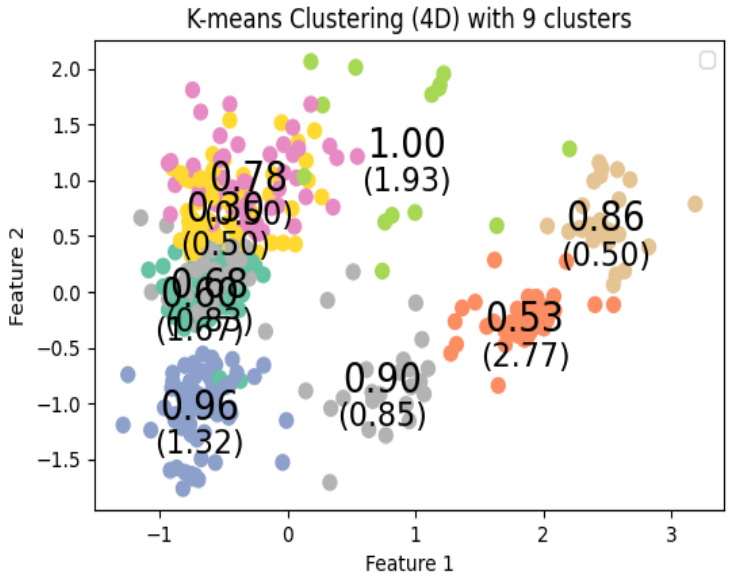


Fig. 13: The clustering of interest.

5.3.1. Orange and beige clusters

The orange and beige clusters are the two right-most clusters on the uncertainty plot and are two of three clusters that form the previously described archipelago structure. Comparing these clusters is of interest as they can be informative regarding the utility of the clustering, and can nonetheless offer insight into what makes these two clusters similar in general location, yet distinctly different. As far as network connectivity goes, the clusters are drastically different, with the orange region having the greatest degree connectivity of all the clusters, 2.77, and the beige having the least, 0.50. The inverse trend can be observed for the predictive performance of the model, with the orange cluster having an F1 as low as 0.53 and the beige 0.86. Their sizes are also drastically different, with the orange network consisting of 138 nodes and the beige of 27 nodes.

The networks of the nodes of the clusters and their first neighborhood are visualized in Figure 14. It is remarkable that both these clusters form a single coherent component, the orange consisting of 186 nodes, and the beige of 53. A commonality for both clusters is that many of their constituent nodes have node 487 in their first neighborhood. In fact, the orange cluster (left subplot), seems to be entirely centered around node 487, with almost all nodes being one or two edges away. This node is not part of any of the clusters themselves, yet is on the receiving end of transactions from most of the nodes. Examining its features gives a more clear picture of its role. Its industry code 64.190 means it is a bank, and is therefore naturally central in many transacting networks. In fact, the node has 145 incoming edges and 11

outgoing edges in the graph, 133 and 2 of which can be found in the orange cluster and its neighborhood respectively, and 24 and 1 in the beige network. Node 773 is also central to the clusters, especially to the beige. The node is a private limited company in an unknown industry. It has 17 incoming and 7 outgoing nodes in the entire network, 8 and 1 of which can be found in the extended network of the beige cluster. The beige neighborhood seems a little less centralized, with a smaller percentage of nodes connected to the central nodes, and a greater average shortest path between the peripheral and central nodes.

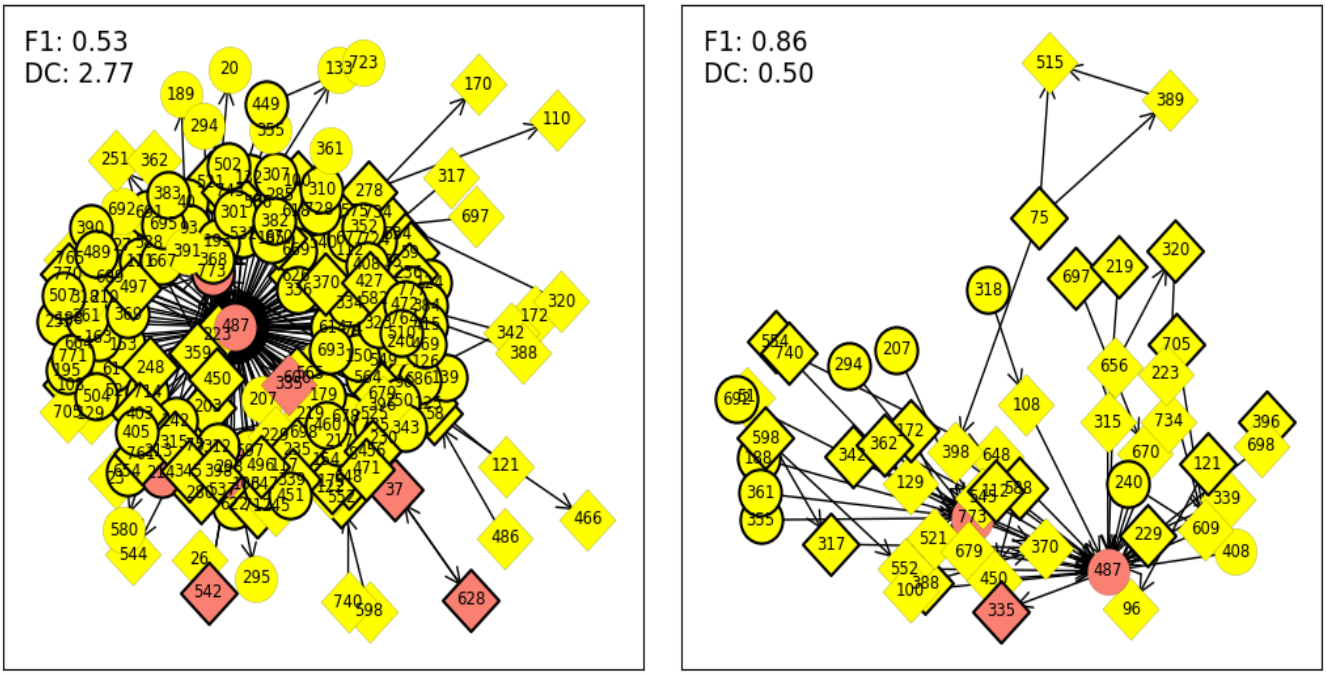


Fig. 14: The orange and beige clusters are visualized on the left and right respectively. Squares denote that the true label is fraudulent, while circle denotes non-fraud. Nodes lacking a black outline are part of the first neighborhood of the clusters and not the cluster themselves.

Both networks also have cycles (colored red) containing fraudulent nodes, which is a point of interest for the analysis. However, both of these cycles pass through the central nodes which decreases the likelihood of them indicating a structure holding any predictive power in a real-life use case, as circular structures involving a large bank are common. Both of the cycles are visualized in Figure 15. The edges are denoted by the latest chunk, i.e. the last two-month period in which the nodes shared this edge. These visualizations further undermine the possibility of these cycles holding any predictive power. For the cycle in the orange cluster, the timings make it hard to construct any narrative of any coordinated or irregular behavior. As for the cycle in the beige region, the timings confirm that the pattern is cyclical, however, the large time window between chunks 4 and 8, and the inclusion of the central nodes 773 and 487, make it rather insignificant. Node 335 is a corporation whose industry code is 08.990, which serves as a catch-all category for mining companies that do not fit any defined sub-categorizations. That may, or may not be of interest to a domain expert.

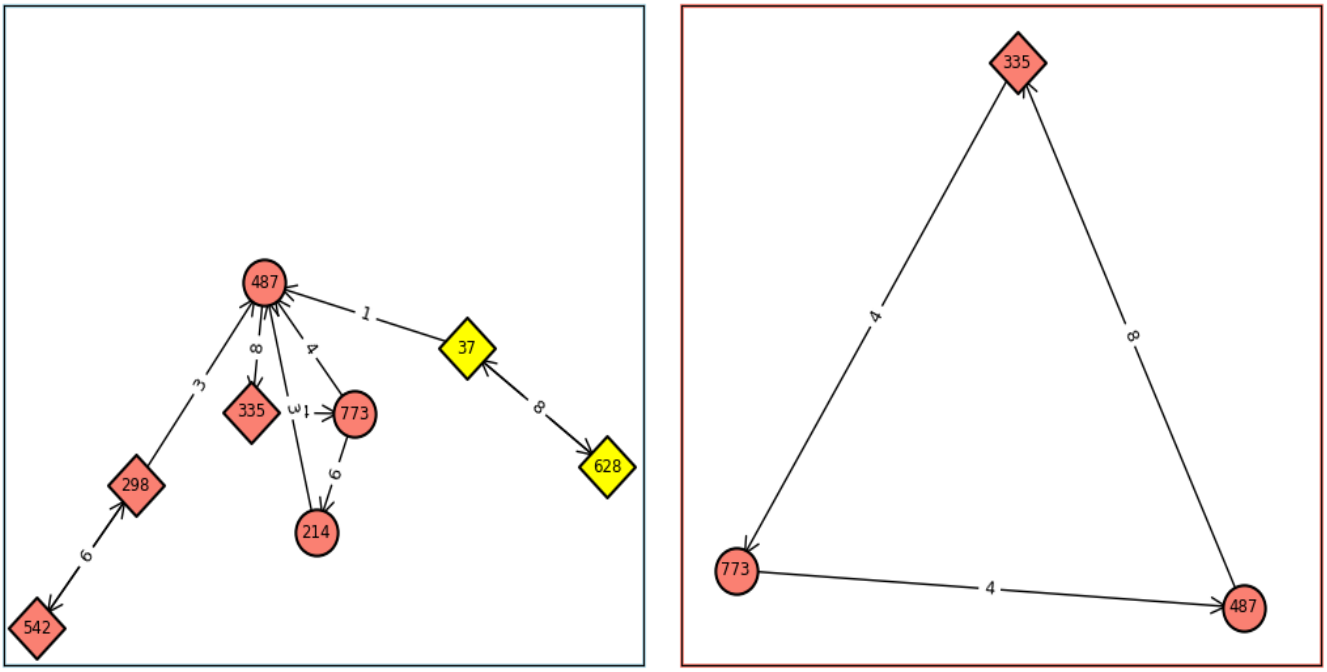


Fig. 15: The cycles in the orange and beige clusters are visualized on the left and right respectively. The edge notations denote the two-month period of the edge

In order to make the analysis more feasible and precise, nodes 487 and 733 were removed from the greater neighborhood. This significantly reduced the connectivity of the clusters and splintered them into small components, the largest of which consisted of five nodes in both clusters. The sizeable components, which we define as components consisting of four or more nodes, totaled to be 4 and 2 in the orange and beige networks respectively. Out of these, two were of particular interest. These are visualized in Figure 17 and 16. The pattern in the second figure shows three fraudulent nodes transacting to node 370, whose industry code only informs us is an international organization. The pattern becomes more interesting when one takes into consideration that it consists of nodes that all have been predicted to be fraudulent. The pattern with the triangle seems to have node 75 as its center, as it transacts to every other node. Yet again, the structure in and of itself is probably rather common, however, combined with the fact that all of these nodes were classified as fraudulent is of interest. In addition, node 515, which is on the receiving end of two trades within this network, is a Somali national, while 389 and 398 are catering and furniture businesses with nationality information lacking. In a setting without the nodes, one could reach both these networks in the same manner just demonstrated, and thereby find reasonable suspicion in the network given that all its nodes are predicted fraudulent and with unusual features. Another thing of note is that the nodes are found across both the red, beige, and green clusters, which can be seen as evidence that the chosen clustering is not ideal.

The covariates visualized in Figure 18 contribute further to understanding the similarities and distinctions between the two clusters. Once again the nodes 487 and 773 are removed as they are outliers among the rest of the nodes. The "Most Transacted Currencies" visualization (plotted on a logarithmic scale) shows that the nodes in the two clusters transact in mostly the same currencies: NOK, EUR, USD, CHF and SEK. However, they deviate in that the beige cluster has a substantial amount of transactions in CAD.

The transacted codes follow the same pattern, with mostly the same codes being transacted, with the

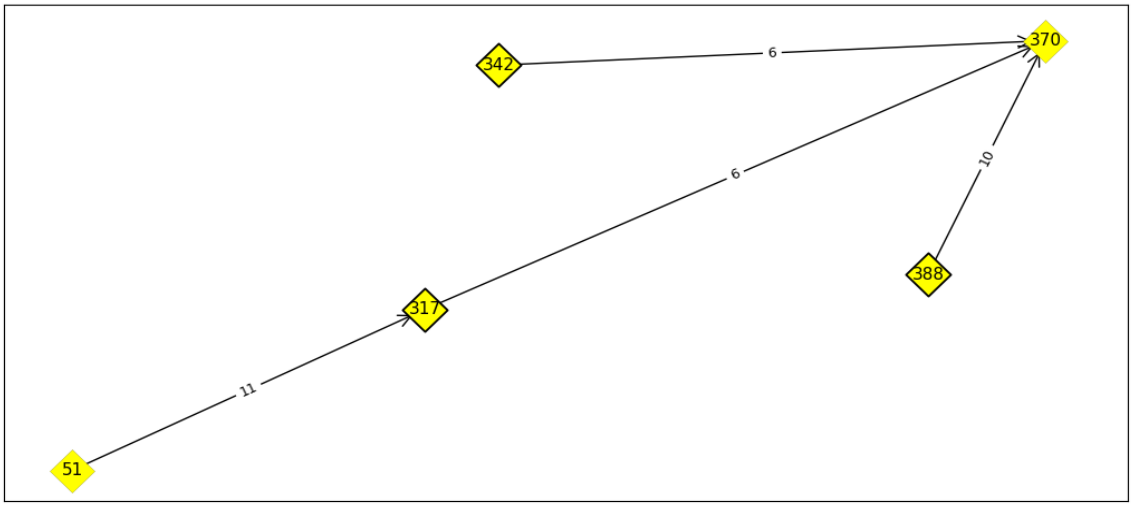


Fig. 16: Network of interest in orange cluster. The edge notations denotes the two-month period of the edge

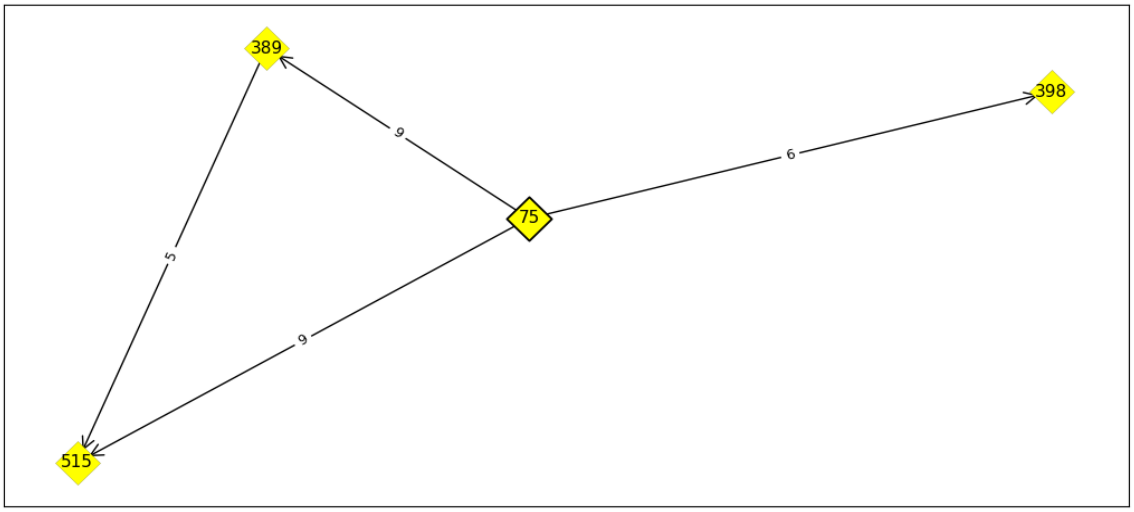


Fig. 17: Network of interest in orange cluster. The edge notations denotes the two-month period of the edge

exception of code 107 which seems to have been transacted exclusively by nodes in the orange cluster. There is also a very different mix of nationalities represented in each cluster beside the majority of Norwegian nodes. The five most frequent industries represented are also very different, the only commonality being the "Other business services not mentioned elsewhere" code 82.990. The most diverging covariate distributions per the Mann-Whitney U test (p-values are in the plot titles) inform us that QAR, CAD, and AUD transactions are only found in the beige cluster, with all of them only being found in a single node each, however, the last two being transacted in 6-digit amounts at a time.

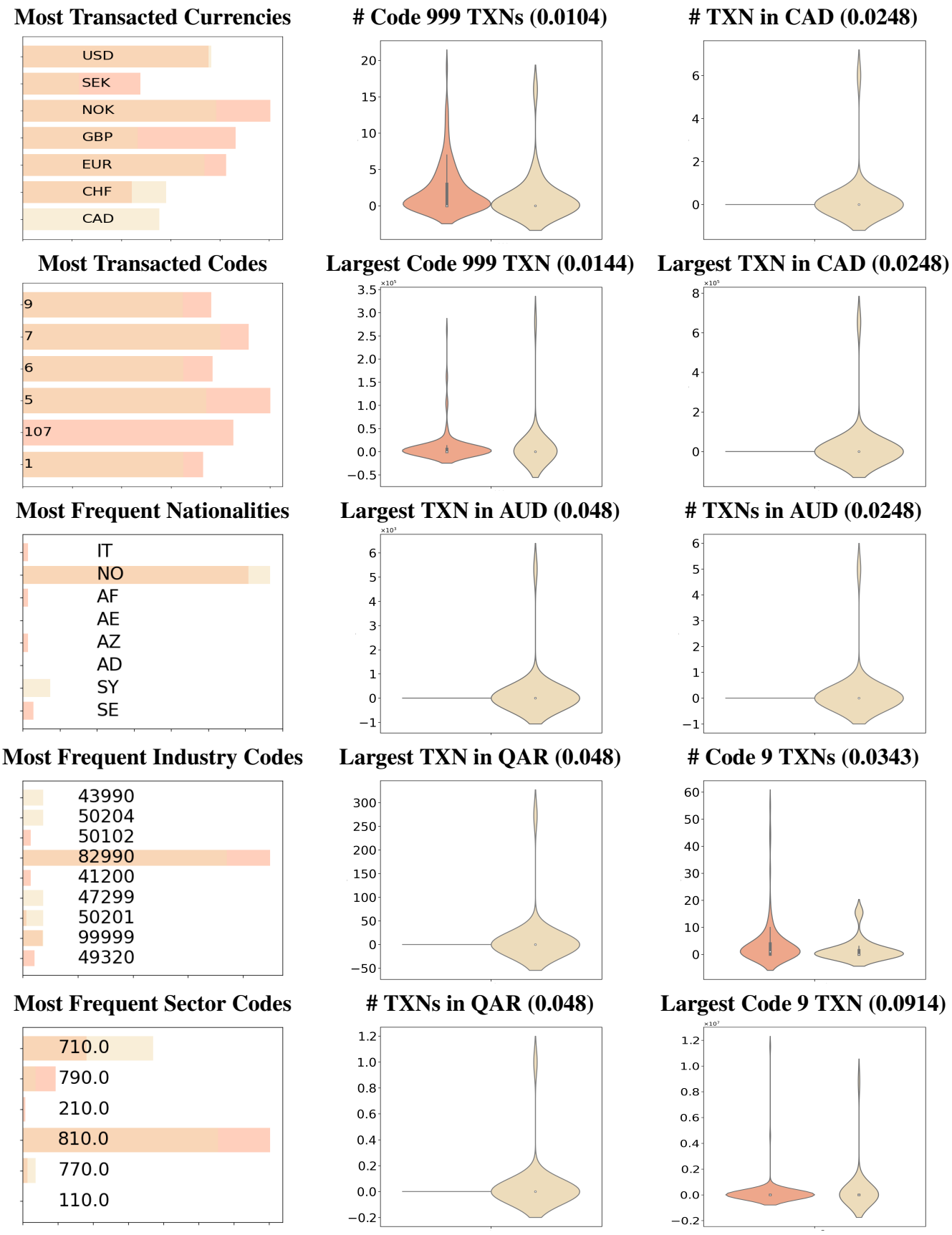


Fig. 18: Covariate visualizations for orange and beige clusters. TXN is an abbreviation for transactions.

5.3.2. Blue and gray clusters

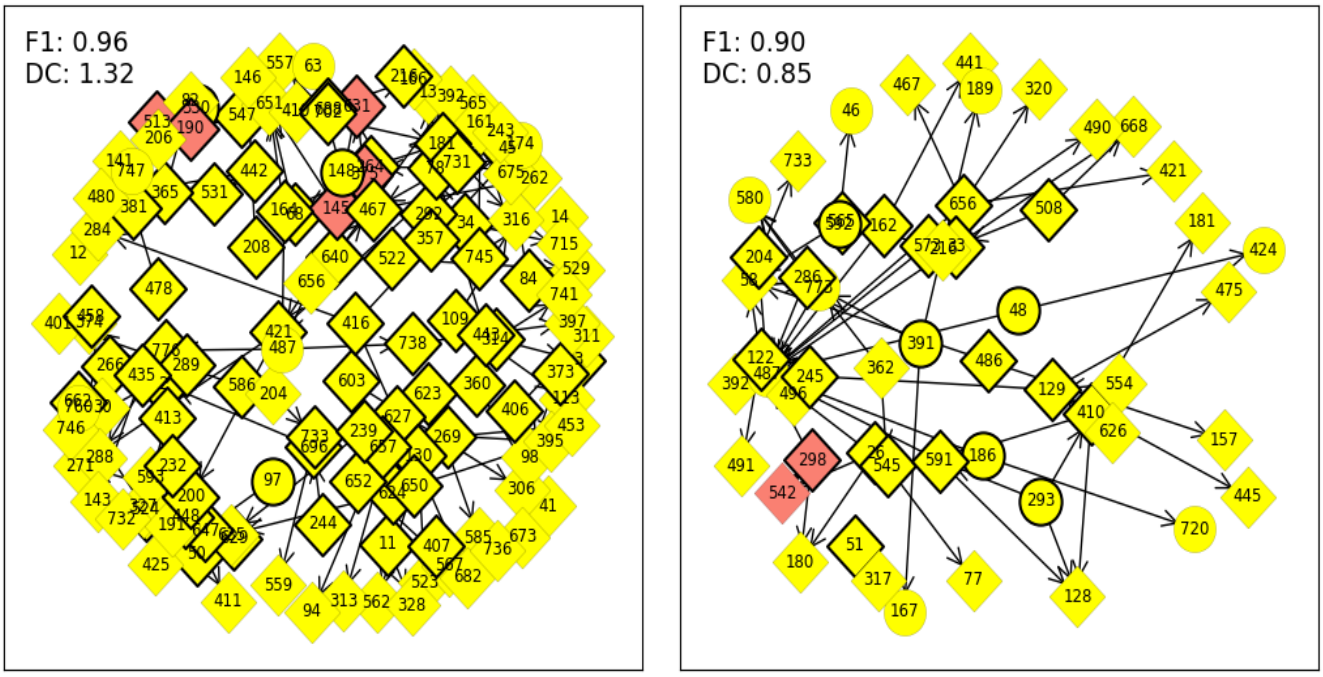


Fig. 19: The blue and gray clusters are visualized on the left and right respectively.

The blue and gray clusters are the bottom-most clusters. They share a lot of similarities, such as a high F1 score and a make-up of almost entirely positive predictions. Yet they are quite distinctively separated in space. The blue cluster has an F1 score as high as 0.96 and the gray has an F1 score of 0.90. The blue cluster also has a higher degree connectivity of 1.32 compared to the 0.85 of the gray cluster. It is also larger, consisting of 74 nodes versus the 23 nodes of the gray cluster. The network of the clusters and their first neighborhood are visualized in Figure 19. Once again, node 487 is present and is responsible for a significant portion of the edges in the gray cluster. Since this node represents a large bank, it is preferable that it is removed once again, as the networks involving it are too general. Node 773 is also removed from the gray cluster, for the same reason of it creating networks that are less distinctive. The networks without nodes 487 and 773 are visualized below it in Figure 24. There is still a decent amount of interconnectivity after removing these nodes, however, one needs to analyze the constituent components to get a more complete picture.

There is a considerable amount of sizable components in the two subgraphs. The blue cluster consists of six components with four or more nodes, the largest two containing eight nodes. The gray cluster on the other hand has five components containing over three nodes, with the largest containing seven. All of these are visualized, either in this subsection or in the Appendix, and all of them are mainly consisting of fraudulent nodes. A consistent theme for the largest components is that they are rather hard to interpret. One of the more interesting patterns is shown in Figure 21. In particular, its high degree of interconnectivity makes it interesting. Although the sequence of transactions makes it hard to construct a sequential narrative, there is clearly a network of four nodes that transact with each other in a short time span. The sequentiality, or lack thereof, making the patterns difficult to interpret is a recurring theme for the large

components. The features yet again help to indicate some suspicion, however. While nodes 128 and 591 are Norwegian nationals, the rest are corporations whose information regarding nationality is not included. Node 181 is a wholesale trader of fuel, while the other two are restaurants or cafes.

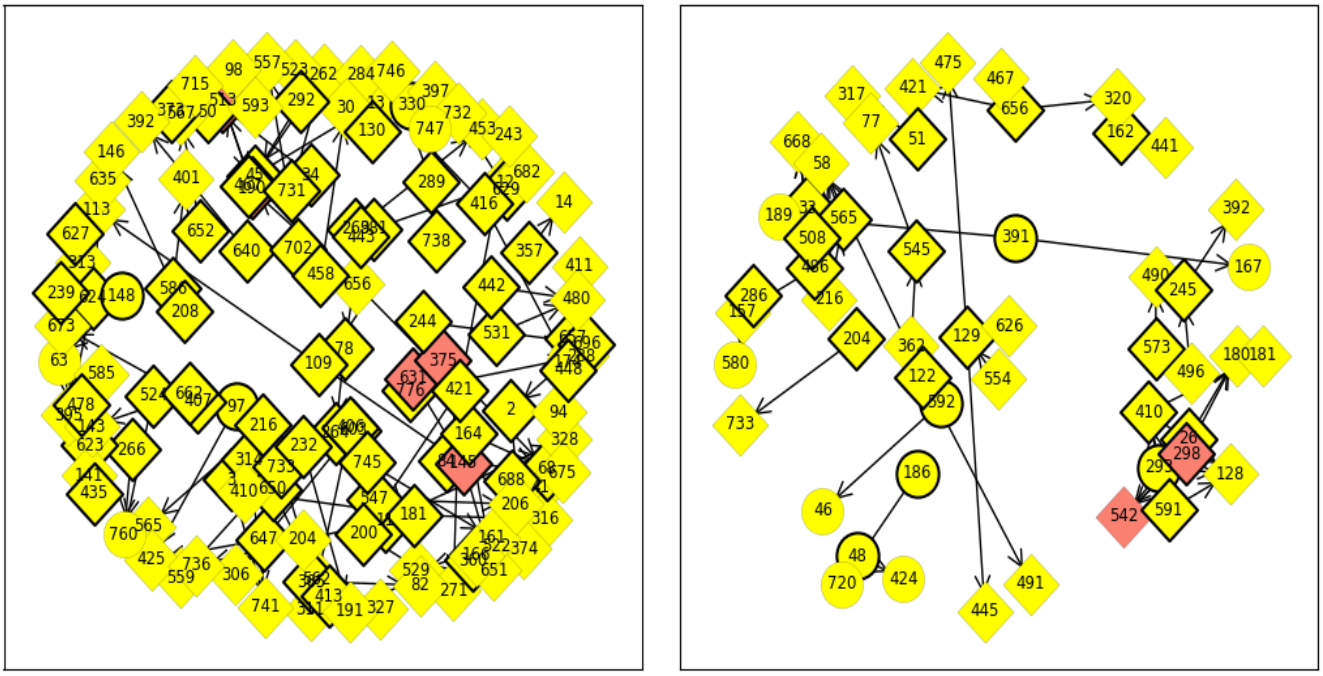


Fig. 20: The blue and gray clusters are visualized on the left and right respectively without the bank node 487. Squares denote that the true label is fraudulent, while circle denotes non-fraud. Nodes lacking a black outline are part of the first neighborhood of the clusters and not the cluster themselves.

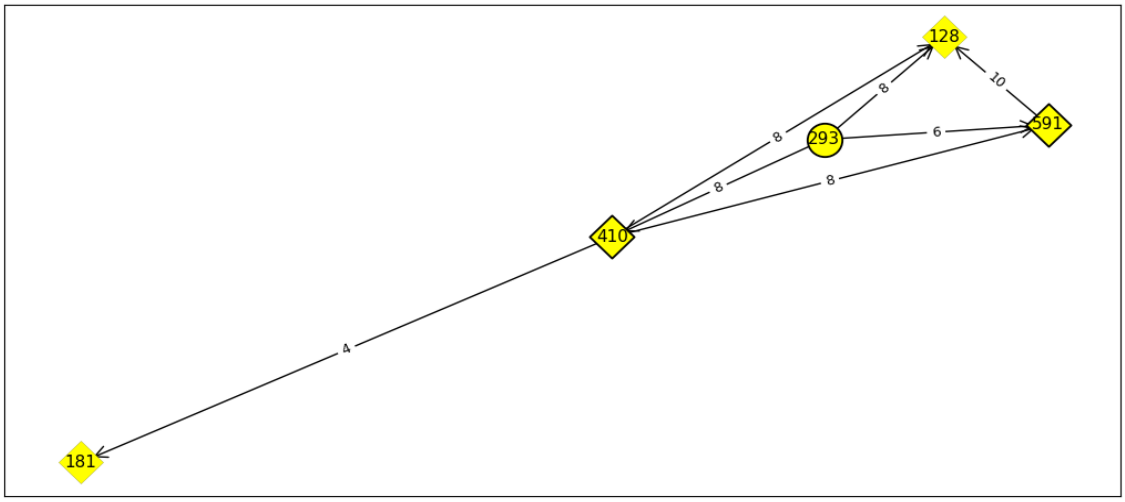


Fig. 21: The edge notations denote the two-month period of the edge

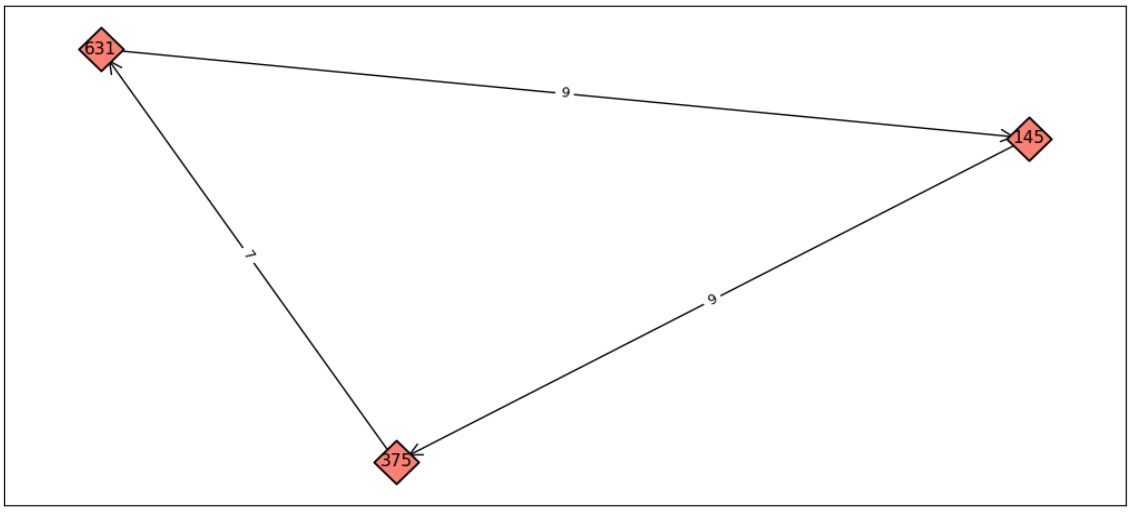


Fig. 22: The cycle in the blue cluster network. The edge notations denote the two-month period of the edge

There is also a three-node cycle present in the blue cluster which is interesting to analyze. The cycle is visualized in Figure 22. Once again, all the nodes in the network of interest are also predicted as fraudulent. Inspecting the features of the nodes in the cycle provides a narrative to the cluster. The entities involved are from three different countries spanning two continents; Norway, Sweden and Iraq, and they transact using both Norwegian and Swedish currency. The only corporation, the Norwegian node, yet again belongs to a very vague industry through its industry code 82.990. This is another common catch-all category whose description is as vague as "Other business services not mentioned elsewhere". This cyclical pattern could have been quite easily discerned even without knowing the labels and could have been directed for further investigation on the additional basis of its suspicious features.

The covariate visualizations are shown in Figure 23. The variation in transacted currencies is surprisingly similar, especially considering the difference in the sizes of the clusters. In fact, the transacted volume in the gray cluster is more spread out than that of its blue counterpart. The blue cluster is much more diverse in terms of nationality, however, this does not come as a surprise given the greater size of the cluster. The industries they operate across are interestingly different, however. The gray cluster is well represented by 56.101, which denotes cafes and restaurants, and codes representing freight transport, such as 49.410 and 50.204 which denote freight transport across road and sea respectively. The blue distribution includes a good share of cleaning services denoted by 81.210, but is apart from that spread out on a number of seemingly unrelated industries. The covariate distributions deemed divergent by the Mann-Whitney U test appear to be greatly affected by outliers. It is clear that the gray cluster has a node that transacts both often and in large quantities using both EUR and USD. Apart from that they display different transaction codes that are better represented in one cluster than the other.

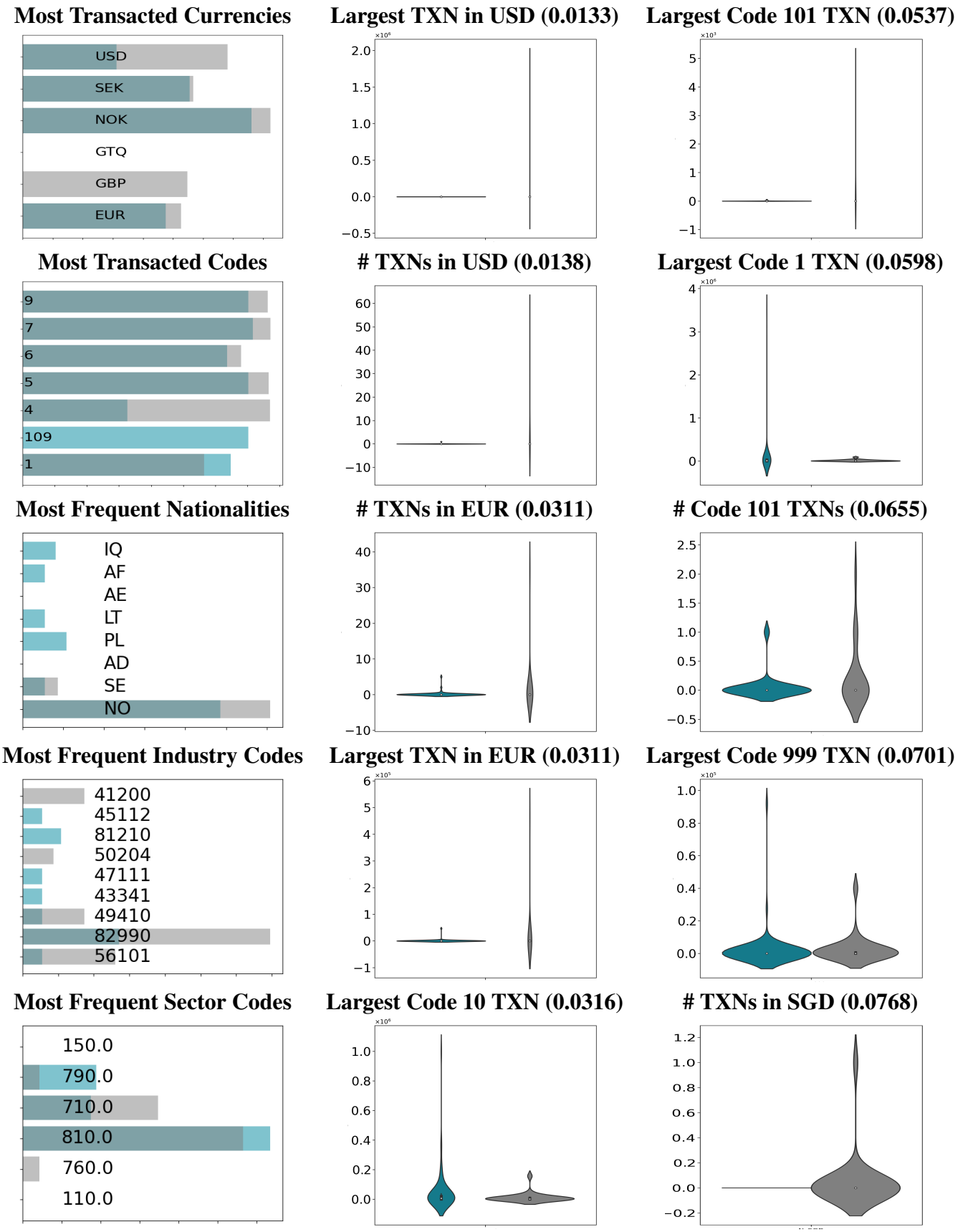


Fig. 23: Covariate visualizations for blue and gray clusters. TXN is an abbreviation for transactions.

5.3.3. *Upper left cluster and green cluster*

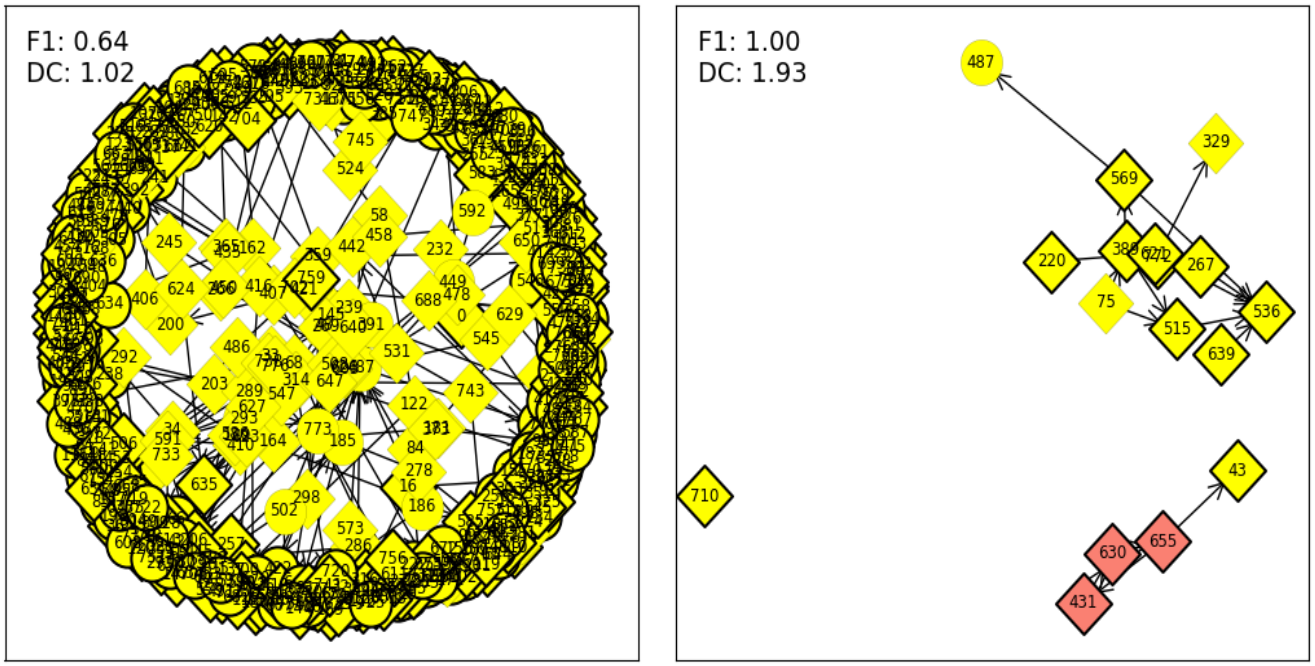


Fig. 24: The upper left and green clusters are visualized on the left and right respectively. Squares denote that the true label is fraudulent, while circle denotes non-fraud. Nodes lacking a black outline are part of the first neighborhood of the clusters and not the cluster themselves.

As previously mentioned, the upper left mixed cluster is of less interest due to its great variability in regard to predictions and uncertainty, and will therefore be treated as a whole instead of looking into the k-means partitions. As a whole, this cluster stands in contrast to the green cluster in terms of F1 performance, the latter of which is predicted perfectly. They also represent the largest and smallest of the clusters analyzed, the upper left consisting of 501 nodes and the green 14 nodes. Their networks are visualized in Figure 24. Node 487 is found in the neighborhood of the clusters but is not of any significant centrality. While the upper left network is too large to make inferences about by simply visualizing it. A quick glance at the network of the green clusters, however, discerns two very succinct components, all of which are in the cluster and not the first neighborhood, and are correctly predicted as fraud. In both cases, the next step is to remove the bank node and inspect the components further. After doing so, the upper left and green networks consist of 483 and 4 components each. Out of the 483 in the upper left network, 8 of the components consist of more than 3 nodes.

The upper left network does include a set of interesting fraudulent components, yet under-delivers relative to its size. 8 sizable components are few when one takes into consideration that the component consists of 501 nodes. For comparison, these components are of equal size and only two more than the components the 74-node large blue cluster and its network mustered. These components also rarely capture large coherent parts of these fraudulent networks if one excludes the first neighborhood of nodes. An interesting network that is an extension of the one in Figure 21 is shown on the left part of Figure 25. Although it was previously discovered in the blue cluster with a much better F1-score and greater certainty and

separability, the upper left cluster manages to uncover a greater part of the network. It is hard to determine the significance of this, however, as it might be coincidental given that the recovered network on the left in Figure 25 just barely forms a coherent component on the sole basis of the first neighborhood nodes connecting, while the nodes that are part of the upper left cluster are far apart. Another component from the upper left cluster is shown in the right part of Figure 25, and it indicates a recurring pattern where a group of three or more fraudulent nodes all transact to a fourth fraudulent one. The receiving node 651 is the only Norwegian individual in the network, while the rest are either of Iraqi or Swedish nationality. The nodes in the network transact in both NOK and SWE. Yet again, however, few of the nodes are part of the cluster itself, as all but one are neighboring nodes. These observations seem to confirm the a priori belief that this larger mixed cluster is of less analytical utility than the others.

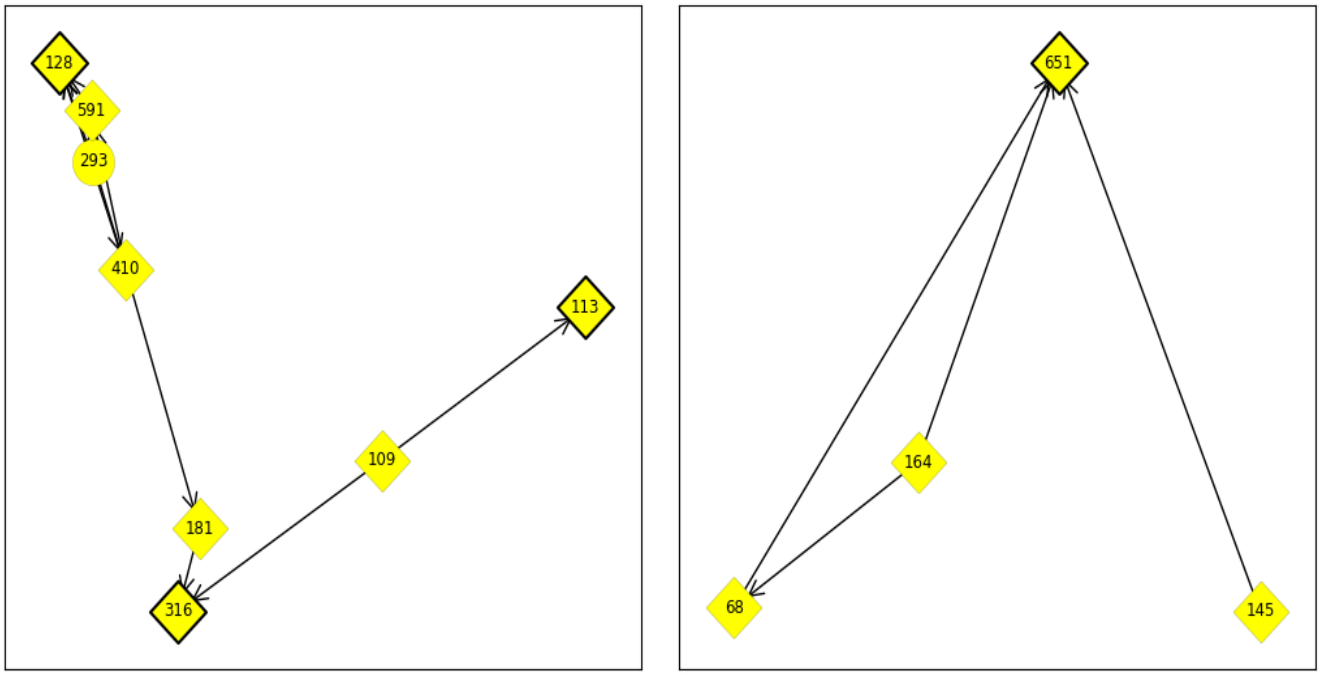


Fig. 25: The two highlighted components from the upper left cluster network.

The green cluster is on the opposite end of the spectrum, and quite unique in how much of the fraudulent networks it manages to capture entirely within the cluster itself. An example is the cycle pattern in the green cluster which is visualized on the left in Figure 26. Analyzing its base features alone makes it is hard to find any suspicion; they are all Norwegian nationals transacting in NOK. They do seem to correlate a lot with regard to the transaction codes, however, it is hard to say how suspicious this is, as the meaning of the transaction codes are not available to us. The other sizable component, the network visualized on the right in Figure 26 is probably the fraudulent network with the greatest amount of nodes captured within the cluster itself. It is another network that extensively captures the recurrent pattern of multiple fraudulent nodes transferring funds to a fraudulent node 536. The features inform that four of the individuals are of Norwegian nationality, while one is of Somali nationality. The nodes have also transacted in a relatively high number of currencies: NOK, SWE, USD, EUR and AED. This cluster contrasts the upper left cluster, as it probably is the most interesting to analyze. In a setting without labels, it is not only a cluster that is easily discernible, but also one that gives an analyst considerable material to work with.

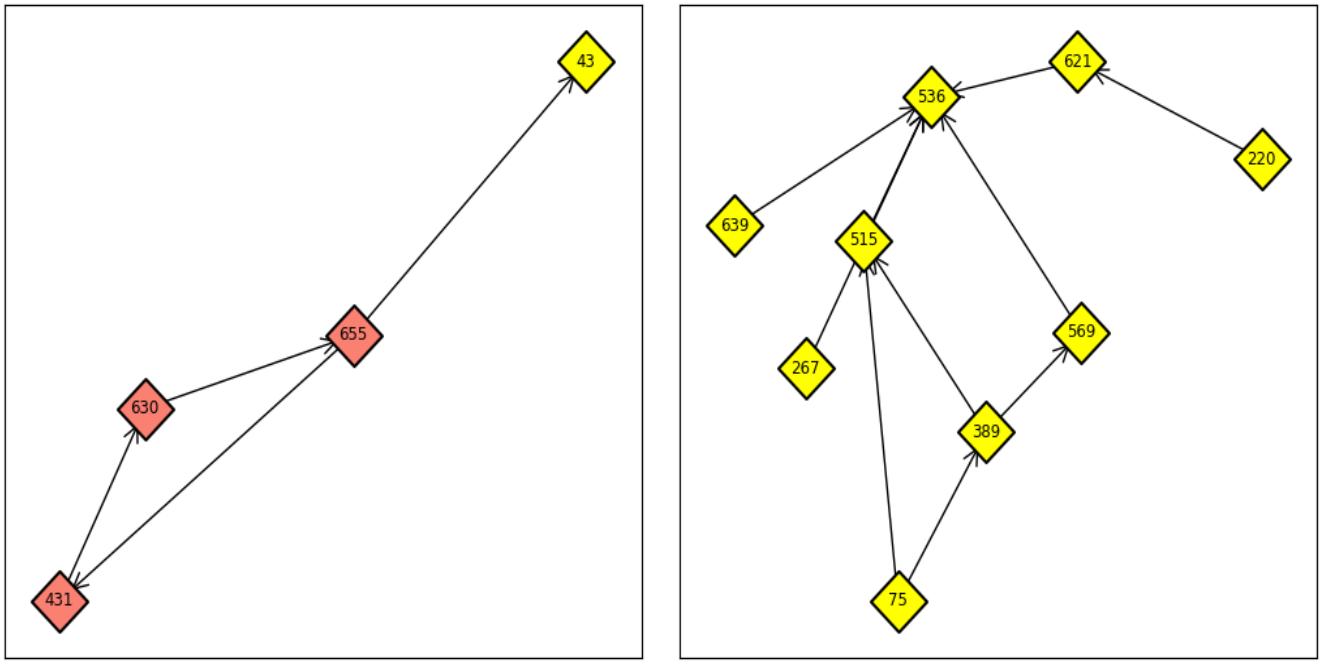


Fig. 26: The cycle and large component in the green cluster network to the left and right, respectively.

The covariates of the clusters are visualized in Figure 27. Note that the gray color here was chosen to symbolize the generality of the cluster, and should not be confounded with the gray cluster described before. The two clusters transact in mostly the same currencies, with the exception being GBP which is only found in the upper left cluster. Of interest is the over-representation of AED in the green cluster. The industries are noticeably different. There is not any pattern, however, besides the upper left cluster containing a decent share of corporations that are cafes/restaurants or involved in naval freight transport. These are reminiscent of the industries represented in the gray cluster, and might indicate that these industries are over-represented in money laundering schemes. The fact that the green cluster is the outlier in the three most divergent covariate distributions is interesting. It seems that the green cluster has a node that is rather extreme in regard to the number of transactions in code 107 and AED.

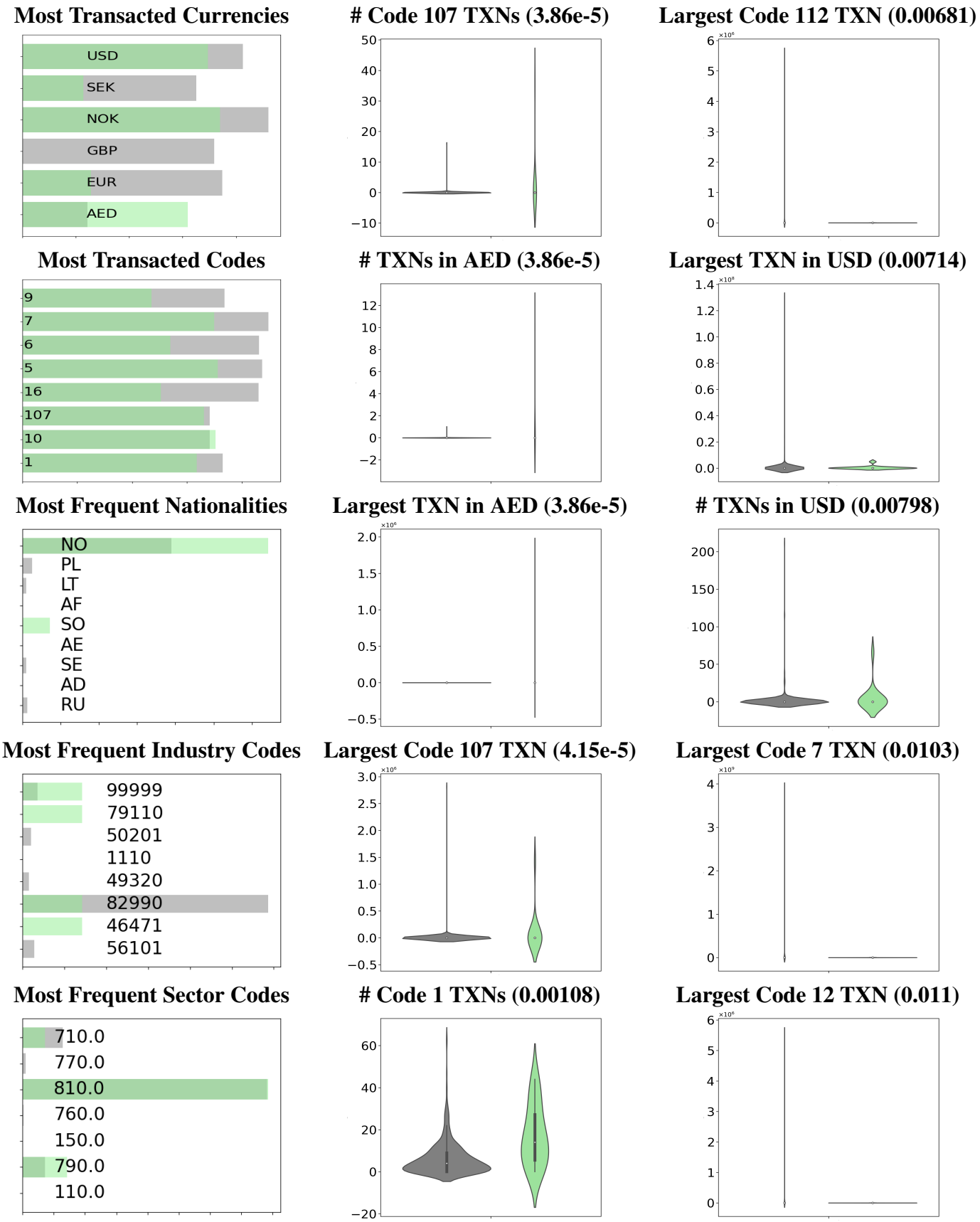


Fig. 27: Covariate visualizations for upper left and green clusters. TXN is an abbreviation for transactions.

5.3.4. Summary of exploratory analysis

1. Incorporating non-spatial dimensions for algorithmic clustering yielded more sensible clustering per the defined criteria. Predicted probability is responsible for the bulk of the improvement, with marginal gains from uncertainty.

2. The removal of very central corporate nodes such as banks seems to be an important step for analysis.

    - Removal as a first step before doing the cluster analysis strengthens the correlation between the cluster F1 score and degree connectivity to 0.432 with a p-value of 5.35e-17.
    - For pattern analysis, the removal of central nodes is an important step of an uncovered heuristic to recover fraudulent networks.

3. The a priori assumption of the upper left region being a difficult cluster to analyze seems to hold, given the relatively few fraudulent components in its network relative to its great size.

4. The idea of clustering with a spread in predictive performance being beneficial for analysis seems to hold some ground. It was possible to recover more fraudulent networks relative to the number of nodes in clusters with greater F1-score. This means that in cases with limited resources for inspection, one could prioritize clusters based on how good one believes their predictive performances are. Either indicated by an a priori assessment of the uncertainty plot, or a correlating measure such as degree connectivity (given that the correlation is still present in a more realistic data set)

5. Recurring patterns in recovered fraudulent networks were cycles and the allocation of funds from multiple parties to one. These patterns were often accompanied by covariates that indicated suspicion such as multiple nationalities and transacted currencies.

6. LIMITATIONS

The model in particular imposed a series of limitations on the work. The first of which is that the model uses drop connect for uncertainity handling. As described in the Theory section, this is merely an approximation of Bayesian UQ, and does not necessarily offer ideal uncertainty handling with regards to the parameters themselves. Beyond poorer model performance, the effects of this cascade downstream and leads to less accurate embedding uncertainties and thereby less sensible clustering. While this is a hinder for GNNs for the time being, it is not so when implementing Bayesian UQ for more conventional forms of ANNs (Hubin and Storvik, 2023). A proposed solution in the context of GNNs would be to apply normalized flows (Skaaret-Lund et al., 2023).

The most significant constraint encountered was the incapacity of the model to effectively learn from imbalanced datasets. Despite many tests with various sampling ratios, the model struggled to learn as soon as class representation became skewed. Any notable AUC observed in the validation dataset appeared to be coincidental, as this performance didn't translate to the training or test datasets, which consistently exhibited an AUC of 0.5. This issue resulted in an unrealistic depiction of the data set for the final model. The process of undersampling led to a scarcity of intact networks from the majority class samples. Practivally all networks of interest, with a few exceptions, were from the positive class. This situation significantly boosted the predictive power of degree connectivity for F1 performance. Therefore, the heuristic approach of removing central nodes and examining components in clusters proved effective to an extent that might not reflect real conditions.

7. DISCUSSION

In this section, we will answer the research questions, as well as discuss how a more ideal setting could yield improved answers to these questions.

1. Can Bayesian Graph Neural Networks and PCA yield node embedding visualizations that show the clustering of nodes of similar predictions and uncertainty?

The uncertainty plot displayed a noticeable differentiation among clusters, although diversity of clusters in regard to predicted class and uncertainty was lacking. Distinct clusters could be readily observed, although the specific number may vary depending on the chosen clustering approach. The work found that a simple k-means clustering that incorporated the non-spatial dimensions was able to yield clusterings that were both intuitive and sensible with regard to our defined criteria. The selected clustering method yielded five discernible clusters that were considered reasonable. Four of those were high-certainty positive clusters (blue, gray, orange and beige), while one was a low-certainty positive cluster (green). The majority of the nodes, however, were situated in a region of the uncertainty plot that was difficult to differentiate, consisting of different predictions with varying degrees of certainty.

Using an improved classifier designed to handle more imbalanced data sets and incorporate uncertainty more effectively, one can hope for a broader range of clusters. The classifier trained in this work had a high recall and a relatively low precision and is biased toward positive clusters. With an improved model, one could hopefully observe negative clusters with varying degrees of certainty too. Studying how the properties of these different types of clusters and their networks diverge could yield interesting results.

2. Can one recover money laundering patterns in the networks of these clustered nodes?

During the exploratory analysis, a heuristic naturally developed that seemed to do well in recovering fraudulent networks. The first step is to retrieve the first neighborhood of the nodes in the cluster. In order to create networks that are more specific to their constituent nodes, one then proceeds by eliminating prominent central nodes, such as banks. This process usually leads to the division of the network into multiple components. If a component exhibits characteristics like cyclic patterns or the concentration of funds towards a specific party, alongside suspicious node attributes, there is an increased probability of that component being fraudulent. Given the artificial imbalance in the data, however, practically any large component was shown to be fraudulent. In a setting with a more realistic data set, a more discerning approach would be necessary, with more emphasis on identifying such patterns and features first.

3. What interpretations can be made from the eventually recovered patterns?

Due to the limited diversity of identified clusters, there is a restricted capacity to draw insightful conclusions regarding money laundering and the methods employed to detect it through cross-examination. In this context, the recovered patterns are of limited value in and of themselves. The analysis has shown that one can interpret networks with cycles or the pooling of funds towards a particular entity, in conjunction with suspicious node characteristics, as reasonably suspicious. However, we can only make that inference because current AML systems already are capable of detecting these types of networks. With an ideal model, one could hope for a more varied set of clusters, especially in regard to uncertainty, in which case comparing the patterns uncovered in the different types of clusters might be more insightful.

One can make some interpretations from the recovered networks with regard to the clustering itself. The number of considerable recovered networks in a cluster relative to its size, can serve as a measure of the analytical utility of the cluster. This means that in situations where inspection resources are limited, it can be efficient to prioritize clusters by this measure of analytical utility. In a setting without labels available, one can attempt to assess this analytical utility by a variety of methods. A priori evaluations of the separability of the clusters can at least be valuable in identifying clusters that are of *little* analytical utility, as is the case with the upper left cluster. Furthermore, clustering in a manner that increases the degree connectivity of the clusters can be beneficial, as the degree connectivity correlates with the predictive performance of the cluster and thereby its analytical utility. However, it remains to see if this correlation still holds in a realistic setting with an ideal model. If so, there would be at least two potential methods that could be utilized to enhance the workflow of AML analysis and improve its effectiveness.

Further Work

Besides the alleviation of the previously described limitations, there are several directions one could pursue onward. One promising approach would be to explore the use of heterogeneous graphs. As opposed to homogeneous graphs, which contain only one type of node and edge, heterogeneous graphs comprise multiple types of nodes and edges, allowing for a richer representation of complex network structures. This could potentially allow for the capture of more nuanced relationships and interactions within the network, thereby improving model performance, as well as allowing for more informative inspection of the recovered networks.

Furthermore, it would be worth investigating oversampling techniques to better handle the imbalanced data set. Although a model that handles the full imbalance of the DNB data set would be ideal, it is unlikely that one could get a good model performance without any form of resampling. By oversampling the minority class nodes, one could ensure that more majority class nodes are given consideration than in the current implementation which only utilizes undersampling.

Considering alternative clustering techniques could also offer further avenues for future work. Ideally one could attempt to make clusters using an Expectation-Maximixing algorithm that takes into account the mixtures of Gaussian and Beta distributions inherent in the data. In addition, one could use more sophisticated evaluation criteria such as the Akaike information criterion for a more "objective" way to select the number of clusters.

References

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080%2F01621459.2017.1285773.

Richard J. Bolton and David J. Hand. Statistical Fraud Detection: A Review. *Statistical Science*, 17 (3):235 – 255, 2002. doi: 10.1214/ss/1042727940. URL https://doi.org/10.1214/ss/1042727940.

Corina-Narcisa Cotoc, Maria Nițu, Mircea Constantin Șcheau, and Adeline-Cristina Cozma. Efficiency of Money Laundering Countermeasures: Case Studies from European Union Member States. *Risks*, 9(6), 2021.

Andrew Gelman, John Carlin, Hal Stern, David Dunson, Aki Vehtari, and Donald Rubin. *Bayesian Data Analysis*. 11 2013. ISBN 9780429113079. doi: 10.1201/b16018.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015. ISSN 1476-4687. doi: 10.1038/nature14541. URL https://doi.org/10.1038/nature14541.

R. Grint, C. O'Driscoll, and S. Patton. New technologies and anti-money laundering compliance report, Aug 2017. URL https://www.fca.org.uk/publications/research/new-technologies-and-anti-money-laundering-compliance-report.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian Graph Neural Networks with Adaptive Connection Sampling, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Aliaksandr Hubin. Using node embedding to obtain information from network based transactions data in a bank. https://app.cristin.no/results/show.jsf?id=1816848, 2019. Presentation at BIG INSIGHT DAY 2019, Oslo, November 14, 2019.

Aliaksandr Hubin and Geir Storvik. Variational Inference for Bayesian Neural Networks under Model and Parameter Uncertainty, 2023.

Fredrik Johannessen. Finding money launderers using heterogeneous graph neural networks. Master's thesis, Department of Mathematics, University of Oslo, 2022.

Ragnhild Johansen. United Nations Office on Drugs and Crime, 2007. URL https://web.archive.org/web/20200308095753/https://www.unodc.org/unodc/en/money-laundering/globalization.html.

Ian T Jolliffe. Principal component analysis and factor analysis. *Principal component analysis*, 2:150–166, 2002.

Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, 23(1): 173–186, 2020.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5. URL https://www.sciencedirect.com/science/article/pii/S0893608005801315.

Wouter H Muller. Anti-money laundering–a short history. *Anti-Money laundering: International law and practice*, page 1, 2007.

Radford M. Neal. Bayesian Learning for Neural Networks. 1995.

Ebberth L. Paula, Marcelo Ladeira, Rommel N. Carvalho, and Thiago Marzagão. Deep Learning Anomaly Detection as Support Fraud Investigation in Brazilian Exports and Anti-Money Laundering. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 954–960, 2016. doi: 10.1109/ICMLA.2016.0172.

Thomas Pierrot, Nicolas Perrin, and Olivier Sigaud. First-order and second-order variants of the gradient descent: a unified framework. *CoRR*, abs/1810.08102, 2018. URL http://arxiv.org/abs/1810.08102.

Peter Reuter. Are estimates of the volume of money laundering either feasible or useful? In *Research handbook on money laundering*, pages 224–231. Edward Elgar Publishing, 2013.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Hong Shi, Xiaomene Zhang, Shizhong Sun, Lin Liu, and Lin Tang. A Survey on Bayesian Graph Neural Networks. In *2021 13th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, pages 158–161, 2021. doi: 10.1109/IHMSC52134.2021.00044.

Lars Skaaret-Lund, Geir Storvik, and Aliaksandr Hubin. Sparsifying Bayesian neural networks with latent binary variables and normalizing flows, 2023.

Agus Sudjianto, Ming Yuan, Daniel Kern, Sheela Nair, Aijun Zhang, and Fernando Cela-Díaz. Statistical Methods for Fighting Financial Crimes. *Technometrics*, 52(1):5–19, 2010. ISSN 00401706, 15372723. URL `http://www.jstor.org/stable/40586676`.

Jun Tang and Jian Yin. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In *2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3453–3457 Vol. 6, 2005. doi: 10.1109/ICMLC.2005.1527539.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, 2018.

Mingzhang Yin and Mingyuan Zhou. ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks, 2019.

Code Listings

```python
class EmbeddingExtractor(nn.Module):
    def __init__(self, model, layers):
        super().__init__()
        self.model = model
        self.layers = layers
        self._features = {layer: torch.empty(0) for layer in layers}

        for layer_id in layers:
            layer = dict([*self.model.named_modules()])[layer_id]
            layer.register_forward_hook(self.save_outputs_hook(layer_id))

    def save_outputs_hook(self, layer_id):
        def fn(_, __, output):
            self._features[layer_id] = output
        return fn

    def forward(self, **kwargs):
        _ = self.model(**kwargs)
        return self._features
```

Listing 1: EmbeddingExtractor class used to extract node embeddings.

```python
def get_n_layers(indexes, adjacency_matrix, n_layer=1):
    # Get n neighbourhood
    full_G = nx.from_scipy_sparse_array(adjacency_matrix, create_using=nx.
       DiGraph())
    connected_nodes = set()
    original_nodes = indexes
    indexes = set(indexes) | connected_nodes
    for _ in range(int(n_layer)):
        for node in indexes:
            neighbors = list(full_G.predecessors(node)) + list(full_G.
               successors(node))  # full_G[node]
            for neighbor in neighbors:
                if neighbor not in indexes:
                    connected_nodes.add(neighbor)
        indexes = set(indexes) | connected_nodes
    indexes = list(indexes)

    # Get subgraph adj matrix
    subgraph_adj_matrix = adjacency_matrix[indexes, :][:, indexes]

    # Create a graph from the subgraph adj matrix
    G = nx.from_scipy_sparse_array(subgraph_adj_matrix)

    # Relabel the nodes to match the original indexes
    mapping = {i: indexes[i] for i in range(len(indexes))}
    G = nx.relabel_nodes(G, mapping)

    return indexes
```

Listing 2: get_n_layer function used to get neighbouring nodes.

Training Curves

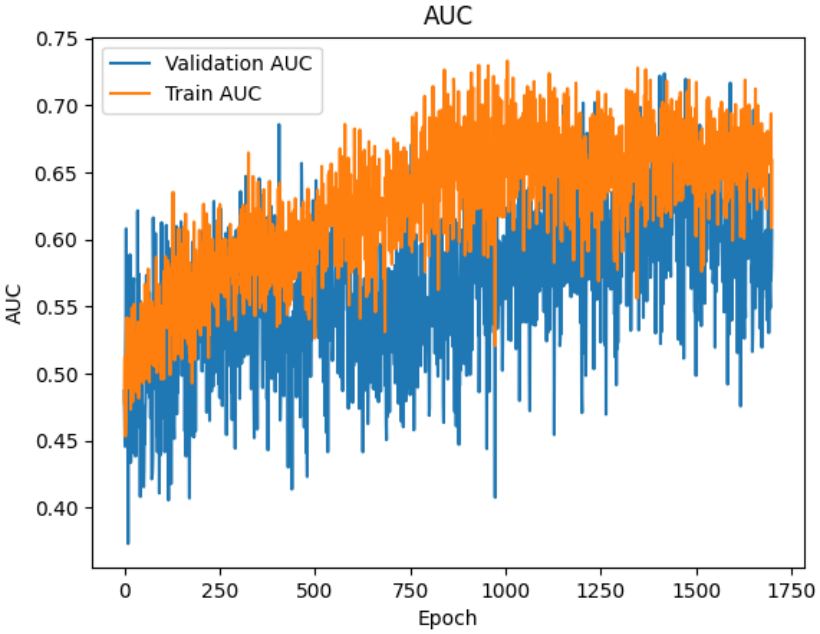


Fig. 28: Training curve for model with 50%-50% class balance.

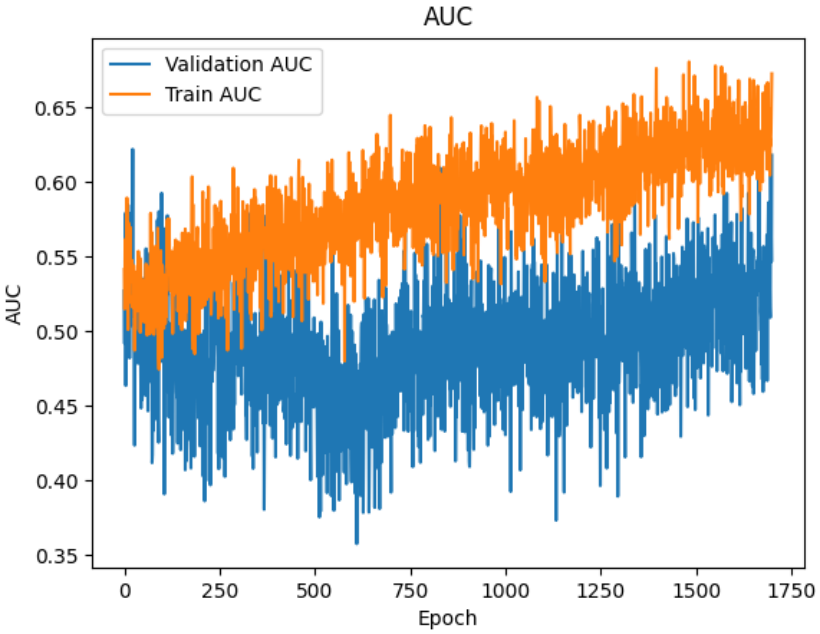


Fig. 29: Training curve for model with 71.4%-28.6% class balance.

Orange Cluster Components

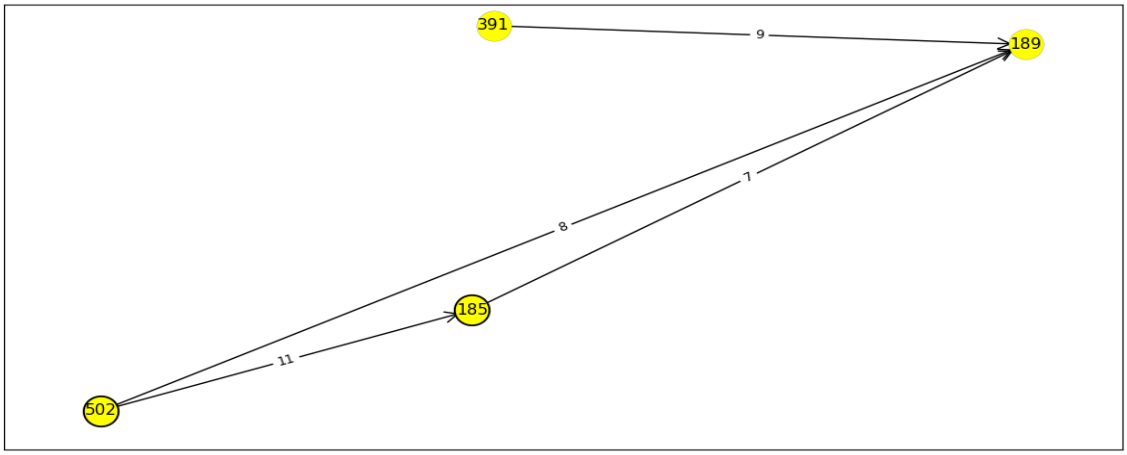


Fig. 30: Component of orange network.

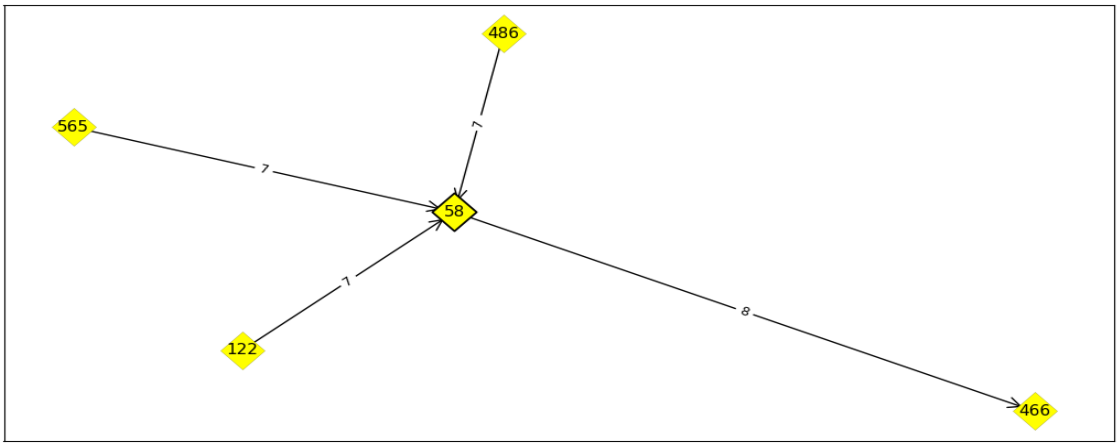


Fig. 31: Component of orange network.

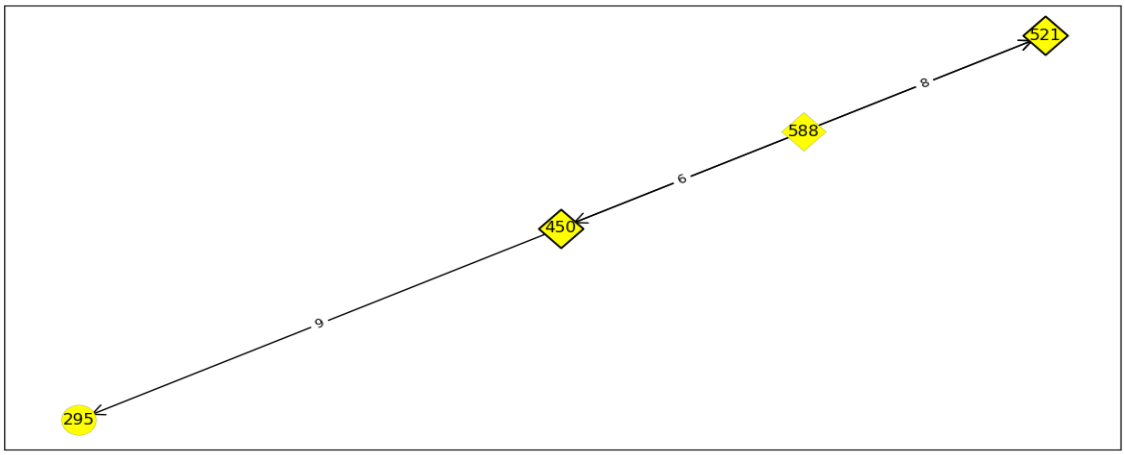


Fig. 32: Component of orange network.

Beige Cluster Components

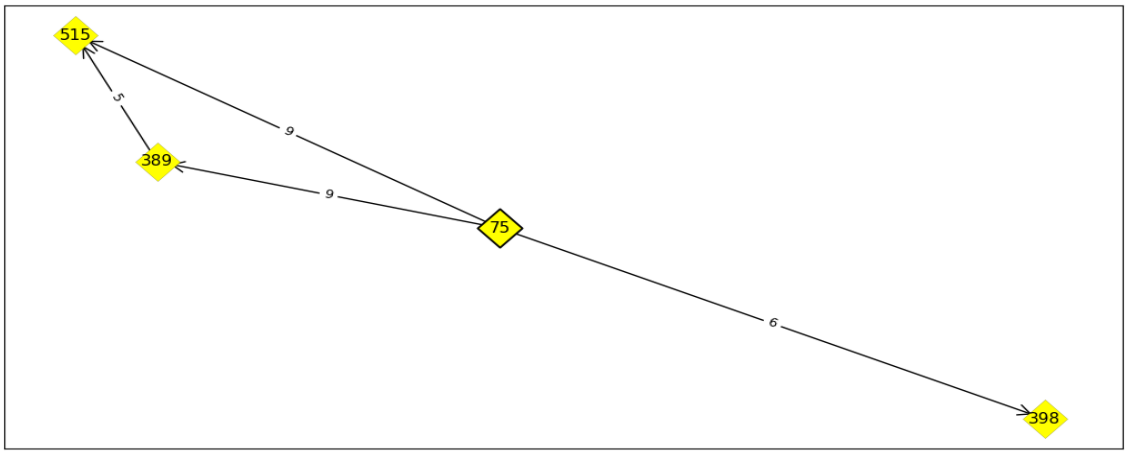


Fig. 33: Component of beige network.

Blue Cluster Components

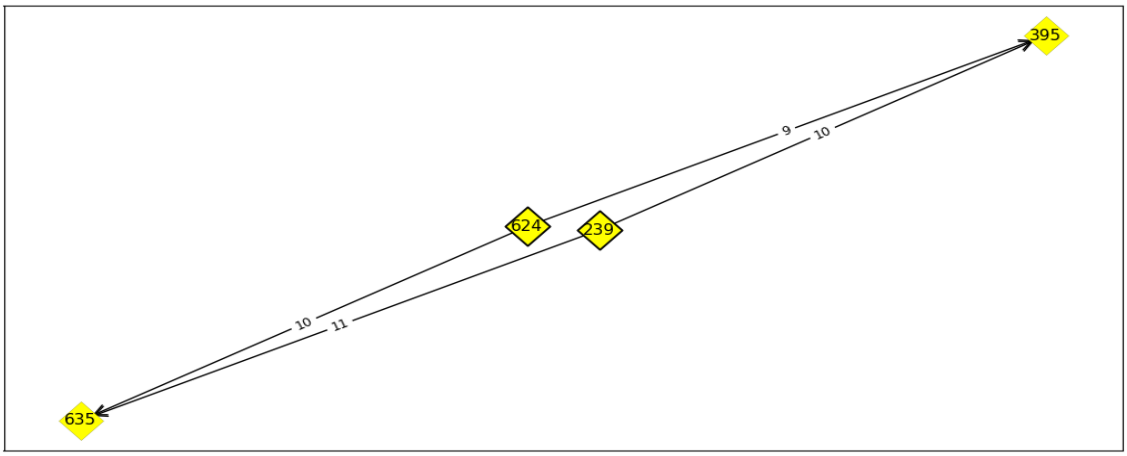


Fig. 34: Component of blue network.

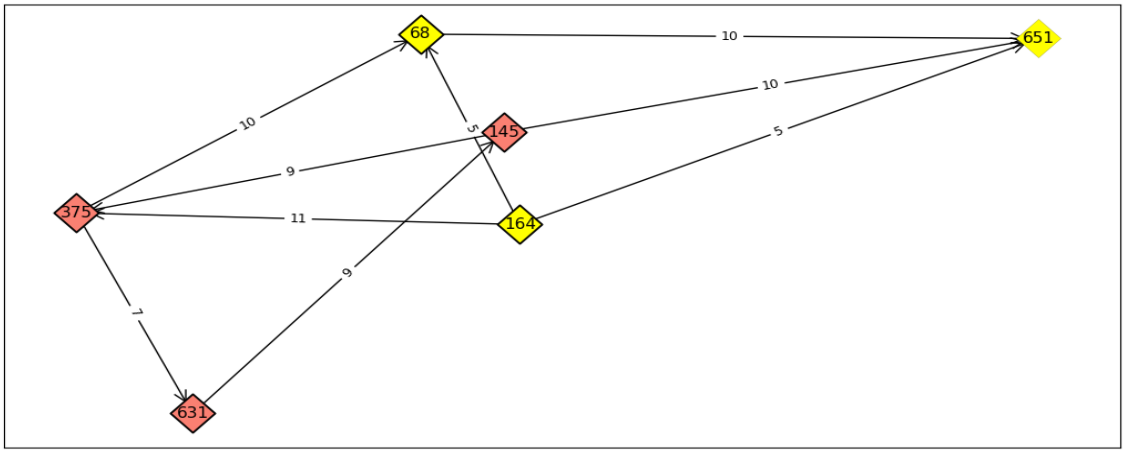


Fig. 35: Component of blue network.

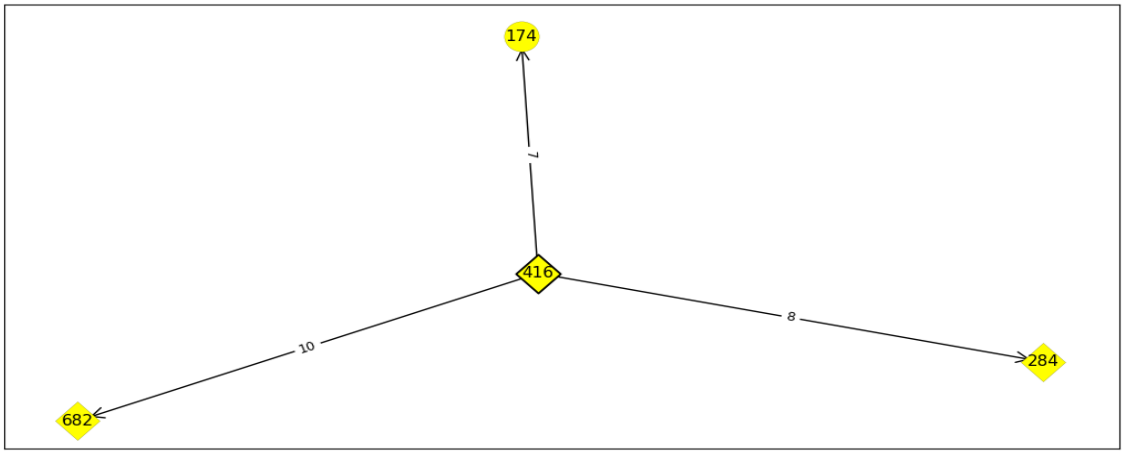


Fig. 36: Component of blue network.

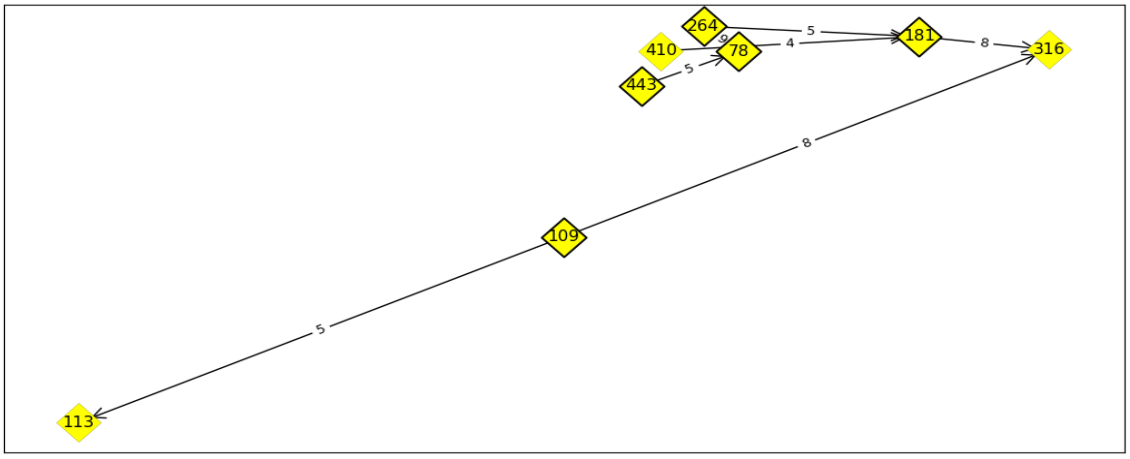


Fig. 37: Component of blue network.

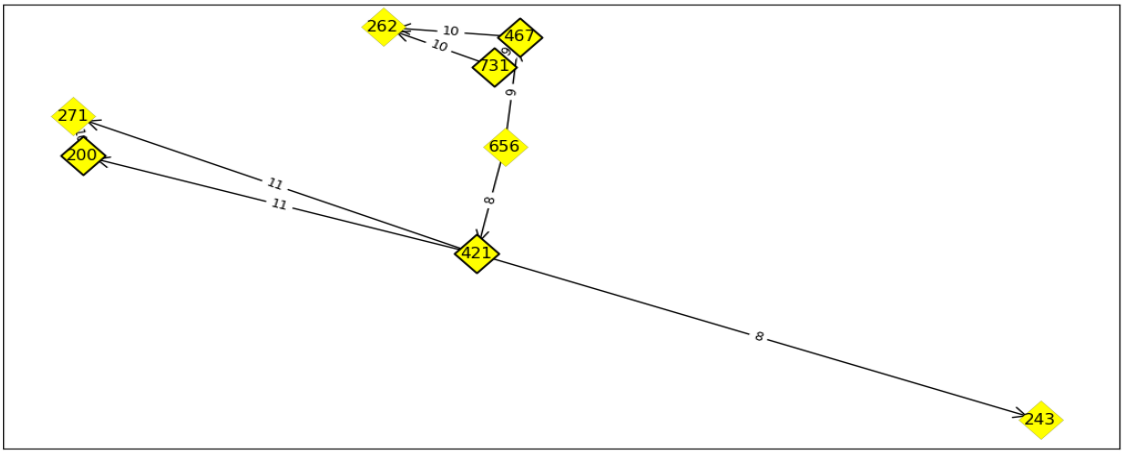


Fig. 38: Component of blue network.

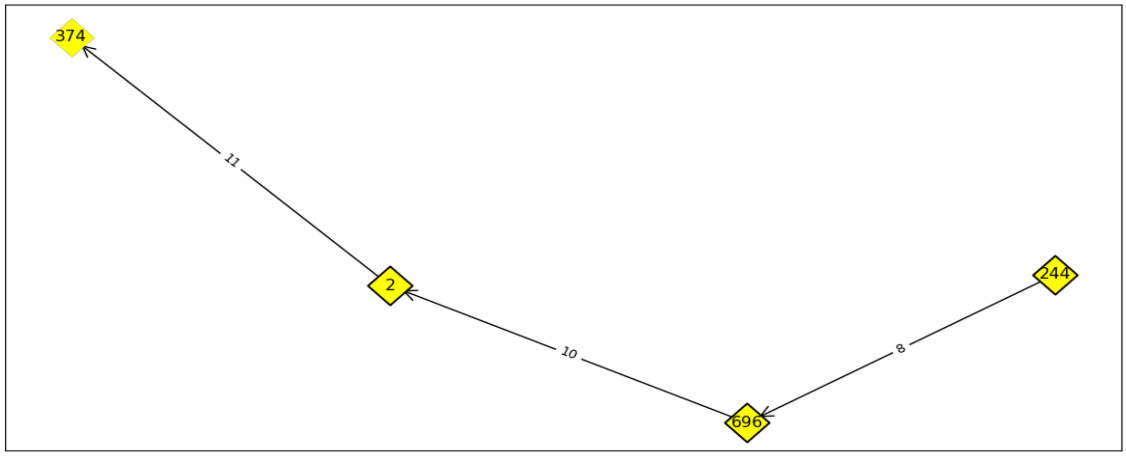


Fig. 39: Component of blue network.

Gray Cluster Components

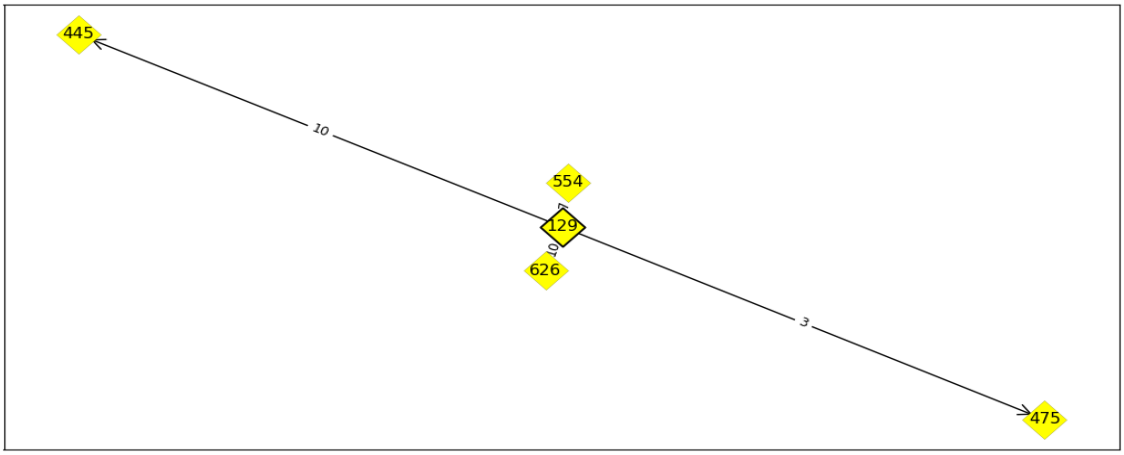


Fig. 40: Component of gray network.

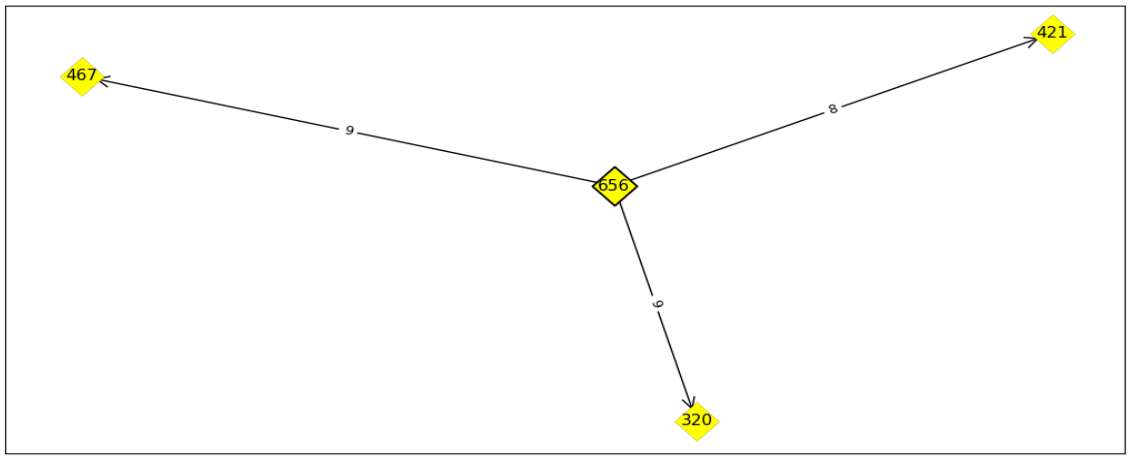


Fig. 41: Component of gray network.

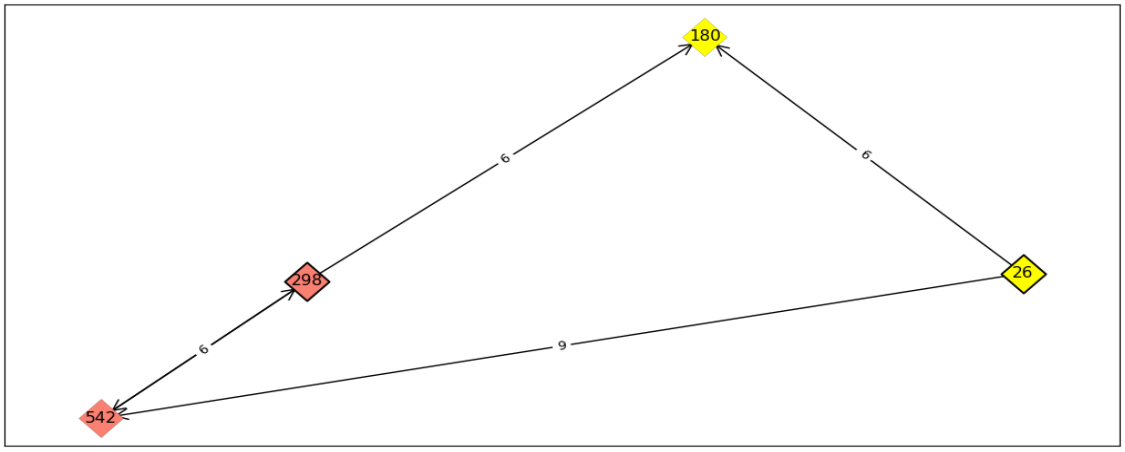


Fig. 42: Component of gray network.

Upper Left Cluster Components

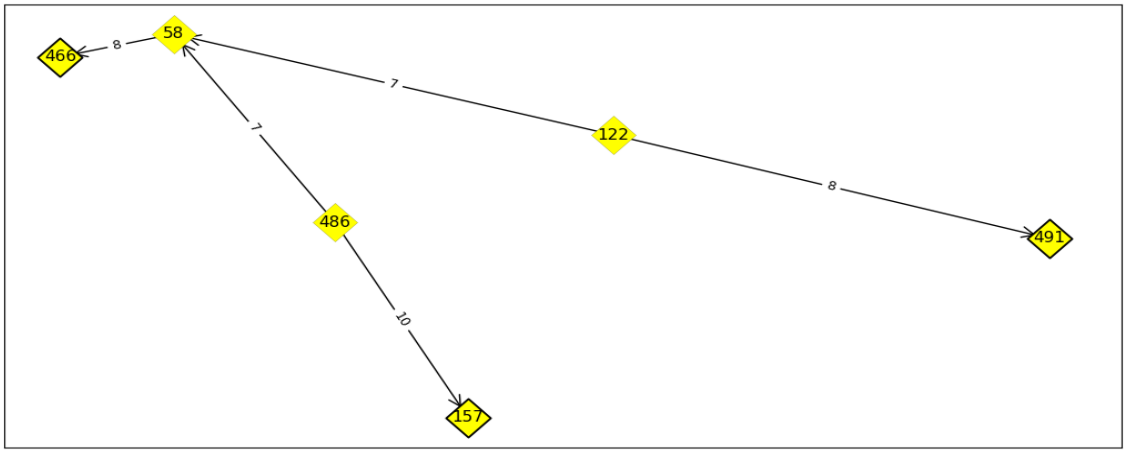


Fig. 43: Component of the upper left network.

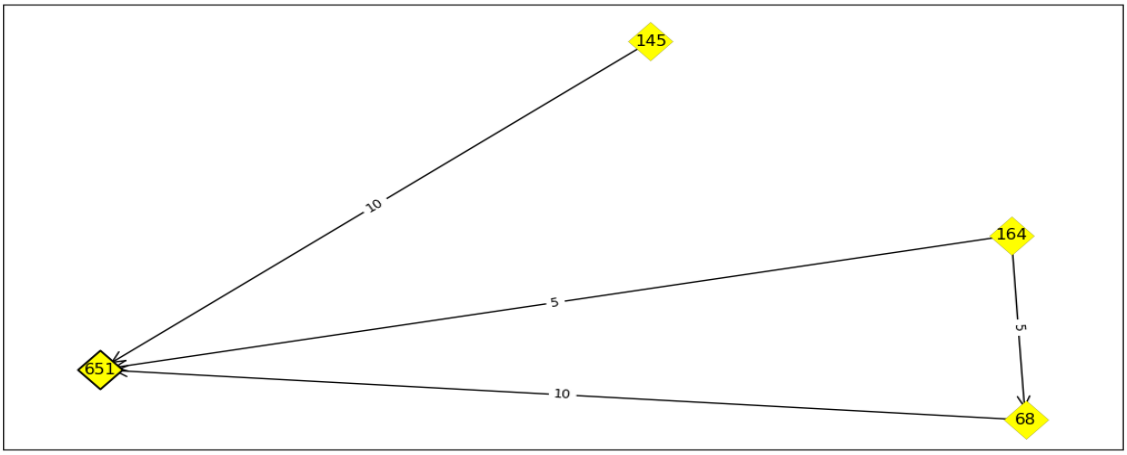


Fig. 44: Component of the upper left network.

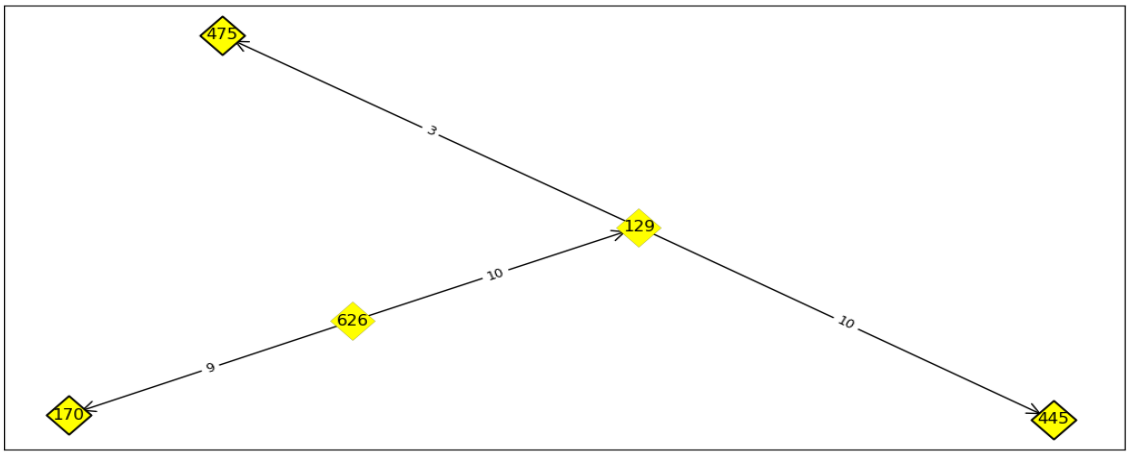


Fig. 45: Component of the upper left network.

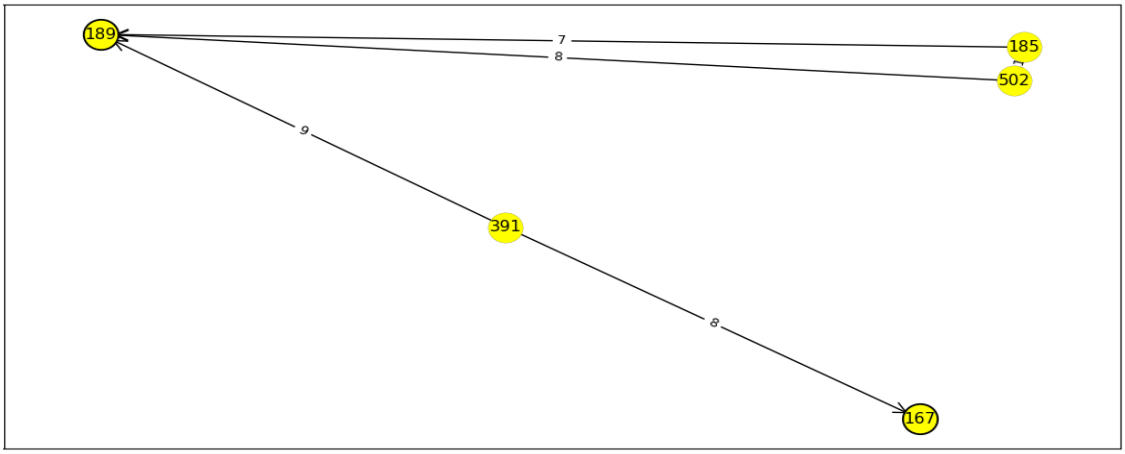


Fig. 46: Component of the upper left network.

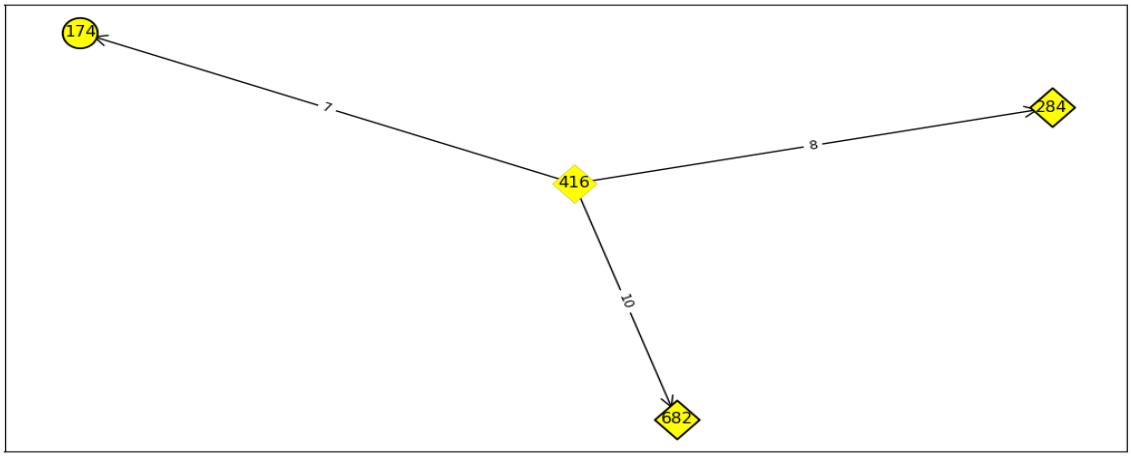


Fig. 47: Component of the upper left network.

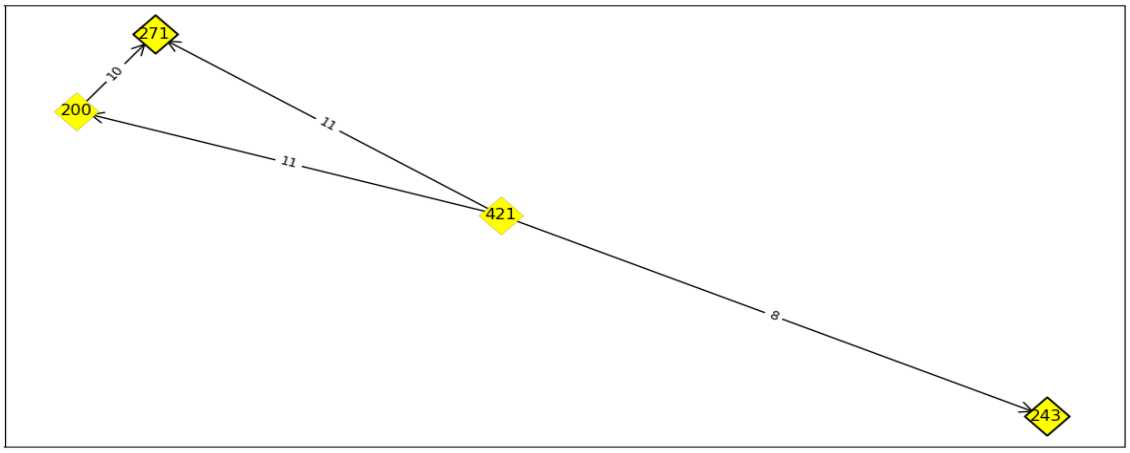


Fig. 48: Component of the upper left network.

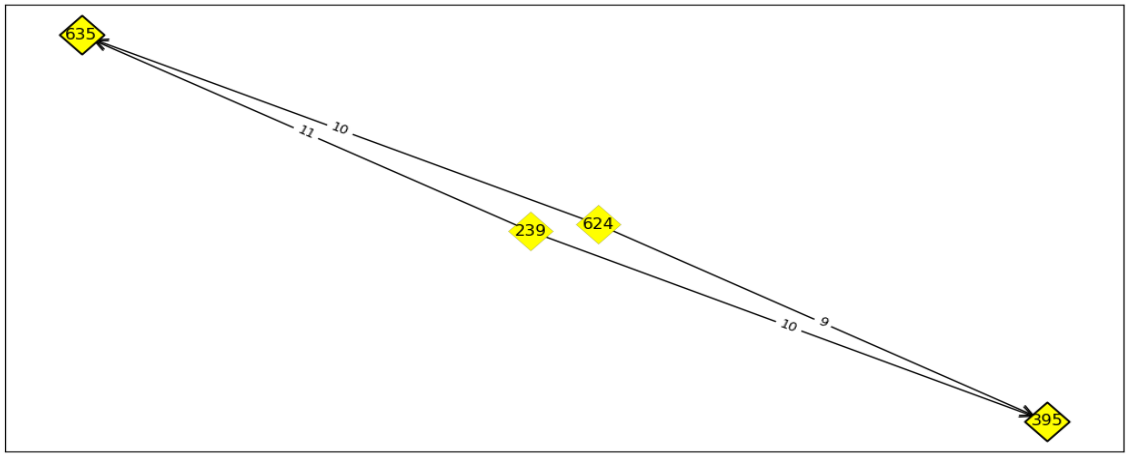
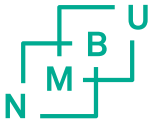


Fig. 49: Component of the upper left network.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway