Learning Latent Customer Representations for Credit Card Fraud Detection

January 5, 2018

Author: Kasper Bouwens

Supervisor University of Amsterdam:
Peter O’Connor

Supervisor University of Oxford:
Luisa Zintgraf
Acknowledgements

First and foremost I would like to thank Luisa Zintgraf and Peter O’connor for supervising this project. Even though none of us were experts on graphical models, we decided to go this route. After a period where I was truly lost in the subject we managed to get on the same page which finally paid off. I feel like both of you exhibited great patience and put a lot of effort in helping me materialise different aspects of the somewhat vague ideas I first had. Without you it truly could not have been realised.

I also want to thank my friends, family and girlfriend. Mostly due to my mental absence, it must not have been easy to deal with me from time to time. In truth, I consider all of you to be my friends and all of you to be by family. Your optimism was an inspiration and kept me going whenever I was feeling pessimistic.
Abstract

A growing amount of consumers are making purchases online. Due to this rise in online retail, online credit card fraud is increasingly becoming a common type of theft. Previously used rule based systems are no longer scalable, because fraudsters can adapt their strategies over time. The advantage of using machine learning is that it does not require an expert to design rules which need to be updated periodically. Furthermore, algorithms can adapt to new fraudulent behaviour by retraining on newer transactions. Nevertheless, fraud detection by means of data mining and machine learning comes with a few challenges as well. The very unbalanced nature of the data and the fact that most payment processing companies only process a fragment of the incoming traffic from merchants, makes it hard to detect reliable patterns. Previously done research has focussed mainly on augmenting the data with useful features in order to improve the detectable patterns. These papers have proven that focussing on customer transaction behavior provides the necessary patterns in order to detect fraudulent behavior. In this thesis we propose several bayesian network models which rely on latent representations of fraudulent transactions, non-fraudulent transactions and customers. These representations are learned using unsupervised learning techniques. We show that the methods proposed in this thesis significantly outperform state-of-the-art models without using elaborate feature engineering strategies. A portion of this thesis focuses on re-implementing two of these feature engineering strategies in order to support this claim. Results from these experiments show that modeling fraudulent and non-fraudulent transactions individually generates the best performance in terms of classification accuracy. In addition, we focus on varying the dimensions of the latent space in order to assess its effect on performance. Our final results show that a higher dimensional latent space does not necessarily improve the performance of our models.
Contents

1 Introduction 4

2 Related Work 5
  2.1 Fraud Detection and Classification ......................... 5
  2.2 Feature Engineering Literature .......................... 7

3 Theoretical Background 8
  3.1 The Expectation Maximization Algorithm ................... 8
  3.2 Probabilistic Graphical Models ........................... 9

4 Problem Statement 10

5 Data 12
  5.1 Preprocessing ........................................ 13
  5.2 Feature Engineering .................................... 13

6 Models 14
  6.1 Model V.0 ............................................. 14
    6.1.1 Prediction with V.0 ............................... 15
  6.2 Model V.1 ............................................. 15
    6.2.1 Prediction with V.1 ............................... 16
  6.3 Model V.2 ............................................. 17
    6.3.1 Prediction with V.2 ............................... 18
  6.4 Model V.3 ............................................. 18
    6.4.1 Prediction with V.3 ............................... 19
  6.5 Parameter Estimation ................................. 19
    6.5.1 Parameter Estimation V.3 ....................... 20

7 Experimental Setup 21
  7.1 Evaluation ........................................... 22

8 Results 24
  8.1 Cluster Quantities .................................... 24
  8.2 Baseline Comparison .................................. 27
  8.3 Comparison with Feature Engineering Strategies .......... 28

9 Conclusion 30

10 Recommendations 31

A Aggregation and Periodic Features 33

B Analysis of Variance (ANOVA) 35
1 Introduction

This thesis studies the detection of online credit card fraud using machine learning techniques. In particular, we use unsupervised learning techniques to learn latent representations that help explain fraudulent transactions. We use these representations in a probabilistic graphical model to detect fraudulent transactions. In this thesis the data is provided by a payment processing company (PPC) that has chosen to remain anonymous. Fraud detection has previously been studied with data provided by banks, and online merchants. In contrast with these data sources, data from payment processing companies can have many sources. To our knowledge, PPC data has not yet been studied in the context of fraud detection. Our objective is to bridge this knowledge gap.

Credit card fraud can be defined as the action of an individual or a group of individuals who use a credit card without the owner’s consent and with no intention of repaying the purchase made (Caneiro et al., 2017). Fraud is a substantial problem for merchants especially for online payments, due to the fact that they are responsible for paying the bill when a fraudster steals goods or services. When a consumer claims that he/she did not receive the products or services requested or if an order was placed by a fraudster, a chargeback can occur. If a merchant cannot prove that he/she provided the goods or services purchased, the money will have to be returned to the consumer’s account and the product might be lost. In addition, a high chargeback rate might lead to chargeback fees imposed on merchants by card associations.

Phua et al. (2010) stated that “it is impossible to be absolutely certain about the legitimacy of and intention behind an application or transaction. Given the reality, the best cost effective option is to tease out possible evidences of fraud from the available data using mathematical algorithms”. For the purpose of this research we will use a combination of supervised and unsupervised learning approaches. An important part of this thesis involves using unsupervised learning techniques in order to model transactions. These models are used in a supervised learning setting. Supervised learning is a branch of machine learning that involves predicting a target variable $\hat{y}$ given an input variable $x$, based on ground truth pairs $(y, x)$. In our case this means that each transaction is labeled with an indicator that says if a transaction is fraudulent or not. Like all data-mining approaches for credit card fraud detection, supervised machine learning comes with a few challenges. The most important challenge is that the data needs to be correctly labeled. In the case of fraud detection, transactions usually get labeled as fraudulent when a customer claims fraud and requests a chargeback. However, customers might also be motivated to report a transaction as being fraudulent for other reasons. Whenever a customer is not satisfied with his/her purchase, a customer can still claim fraud even though no actual fraud occurred. Oftentimes it is hard to prove that a customer might not be telling the truth. Besides these issues, Phua et al. (2010) also point out that the lack of publicly available data and well-researched methods are the two main difficulties for data mining-based fraud detection approaches.

Fortunately, a far greater number of legitimate transactions are made on a daily basis compared to fraudulent transactions. Nevertheless, this also poses a considerable challenge for data scientists trying to invent automated methods for detecting fraud. The main challenge, known as class imbalance, occurs when very few instances of a class (in our case fraud) are present in the data. When predicting whether a set of transactions is
fraudulent or not, a system could perform very well in terms of percentage of correctly predicted fraud values, by always predicting 'non-fraudulent'. Because of this, fraud detection often requires rebalancing of the data. Furthermore, fraudsters are always trying to make their behavior seem legitimate. This means that predefining fraudulent behavior in order to detect it, is not possible. Fraudulent behavior may also depend on the type of merchant that is being targeted. A diamond stealing fraudster might want to make one large purchase where someone stealing from a merchant who sells cheaper goods might want to make a large number of smaller purchases. Due to the highly dynamic behavior of fraudsters, fraud detection systems have to be dynamic as well to be able to adapt to variety of ever evolving fraud patterns.

In this thesis, we propose multiple fraud detection algorithms. As we will demonstrate, we are able to use unsupervised learning techniques in order to model fraudulent and non-fraudulent transactions separately. The models that represent these two classes of transactions are then used in a supervised setting in order to detect fraudulent transactions. Provided that these separate models are learned, we demonstrate that any rebalancing of the data can be avoided. Additionally, we propose a novel approach for incorporating individual client information into the fraud detection procedure.

This document is structured into eight more sections. We start by introducing related work in the credit card fraud detection literature (section 2). Section 3 goes into further detail on the various techniques that are used in this thesis. Following this theoretical explanation we briefly discuss our proposed approach (section 4) and the problems we aim to solve. Section 5 presents a summary the data provided to us and the steps taken in order to prepare the data. Section 6 goes into further detail about the implementation of the proposed models. Section 7 covers the experimental setup and details which methods were implemented to compare our algorithms to. In section 8 we summarize our results and in section 9 we conclude this thesis. Finally, in section 10 we make some suggestions for future work.

2 Related Work

Credit card fraud is an intensively studied data mining application that has a long history of research dedicated to its prevention. The credit card fraud detection literature can be partitioned in various ways. When reading through the literature we found that several papers mainly focus on feature engineering in contrast to others that focus more on the development of novel detection algorithms. This section is intended to provide an overview of previous work done in both areas, starting with the research done in the latter field. Note that similarly to this thesis, some of these papers also describe a feature engineering strategy. However these papers do not state feature engineering as their main focus.

2.1 Fraud Detection and Classification

Over the years several data mining/machine learning approaches have been researched that aim to outperform static rule based approaches in detecting fraudulent credit card transactions. To this end, Maes et al. (2002) compare artificial neural networks and
Bayesian networks. Bayesian networks are a type of probabilistic graphical model that represents the dependence between variables via a directed acyclic graph. These variables are connected via directed edges that represent conditional dependencies. The paper distinguishes two learning tasks: Identifying the topology of a network and learning the numerical parameters (the prior and conditional probabilities). Since this paper focuses on a problem where all variables are observed during training the latter is considered trivial and therefore not the focus of the paper. To learn the network topology, the authors used a global optimization method called STAGE to find the best possible configuration of the network. In contrast to Maes et al. (2002), this thesis explores Bayesian networks in which a portion of the variables is unobserved, making parameter learning more complex. This will be discussed further on in this thesis. As mentioned earlier, the paper also describes experiments with artificial neural networks (ANN). In this work the authors implemented a feed forward multi-layer perceptron. The paper concludes however that they obtain better results by using Bayesian networks. A more detailed description of Bayesian networks is set out in section 3.

The paper by Maes et al. (2002) is somewhat similar to the research conducted by Taniguchi et al. (1998), albeit in a different domain. In their research, they also compared ANNs with Bayesian networks in order to detect communication network fraud. Furthermore they have used unsupervised methods to do probability density estimation. In their research Bayesian networks are also reported to outperform neural networks. Interestingly, the paper states that a combination of unsupervised and supervised techniques could further improve results.

Fu et al. (2016) used a convolutional neural network (CNN) based approach for mining latent fraud patterns. In order to use CNNs for transaction data they use a feature engineering strategy that partitions the data in time windows and calculate the average amount or total amount spent by the customers in each time window. They also introduce the notion of merchant entropy which is a way of calculating the proportion of total money spent at each merchant type. By comparing the entropy of an incoming transaction with the entropy from a given time window they obtain a value which they call trading entropy. Furthermore, they devise a technique for oversampling fraudulent transactions which they call cost-based sampling. The resulting model creates a transaction matrix from all time windows and uses cost-based sampling in order to train their CNN. The paper concludes that the cost-based sampling method effectively combats the class imbalance. Moreover, whether the CNN outperforms other traditional models depends on the degree to which the data is oversampled.

Correa Bahnsen et al. (2013) propose a cost-sensitive comparison measure which weighs the cost of a false negative by the amount of money being transferred. In this thesis we will use this measure to assess the performance of our algorithms as well. A more detailed description can be found in the evaluation section (6.1). Additionally, Correa Bahnsen et al. (2013) use this cost measure to develop a cost-sensitive fraud detection algorithm based on bayes minimum risk. The algorithm quantifies tradeoffs between predicting fraud an non-fraud by using probabilities and the costs associated with each prediction. They report a 23% cost improvement compared to more traditional classifiers. Correa Bahnsen et al. (2015) make use of this cost-sensitive comparison measure in order to develop cost-sensitive decision trees. Where standard decision tree algorithms aim to maximize accuracy, this paper seeks to factor in the cost by using...
cost based impurity measures and a pruning method which aims to minimize costs and uses this as a pruning criterion.

Vlasselaer et al. (2015) developed a fraud detection system that looks at past customer behavior in order to make predictions on future transactions. In order to build a dynamic system that adapts to changing fraud patterns, two sliding windows are used that identify transactions on short and long-term customer behaviors. It also uses two featureization steps. The first step is called the intrinsic feature extraction in which the model compares the current transaction to past transactions that fall within the two windows. The second step is a network-based feature extraction step which utilizes relationships between cardholders and merchants in order to measure the exposure of each entity to fraud.

2.2 Feature Engineering Literature

A method developed by Whitrow et al. (2008) focuses on individual customer spending behavior in order to construct useful features. This feature engineering procedure involves grouping transactions made during different fixed time intervals by the same customer. In this work, transactions are grouped by card or account number, transaction type, merchant group, country or other categories. After grouping these transactions, the total amount spent in each group of transactions are aggregated. The newly computed features are then passed to multiple different machine learning algorithms including a random forest classifier, logistic regression and a support vector machine. The interesting novelty of this approach is that by aggregating transactions over customers, their models do not predict fraud on a transactional level but rather attempt to predict whether a customer account is compromised. It is reported that the random forest classifier consistently performs best on both datasets used in this experiment. The experiments that are performed on the two datasets yield very different results, suggesting that the success of aggregation methods are very case specific.

In addition to aggregating the transactions in order to create features that capture the spending behavioral patterns of customers, Correa Bahnsen et al. (2016) propose a new set of features based on analyzing the periodic transaction behavior of a customer by looking at the time of a transaction and comparing it with a confidence interval. The confidence interval is calculated by looking at past transactions from one account and using the von Mises distribution to define normal customer behavior. The von Mises distribution, used to model the time of a transaction as a periodic variable, is a distribution of a wrapped normal distributed variable across a circle. The rationale for this method is that customers are expected to make transactions around similar hours. So by drawing a distribution over all past transactions from one account one may compare each new transaction with this distribution to see whether it falls within the normal time range the customer is expected to make a purchase. The features derived from these comparisons are expressed as booleans that indicate if a transaction falls within a certain confidence interval or not. In addition, the paper describes a new cost function that takes into account the financial cost of misclassifying a transaction. The cost of a false negative is set to be equal to the amount of the transaction.

The previously mentioned methods do not use data from one single merchant but focus rather on transactions made with multitude of merchants. In Caneiro et al. (2017)
fraud detection for credit card payments is studied for the case when the only data available is data from one single online retail merchant. This means that the transaction history for each customer is incomplete and therefore that fraud detection models require a different set of features to achieve the desired predictive performance. To achieve this, some categorical variables were transformed to reduce the number of categories. For instance, countries were clustered by fraud risk. This fraud risk was calculated by the ratio of fraudulent orders over the total number of orders for each country. In order to obtain some variables that strengthen the predictive power of the models, a binary function was created to represent the degree of similarity of certain pairs of categorical variables like (billing country, shipping country) and (billing country, card country).

3 Theoretical Background

This section provides a high-level overview of some of the techniques employed in this thesis. Namely, it aims to deliver some background knowledge on bayesian networks and the expectation maximization algorithm (EM). In this thesis, we use EM in order to learn separate latent representations for fraudulent and non-fraudulent transactions. These representations are captured in variables that are used in a bayesian network with the purpose of doing inference on the probability of fraud. In general, EM is used to find maximum likelihood (MLE) or maximum a posteriori (MAP) estimates of parameters where a probabilistic model such as a bayesian network depends on unobserved latent variables (for a more general introduction, see Bishop (2006)).

3.1 The Expectation Maximization Algorithm

To illustrate the EM algorithm, we denote the set of all observed data by $X$, and the set of all latent variables by $Z$. The set of all model parameters is denoted by $\theta$. The current estimate for the parameters is denoted $\theta_{t-1}$ and the subsequent estimation, obtained from computing the E and M-step of the EM algorithm is denoted by $\theta_t$. The log likelihood function is given by

$$\ln p(X \mid \theta) = \ln \sum_Z p(X, Z \mid \theta).$$

(1)

In this example, we assume $Z$ to be discrete. If $Z$ were continuous, we would have to take the integral over $Z$ instead of the sum. Since $Z$ is unobserved, we cannot use the complete-data log likelihood to estimate $\theta$. As a substitute, in the the E-step we consider its expected value under the posterior distribution of the latent variable. This expectation $Q(\theta, \theta^{t-1})$ is given by

$$Q(\theta, \theta^{t-1}) = \sum_Z p(Z \mid X, \theta^{t-1}) \ln p(X, Z \mid \theta)$$

(2)

In the following M-step we find the parameters that maximize this expected value:

$$\theta = \arg \max_{\theta} Q(\theta \mid \theta^{t-1}).$$

(3)
By repeating these two steps the algorithm improves its estimation of \( \ln p(X \mid \theta) \) by improving \( Q(\theta \mid \theta^{t-1}) \). Formal proofs have been drafted, proving that the log-likelihood \( \ln p(X \mid \theta) \) does indeed increase. Generally, the algorithm is used for data clustering in order to approximate the true distribution over latent parameters. Since we do not know what these parameters represent exactly, we denote them simply as clusters. Subsequently, we will mutually use the terms latent variables and clusters.

### 3.2 Probabilistic Graphical Models

A probabilistic graphical model is a graph that contains nodes which represent random variables and edges which symbolize probabilistic relations between the variables. Bayesian networks are directed probabilistic graphical models where each edge has a certain direction symbolized by an arrow. A directed edge from node \( A \) going into \( B \) is used to denote a conditional dependence that can be read as: if \( A \) then \( B \) with associated conditional probability \( p(B \mid A) \).

An often used example to illustrate the usage of bayesian networks is the sprinkler network (Cooney, 2015). The sprinkler network models a scenario in which there are three boolean variables: \( R \) represents whether or not it is raining, \( S \) represents if the lawn sprinkler is turned on and \( G \) represents whether the grass is wet. The joint probability distribution associated with these variables is \( p(R, S, G) \). By using the product rule we can rewrite the joint distribution in the form:

\[
p(R, S, G) = p(G \mid R, S)p(R, S).
\] (4)

By reapplying the product rule on the second term on the right of the right hand side of the equation we get:

\[
p(R, S, G) = p(G \mid R, S)p(S \mid R)p(R).
\] (5)

The first term on the right hand side indicates that the probability of the grass being wet is dependent on the probability of rain and the probability of the sprinkler being turned on. The middle term denotes that the chances of the sprinkler being turned is dependent on the probability of rain. In this example the left-hand side of the second equation is symmetrical with respect to the variables \( R, S, G \) while the right-hand side is not. We have made an explicit choice to decompose the probabilities in a particular order \( R, S, G \) which represents our sprinkler scenario shown in Figure 1. Any other decomposition of the joint probability would have resulted in a different scenario with a different graphical representation.

In this thesis we use a similar decomposed joint probability as used in the sprinkler network scenario to model credit card fraud. To illustrate this, it is useful to make use of three variables to discuss the fraud detection scenario: \( F \) indicates whether or not a fraud was committed, and \( X \) represents a transaction. Besides these variables we use \( C \) to denote a distribution over our latent variables (i.e. latent clusters). In order to have an idea of what \( C \) means, we shall think of \( C \) as some underlying process that is unobserved, but causes our transactions to be fraudulent or non-fraudulent. Similar to the sprinkler network, we can now decompose the joint probability \( p(F, C, X) \) into an intuitive network of dependencies. As it turns out, we can use \( F \) and \( C \) to create a framework...
in which the conditional probability \( p(X|C, F) \) combines two separate models. Specifically, we assume that the probability of a transaction is simultaneously influenced by the probability of fraud and some unobserved process \( C \). For this example, let us assume that the distribution \( C \) is obtained from all non-fraudulent transactions. We can thus assume that \( C \) models the underlying mechanics behind non-fraudulent behavior and that the probability of \( C \) is marginally independent of \( F \). If hypothetically, we were to know that a particular transaction was fraudulent, we could infer that the probability that \( X \) was caused by our non-fraudulent variable \( C \) would decrease. Correspondingly, if the probability of \( X \) being explained by \( F \) would increase we could infer that the probability of \( X \) given non-fraudulent behavior would decrease.

In our example the probabilities \( p(X|C) \) and \( p(X|F) \) are negatively correlated. In general however, we can state that \( C \) and \( F \) are marginally independent but conditionally dependent. Once these models are put together, we can take advantage of this conditional dependence between \( F \) and \( C \) to infer the probability of one, if the other is known. This type of probabilistic reasoning is known as “explaining away”. Furthermore, by extending the network to also model each individual customer, the model becomes more adapt at discriminating between normal transaction patterns and possibly fraudulent ones. This thesis proposes three models that have been implemented in an attempt to study these claims. Further details of the network topologies shall be discussed in more detail in section 5.

4 Problem Statement

An important objective of this research is to bridge the knowledge gap that exists within the fraud detection literature when it comes to PPCs. PPCs process payments from a large number of different merchants but do not necessarily process all payments for each merchant. Therefore, previously studied techniques such as periodic features cannot be blindly relied on. In fact, most historical data provided by PPCs is incomplete, due to
the competitive market in which large merchants often hire multiple PPCs to balance the transaction load. This incompleteness requires novel approaches to accurately make use of this fragmentary data in order to improve fraud detection. To this end, we first learn latent representations for fraudulent and non-fraudulent transactions. We then extend upon this idea by incorporating the fact that we know which customers made the transactions. By using unsupervised learning, we aim to model individual customers by making use of the incomplete historical data we have.

In this thesis we have chosen to utilize bayesian networks because they offer a flexible framework where different aspects of a transaction can be modelled (customers, latent variables etc.). Furthermore, by using a probabilistic model we avoid the typical problems that arise when feature engineering is applied. Namely, that hardcoded features can sometimes be costly to compute and that there are no best practices when it comes to deciding on which ones to use. In order to examine these claims, we formulated multiple questions which we aim to answer in this thesis.

**Q1** Can we increase the performance of a bayesian network model by separately modelling fraudulent and non-fraudulent transactions? And if so, how does this compare to some of the current state-of-the-art models?

Due to the problems that arise when dealing with severe class imbalance, our hypothesis is that we can improve fraud detection scores with separate models that learn representations for fraudulent and non-fraudulent transactions because they suffer less from overfitting on the dominant class. Both models are essentially ignorant of one another and only become dependent once conditioned on a particular transaction.

**Q2** How does the number of dimensions in the latent space affect the overall performance of the models?

Can we represent valuable information in a way that is fine-grained enough to capture different relevant aspects of a transaction and make it expressive enough to use for prediction? As explained in section 3, the method used in this thesis for learning these representations is the unsupervised soft clustering technique EM which provides us with a representation in the form of some distribution over clusters. What these clusters mean is not clear a priori. Therefore our hypothesis is that there is middle ground between having too little clusters in which case some information will be lost, versus having too many clusters in which case clusters essentially lose their meaning.

**Q3** Is it possible to create a model that incorporates individual customer transaction behavior and how would it affect the overall performance of our model?

By conditioning a transaction on its own customer history we want to build a model that explains a transaction by looking at previous transactions. Model V.3, introduced in section 6.4, was built in an attempt to do this. Observing whether a transaction falls within a time interval where the customer is expected to make transactions, can provide fraud detection algorithms with valuable information. This has been demonstrated by Correa Bahnsen et al. (2016). In contrast, we aim
to observe and represent client behaviour in general, without using a feature engineering strategy to do so. It is especially important to mention that this thesis builds upon the idea introduced by Correa Bahnsen et al. (2016), of observing individual customer transaction patterns. However, since the data used in this thesis does not contain complete customer transaction histories, using their approach could lead to inaccurate but hard cutoffs. It is for this reason that we have chosen a probabilistic approach which combines soft-clustering and probabilistic inference to tackle this problem.

Q4 How do the probabilistic models proposed in this thesis compare to feature engineering strategies such as aggregation and periodic features?

One of the goals of this thesis is to assess whether or not a good performance can be achieved without much feature engineering. As mentioned earlier, Correa Bahnsen et al. (2016) aggregated features across customers in order to significantly boost the performance of several algorithms. Furthermore, they used periodic features in order to analyze the transaction times of individual customers. Not only did they demonstrate that both strategies improved the prediction accuracy of their models, they also showed that the performance was even greater when used together. However, the use of these techniques comes with significant computational costs due to the fact that the data has to be rigorously preprocessed. In order to analyse whether feature engineering outperforms conditioning latent variables on customers, both feature engineering strategies have been reimplemented and evaluated.

By answering these questions we contribute to the existing fraud detection literature in multiple ways. Firstly, we show how fraudulent and non-fraudulent transactions can be modeled independently, by employing the well-known EM algorithm. More specifically, we prove that a significantly larger amount of money can be saved by using these representations in a probabilistic model. Building on these findings, we extend our probabilistic models which contain latent representations by factoring in the customer that made the transaction. Our approach shows that despite the fact that little may be known about a customer, it is useful to condition the probability of a transaction on the customer involved.

5 Data

Aside from providing some information on the raw data we received from the PPC, this section is also intended to describe how the data was preprocessed. Carefully constructing a dataset is a vital part of this type of data mining project, partly because raw transaction contain little information. Furthermore, all categorical variables need to be transformed into numerical ones in order to make them compatible with our algorithms. For instance, each currency type was converted by tuning it into a one-hot representation.

The raw data provided to us consists of 90,556 transactions made over a twelve month period. Additionally, a chargeback report was provided, indicating for which
transactions a chargeback occurred and what the motivation for the chargeback was. The dataset contains 1,163 transactions that were reported as fraud. The transaction amounts range from one cent to approximately 7,500 Dollars. The total amount of money spent in legitimate transactions equals 28,116,549.78 Dollars, whereas amount spent in fraudulent transactions equals 76,786.50 Dollars. Six different currency types are present in the data set, 25 unique merchants, the country where the card was issued, the country where the purchase was made and 54,762 unique credit cards.

5.1 Preprocessing

The first and most important step in preprocessing was to define classes for each data point. Since we were dealing with a binary classification problem, we used the chargeback reports to label each transaction listed in the report as a fraud. Customers could mislead merchants by claiming fraud when they were not satisfied with the goods received. However, since we had no way to distinguish these transactions from genuine frauds we decided to ignore this. Similarly, because unreported frauds could not be taken into account, all transactions not reported were marked as non-frauds.

A crucial issue that needed to be addressed was that each transaction time was recorded in pacific time. Correa Bahnsen et al. (2016) note that transaction times are an essential part in characterising customer spending behavior. Hence we converted pacific times to local times to see at what hour of the day a customer had made the transaction. Next, we converted all amounts to US Dollar in order to accurately compare transaction amounts. Taking example after Caneiro et al. (2017) we used the most recent 20% of transactions for testing. As they state: "Dividing by date was due to the fact that no pattern of fraud could be learned a priori from the date, but fraud behavior may change". The authors further argue that by doing this, the performance estimations should be more conservative as more recent orders should better emulate new orders.

For our data this split resulted in a training set containing 72,445 transactions with 1,051 frauds and a testing set containing 18,111 transactions with 112 frauds.

5.2 Feature Engineering

A number of straightforward transformations were made to convert all variables to numerical variables. The phone numbers where replaced by booleans indicating whether a valid phone number was written down. In addition, we made one-hot encodings for both the currencies and the merchants. One-hot encodings are binary vectors with columns for each of the categories of our original variable. Since both did not have many categories, the resulting one-hot vectors where relatively dense.

Inspired by Caneiro et al. (2017) we also constructed a few abstract features to add more useful information to our transactions in order to train the models. A feature that was constructed through such abstraction was used to compare the user’s email address and full name. This was done by computing the n-gram similarity which outputs a continuous value in [0,1]. To encode the country and GeoCode, the fraud risks for each where calculated per country by dividing the number of fraudulent transactions by the total number of transactions:
Fraud Ratio per country = \frac{\text{Fraudulent Transactions per country}}{\text{Total Transactions}}. \quad (6)

Furthermore, the time difference was measured between the current transaction and the first transaction by the same card at the same merchant. And finally, the number of cards used per full name, previous to the current transaction was added to each transaction. Table 1 shows the final list of features used in our experiments.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card</td>
<td>string</td>
<td>Card number. Used to identify a cardholder</td>
</tr>
<tr>
<td>Phone</td>
<td>boolean</td>
<td>A boolean indicating whether a valid phone number was given.</td>
</tr>
<tr>
<td>sin time</td>
<td>float</td>
<td>The sinus of a transaction time on the unit circle, scaled to 24 hours.</td>
</tr>
<tr>
<td>cos time</td>
<td>float</td>
<td>The sinus of a transaction time on the unit circle, scaled to 24 hours.</td>
</tr>
<tr>
<td>week day</td>
<td>int</td>
<td>The number of the day of the week {0, \ldots, 6}.</td>
</tr>
<tr>
<td>month</td>
<td>int</td>
<td>The number of the day of the month {0, \ldots, 11}.</td>
</tr>
<tr>
<td>Dev Country</td>
<td>boolean</td>
<td>A boolean indicating whether or not a country and geocode match.</td>
</tr>
<tr>
<td>Dev Name Mail</td>
<td>float</td>
<td>The n-gram similarity between the person's email and full name.</td>
</tr>
<tr>
<td>TimeSinceFirstOrder</td>
<td>float</td>
<td>The time that expired between now and the first transaction.</td>
</tr>
<tr>
<td>Cards Used</td>
<td>int</td>
<td>How many cards were used by the same full name.</td>
</tr>
<tr>
<td>Country_risk</td>
<td>float</td>
<td>Risk country (x = (#\text{frauds committed in country }x) / (#\text{frauds committed}))</td>
</tr>
<tr>
<td>Geocode_risk</td>
<td>float</td>
<td>Risk geocode (x = (#\text{frauds committed with geocode }x) / (#\text{frauds committed}))</td>
</tr>
<tr>
<td>Merchant</td>
<td>one-hot</td>
<td>One-hot vector indicating which merchant made the sale.</td>
</tr>
<tr>
<td>Currency</td>
<td>one-hot</td>
<td>One-hot vector indicating which currency was used.</td>
</tr>
</tbody>
</table>

Table 1: Table containing all input variables

6 Models

As previously discussed, the aim of this research is to explore the use of latent variables in order to explain fraudulent and non-fraudulent transactions separately. In this section we describe the different variants of the model that were implemented ranging from the simplest version (V.0) where no latent variables were used to the most complex version (V.3) where we aim to learn individual customer representations. For each model we outline its decomposition and explain how it can be used to make predictions on the probability of a new transaction being fraudulent.

6.1 Model V.0

For data set \(\mathcal{D} = \{(\mathbf{X}_n, F_n)\}_{n=1}^N\) with transaction vectors \(\mathbf{X}_n = (x_{n1}, \ldots, x_{nD}) \in \mathbb{R}^D\) and fraud labels \(F_n \in \{0, 1\}\) we have a set of distributions \(p(F; \theta_f)\) with \(f \in |F|\), and \(p(\mathbf{X}; \theta_x)\) which are the observed marginal probabilities of \(\mathbf{X}\) and \(F\). \(N \in \mathbb{N}\) stands for the total number of transactions with \(n \in (1, \ldots, N)\) whereas \(D \in \mathbb{N}\) is used to denote the dimensions of vector \(\mathbf{X}_n\) with \(d \in (1, \ldots, D)\).

With these variables we implemented a straightforward probabilistic classifier which uses no latent variables, known as a naive bayes classifier. A naive bayes classifier is a probabilistic model that assumes complete independence between input features. The model can be summarised by saying that the probability of a fraud label \(F\) is conditioned on the respective transaction vector \(\mathbf{X}\). Since all variables in \(\mathbf{X}\) are independent we can write
$$p(X \mid F; \theta_{f,x}) = \prod_{d=1}^{D} p(x_d \mid F; \theta_{f,x}).$$  (7)

$$p(F \mid X) = \sum_F p(F) p(X \mid F; \theta_{f,x}),$$  (8)

where the probability $p(X)$ can be viewed as a normalizing constant once $p(X)$ is known.

Finally, since each misclassification error has different implications we use a cost-sensitive class boundary inspired by Elkan (2001). Whenever our prediction exceeds this threshold, we predict fraud. In this thesis we have chosen this class boundary to be the cost of a true positive divided by the cost of a false negative,

$$\text{threshold} = \frac{C_{FP}}{C_{FN}}.$$  (9)

Table 2 (Section 7.1) gives an overview of the costs associated with each type of error. Each subsequent model described in this section uses this class boundary in order to factor in the cost at prediction time.

### 6.2 Model V.1

The approach taken in model V.1 relies on the assumption that there is some unobserved variable $C_{\neg F}$ that explains the non-fraudulent transaction variables in $X$ whereas the fraudulent transactions are explained by the observing $X$ when $F = 1$. In order to represent this supplementary unobserved variable we use $C_{\neg F} = (c_1, \ldots, c_K)$ where $K \in \mathbb{N}$ indicates the length of vector $C_{\neg F}$. The decomposition of the joint probability (10) is, to a certain degree, similar to the sprinkler network decomposition discussed earlier (equation 5). The decomposition of the model V.1 is as follows:
\[ p(F, C_{\neg F}, X) = p(X | C_{\neg F}, F)p(C_{\neg F}, F). \] (10)

The probability \( p(X | F, C_{\neg F}) \) is defined as a switch between two separate models. Namely the probabilities of \( X \) trained on all fraudulent transactions and the conditional probability on \( C_{\neg F} \) obtained from all non-fraudulent transactions.

\[ p(X | F, C_{\neg F}) = \begin{cases} \hat{p}(X | C_{\neg F}; \theta_c, x) & \text{if } F = 0 \\ \hat{p}(X; \theta_x) & \text{if } F = 1 \end{cases} \] (11)

Due to the independence assumption we can write the probability of \( X \) conditioned on \( C_{\neg F} \) as:

\[ \hat{p}(X | C_{\neg F}; \theta_{c,x}) = \prod_{d=1}^{D} \hat{p}(x_d | C_{\neg F}; \theta_{c,x}) \] (12)

Note that in the switch (11), the two probabilities are marked with a hat in order to denote that they represent separate models. The resulting network is displayed in Figure 3.

We can see in Figure 3 that the decomposition results in a graph where each transaction is simultaneously explained by the fraud variable and our latent variable \( C_{\neg F} \) where \( C_{\neg F} \) is only trained on non-fraudulent transactions. The intuition behind this approach is that fraudulent and non-fraudulent transactions are explained by different factors which we try to capture separately in \( F \) (fraud or not fraud) and \( C_{\neg F} \) (non-fraudulent customer behavior).

![Bayesian Network V.1](image)

**Figure 3: Bayesian Network V.1**

### 6.2.1 Prediction with V.1

In order to predict whether a transaction is a fraudulent one or not we want to compute \( p(F | X) \). Equations 10 and 11 already demonstrated how to compute the joint probability. By using Bayes rule and marginalizing out \( F \) and \( C_{\neg F} \) the joint probability can be used to obtain
\[ p(F \mid X) = \frac{p(F, X)}{p(X)} = \frac{\sum_{c \neq F} p(F, c_F, X)}{\sum_{c \neq F} \sum_f p(F, c_F, X)} = \frac{\sum_{c \neq F} p(X \mid c_F, F) p(c_F) p(F)}{\sum_{c \neq F \in \text{K}} \sum_f p(X \mid c_F, F) p(c_F, F)} \]

(13)

6.3 Model V.2

The third model we implemented relies on two latent variables \( C_{\neq F} \) and \( C_F \) where we aim to explain non-fraudulent transactions by using \( C_{\neq F} \) and fraudulent transactions by using \( C_F \). Evidently, \( C_{\neq F} \) is trained all non-fraudulent transactions and \( C_F \) is trained on the fraudulent ones. We make a slight alteration in order to model the joint probability

\[ p(F, c_F, c_{\neq F}, X) = p(X \mid c_F, c_{\neq F}, F) p(c_F, c_{\neq F}, F). \]

(14)

Where we define the conditional probability as a similar switch to (11):

\[ p(X \mid F, c_F, c_{\neq F}) = \begin{cases} \hat{p}(X \mid c_{\neq F}; \theta_{c_{\neq F}, x}) & \text{if } F = 0 \\ \hat{p}(X \mid c_F; \theta_{c_F, x}) & \text{if } F = 1 \end{cases} \]

(15)

Note that the decomposition of this model is almost identical to the previous model. The main difference in that we have added another variable \( C_F \) which is learned from fraudulent transactions. \( C_F \) allows us to add another model to our network. Once more, due to the independence assumption on \( X \) this model equates to

\[ \hat{p}(X \mid c_F; \theta_{c_F, x}) = \prod_{d=1}^{D} \hat{p}(x_d \mid C_F; \theta_{c_F, x}). \]

(16)

Figure 4 shows the graphical representation of this model.

![Bayesian Network V.2](image-url)
6.3.1 Prediction with V.2

Prediction with model V.2 is almost identical to the procedure used in V.1. Since we have made a slight alteration in order to model the joint probability $p(F, C_F, C_{\neg F}, X)$, the model has one extra variable that needs to be marginalized in order to compute $p(F | X)$. The resulting equation becomes

$$p(F | X) = \frac{p(F, X)}{p(X)} = \frac{\sum_{c_F} \sum_{c_{\neg F}} p(F, C_F, C_{\neg F}, X)}{\sum_{c_F} \sum_{c_{\neg F}} p(F, C_F, C_{\neg F}, X)} = \frac{\sum_{c_F} \sum_{c_{\neg F}} p(X | C_F, C_{\neg F}, F) p(C_F, C_{\neg F}, F)}{\sum_{c_F} \sum_{c_{\neg F}} \sum_f p(X | C_F, C_{\neg F}, F) p(C_F, C_{\neg F}, F)}.$$ (17)

6.4 Model V.3

With model V.3, the aim is to incorporate customer information without having to rely on feature engineering. By soft-clustering transactions and customers instead of using hard confidence intervals, this model works towards making use of transaction data without having the need for complete customer transaction histories. Previously we aimed to model fraudulent and non-fraudulent transactions individually. In this model our latent variables $C$ are trained on all transactions.

The model is similar to V.1 with an added variable $Q$ which contains the customer identification of transaction $n$, $id^{(n)} \in ids$. Variable $Q$ makes sure that our latent variable $C$ is always conditioned on the customer making the transaction. Therefore, the joint distribution with the customer factored in decomposes to

$$p(X, C, F, Q) = p(X | C, F) p(C | Q) p(Q) p(F).$$ (18)

The decomposition of this model is almost identical to the model V.1. The main difference is that we have added another variable $Q$ along with the conditional probability $p(C | Q)$. Since the model does not know all customers a priori, it has to account for scenarios in which it will eventually come across new customer IDs. Thus, the conditional probability of the latent variable $C$ given the customer $Q$ becomes

$$p(C = c | Q = q; \theta_{c,q}) = \theta_{c,q} \text{ if } q \in ids \text{ else } \frac{1}{|ids|}.$$ (19)

As can be seen in this equation, the model uses its learned parameters whenever a customer is known. Whenever it observes a new customer the conditional probability becomes $U \in \mathbb{R}^K$ which denotes the uniform distribution of over $C$. Figure 5 shows the graphical representation of this model.
6.4.1 Prediction with V.3

Prediction with model V.3 predominantly involves applying the same marginalisations as in previously introduced models and using bayes rule. Nevertheless, with this model the goal is to infer \( p(F | X, Q) \). Instead of marginalizing over \( Q \) we know the ID of the customer for each transaction and so we can use that to make predictions. Globally, the procedure comes down to

\[
p(F | X, Q = q) = \frac{p(F, X, Q = q)}{\sum_c \sum_f p(F, C, X, Q = q)}.
\]

(20)

6.5 Parameter Estimation

In section 3 we briefly introduced the EM algorithm. As V.0 incorporates observed variables only, EM is not used for this model. Instead, maximum likelihood estimation can be done to directly infer the full-data likelihood. For all subsequent models introduced in this thesis, we use the EM algorithm in order to estimate the latent variables \( C \) on the complete training set, \( C_F \) on the fraudulent transactions and \( C_{\neg F} \) on the non-fraudulent transactions. This section is intended to give a more detailed explanation on how the EM algorithm is used to estimate \( C \), \( C_{\neg F} \) and \( C_F \). For the sake of avoiding redundancy, we have only used \( C_{\neg F} \) in our explanation. However, \( C_F \) can freely be replaced by \( C_{\neg F} \) since both are learned in identical fashion. Model V.3 employs a slight variation in order to factor in customer identification during training. Therefore we shall explain the estimation of \( C \) separately. For V.1 and V.2 the EM algorithm is utilized in order to estimate the marginal probability \( p(C_F; \theta_c) \) and the conditional probability \( p(X | C; \theta_{c,x}) \). Where \( \theta_c[k] = p(c_k) \) and \( \theta_{c,x}[d,k] = p(x_d | c_k) \). As explained earlier, we use integer \( K \) to denote the number of clusters. \( T \) is used to denote the number of iterations of the algorithm. The algorithm starts by initializing \( \theta_c^0 \) and \( \theta_{c,x}^0 \) as some initial variables (e.g., random values). These variables must satisfy the constraints:

- \( \theta_c^0[k] \geq 0 \) for all \( k \in \{1 \ldots K\} \) and \( \sum_{k=1}^{K} \theta_c^0(k) = 1 \).
- For all \( k, d \), \( \theta_{c,x}^0[d,k] \geq 0 \). For all \( k \in \{1 \ldots K\} \), for all \( d \) in \( \{1 \ldots D\} \), \( \sum_{d=1}^{D} \theta_{c,x}^0 = 1 \).
Then, for \( t = 1 \ldots T \):

1. For \( n = 1 \ldots N \), for \( k = 1 \ldots K \):

\[
\delta(k \mid n) = p(c_k \mid X^{(i)}; \theta^{(t-1)}_{\ell,c}) = \frac{\theta^{(t-1)}_{\ell,c}[k] \prod_{d=1}^{D} \theta^{(t-1)}_{\ell,c,d}[d,k]}{\sum_{k=1}^{K} \theta^{(t-1)}_{\ell,c}[k] \prod_{d=1}^{D} \theta^{(t-1)}_{\ell,c,d}[d,k]}
\]

(21)

2. Then the new parameter values become:

\[
\theta'_{\ell,c} = \frac{1}{n} \sum_{n=1}^{N} \delta(k \mid n) \\
\theta'_{\ell,c,d} = \frac{\sum_{p,q} \delta(k \mid n)}{\sum_{n} \delta(k \mid n)}
\]

(22a)

(22b)

For completeness, it is worth noting that we modified equation (21) to use log probabilities. This was purely done for computational stability and is by no means a mandatory aspect of the algorithm. For similar reasons we also used laplace smoothing with smoothing parameter \( \alpha = 0.0001 \) in order to prevent zero probabilities.

The supervised portion of this learning algorithm comes into play when estimating \( p(X; \theta) \). Since \( X \) is observed we can use maximum likelihood estimation (MLE) to estimate the probability \( p(X) \) for all fraudulent transactions. In this thesis, we model all our variables to be discrete random variables in which case MLE becomes a straightforward averaging procedure

\[
\theta'[d] = \frac{1}{n} \sum_{n=1}^{N} x_{d,n}.
\]

(23)

### 6.5.1 Parameter Estimation V.3

In order to estimate the parameters in model V.3, we used the decomposition in (18) to directly solve for \( p(C \mid X, F, Q) \). In the expectation step we compute the expected value of the likelihood function, with respect to the conditional distribution of \( C \) given \( F \), \( Q \) and \( X \), under the previous estimate of the parameters \( \theta^{t-1}_{\ell,c,f} \) and \( \theta^{t-1}_{\ell,c} \).

\[
\delta = \frac{p(C \mid X, F, Q)}{\sum_{c} p(C \mid X, F, Q)} \\
= \frac{p(C \mid X, F; \theta^{t-1}_{\ell,c, f}) p(C \mid Q; \theta^{t-1}_{\ell,c}) p(F) p(Q)}{\sum_{c} p(C \mid X, F; \theta^{t-1}_{\ell,c, f}) p(C \mid Q; \theta^{t-1}_{\ell,c}) p(F) p(Q)}
\]

(24)

\[
= \frac{\theta^{t-1}_{\ell,c, f}[k, id(n)] \prod_{d=1}^{D} \theta^{t-1}_{\ell,c, d}[d, k, f]}{\sum_{k=1}^{K} \theta^{t-1}_{\ell,c, f}[k, id(n)] \prod_{d=1}^{D} \theta^{t-1}_{\ell,c, d}[d, k, f]}
\]

Since \( Q \) and \( F \) are observed, \( p(Q) \) and \( p(F) \) can also be observed a priori. Thus, in the M step we update our parameters \( \theta^{t-1}_{\ell,c, f} \) and \( \theta^{t-1}_{\ell,c} \) in order to maximize the expected value of the likelihood function.
7 Experimental Setup

In order to answer the research questions asked earlier in this thesis, multiple experiments needed to be performed. Specifically two experiments were performed in order to answer the following questions:

Q1 Can we increase the performance of a bayesian network model by separately modelling fraudulent and non-fraudulent transactions? And if so, how does this compare to some of the current state-of-the-art models?

Q2 How does the number of dimensions in the latent space affect the overall performance of the models?

Q3 Is it possible to create a model that incorporates individual customer transaction behavior and how would it affect the overall performance of our model?

Q4 How do the probabilistic models proposed in this thesis compare to feature engineering strategies such as aggregation and periodic features?

In view of research Q1, we computed the results from model V.1 and V.2. Since all models that used latent variables produced varying scores, we chose to compare the best test scores for each number of clusters $K$. Since this required that we also compute scores for model V.3, it also gave us the opportunity to assess Q3.

The second research question was answered by systematically varying the number of dimensions of the latent variables. This was done for each model with cluster quantity $K$ ranging from two to six clusters. Since our implementation of the EM algorithm was not always guaranteed to converge to the same log likelihood due to it being initialized randomly, we decided to run each model with cluster size $K$ one hundred times in order to average over results. By doing this we can statistically substantiate or reject the hypothesis that changing the number of dimensions has an effect on our scores. Furthermore, executing one hundred runs for each model also provided the optimal results we needed in order to answer Q1.

One key issue within the fraud detection research domain is that there are no standardized data sets to test on and compare results. In order to create some sort of baseline we chose to compare our models to the best performing cost-sensitive models also used in Correa Bahn sen et al. (2016). These are cost-sensitive logistic regression (CSLR) and cost-sensitive decision trees (CSDT). These models were implemented using the python library (Correa Bahn sen (2016) CostCla. Version 0.5. Jan. 29, 2016. URL:
http://albahnsen.com/CostSensitiveClassification), which was created by one of the authors. For completeness we also compared our models to cost-insensitive logistic regression (LR) and cost-insensitive random forest (RF). Each model was implemented using hyper parameters reported in the paper. Due to the fact that the results for the baseline models can vary, we chose to execute 100 runs for each model and picked the best results for comparison.

Lastly, to answer Q4, two feature engineering strategies, used by Correa Bahnsen et al. (2016) where applied to the data. Namely, aggregation and periodic features. In order to further assess the quality of the models V.1, V.2, and V.3, we compared them with the CSLR, CSRF, LR and RF models, trained on data sets that were augmented with these features. As previous experiments, each model was trained and tested 100 times in order to compute the optimal results, which we used for comparison. Due to the fact that both feature aggregation and periodic features have been discussed in section 2.2, we shall remain brief in our explanation here.

The aggregation features employed by Whitrow et al. (2008) involved counting how many times a feature from the current transaction could be observed for the same customer in several predefined time intervals. Additionally, the amount of money spent in the transactions that exhibited these same features was aggregated. Additionally, we used the method employed by Correa Bahnsen et al. (2016) who expanded upon this approach through the aggregation of feature co-occurrences. This meant for instance, that if the current transaction was made in France and was made at merchant A, we would count the number of transaction made by the same customer that exhibited these same traits. Furthermore, the total amount of money spent in these past transactions would also be aggregated and used to augment the current transaction.

The periodic features where constructed by using the set of transactions made by the same customer during a number of predefined time intervals. The von Mises distribution was then fitted on these transactions in order to compute a confidence interval. Afterwards we assessed whether the current transaction time occurred within the confidence interval in order to create a boolean feature. More detail on the implementation of both the aggregation features and the periodic features is included in Appendix A.

7.1 Evaluation

There are multiple methods for scoring the performance of a binary classification algorithm. First, we may simply look at the ratio of correctly predicted transactions. Due to the severe class imbalance mentioned earlier, this will always produce very high scores even if zero true positives are produced. In other words, our system can do well by always predicting 'non-fraud'. Clearly, this is not a good evaluation metric.

A commonly used way of scoring a binary classification algorithm is the F₁ score. An advantage of this measure is that it takes both recall and precision into account. The F₁ is defined as the harmonic average between precision and recall (26) which becomes 1 in case of perfect precision and perfect recall and zero in case of zero precision and zero recall.

\[
F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]  \hspace{1cm} (26)
A disadvantage of this method is that it does not take into account how accurate the system predicts negative data points. Due to the fact that the data is severely unbalanced, \( F_1 \) might not be very indicative of the overall performance of the model. Another performance measure which is better suited for the problem is the balanced classification rate (BCR), shown in equation 27. The advantage of using BCR is that it takes both the recall for positives and negatives into account.

\[
BCR = \frac{1}{2}(\text{recall} + \text{specificity}).
\]  

(27)

By looking at both positive and negative recall, BCR provides additional insight into the overall performance of the classifier.

Lastly, a useful metric for the fraud detection problem is to take into account the amount of money that is actually saved by a particular algorithm. As previously stated, Correa Bahnsen et al. (2015) implemented this measure and even optimized their algorithms with respect to a cost function that took the amount of money lost in a transaction into account. The monetary cost was factored into the evaluation function by using the cost matrix shown in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>Actual positive</th>
<th>Actual negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicted positive</td>
<td>( C_{TP_n} = Cost_a )</td>
<td>( C_{FP_n} = Cost_a )</td>
</tr>
<tr>
<td>predicted negative</td>
<td>( C_{FN_n} = Amt_i )</td>
<td>( C_{TN_n} = 0 )</td>
</tr>
</tbody>
</table>

Table 2: Credit card fraud cost matrix (Correa Bahnsen et al., 2013)

According to this table all predicted positives (whether actually true or false) are associated with an administrative cost \( Cost_a \). This administrative cost is a PPC dependent cost for analyzing a transaction and contacting the card holder. The cost matrix also defines the cost of a false negative to be the amount \( C_{FN_n} = Amt_i \) of the transaction \( n \) and the cost of a true negative 0. The cost of is calculated by

\[
Cost = \sum_{n=1}^{N} y_n(1 - c_n)Amt_n + c_nCost_a.
\]  

(28)

Where \( y_n \in \{0, 1\} \) denotes the class label and \( c_n \) denotes the predicted label for transaction \( n \). With this equation, we can simply calculate the cost of using no algorithm which is defined as

\[
Cost_L = \min\{Cost_0, Cost_1\}.
\]  

(29)

In this equation \( Cost_0 \) denotes always predicting non-fraud and \( Cost_1 \) denotes always predicting fraud. In other words, \( Cost_L \) refers to the least costly option of always predicting fraud or always predicting non-fraud. To assess the cost-wise improvement of an algorithm we calculate the normalized difference between the cost of using that algorithm and \( Cost_L \).

\[
Savings = \frac{Cost_L - Cost}{Cost_L}.
\]  

(30)
The motive for penalizing predictions in this manner is that losses can range from a very small amount to thousands of Dollars. Therefore, it is much more realistic to take into account the example-dependent financial costs instead of using a fixed cost.

For the purpose of this research we will use both the balanced classification rate, the $F_1$ score and the amount of money saved to assess the performance of the three implemented algorithms.

8 Results

In this section we present experimental results we obtained from the models introduced in this thesis. We first discuss the average results for each model we computed in order to answer Q2. Afterwards, we present the optimal scores from each of our models and compare them to the optimal scores obtained from the baseline models. This allows us to answer Q1 and Q3. Lastly, we present results from two feature engineering strategies. These are used to assess whether feature engineering strategies can outperform our models (Q4).

8.1 Cluster Quantities

Earlier in this thesis we explained that we are interested in knowing if the number of clusters $K$, has any influence on the performance of the models. As explained in the experimental setup section, we trained and tested each model one hundred times with different values for $K$. Below we show results for all models that use latent variables. For simplicity, in V.2, we used the same number of latent clusters for $C_F$ and $C_{\neg F}$.

<table>
<thead>
<tr>
<th>$K$</th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>$F_1$</th>
<th>BCR</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.6049</td>
<td>0.0146</td>
<td>0.7674</td>
<td>0.0285</td>
<td>0.6862</td>
<td>0.3207</td>
</tr>
<tr>
<td>3</td>
<td>0.1821</td>
<td>0.0057</td>
<td>0.8370</td>
<td>0.0111</td>
<td>0.5096</td>
<td>-0.0472</td>
</tr>
<tr>
<td>4</td>
<td>0.0607</td>
<td>0.0033</td>
<td>0.8769</td>
<td>0.0063</td>
<td>0.4688</td>
<td>-0.1623</td>
</tr>
<tr>
<td>5</td>
<td>0.0376</td>
<td>0.0026</td>
<td>0.9125</td>
<td>0.0049</td>
<td>0.4751</td>
<td>-0.1836</td>
</tr>
<tr>
<td>6</td>
<td>0.0232</td>
<td>0.0022</td>
<td>0.9327</td>
<td>0.0040</td>
<td>0.4780</td>
<td>-0.2231</td>
</tr>
</tbody>
</table>

Table 3: Evaluation metrics for different cluster quantities for model V.1

By looking at Table 3 which shows scores for model V.1, it is obvious that the number of clusters affect the performance of the model. Notably, increasing the cluster quantity seems to have a negative effect on recall and precision. As a consequence, the savings measure decreases as well. As we will show, the best scores obtained from each model do not display a similar trend. This leads us to believe that the number of clusters has little effect on the maximal performance that can be obtained from the model. It does however affect how likely it is to converge to an optimal score. This is further supported by the fact that both $K = 4$ and $K = 5$ achieve identical maximum scores. However with $K = 5$ this score is only obtained once. While with $K = 6$ it is much more common, which results in higher average scores.
Recall | Precision | Specificity | F₁ | BCR | Savings
--- | --- | --- | --- | --- | ---
2 | 0.8487 | 0.0116 | 0.5423 | 0.0229 | 0.6955 | 0.3561
3 | 0.7955 | 0.0100 | 0.4706 | 0.0198 | 0.6331 | 0.3322
4 | 0.8105 | 0.0093 | 0.4447 | 0.0184 | 0.6276 | 0.3508
5 | 0.8366 | 0.0097 | 0.4550 | 0.0192 | 0.6458 | 0.3594
6 | 0.8356 | 0.0106 | 0.4887 | 0.0209 | 0.6622 | 0.3625

Table 4: Evaluation metrics for different cluster quantities for the model V.2

Table 4 shows the average results for model V.2 for different K. At first sight, it is not entirely obvious if the number of clusters has any effect. As with model V.1, we shall see in the next subsection that these averages are not indicative of the best results obtained with each cluster size. If we look at Table 4, we can see that the number of clusters that produces the highest F₁ and BCR scores (K = 2), does not produce the highest savings score. This seems counter intuitive. However, as mentioned earlier, the transaction amount can vary from one cent to 7,500 Dollars. If two algorithms detect the same number of fraudulent transactions with different transaction amounts, the savings scores won’t be identical. Therefore, a higher classification accuracy is not a guarantee for better savings scores. As with model V.1, the scores in this table mostly reflect the ability to converge to a global optimum of the likelihood function, rather than telling us something about the desired granularity of our latent space needed to maximize results.

Recall | Precision | Specificity | F₁ | BCR | Savings
--- | --- | --- | --- | --- | ---
2 | 0.8408 | 0.0137 | 0.6113 | 0.0271 | 0.7260 | 0.4258
3 | 0.8095 | 0.0114 | 0.5204 | 0.0225 | 0.6649 | 0.4089
4 | 0.8011 | 0.0107 | 0.5145 | 0.0211 | 0.6578 | 0.4218
5 | 0.8064 | 0.0104 | 0.5149 | 0.0205 | 0.6607 | 0.4350
6 | 0.8096 | 0.0110 | 0.5381 | 0.0218 | 0.6739 | 0.4468

Table 5: Evaluation metrics for different cluster quantities for model V.3

The results from model V.3 where latent variables are conditioned on customers, are shown in Table 5. Similarly to model V.2, K seems to have little influence on the performance of model V.3. Since both models showed very little changes in savings, we decided to perform a one-way analysis of variance (ANOVA) on the savings scores. ANOVA is a method which is used to compare the means of two or more groups. We found that there is no statistically significant difference between savings scores, obtained for different values of K. This means that on average, cluster size has no significant effect on the performance of our model when it comes to amount of money saved. However, because we are dealing with millions of Dollars, statistical significance might be of lesser relevance to PPCs. For more detail on the ANOVA procedure we refer to appendix B.

If we look at these results individually, it is not very easy to perceive a trend when it comes to altering the number of clusters. To get a better understanding of the effect of altering the size of K, we plotted the evaluation scores for each model side by side.
in Figure 6. If we compare the scores from each model, we can clearly see that for V.2 and V.3, each metric consistently follows the same trend.

For model V.1 we only clustered non-fraudulent transactions. As a result, V.1 seems to gradually overfit on non-fraudulent transactions if $K$ is increased. In contrast, we clustered both fraudulent and non-fraudulent transactions for V.2 and V.3. This seems to prevent the models from overfitting on one class.

![Figure 6: Evaluation metrics for different cluster quantities](image-url)
8.2 Baseline Comparison

In this section we present a comparison between best scores from the baseline models and the best scores obtained from our models. All models were trained and evaluated one hundred times before selecting the best scores in terms of savings for each model. The results are summarised in Table 6. The main motivation for reporting the optimal scores of each model is that in practice, we can run this model until we find the initialization that satisfies our requirements the most. To most financial institutes we assume savings has highest priority. Because the maximum scores from model V.1 V.2 and V.3 fluctuated when $K$ was changed, we decided to display the scores for different cluster quantities separately.

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>$F_1$</th>
<th>BCR</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost$_0$</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.5000</td>
<td>-0.5128</td>
</tr>
<tr>
<td>Cost$_1$</td>
<td><strong>1.0000</strong></td>
<td>0.0062</td>
<td>0.0000</td>
<td>0.0123</td>
<td>0.5000</td>
<td>0.0000</td>
</tr>
<tr>
<td>RF</td>
<td>0.0268</td>
<td><strong>0.3000</strong></td>
<td>0.9996</td>
<td>0.0492</td>
<td>0.5132</td>
<td>-0.4832</td>
</tr>
<tr>
<td>LR</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.9999</td>
<td>0.0000</td>
<td>0.5000</td>
<td>-0.5128</td>
</tr>
<tr>
<td>CSDT</td>
<td>0.3839</td>
<td>0.0092</td>
<td>0.7423</td>
<td>0.0180</td>
<td>0.5631</td>
<td>-0.1790</td>
</tr>
<tr>
<td>CSLR</td>
<td>0.7768</td>
<td>0.0073</td>
<td>0.3411</td>
<td>0.0145</td>
<td>0.5590</td>
<td>0.0814</td>
</tr>
<tr>
<td>V.0</td>
<td>0.8750</td>
<td>0.0225</td>
<td>0.7635</td>
<td>0.0439</td>
<td>0.8193</td>
<td>0.4513</td>
</tr>
<tr>
<td>V.1, $K=2$</td>
<td>0.7589</td>
<td>0.0177</td>
<td>0.7378</td>
<td>0.0346</td>
<td>0.7484</td>
<td>0.4708</td>
</tr>
<tr>
<td>V.1, $K=3$</td>
<td>0.7232</td>
<td>0.0161</td>
<td>0.7256</td>
<td>0.0315</td>
<td>0.7244</td>
<td>0.4279</td>
</tr>
<tr>
<td>V.1, $K=4$</td>
<td>0.6875</td>
<td>0.0288</td>
<td>0.8555</td>
<td>0.0553</td>
<td>0.7715</td>
<td>0.4407</td>
</tr>
<tr>
<td>V.1, $K=5$</td>
<td>0.6875</td>
<td>0.0288</td>
<td>0.8555</td>
<td>0.0553</td>
<td>0.7715</td>
<td>0.4407</td>
</tr>
<tr>
<td>V.1, $K=6$</td>
<td>0.0268</td>
<td>0.0026</td>
<td>0.9349</td>
<td>0.0047</td>
<td>0.4809</td>
<td>-0.1662</td>
</tr>
<tr>
<td>V.2, $K=2$</td>
<td>0.9196</td>
<td>0.0109</td>
<td>0.4816</td>
<td>0.0215</td>
<td>0.7006</td>
<td>0.4009</td>
</tr>
<tr>
<td>V.2, $K=3$</td>
<td>0.8929</td>
<td>0.0181</td>
<td>0.6990</td>
<td>0.0355</td>
<td>0.7959</td>
<td>0.5429</td>
</tr>
<tr>
<td>V.2, $K=4$</td>
<td>0.8929</td>
<td>0.0162</td>
<td>0.6636</td>
<td>0.0318</td>
<td>0.7783</td>
<td>0.5075</td>
</tr>
<tr>
<td>V.2, $K=5$</td>
<td>0.9286</td>
<td>0.0119</td>
<td>0.5220</td>
<td>0.0235</td>
<td>0.7253</td>
<td>0.5147</td>
</tr>
<tr>
<td>V.2, $K=6$</td>
<td>0.8750</td>
<td>0.0276</td>
<td>0.8078</td>
<td>0.0534</td>
<td><strong>0.8414</strong></td>
<td>0.4952</td>
</tr>
<tr>
<td>V.3, $K=2$</td>
<td>0.7321</td>
<td>0.0170</td>
<td>0.7373</td>
<td>0.0333</td>
<td>0.7347</td>
<td>0.5143</td>
</tr>
<tr>
<td>V.3, $K=3$</td>
<td>0.7232</td>
<td>0.0239</td>
<td>0.8158</td>
<td>0.0462</td>
<td>0.7695</td>
<td>0.5312</td>
</tr>
<tr>
<td>V.3, $K=4$</td>
<td>0.8125</td>
<td>0.0260</td>
<td>0.8105</td>
<td>0.0504</td>
<td>0.8115</td>
<td><strong>0.5729</strong></td>
</tr>
<tr>
<td>V.3, $K=5$</td>
<td>0.8571</td>
<td>0.0165</td>
<td>0.6829</td>
<td>0.0325</td>
<td>0.7700</td>
<td>0.5672</td>
</tr>
<tr>
<td>V.3, $K=6$</td>
<td>0.8036</td>
<td>0.0195</td>
<td>0.7481</td>
<td>0.0380</td>
<td>0.7758</td>
<td>0.5359</td>
</tr>
</tbody>
</table>

Table 6: Comparison with Baseline Results. Cost$_0$ and Cost$_1$ refer to the costs of always predicting zero and one respectively.

It is easy to observe from Table 6 that V.2 outperforms V.1 and V.0 for all $K$ except $K = 2$. Model V.1 with $K = 2$ does outperform V.0 in terms of savings, however, it is surpassed by V.0 in terms of classification accuracy. This indicates that V.1 with $K = 2$ is more accurate with transactions with higher amounts.

For reference, we have displayed the cost of always predicting fraud (Cost$_1$) and always predicting non-fraud (Cost$_0$). The RF, LR, and CSDT obtain savings $< 0$. This means that they perform worse in maximizing savings, than always predicting fraud.
It is worth observing that while V.3 with $K = 4$ outperformed all other models concerning savings, it is surpassed by V.2 with $K = 6$ in terms of pure classification accuracy. This can be observed by looking at the $F_1$ and BCR scores, where V.2 with $K = 6$ nearly surpassed all models. Only V.1 with $K = 4$ and $K = 5$, which produced identical maximum scores, surpassed V.2 with $K = 6$ in terms of $F_1$ score, due to their high precision.

The results shown in Table 6 suggest that we have successfully built models that suffer much less from class imbalance than the baseline models. An important question that comes to mind is how much of this can be accredited to our use of the cost-sensitive threshold, introduced in Equation 9. In order to make a fair assessment of our models, we have used the thresholding method in conjunction with the baseline models to see whether the models introduced in this paper still perform better. Table 7 displays the improved baseline scores when cost-sensitive thresholding is applied. As expected, most models experience a steep increase in performance. Most notably, the LR model now produces a significant positive saving score. In contrast, the CSLR model is negatively affected by the use of the cost-sensitive threshold.

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>$F_1$</th>
<th>BCR</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>0.4732</td>
<td>0.0323</td>
<td>0.9119</td>
<td>0.0605</td>
<td>0.6926</td>
<td>-0.1449</td>
</tr>
<tr>
<td>LR</td>
<td>0.8214</td>
<td>0.0099</td>
<td>0.4890</td>
<td>0.0196</td>
<td>0.6552</td>
<td>0.4152</td>
</tr>
<tr>
<td>CSDT</td>
<td>0.4018</td>
<td>0.0074</td>
<td>0.6661</td>
<td>0.0145</td>
<td>0.5340</td>
<td>0.0961</td>
</tr>
<tr>
<td>CSLR</td>
<td>1.0</td>
<td>0.0064</td>
<td>0.0286</td>
<td>0.0127</td>
<td>0.5143</td>
<td>0.0284</td>
</tr>
</tbody>
</table>

Table 7: Baseline Results with Cost-Sensitive Threshold

In general, we can establish that the cost-sensitive prediction threshold improves results significantly. We have also shown that model V.0, V.1, V.2 and V.3 outperform the baseline models, regardless of the cost-sensitive prediction threshold. Modelling fraudulent and non-fraudulent transactions separately, by using latent variables (V.2), produces the best results in terms of BCR scores. This seems to demonstrate that this approach results in a model that is capable of accurately discriminating between fraudulent and non-fraudulent transactions. In contrast, model V.3 seems to produce high savings scores without achieving the highest classification accuracy. This result suggests that factoring in the customer yields better classification accuracy among high value transactions.

8.3 Comparison with Feature Engineering Strategies

The feature engineering strategies (aggregation features and periodic features) that have been implemented were tested with four models. We evaluated the scores these models produced with only the aggregated features, only the periodic features and the two combined. As reported by Correa Bahnsen et al. (2016) the combination of both sets of features should yield improved results.
As can be seen in Table 8, feature aggregation dramatically improves the results obtained from the cost-sensitive models. Nonetheless, compared to the results obtained from model V.2, displayed in Table 6, the use of this feature engineering strategy still bears inferior results.

Table 8: Results with Aggregation Features

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
<th>BCR</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>0.0089</td>
<td>0.0370</td>
<td>0.9986</td>
<td>0.0143</td>
<td>0.50375</td>
<td>-0.5042</td>
</tr>
<tr>
<td>LR</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.9998</td>
<td>0.0000</td>
<td>0.4999</td>
<td>-0.5130</td>
</tr>
<tr>
<td>CSDT</td>
<td>0.9107</td>
<td>0.0130</td>
<td>0.5687</td>
<td>0.0256</td>
<td>0.7397</td>
<td>0.2105</td>
</tr>
<tr>
<td>CSLR</td>
<td>1.0</td>
<td>0.0062</td>
<td>0.0052</td>
<td>0.0123</td>
<td>0.5026</td>
<td>0.0051</td>
</tr>
</tbody>
</table>

As can be seen in Table 8, feature aggregation dramatically improves the results obtained from the cost-sensitive models. Nonetheless, compared to the results obtained from model V.2, displayed in Table 6, the use of this feature engineering strategy still bears inferior results.

Table 9: Results with Periodic Features

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
<th>BCR</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>0.0268</td>
<td>0.3750</td>
<td>0.9997</td>
<td>0.0500</td>
<td>0.5133</td>
<td>-0.4822</td>
</tr>
<tr>
<td>LR</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.9998</td>
<td>0.0000</td>
<td>0.4999</td>
<td>-0.5130</td>
</tr>
<tr>
<td>CSDT</td>
<td>0.9018</td>
<td>0.0131</td>
<td>0.5778</td>
<td>0.0258</td>
<td>0.7398</td>
<td>0.2011</td>
</tr>
<tr>
<td>CSLR</td>
<td>1.0</td>
<td>0.0062</td>
<td>0.0053</td>
<td>0.0123</td>
<td>0.5027</td>
<td>0.0053</td>
</tr>
</tbody>
</table>

The use of periodic features without feature aggregation is slightly less effective than using feature aggregation alone. This is in line with the results obtained by Correa Bahnsen et al. (2016). The reason for this decline in performance is simply that the aggregation features offer a far richer and detailed account of past transactions then the periodic features. However, when examining these results one must bear in mind that the aggregation strategy adds a total of 240 features to the original 14 features displayed in Table 1, whereas the periodic strategy only adds 4.

Table 10: Results with Aggregation and Periodic Features

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
<th>Precision</th>
<th>Specificity</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
<th>BCR</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>0.0089</td>
<td>0.1111</td>
<td>0.9996</td>
<td>0.0165</td>
<td>0.5043</td>
<td>-0.4948</td>
</tr>
<tr>
<td>LR</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.9998</td>
<td>0.0000</td>
<td>0.4999</td>
<td>-0.5130</td>
</tr>
<tr>
<td>CSDT</td>
<td>0.9018</td>
<td>0.0131</td>
<td>0.5778</td>
<td>0.0258</td>
<td>0.7398</td>
<td>0.2011</td>
</tr>
<tr>
<td>CSLR</td>
<td>0.8214</td>
<td>0.0139</td>
<td>0.6383</td>
<td>0.0273</td>
<td>0.7299</td>
<td>0.3517</td>
</tr>
</tbody>
</table>

When combining the two sets of features, some peculiar behavior can be observed. Firstly, the CSDT algorithm obtains exactly the same results as it did when trained with only the periodic features. Furthermore, because the results were better with only the aggregated features, we expected an improvement in the results. Another unexpected result was that the CSLR dramatically improved its performance, compared to CSLR with the two feature engineering strategies separately. Interestingly, where first it almost exclusively predicted non-fraud, it suddenly outperformed all other baseline models. Regardless we can clearly see that in terms of F<sub>1</sub>, BCR and savings scores all feature engineering strategies are outperformed by V.2 model.
9 Conclusion

In this thesis, we introduced a number of different bayesian network based methods for credit card fraud detection. For the purpose of analysis, we performed several experiments. First, we varied the number of clusters $K$ in our latent space in order to assess its effect on the performance of our models. The experimental results show that for model V.1 less clusters result in a better performance. A likely explanation is that this can be attributed to the fact that a higher value for $K$ results in a likelihood function with more local optima. This is confirmed by the fact that the differences in performance are reduced when looking at the maximum scores (Table 6). Table 3 shows that on average, more clusters results in a model that is less likely to predict fraud. In other words, learning latent representations for non-fraudulent transactions only, seems to be prone to overfitting.

In model V.2 and V.3, both fraudulent an non-fraudulent transactions are explained by latent variables. In V.2 we trained a separate model on fraudulent transactions. In V.3, the model learned one latent variable from all transactions but we conditioned the latent variable on the customer. Since with both models, the latent variables were trained on all variables, these models seem to be less prone to overfitting. This is why on average, the savings scores fluctuate less significantly when $K$ is changed.

Although these averages give some indication on how $K$ influences performance, they hide a significant amount of information. When we look at the optimal results, the saving scores obtained from V.1, V.2 and V.3 fluctuate significantly. As demonstrated in Table 6, model V.3 with 4 clusters outperforms all other models in terms of savings and V.2 with 6 clusters outperforms all other models in terms of BCR scores while obtaining the second highest $F_1$ scores.

In general, these experimental results lead us to conclude that our proposed models outperform all baseline methods. Moreover, we have succeeded in training models that reduce savings up to 57.29% without rebalancing the data. This resilience against class imbalance, however, is not a virtue of the bayesian networks. Instead, it can be attributed to the cost-sensitive threshold, employed for prediction (Equation 9). This is exemplified by the fact that the LR model reached much higher saving scores when combined with the cost-sensitive threshold. Nevertheless, we have seen that the bayesian network models introduced in this paper, still outperform all baseline models and feature engineering strategies in terms of savings. Therefore we conclude that modelling fraudulent and non-fraudulent transactions separately, yields very positive effects on the performance of our models.

The most straightforward explanation for this enhanced performance is that our latent variables capture the underlying processes that cause transactions to be fraudulent or non-fraudulent. Additionally, the topology of the networks represent a probabilistic reasoning framework in which the fraudulent latent variables are negatively correlated with the non-fraudulent variables. In other words, if the latent variables, trained on all fraudulent transactions explain a transaction, the other latent variables trained on the non-fraudulent transactions explain it less.

Model V.3 fine-tunes these ideas by conditioning the latent variable $C$ on the client ID. This results in a model where each client has its own latent representation which is equal to the distribution over the latent variables $C$. These probabilistic representa-
tions contribute to the detection of fraudulent transactions of high value. As a result, this model shows a significant improvement compared to other models from a financial standpoint. However, from an accuracy standpoint, model V.2 shows more promise.

10 Recommendations

In the course of this research many extensions have been thought of, that could contribute to the current research. There are a number of reasons that have lead us not to pursue these ideas. Sadly, time played a big part in these decisions. The main topics that we would be interested in further exploring are cluster analysis, using mixture of gaussians and improving network topology.

Due to time constraints, determining what clusters actually mean was beyond the scope of this thesis. An interesting addition to our research would be to analyze each cluster of data points in order to assess what common features are observed in each cluster. EM is a soft clustering method which produces distributions over latent variables. These distributions are likely to be different for fraudulent datapoints and non-fraudulent datapoints. Figuring out if they are different and how they are different would help us better understand the use of latent variables in the fraud detection domain.

Another aspect that has not been the topic of research in this thesis, was the choice of distributions as mixture components in EM. In this research we have modeled all variables as discrete variables. Because the choice of mixture components was not the subject of this research, we chose to use easily implementable building blocks for our algorithms. Instead of focussing on the choice of components, we wanted to test specific hypotheses, which we expressed with various network topologies. However, by using a mixture of gaussians we would better model continuous variables. Ultimately, this could further improve the techniques introduced in this paper.

Lastly, there are several ways to further improve the topologies of the networks. Part of this research aimed to condition latent variables on customers. Likewise, a similar approach could be taken in order to condition latent variables on merchants. Adding the merchant ID to our bayesian networks could have an impact in the same manner that customers impact the probability of fraud. Furthermore, we made the independence assumption in order to build our models. In other words, we have assumed that all input variables \( x_n \in X \) are independent of each other given the variable on which they are conditioned. In reality however, this assumption is wrong. For instance, the sine and cosine of the clock time we used to represent the time of day, are dependent. Improving upon the network topology in order to model certain dependencies, would therefore be an useful addition to this research.
References


A Aggregation and Periodic Features

This appendix offers an overview of all the features that were created as periodic and aggregation features. The techniques have presented here have all been described in Correa Bahnsen et al. (2016), and are exactly implemented as explained. However since we had no way of knowing which features had been used in their paper, we have chosen to make our own selection of features.

In order to calculate the aggregated features, six time intervals were selected to aggregate the feature over. The time intervals took place between the transaction time and 1, 3, 6, 12, 18, 24, 72 and 168 hours prior to the transaction. The features that were aggregated are GeoCode, Merchant, Country, Email and Phone No. For each transaction this meant that counts were made of how many times the same customer had made a transaction with the same feature value, within the time interval. Furthermore, the amount of money spent in these transactions was also aggregated and put in a second feature. With these five features this meant that for 8 time intervals and the aggregated amount, \(5 \cdot 8 \cdot 2 = 80\) features were created.

Secondly feature co-occurences were also aggregated. This meant that essentially the same operation was perfomed. Only now each combination of the five features mentioned earlier was counted. Table 11 shows the combinations of features that were used. The amount of features added by aggregating these co-occuring features was \(10 \cdot 8 \cdot 2 = 160\). In total 240 features were created.

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merchant</td>
<td>Email</td>
</tr>
<tr>
<td>Merchant</td>
<td>Country</td>
</tr>
<tr>
<td>Merchant</td>
<td>GeoCode</td>
</tr>
<tr>
<td>Country</td>
<td>GeoCode</td>
</tr>
<tr>
<td>Country</td>
<td>Email</td>
</tr>
<tr>
<td>Email</td>
<td>GeoCode</td>
</tr>
<tr>
<td>Phone No</td>
<td>Email</td>
</tr>
<tr>
<td>Phone No</td>
<td>Country</td>
</tr>
<tr>
<td>Phone No</td>
<td>Merchant</td>
</tr>
<tr>
<td>Phone No</td>
<td>GeoCode</td>
</tr>
</tbody>
</table>

Table 11: Feature Combinations

The periodic features were made with four intervals. Namely, 7, 30, 60 and 90 hour intervals. For each customer, transactions that fell within these intervals were grouped. Then the von Mises distribution was then used in order to create a periodic variable. The von Mises distribution is defined as

\[
f(t_x|\mu_v, \sigma_v) = \frac{e^{\frac{1}{2}\sigma_v \cos(t_x - \mu_v)}}{2\pi I_0(\frac{1}{\sigma_v})}.
\]  

(31)

Where \(I_0\) is the modified Bessel function of order 0, and \(\mu_v\) and \(\sigma_v\) are the periodic mean and periodic standard deviation. We used the probability of the current transaction given
by the von mises to infer whether or not this transaction was expected. This resulted in a boolean variable which was set to true if the probability of the current transaction was greater than a certain threshold. Multiple thresholds were empirically tested in order to produce the optimal results. The resulting sets of features were added on top of initial set of features described in Table 1.
Analysis of Variance (ANOVA)

The ANOVA is a generalisation of the t-test. More specifically, it is a method for testing whether the means of multiple groups are equal. In this thesis, we are using it to test whether the number of clusters has any effect on savings. The null hypothesis (the number of clusters has no effect) is given by

$$H_0 = \mu_2 = \mu_3 = \mu_4 \cdots = \mu_K.$$ \hspace{1cm} (32)

Here, \(\mu\) represents the group mean. In this setting, \(K\) denotes the number of clusters with \(K - 1\) being the number of groups (\(K = 1\) is not used). In this thesis we have conducted a one-way ANOVA, meaning that only one independent variable and one dependent variable were involved. In our analysis we calculated the savings measure for model V.1, V.2 and V.3 with 6 different cluster quantities \(k = \{2, 3, 4, 5, 6\}\). Each model and cluster was trained and tested 100 times to create six groups of 100 observations.

**Step 1** The first step is to calculate the mean of each group. Since each experiment was repeated 100 hundred times for 6 different cluster quantities \(k\), 6 averages are calculated over 100 observations.

$$\mu_k = \frac{1}{100} \sum_{i=1}^{100} savings_{k,i} $$ \hspace{1cm} (33)

The averages obtained from each group are displayed in Table 4.

**Step 2** The subsequent step is to calculate the overall mean. In other words, averaging over all \(\mu_k\).

$$\bar{\mu} = \frac{\sum_{k=2}^{K} \mu_k}{K - 1}$$ \hspace{1cm} (34)

**Step 3** After this, the between group sum of squares is calculated by

$$BSS = \sum_{k=2}^{K} (K - 1)(\mu_k - \bar{\mu}).$$ \hspace{1cm} (35)

The between-group degrees of freedom is equal to the number of groups, minus one \(f_b = 4\). The between-group mean square value is equal to

$$MS_B = \frac{BSS}{f_b}$$ \hspace{1cm} (36)

**Step 4** In order to calculate the within-group sum of squares we must center the data by subtracting each group mean \(\mu_k\) from the observations in each corresponding group. By squaring all these values and summing them obtain the within group sum of squares

$$WSS = \sum_{i=1}^{100} \sum_{k} (savings_{k,i} - \mu_k)^2.$$ \hspace{1cm} (37)
The within group degrees of freedom is equal to the number of observations, minus one, times the number of groups

\[ f_w = a(n - 1) = (K - 1)(100 - 1) \]  

(38)

And the within-group mean square value is

\[ MS_W = \frac{WSS}{f_w} \]  

(39)

**Step 5** Lastly, the F-ratio is

\[ F = \frac{MS_B}{MS_W} \]  

(40)