COMBATING FRAUD BY LEVERAGING MACHINE LEARNING

Casey Astiz

Adviser: Professor Michael Linderman

A Thesis

Presented to the Faculty of the Computer Science Department

of Middlebury College

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Arts

December 2018

ABSTRACT

Credit card fraud in the United States has been on the rise, reaching over $8 billion in 2018 [8]. The majority of these are “card not present” transactions, which have become more common with the growing popularity of online transactions. Fraudulent transactions only make up 1-3% of transactions. Therefore, it is difficult to create unbiased classifiers for fraud detection systems. In this thesis, I summarize related works on the topic of anomaly detection, feature engineering for classifying fraud, and game theory. I interviewed practitioners in the fraud detection space to compare their knowledge and day-to-day methods with current research. I implemented my own fraud detection system using a synthetic mobile payment transaction dataset. I discuss how these anomaly detection methods can be used in other applications. I find decision trees to be the most effective classifier given my dataset, which is consistent with interviewee’s suggestion of rule based systems.

ACKNOWLEDGEMENTS

I would like to thank Professor Linderman and Professor Scharstein first and foremost for their help in this Computer Science Thesis. Without them, this thesis would not exist. I would also like to thank the Computer Science department at Middlebury College for giving me the opportunity to learn and develop as a computer scientist. Last, I would like to thank all my unnamed interviewees who were willing to discuss their expertise about fraud detection with me, in addition to my friends and family who supported me through the arduous thesis writing process. I truly appreciate all who helped create this thesis, and add to my knowledge!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Every day there are new ways to pay for goods or services. As shown in Figure 1.1, cash is a way of the past; mobile, peer to peer, and online payments through credit cards are the way of the future [7]. While these convenient payment methods are very exciting, the additional methods of payment lead to increased opportunities for fraud. Credit card fraud is on the rise worldwide, as shown in Figure 1.2 reaching $8 billion in the United States in 2018 [8]. For example, credit card fraud through online payments in Australia hit $476 million in 2017, rising $58 million from the year before [28]. Credit card fraud is happening all over the world, and it is necessary to find more efficient and effective methods to catch fraudsters to keep everyday users safe unaffected by fraud.

Financial fraud increases costs in several ways. With any type of financial fraud, there is a certain level of financial burden for the company or bank that has to pay out. Other than the cost of the fraudulent transaction itself, having high levels of fraud leads to uncertainty for both the customers and bank trustees. This uncertainty then creates higher transaction costs for all and less efficient markets. When banks lose money to fraudsters, card holders partially or entirely pay for this loss through higher interest rates, fees, or reduced rewards and benefits for customers. Without automatic fraud detection, a significant amount of overhead exists because of the cost of investigating transactions and employing these investigators [13]. Credit card fraud has other societal costs, as “credit card frauds have important social consequences and ramifications, as they support organized crime, terrorism funding, and international narcotics trafficking” [31].

Fraud detection is an instance of anomaly detection. Fraud detection and anomaly detection more broadly are difficult problems because of the heavily skewed classes and potential for undistinguishable classes. These two factors mean that there are few

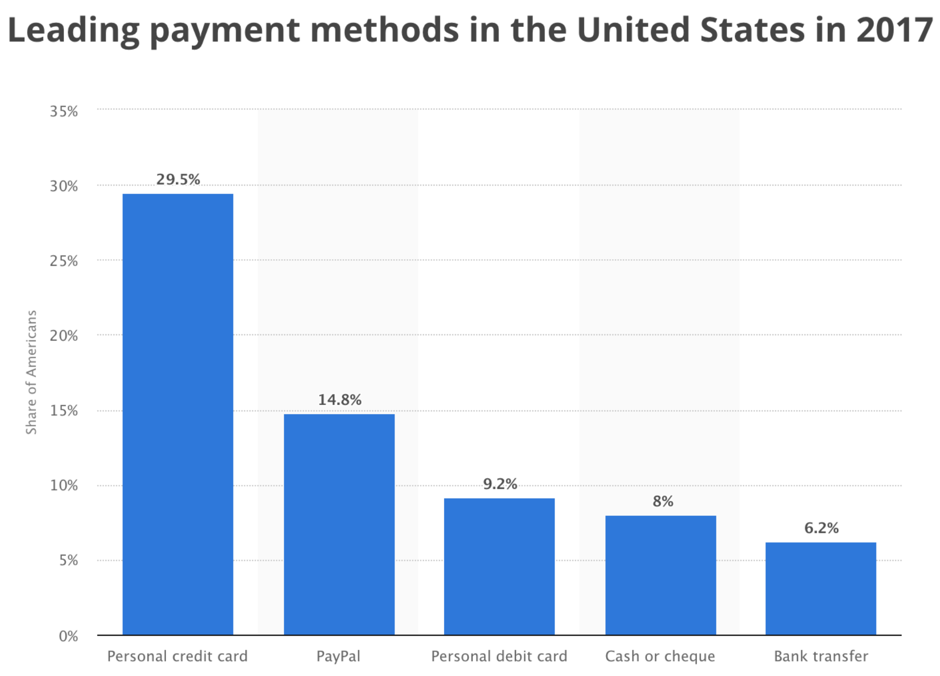


Figure 1.1: Leading payment methods in the United States in 2017 [7]

examples for classifiers to learn from, and there may not be a clear division between the classes. Further, an effective fraud detector must be adaptive to new patterns and categories of fraud. The main goal of this thesis is to investigate current methods of financial fraud detection and implement one or more of those methods. I will describe the most prevalent methods for fraud detection in current use, as well as past fraud detection methods.

Chapter 2 reviews different methods used in the industry in the past and present for fraud detection. The background section will combine information from research papers with Chapter 3, which synthesizes information I learned from interviewing people at different companies in this field to provide some context for the literature review. My own implementation of a fraud detection system is described in Chapter 4 using a synthetic credit card transaction dataset. Comparing business fraud detection systems to published research in the field allowed me to analyze why some systems would be successful in research settings but not business settings. Chapter 5 will discuss other

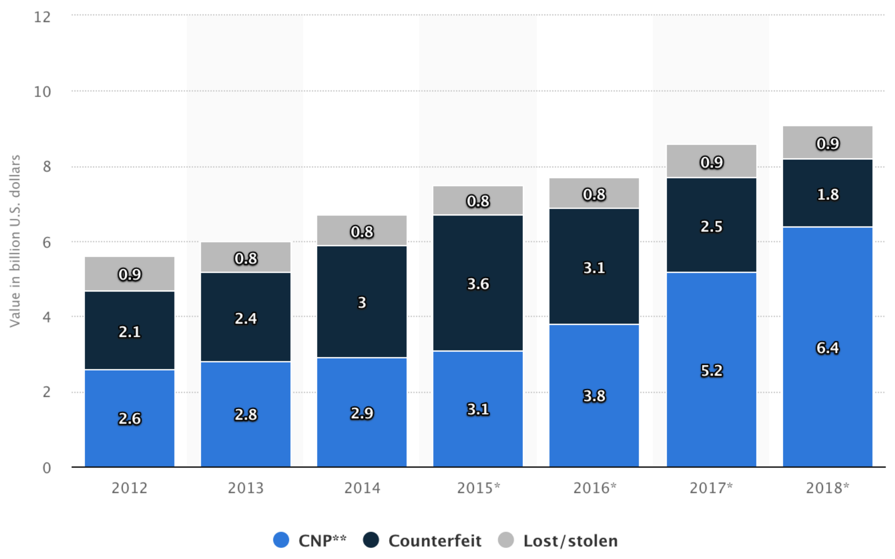


Figure 1.2: Value of payment card fraud losses in the United States from 2012 to 2018, by type (in billion U.S. dollars) [8]

applications for successful fraud detectors. Chapter 6 concludes.

CHAPTER 2

FRAUD DETECTION

Here we define fraud as “wrongful deception with the intent to gain personally or financially,” or intentionally deceiving another person to obtain their property [4]. Fraud can either be a civil or criminal offense, depending on the state. There are a multitude of types of frauds, from credit card fraud and website misdirection to pyramid schemes and insurance fraud.

It is important for businesses to balance finding all fraud and potential overhead for investigations and lost business. A company may choose to accept some level of fraud in order to minimize the number of legitimate users being classified as fraudsters. For example, in subscription based services, it can be financially advantageous for companies to allow all potential users to create new accounts because the cost of losing a potential client can be higher than the cost of allowing a small portion of fraudulent accounts. Because companies are normally profit maximizing, fraud detection in a business context will most likely always involve weighing the costs of false positives with the cost of the fraud itself.

Researchers have more flexibility than a financial institution in their pursuit of fraud detection. Researchers are able to manipulate datasets by artificially changing fraud versus non-fraud proportions, and test different combinations for the best performance. Researchers also have no time constraints as they do not need to make a split second decision of whether or not to allow or deny a transaction. Though some researchers focus on increasing accuracy in real time fraud detection systems, most researchers can build systems that make decisions in any amount of time.

2.1 Imbalanced Classes

Financial fraud detection falls under the category of anomaly detection because most transactions are not fraud. Some datasets are perfectly split between two classes. This means that there are equal amounts of positive and negative examples for a classifier to be trained with. Imbalanced classes happen when there are more examples of one class over another, and is commonly used to refer to large differences in number of examples. This imbalance makes classifying the minority class more difficult because the classifier has not seen as many examples of that class, and therefore the model will be biased towards the majority class. Anomaly detection is an extreme example of skewed classes because the minority could be less than 1% of the examples.

There are several ways to address skewed data. One strategy is to get more data. With more examples of the anomaly, it is easier to detect a pattern between them. However, acquiring more data is not always an option, especially when working with data collected in the real world. Another popular method for dealing with data imbalance is to artificially balance the data. In a study by Chan and Stolfo from 1998, the authors find that artificially balancing the data during training leads to more accurate classifiers [13]. To balance the classes, one can under sample the majority class or over sample the minority class by creating more instances of minorities.

$$Precision = \frac{TruePositives}{(TruePositives + FalsePositives)} \tag{2.1}$$

$$Recall = \frac{TruePositives}{(TruePositives + FalseNegatives)} \tag{2.2}$$

$$F1 = \frac{2(Precision * Recall)}{(Precision + Recall)} \tag{2.3}$$

In most machine learning problems, the default metric to monitor a model is accuracy of a system, defined as the number of correct predictions divided by the total

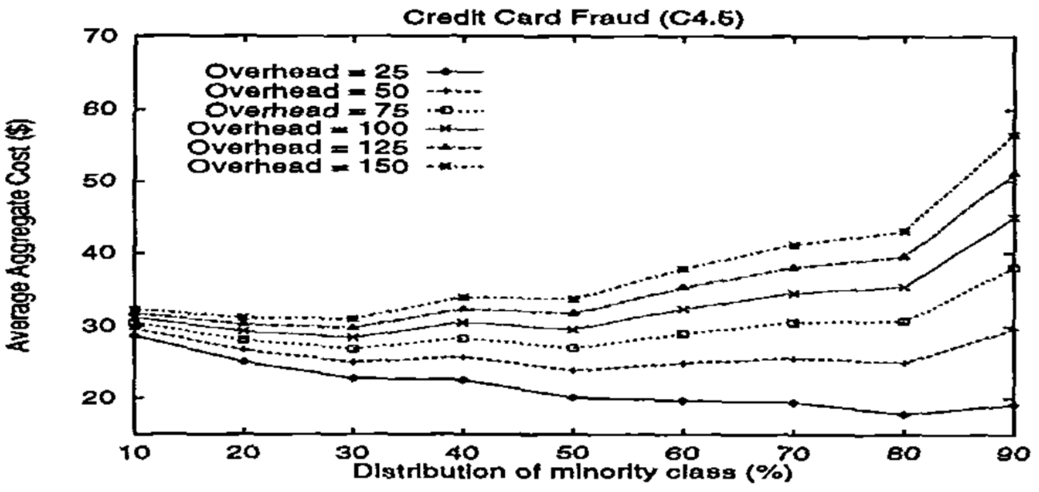


Figure 1: Training distribution vs. the credit card fraud cost model

Figure 2.1: Training Distribution vs. the credit card fraud cost model [13]

number of examples. However, this is not an effective measure when discussing anomalies. For example, if less than 1% of a dataset are instances of fraud, and if a classifier were to predict non-fraud for each transaction, the accuracy would be over 99%, but would still not be catching any of the fraud. Metrics that are often used to measure skewed class problems are precision, recall, and F1 score. These all analyze the success in predicting the minority class rather than the majority class through the proportions of true positives, false positives, true negatives, and false negatives. The definition for precision, recall and F1 are shown in equations 2.1, 2.2, and 2.3, respectively. Chan and Stolfo take a different approach, and instead use the average aggregate cost of fraud as their metric for evaluating their model. Here, the authors define costs for true positives, false positives, true negatives, and false negatives. Each scenario is defined as the cost of the transaction plus the cost of investigating the transaction if it is predicted as fraud. The cost of the investigation and fraud analysts is called the overhead.

Figure 2.1 shows the results of varying the distribution of fraudulent transactions in the training set and different overhead against the average aggregate cost [13]. While Chan and Stolfo find that larger overhead leads to higher overall cost, they also observe that when overhead is smaller, cost is minimized by including larger percentage of fraudulent transactions in the training set to increase model sensitivity. When overhead is smaller, a bank can afford to have a higher false positive rate, as the false positives can be investigated relatively inexpensively. When overhead is larger, a bank cannot afford as many false positives, as it is expensive to investigate positive fraud, and so benefits from a more specific model. These results are helpful to banks who may be deciding on how sensitive to make their system, given their overhead costs.

A 1997 paper by Stolfo et al. uses Meta-Learning for credit card fraud detection [22]. Here, the authors similarly argue that false positive rates and true positive rates are much better metrics than overall accuracy when evaluating these learned fraud classifiers. Their proposed meta-learning system allows financial institutions to share their models of fraudulent transactions by exchanging classifier agents in a secured agent infrastructure. This is a useful tool for financial institutions to be able to work together to solve similar fraud problems.

For this study, the authors used a database of 500,000 transaction records from the Financial Services Technology Consortium, each containing 30 fields. In their experiment, the authors tested various machine learning and meta-learning models with different proportions of fraud to non-fraud training data. Based on their experiments, they find that a 50%/50% distribution of fraud to non-fraud training data generates classifiers with the highest true positive rate (80%) and the lowest false positive rate (13%). The best method the authors discovered is using meta-learning with BAYES as a "meta-learner to combine base classifiers" [22]. BAYES calculates the probability of an event based on prior knowledge of factors that may affect that event.

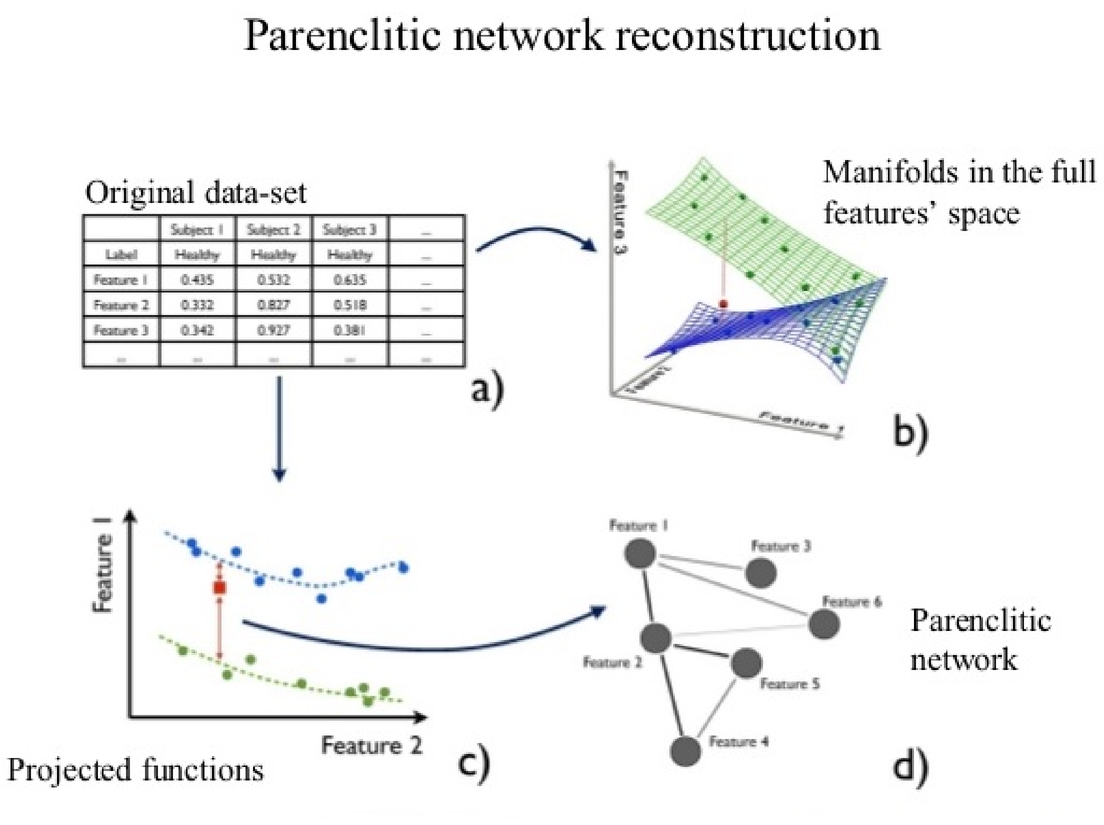


Figure 2.2: Parenclitic Network Formation [11]

2.2 Credit Card Fraud

The papers in the previous section mainly focus on redistributing the imbalanced classes as a way to improve fraud detection rates. In a different approach, Zanin et al. use parenclitic network analysis, which is a hybrid data mining and network classification algorithm [31]. As Zanin et al. write, "while data mining algorithms are able to detect hidden patterns in data, they usually lack the capacity of synthesizing metrics describing the global structure created by the interactions between the different features" [31]. The authors focus on parenclitic networks, aiming to represent the data in new ways in order to find other patterns of describing the data.

Parenclitic networks look for relationships between features of a transaction. Each transaction t has a network $G(t)$ of k nodes based on the number of features in the

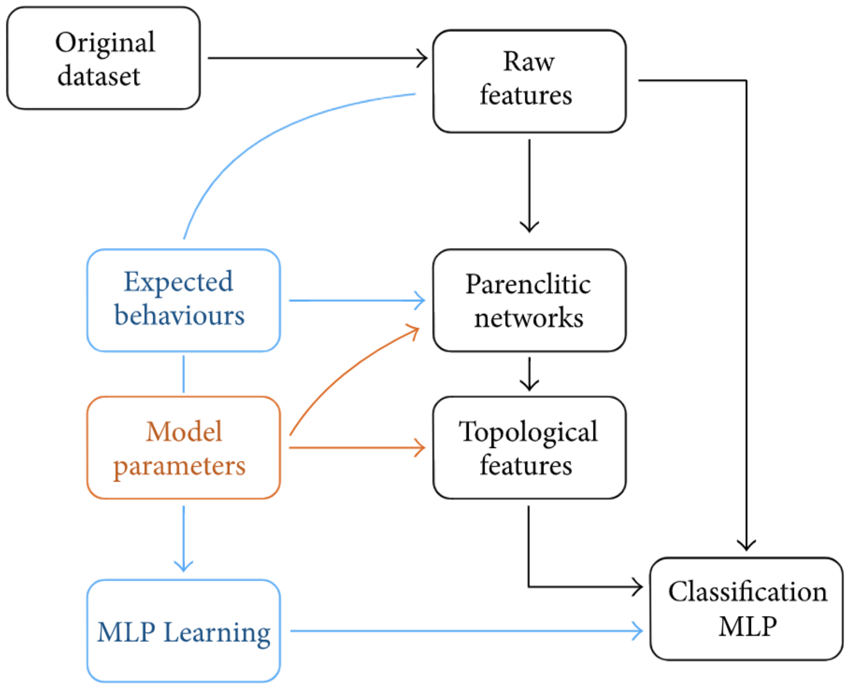


Figure 2.3: System incorporating parenclitic networks [31]

dataset. The connections between each node corresponds to a measure of compatibility between those two features, measured with linear regression. Parenclitic networks are constructed by fitting a line between every pair of features. An edge is created between that pair of features if the distance of the unlabeled transaction to the relationship measure is greater than some threshold. Because these networks are looking for differences between licit transactions and illicit transactions, "parenclitic network $G(t)$ summarizes the [pairs] of features whose correlation strongly differs from a typical licit transaction; the structure of this network thus contains valuable information about the (abnormal) correlation of features in the credit card transaction" [31]. The measure of correlation of these features are then extracted from the network, and create features themselves. An example of parenclitic network creation is shown in Figure 2.2.

To test their parenclitic network approach, the authors used an anonymized dataset of all credit and debit card transactions from clients of a Spanish Bank, BBVA, from

January 2011 to December 2012. This dataset includes standard fields about transactions such as amount, origin and destination. The authors have also synthesized information like average transaction size for a user from the given information. They use multi-layer perceptrons, a type of artificial neural network, as the model for classifying their transactions. These multi-layer perceptrons are represented by a set of connected nodes in which each connection has a weight associated with it that can be adjusted. The overall system that the authors create is displayed in Figure 2.3. Zanin et al. find that parenclitic networks alone are not enough to significantly lower the classification error. However, the addition of parenclitic features to the raw data set increases the overall information a model receives as input and decreases the classification error rate from 19.2% to 12.23% [31].

In a 2017 paper, Wedge et al. have attempted to solve the "false positive" problem in fraud prediction at an industrial scale [29]. The goal of this paper is to use a feature engineering approach to reduce false positives. It is estimated that only 1 in 5 transactions that are predicted to be fraud are actually fraud, and analysts have found that these false positives are costing more than the fraud itself [29]. However, improving false positive rates with human involvement is very costly, prompting Wedge et al. to study the automatic method by the name of deep feature synthesis.

The authors create their features using only transaction information and aggregated historical information. Each transaction has a set of features associated with it. The authors use deep feature synthesis in order to relate the current set of features to the historical information known about that user's transactions. Deep feature synthesis applies two types of functions to information known about a transaction to create these features. The first is a transform function, which applies a function to the data known about a transaction and creates a new feature. For example, given dates, a transform feature could be an indicator for whether it was a weekend date. The other function is an aggregation, where features are made by analyzing the user's past behavior in comparison to the example at hand.

These transactions were classified using scikit-learn's random tree classifier with 100 trees. Decision trees are effective at calculating relative feature importances. The 236 original and deep feature synthesized features were then fed to the decision tree classifier. Through deep feature synthesis, they were able to reduce false positives by 54% and increase precision by 91.4% compared to the current BBVA solution [29].

They found that their solution can maintain high accuracy when the historical features of a card are computed once every 35 days, losing 0.05 points of precision compared to a daily computation. This eliminates the need for streaming computing for the feature set because the features can be computed offline at a regular interval instead of constantly updating these synthesized features after a new transaction. The author's system reduced the false positive rate, and reached an F1 score of 0.56 versus the BBVA system of 0.20. In addition, Wedge et al.'s system still reduced total cost to this bank by 190,000 euros. This research is helpful because it shows that features do not need to be recomputed with each new transaction, and thus banks can reduce their false positives without increasing the latency for their users.

2.2.1 Financial Fraud

Credit card fraud is an instance of financial fraud. There are many other types of financial fraud including anything related to a financial transaction and any transaction a bank may be responsible for. In a 2011 paper, Johan Perols analyzed the best statistical and machine learning algorithms for financial quarterly statement fraud detection [19].

As is the case with credit card fraud, there is a high imbalance between fraudulent financial statements and non-fraudulent statements. With financial statements, it is much more costly to a bank or a firm to classify a fraudulent firm as a legitimate firm than it is to classify a legitimate firm as a fraud firm [19]. This is because fraudsters will continue to cost a company or bank money until they are caught during an audit. It is difficult to find fraud because fraudsters are actively trying to cover up their fraud, making fraud detection increasingly more difficult. For these reasons, Perols is looking for the best classification method with the lowest overall error rate and highest true positive rate.

The results of this paper are of particular use to institutions like the Securities and Exchange Commission as well as other auditors. In this study, Perols uses open source classification algorithms from the data mining tool Weka [19]. From Weka, six algorithms were chosen for analysis: artificial neural networks (ANN), support vector machines (SVM), C4.5, logistic regression, stacking, and bagging. The experiment was conducted using a dataset consisting of fraud investigations reported in the Accounting and Auditing Enforcement Releases from 1998 through 2005. Similar to other studies referenced in this thesis, the author reinforces the necessity to measure the success of an algorithm for a skewed class problem with metrics other than accuracy.

To test the different models, the author uses a ten-fold cross validation. This is unlike most studies that split the dataset into 3 sections, training set, evaluation set, and cross validation set. Instead, the authors have a training set, test set, and 10 cross validation sets. Here the author finds that SVMs outperforms the other classification algorithms, followed by logistic regression then bagging. In addition, "out of 42 predictors examined, only six are consistently selected and used by different classification algorithms: auditor turnover, total discretionary accruals, Big 4 auditor, accounts receivable, meeting or beating analyst forecasts, and unexpected employee productivity" [19]. Though these models were all given the same features, the training algorithms only picked out a small portion of the same features to judge transactions on, potentially implying that only a subset of features were informative.

2.3 Other Fraud

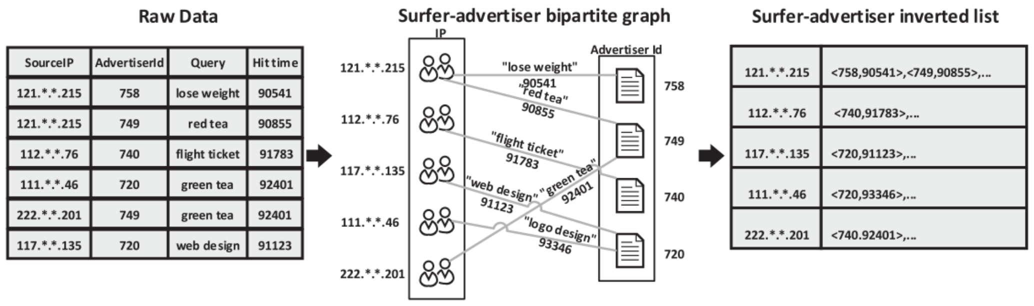
Other forms of fraud detection face similar challenges to credit card fraud detection, and their results are therefore applicable to credit card fraud detection. An example of this is a paper by Fawcett and Provost from 1996. This paper describes automatic methods for fraud detection based on profiling customer behavior [15]. Instead of focusing on credit card data, this paper focuses on cellphone cloning, which the authors claim is more expensive than credit card fraud. This is when a customer's Mobile Identification Number and Electronic Serial Number are cloned and programmed into a cellphone that does not belong to that customer.

In this paper, the authors present a framework for automatically generating fraud detectors. Some of the standard methods used to detect fraud in these calls are to search for collisions or overlapping calls between the original user and the fraudster, or to search for calls close in time that could not have possibly been placed by the same user. The profilers the paper describe capture the typical behavior of an account, forming a set of features for a user which are used calculate how far an account is from typical behavior.

Given this information, the authors use rule learning programs, which search for rules with certainty factors above a user-defined threshold. Each call has 313 attributes that allow for partitions of the calls. Here, each account also has its own set of rules. Through their data mining process, 3630 rules were created for detecting fraud, which were then narrowed down to 99 rules. The conclusions of this paper were that the authors had to sacrifice accuracy by about 1% in order to reduce the total cost of the system, but that their method was overall effective, lowering costs from $20,000 when predicting all as fraud to approximately $3,303 using their system [15]. Like other papers, the authors highlight the need to build adaptive systems to account for evolving fraud.

Crowdsourcing is another channel of fraud, introducing new types of malpractices into Internet advertising. In a paper by Tian et al., authors attempt to detect crowd fraud in internet advertising [25]. In this paper, authors are focusing on detecting when malicious crowdsourcing platforms attack other advertisers. These types of attacks are much harder to detect than automated attacks because they are human generated crowd frauds, and ever changing. Crowd fraud is defined by a few characteristics: moderateness, synchronicity, and dispersivity. Crowd fraud often arises from a vast number of attacking sources, but each source has a low amount of fraudulent traffic. These attacks are meant to hurt advertisers by raising their advertising expenses through manipulating pay per click advertisements. There are similar paradigms in credit card fraud, where fraudsters use multiple fake identities and credit cards to make many low cost transactions. As of the time of this paper in 2015, current methods for crowd fraud detection rely on previously known crowd fraud patterns and rules based on suspicious queries. However, creating a set of rules is very labor intensive.

To detect crowd fraud automatically, Tian et al. use an enhanced graph model based on anomaly detection methods for detection of coalitions, or groups attacking the same target [25]. First, the authors construct a surfer-advertiser bipartie graph, where each edge represents a click log as shown in Figure 2.4. Then, the authors look for clusters to find surfer coalitions that exhibit synchronicity, and then filter out the large coalitions. Once these large coalitions are determined, they can be removed from the domain of the centralized advertisers. Before constructing these graphs, the authors also pre-filter to remove sections of non-fraudulent data; in this case they remove more than 70% of the click logs. The authors test their nonparametric clustering algorithm on real world data and find that the system converges and scales at a linear rate, with a 98.7% accuracy in finding malicious coalitions [25]. Through converting this coalition detection into a clustering problem, the authors are able to build a scalable and accurate crowd fraud detection system.


Raw Data

SourceIP	AdvertiserId	Query	Hit time
121.*.*.215	758	lose weight	90541
121.*.*.215	749	red tea	90855
112.*.*.76	740	flight ticket	91783
111.*.*.46	720	green tea	92401
222.*.*.201	749	green tea	92401
117.*.*.135	720	web design	91123

Surfer-advertiser inverted list

121.*.*.215	<758,90541>,<749,90855>,...
112.*.*.76	<740,91783>,...
117.*.*.135	<720,91123>,...
111.*.*.46	<720,93346>,...
222.*.*.201	<740.92401>,...

Figure 2.4: Bipartie graph visualization [25]

Another type of fraud found in everyday places is ranking fraud for mobile apps. This ranking fraud is committed in order to move apps up the ranking lists. With over 1.6 million apps in the Apple App Store and Google Play, app leader boards are an important marketing tool for developers. However, developers can manipulate their ratings by implementing “bot farms” or “human water armies” which increase app statistics like downloads and ratings in a very short time [32]. App ranking fraud detection is particularly difficult because the fraud can occur at any point of the app’s life cycle, which is why the authors focus on local fraud detection instead of the global anomaly of mobile apps.

In this study by Zhu et al., local fraud refers to short term instances of ranking fraud, and global fraud refers to fraud committed over the entire lifetime of the app. A challenge for this problem is that there are a vast number of apps and no easy way to determine the ones that have committed fraud, enforcing the need for automatic fraud detection methods. Additionally, the authors must find hidden fraud patterns as evidence for ranking fraud because of the dynamic nature of mobile app rankings. Zhu et al. investigate ways of accurately locating the ranking fraud by mining the active periods, mainly focusing on detecting local anomalies versus global anomalies of app rankings [32].

In order to detect leader board fraud, the authors look for active periods or ‘leading sessions' of an app, and find that fraudulent apps' leading session ranking patterns have large spikes unlike normal apps. Zhu et al. use statistical hypothesis tests to find evidences for ranking fraud, then use an unsupervised evidence-aggregation method to combine the three types of evidences. The three types of evidences the authors focus on are ranking based, rating based, and review based, with each of these evidences having several features built into them. By testing their model with different permutations of evidences, the authors find that their combination of all three evidences outperforms single evidence models and other baseline models. This may be because the app ranking fraud does not necessarily cause app rankings to increase, but may lead to higher downloads or reviews. Therefore, it is more important to look at all the features rather than any one of the individual features [32]. Through their experiments, the authors showed that mining for features, and modeling them with hypothesis tests, creates an effective fraud detector that can be extended in other fraud detection systems.

2.4 The Human Element

Credit card fraud detection, while automated by machines, benefits from the ingenuity of humans. Fraud and fraud detection is a cat and mouse game, where each player is looking to outwit the other. Fraud systems adapt to fraudsters' patterns, while fraudsters adapt once they learn the rules of the system. This back and forth creates a complex and ever changing game between the two parties. Modeling formally how a fraudster would attempt to commit fraud will help build better fraud detectors.

2.4.1 Credit Card Fraud Detection with Game Theory

By using game theory, companies can start to predict future attacks while also providing possible courses of action to defend against them. In a paper by Vasta et al., a rule based system is combined with Game-theory [26]. With credit card fraud, the fraudster and fraud detecter have conflicting motives, each trying to maximize their own payoff through a multistage game between the two players. Here, fraudsters are considered to be rational adversaries, meaning that fraudsters are attempting to maximize their gain without being detected. Companies are trying to minimize their losses to fraud, and therefore the game between the two parties becomes a maximization-minimization problem.

Credit card fraud detection is not like a traditional game because there is unlimited moves and unlimited potential moves. With credit card fraud, an attacker can make multiple moves. The fraudster can continue to commit fraud until they are caught or until they notice a change in the fraud detection system. In addition, the attacker can act without time constraints; a fraudster can calculate their move for as long as possible. Lastly, the possible set of moves in this game is not well defined because of the adaptive nature of fraud and fraud detection. This is very similar to database intrusion detection as well.

Information is the main currency in this game. To begin, there is a lack of information for both parties: a detection system does not know whether or not a party is a fraudster, and a fraudster is not aware of the exact rules of the system. A fraudster will launch attacks, aiming to learn about the rules of the system, while the detection system learns from the attacks being launched at it. In a repeated game environment, a fraudster can gain information to arrive at a Nash Equilibrium, where they cannot play anything better given the current strategy of the fraud detection system. Because of the lack of information present, this game is called information warfare.

In Vasta et al.'s system, a transaction will go through two tiers before a final decision is made: a rule based tier and a game theory tier. The rule based tier calculates a suspicion score for each transaction; if the transaction fails the rule system, it is then

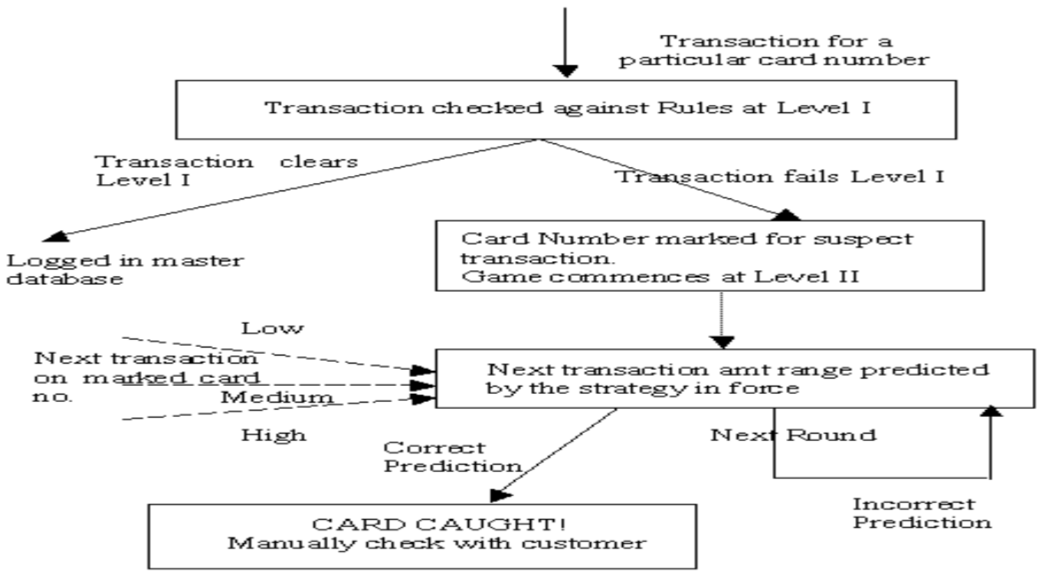

Transaction for a particular card number
Transaction checked against Rules at Level I
Transaction clears Level I
Transaction fails Level I
Logged in master database
Card Number marked for suspect transaction. Game commences at Level II
Low
Next transaction on marked card no.
Medium
High
Next transaction amt range predicted by the strategy in force
Correct Prediction
Next Round
Incorrect Prediction
CARD CAUGHT! Manually check with customer

Figure 2.5: Rule Based Game Theoretic Approach: Search Structure [26]

sent through the second tier. In the second tier, a heuristic is calculated by going through all the potential next moves. The authors model the problem based on parameters about the type of fraudster it is, the strategy space for both parties and the action space for both parties. The structure of how the heuristic is calculated is similar to a decision tree, as shown in Figure 2.5. The benefit of using this system is that if the system thinks that the fraudster is learning its strategy, it can change strategies, or can strengthen its beliefs if it thinks its learning the type of fraudster.

Panigrahi et al. combine game theory with a rule based filter to create a fraud detection system. This system varies from Vasta et al.'s because it has 4 distinct phases. The system begins with a rule based filter, similar to those perviously described. Rules include an address check and outlier detection, where the system measures the difference of the transaction against the transaction's user profile as well as general rules. From this rule based system, transactions are put into three categories: fraud, non-fraud, and suspicious [17].

To combine these rules, Panigrahi et al. use Dempster-Shafer theory, followed by a transaction history database, and a Bayesian learner [17]. The Dempster-Shafer theory assigns a fraud probability to an event based on the classification the transaction has from the rule based system. Dempster-Shafer theory is defined below in Equation 2.4. For example, if a transaction is classified as non-fraud based on address verification, it will have 0 added to its overall fraud probability prediction. Those results are then fed to a transaction history database, which compares the current transaction with the user's past transactions as well as a generic database of fraud.

$$m(h) = m_1(h) \oplus m_2(h) = \frac{\sum_{x \cap y = h} m_1(x) * m_2(y)}{1 - \sum_{x \cap y = \emptyset} m_1(x) * m_2(y)} \tag{2.4}$$

Information about how much the transaction is deviating from normal behavior in 4 mutually exclusive cases is then sent to the Bayesian learner. The first mutually exclusive case is defined as "the occurrence of a transaction on the same card within 8 h of the last transaction," where h is defined as the hypothesis that a transaction is fraud. The other three cases are defined between 8 and 16 h, 16 and 24 h and over 24 h [17]. The Bayesian learner then calculates a suspicion score, which is combined with the earlier suspicion score using Dempster-Shafer again. Based on a threshold, a transaction is then classified as fraud or non-fraud. A visualization of this system can be seen in Figure 2.6. Game theory is incorporated into the rule based filter most prominently in this model. Through their experiments, Panigrahi et al. find that 0.3 and 0.7 are the most accurate lower and upper threshold bounds, respectively, and that their system increased the true positive rate by about 15-20% [17].

2.4.2 Auditing with Game Theory

Game theory concepts can also be leveraged in the auditing context. Wilks and Zimbelman study the current financial auditing methods and offer suggestions for improving systems by adding game theory. The authors describe that there are several levels of

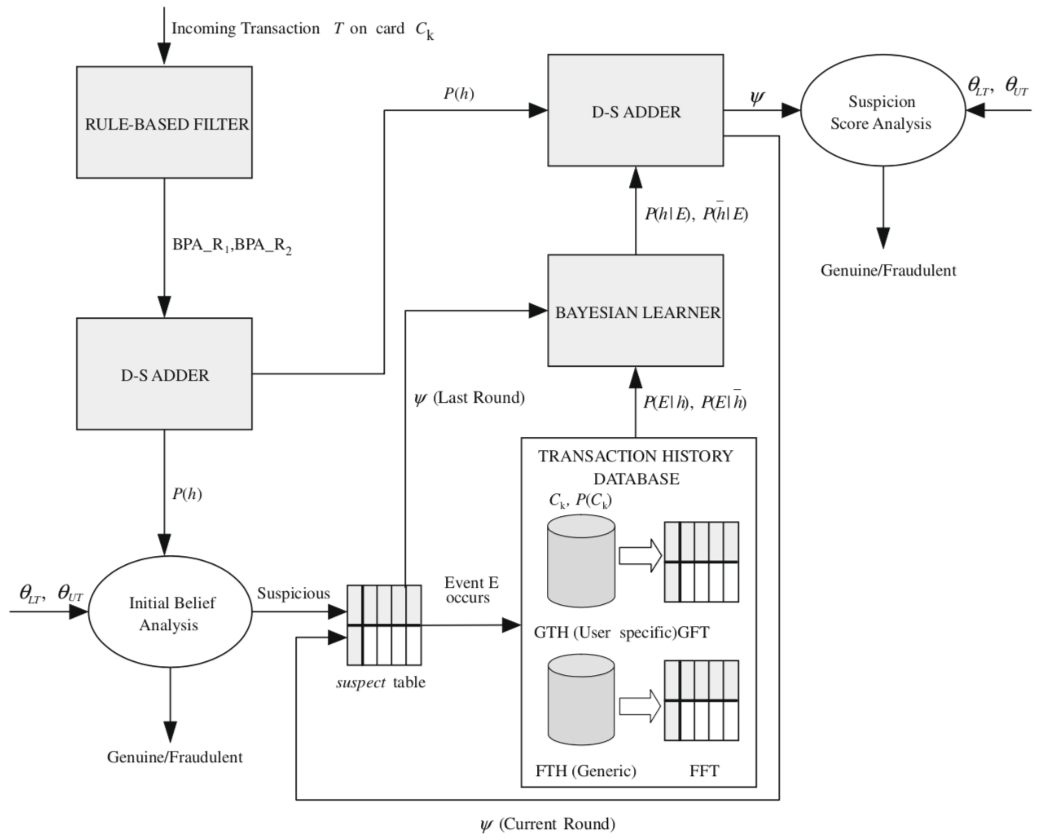


Figure 2.6: Architecture of Rule Based Filter, Dempster-Shafer Adder, Transaction History Database, and Bayesian Learner

reasoning: zero order, first order, and higher order [30]. Zero order logic occurs when a user is only considering conditions affecting themselves. First order is when a user considers some conditions, and higher order is when a user considers multiple levels of reasoning and layers of complexity. The response of a player can change based on the levels of reasoning the player is using. Since fraud and fraud detection are games highly reliant on opponent's moves, strategy can change drastically based on response from the other player. This is inherently difficult because either parties' next move is highly dependent on the other's move.

Currently, the best practices of the auditing field require auditors to go through a checklist of "red flags." While this is very effective for first order reasoning, it does not catch any fraudsters that are reacting to changes in the "red flag" checklist. The authors point out that these checklists have been shown to be less sensitive than intuitive fraud risk models. This is in part because of the dilution effect, where an auditor is focusing on too many irrelevant cues instead of the relevant cues [30].

Instead, the authors suggest decomposing the decision into multiple components, making a judgement on each of these components, and recombining to make a global decision. The three elements the authors contribute to fraud are known as the "fraud triangle": incentive, opportunity, and attitude [30]. The authors also suggest that having auditors meet with their clients and look for in person cues helps auditors learn cues and potentially fraudulent behaviors. Through this process, Wilks and Zimbelman find that accuracy improved in low risk fraud situations, but did not improve high risk situations. They also found that attitude cues are easily concealed, and that auditors often were anchored to their initial judgements, and were unlikely to change their minds given new information.

Through their experiments, the authors offer advice that can be applied to audits as well as credit card fraud detection. The two main components the authors suggest adding are unpredictability and reasoning based on how a fraudster might act. Unpredictability can be achieved by changing the time line of the fraud detection or audit, as well as changing the way the system works from time to time. This unpredictability can help catch fraudsters that were not anticipating an audit, and therefore might not have hidden their work as well [30]. The authors also suggest adding opportunities for auditors to understand how a fraudster may attempt to cover their tracks.

While this is true of humans, it is also true of fraud detection systems, as models also need the opportunity to engage with new fraudulent behavior to learn new patterns. Through incorporating human intuition into computerized systems, fraud detection systems can continue to move towards decreasing the number of analysts in the process of fraud detection. Without it, companies will continue to argue that humans have intuition machines do not, and will fail to lower their overhead costs from investigation into fraudulent transactions.

CHAPTER 3

FRAUD IN CONTEXT

I conducted interviews with four fraud detection professionals in an attempt to understand the differences between related works in fraud detection and companies detecting fraud in the present. I hypothesized that these professionals would have been using systems that are more advanced than the systems discussed in related works. However, I found the opposite to be true; most of the interviewees pointed to simple rule based systems as their main fraud detection method. This may be due to lack of technical knowledge, the computational cost, or that these systems work well enough for their purposes. Through these interviews, I gained more understanding of how fraud is perpetrated and what fraud analysts have found to be most successful in their experience.

3.1 Interviews from Professionals

To gather a sense of fraud detection methods in business contexts, I interviewed four professionals in the industry. Two of the people I interviewed have directly dealt with credit card fraud in their line of work. The third person I interviewed has worked in the British government, then transitioned into insurance fraud detection, and has spent large amounts of time tracing insurance fraudsters. The last interviewee has a doctorate in engineering, but has worked in fraud detection for many different industries. To protect their privacy, I will not be referencing these participants by name or by organization, and will assign them the pseudonyms CC1, CC2, CC3, and CC4. Through these interviews, I found that fraud is handled slightly differently depending on the principles and restrictions of a business.

3.1.1 Interview with CC1

The first credit card fraud expert, CC1, began by explaining the different forms of fraud that a company looks out for. There are two main types of transactions that credit card companies are concerned with: physical and non-physical. There are a plethora of ways to spend money online that do not result in physical object purchases, and each type of purchase comes with its own set of features for detecting fraud. The rules-based method of detecting fraud utilizes the billing address zipcode, CVV, and computer timezone. When purchasing online, the first step is to verify that a CVV does indeed match with the credit card number. After this, one can check that the timezone of the zipcode is close to the timezone the computer is in. While a credit card will be declined if the CVV or the zipcode provided by a business does not correspond to credit card company's stored information, the timezone on a computer is often faked, misleading the fraud detection system.

With physical goods being sent to an address, one way of detecting whether it is fraud is to check if the delivery address is a normal building or residence, and not an abandoned warehouse. Location can also be verified in other contexts such as airline credit cards and banking apps. Airline cards for example can keep information about when you are going to fly places, and not cancel your credit card when you appear in a different country after only a few hours. Banking apps can pull out location information from an app request independent of a transaction to verify that the user is in the same place as their last transaction.

Fraudsters have found sly paths around many fraud detection systems. While it would be optimal to cut off known fraudulent credit cards, new credit card information is continually circulated and sold to fraudsters, making it impossible to find all fraudulent credit cards. However, these fraudsters don't know which cards are currently valid. One way fraudsters have found to solve this problem is through leveraging free trials.

A fraudster can sign up on Spotify or Netflix for example for a free trial. If the card information is good enough to pass these slightly less stringent free trial gates, that means the card information is good to use in other situations. Once the card is used for a nonzero valued transaction, the owner of that card gets the charge. To make these fraudulent dealings look more realistic, fraudsters have started using computers from the area associated with the card and using that computer's geoIP as a proxy for their transactions. Through this process, these transactions look more legitimate because the locations seem plausible.

While fraudsters have technical solutions to help hide themselves, there are methods of detecting this behavior. When a fraudster gets a list of credit cards, they usually use a bot to go through the list of cards and do the process mentioned above. These series of transactions normally take too little time for a human to have completed, and come from the same IP address. CC1's company analyzed these sort of bot transactions, looking for the same bot trying to do similar fraudulent transactions on multiple sites.

Many companies choose to buy fraud detection software rather than developing in house, possibly because of cost or technical capabilities. Banks for example use Falcon software, which analyzes the flow of a user's transactions in order to pinpoint suspicious transactions. Banks are an interesting case because they have such a high volume of transactions that they have data for large analyses and fraud detection systems to train on. Conversely, most physical goods merchants don't use much fraud detection at all and rely on the chip and the fraud detection systems of the credit card company itself. This may be because in the past a merchant would not be responsible for a fraudulent transaction if they had a swipe of a credit card and a signatures.

The complexity of fraud detection systems are also shaped by the risk versus reward of what you are selling. If you are selling diamonds, you may be more concerned with fraudulent transactions than if you are selling Beanie Babies. There are many filters for online transaction fraud such as Retail Decisions [5] and 41st Parameter [1]. These filters check for many red flags, including if the same browser is buying again and again. Companies with longer term transactions have the flexibility to involve customer service members in the fraud detection process, as they have time to verify a transaction for a tangible good. For intangible goods such as Netflix, there is less flexibility as it is an instant decision.

One last example the interviewee mentioned explains how fake physical credit cards are constructed. Once you know you have a valid card, you can magnetically create a card using a hotel room key. While this key does not look real, you can go buy gas with it and if people are not paying attention use it at a store. If you are more sophisticated, you can clone the card using magnetic strip ID card printers. This is much harder to do with the chip.

An online fraud example involves selling iPhones. A fraudulent user will buy an iPhone from an online vendor such as Amazon, and have the phone shipped to them. The fraudster will somehow intercept the package, and claim to have never gotten the package. The fraudster will then receive a refund from the merchant, and the value of the iPhone after selling it. Then, Amazon or the merchant has lost the cost of that item, and the fraudster has doubled their gain. These plans work effectively only if not caught by local agencies or the Counterfeiting and Banking sector of the Secret Service, which is more or less likely depending on the magnitude of the fraud.

3.1.2 Interview with CC2

CC2 is part of a broader fraud detection team at a company with multiple product lines. The specific team works to detect and handle fraud and abuse across all of the company. However, he in particular helps in the credit card fraud detection sector. Since this company is larger, it has the resources to create its own fraud detection systems, which are heavily biased towards applied machine learning techniques. While this is a modern company, according to CC2, they use older machine learning techniques and apply them to new problems. They are not necessarily looking to upgrade their systems when new research comes out. Instead, they look for ways to enhance their system based on the problems that come up. They focus on supervised and reinforced learning for their fraud detection.

Like some of the related work, CC2 emphasizes how fraud and security are very dynamic problems because the problem is always changing. A contrary example to fraud detection is training a model to detect cancer using MRIs and scans. Here, once the classifier is trained, it will consistently have high accuracy and F1 score in its niche. With fraud detection, using a model that has been trained once will not work, as fraudsters will change their behavior accordingly. Therefore, one classifier will never suffice for long periods of time. Because of the heavy interactions with humans with poor intentions, it is best to combine the machine learning with game theory in order to both catch and block fraudsters. For example, while attempting to pursue a fraudster and cut off their fraud channel, there is a risk of losing any trace to that fraudster again. According to CC2, it is easier to trace a perpetrator by not changing models too frequently.

Though model choice can affect results, CC2 argues that it is not the most important element to accuracy gains. There are large differences in the overarching techniques like convergence models, linear models and others, but there is not a particular model that is perfect. Since the problem is changing so quickly, it is best to find the model that fits the problem without trying to find the perfect solution. Instead, the dataset is much more important than the actual model. With changing fraudulent behavior, it is better to implement new parameters by training with a new dataset rather than change the model.

However, there are constraints on the data a company can have as well as privacy concerns. For example, this company does not track users locations for use in fraud detection, and redacts other elements of the data before it ever reaches the fraud detection team. The main goal is to put the fraud detectors in a position where it would be impossible to invade a user's privacy in the future based on the data they have. Instead of being stored on company servers, some of the restricted data can be used by running the models on the device, meaning the fraud team or company as a whole never sees restricted elements of data. These restraints can make fraud detection difficult because the models have to learn user's behavior without seeing the data and gauge their "health" in the system, defined as the potential fraud occurring on a device. Through these constraints, fraud detection becomes an even more difficult problem.

3.1.3 Interview with CC3

CC3 has had a large range of jobs, investigating, analyzing, managing, and developing intelligence in all permutations of fraud and corruption. While he is focused in insurance fraud at the current moment, he has worked in welfare fraud, investment banking "Rogue Trader" profiling, and terrorist funding to name a few. His team and his work in investigating terrorist funding has actually prevented a few bombing attacks on London, emphasizing again the importance of fraud detection.

In the industries CC3 has worked in, he has become a proponent of fraud analytic systems in rule based systems with human input. In his experience rule based systems are less expensive and somewhat more effective than feature based systems. Feature based systems are expensive because of the work to continually increase and change the feature space. Rule based systems are easy to change and with a field investigator looking over the results is a highly effective and inexpensive solution. Fair Issac Company (FICO) for example has a rule based engine for its fraud products [3]. For any of these systems, in Britain, there is a large repository of information about people which is incorporated with census data. This information can be leveraged by British companies for fraud detection. While insurance fraud is different than financial fraud because detection is always after the fraud has occurred and financial fraud is trying to be stopped before the transaction is approved, they still have of overlapping methods.

3.1.4 Interview with CC4

CC3 referred me to CC4, who has done work in the behavioral analytics space. Her form of fraud detection used behavioral analytics to try to spot fraud before the fraud even occurs. The system measures the likelihood of someone committing fraud at a high accuracy, and the system was able to reduce deductions in payouts for insurance claims by about 12%. However, this sort of system is the “Rolls Royce of fraud detection” as it is very expensive according to CC3. In insurance fraud expert’s experience, organized crime will employ their best people to combat the fraud detection systems and threaten people on the inside to gain information about the system and achieve their goals. Therefore, it is quicker to get a rule based system up to speed in the short term, though feature based systems would most likely pay off in the long term.

In addition to the behavioral analytics, CC4 has also worked in banking, insurance, and online gambling fraud. Each of these fields are somewhat different because of the levels of security and data. Banking is probably the most traditional and secure sector. They are very slow to grant access to their data and need to anonymize all data before taking it out of their premises. While all industries have certain levels of regulations, banking is particular strict with their security and users’ privacy. However, because of the amount of transactions banks handle, they can provide researchers with much more data than insurance. In addition, for regulatory reasons, if a bank predicts a transaction to be fraudulent, there must be a reason for this prediction, which can be difficult to describe when using machine learning.

Insurance and gambling are slightly different than banking. One of the major issues with fraud detection in insurance is that sometimes there are no fraudulent transactions in a data set, and the actual amount of data is small. To get more data, insurance companies are attempting to get users to use their apps and extract more information from there. Gambling is different than either of these fields because it is the most lenient and have more relaxed laws than banks or insurance companies. Because of this, analysts in this space can work faster to detect fraud with less constraints. Most techniques are transferable between these industries, but it is still important to have experienced people working on the problem, as CC3 also referenced.

3.2 Research Versus Reality

Through talking with these professionals, one point that was frequently emphasized was the cost of false positives versus false negatives. This was the topic of discussion in many of the papers I read, and the professionals I talked to continued to stress the point. What I learned is that companies generally accept some amount of fraud in their business, with about 1% of transactions being an acceptable amount of fraud for a company. While 1% is quite small, this is a large amount of money when you consider some companies may be having millions to billions of transactions go through their system every day. Because of this, it seems that using a cost equation based on cost of the size of a transaction versus the number of correct and incorrect transactions would be beneficial. This way, the system would learn to pick up on high valued fraud instead of many small transactions.

The experts I interviewed all incorporated more human insight into their detection system than much of the research would suggest. Based on the results the related work has achieved, it may seem that companies would be able to fully automize their fraud systems. However, human intuition has continually trumped automatic systems according to the four interviewees. Human involvement seems to have three factors associated with it: intuition, cost, and regulations. The intuition a human can bring to a fraud detection system relates to game theory. Fraudsters and analysts are essentially playing a game where both parties are profit maximizing, and are in a cat and mouse situation. Using human logic about how a fraudster will act is something only expensive machine algorithms may be able to do. A human can set up a rule based system using knowledge from past fraudulent behavior and expectations of where weaknesses in the system might lie. A behavioral machine learning algorithm is more computationally expensive, meaning that companies may not want to invest the money into a complex system.

Lastly, humans will most likely always have to be part of the process because of regulations. When a customer is refused for a loan or creating a new credit card, there must be a reason associated with that decision to ensure there is no bias in the institution [2]. Unlike somewhat blackbox machine learning algorithms, a fraud analyst can easily point to how a rule based system created a denial decision. Through these interviews, I have found the reverse of what I expected: companies are actually not keeping up with newly researched methods but are actually opting for less expensive systems that have been in place for years.

CHAPTER 4

IMPLEMENTING A FRAUD DETECTION SYSTEM

To understand the complexity of creating a highly accurate fraud detection system, I implemented my own system based on the approaches discussed in Chapter 2. I constructed a series of machine learning models, and reported accuracy and F1 scores based on different proportions of fraud to non-fraud data. For all experimentation, I used open source tools built for Python. I will discuss the model and the experiment in depth in the sections below. The overall implementation is broken down into subsections under Experiment: Hypothesis, Data, Methods, Results, and Discussion.

4.1 Model

For this project, I implemented several different machine learning models. I followed Perols et al. and will be comparing the performance of several popular classification models [19]. In this section, I give an overview of the methods I use in my experiments.

Linear regression is a way to estimate the best fit line through a set of points. The best fit line is the line that minimizes the squared error of each point in relation to the estimation on the line. This algorithm is used to create a relationship between points, and to analyze the relationship between points [24]. As more features are added to

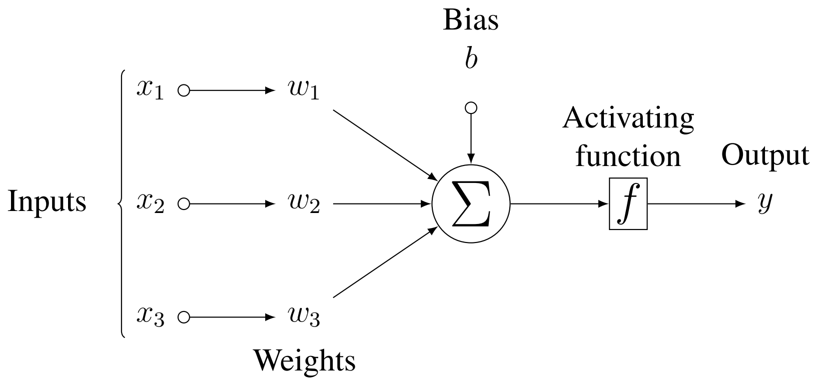


Figure 4.1: Logistic Regression

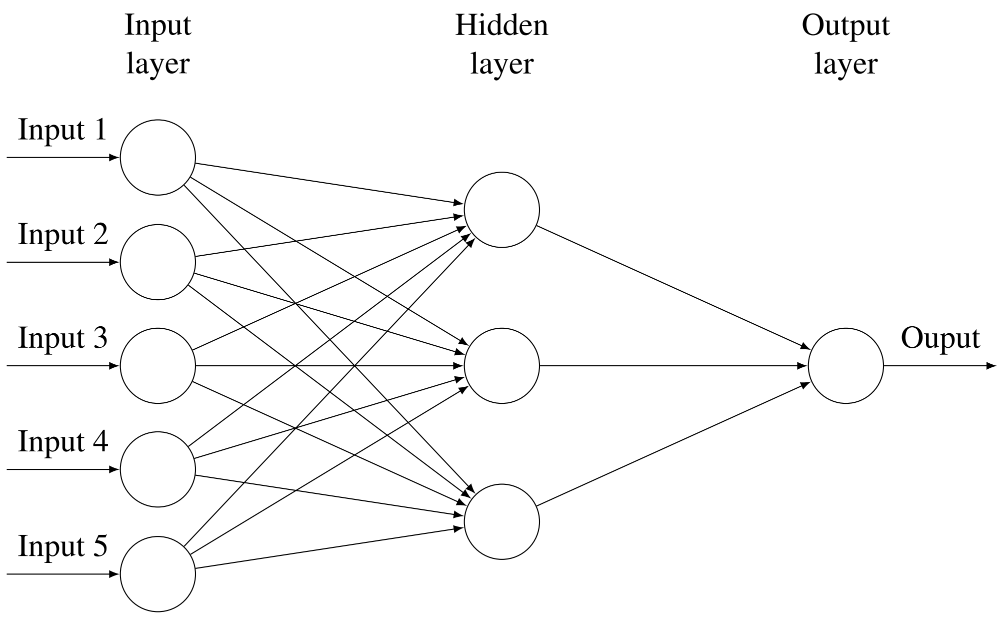


Figure 4.2: Neural Network

the estimation, the regression increases in complexity. Logistic regression is different from linear regression only in the equation used to estimate the relationship between the points. Logistic regression is a curve that has limits at 0 and 1. Along the curve represents probabilities of an event given another feature. For example, logistic regression can model studying for a test: based on the number of hours you study, you can calculate the likelihood of passing that test. It is an apt method for classifying a binary variable because it calculates the probability that something is equal to 1 based on the given inputs. A diagram of logistic regression can be seen in Figure 4.1.

Artificial neural networks leverage logistic regression to build systems that can solve more complex problems. Neural networks are created by combining multiple nodes in different layers to create a network [23]. Here, nodes represent logistic regression, as they take in inputs, estimate a prediction based on those inputs, and output an answer. The general structure of a neural network is to have an input layer, one or more hidden layers which can be of any size, and an output layer, as seen in Figure 4.2. Information is propagated forward through the network, and then back propagated again to adjust the parameter estimates of each node. Through this forward and backward process, the system eventually reaches a minimum, where the difference between the predictions and the real values is minimized. This system is extremely powerful, and can be tuned based on learning rate, hidden layer numbers, activation functions and other features of the network.

Decision trees make decisions by branching at features and predict once a terminal node is reached. Each parent node represents a decision or feature and the children nodes represent the potential outputs from that node. Cost functions are used to find the most homogeneous branches, or branches with similar inputs. Decision trees can grow arbitrarily large, and sometimes require pruning to avoid developing a large tree. In addition, decision trees can be combined to create decision forests, further enlarging the architecture and complexity of the learning model [16]. Like neural networks, there are also many parameters to tune, and architecture elements to adjust such as maximum depth and maximum leaf nodes.

A slightly different approach to this classification problem is using probability distributions such as a Gaussian distribution. This method takes advantage of the large class skew in the data, as there are few instances of the minority class. Therefore, assume all the data you have is the majority class. If this data is approximately normally distributed, then you can calculate the probability of a new example being part of the majority class. Since anomalies are presumably located far from the mean, they have low probabilities of being part of the majority class. The last step of this approach is to choose a threshold for how likely an example must be to be classified as an anomaly.

The last classification method I implemented was support vector machines or SVMs. SVMs classify data into categories by finding the most accurate boundary hyperplane to divide the data [12]. Here, we use kernels to calculate the plane or division that best splits the data with the lowest error rate. Like neural networks, to find the most accurate split between the data, we implement stochastic gradient descent to find the minimum cost points. Once this division is created, predictions are made based on the location of a point of interest in regard to the line dividing the data.

4.2 Experiments

I experimented with the classification algorithms listed above to model the best classifying algorithm possible. The experiments were two fold. I experimented with the distribution of the data and with the parameters and hyper-parameters of the data. In Perols' paper, he finds the best results when looking at an equal split of fraudulent and non-fraudulent transactions [19]. I wanted to replicate his study by altering the proportion of fraud to non-fraud in the training distribution, and compare results. I also experimented with the structure of the models themselves. As discussed above, each model has several elements that can be tuned to find the best results. Neural networks in particular can take many forms, and require certain levels of manual experimentation and tweaking to find the best results. Through these two axis, I compared the performance of each model both to other models in this thesis and performance in Perols' paper. [1]

4.2.1 Hypothesis

Based on Perols' paper, I hypothesize that the best results will come with more equally distributed data. I think that neural networks will perform best because they are able to capture patterns in the data unseen by an analyst, and may perform better than the other models I have selected. However, given enough data and the correct tuning, all of the

[1] All code located at: https://github.com/castiz/CSThesis

tested models should theoretically approach the same accuracy levels.

4.2.2 Data

The data used from this project was obtained from Kaggle.com [18]. I used the "Synthetic Financial Datasets for Fraud Detection" which was generated by the PaySim mobile money simulator and is scaled down to a quarter of the original dataset, which was approximately 25,000,000 observations [18]. The dataset includes attributes about type of the transaction, amount, the customer that started the transaction, the initial balance of the sender, the balance after the transaction of the sender, whether its fraud, and other variables. The summary statistics are shown in Table 4.1, and example of the information displayed in one transaction is displayed in Table 4.2.

Limitations

This project was limited by the dataset I had access to. In the real world, a transaction would include much more information that is not available with the fields of Paysim. For example, banks and credit card companies have the ability to see exactly where a transaction took place. As discussed in Chapter 3, banks and companies have strict policies they have to follow in regards to sharing user data. Therefore, there are no readily available datasets online. For reference, in the paper by Chan et al. referenced above, their dataset had 30 fields per transaction [13]. Some banks pull information from their own banking app on a user's phone and compare the location of the user's phone to the location of the past transaction.

With credit card information, companies can detect fraud based on whether transactions are occurring in two places very far away from each other. Since I do not have any sort of location information, I cannot leverage this type of rule based fraud detection. In addition, I do not have access to the name of the business a transaction is taking place at. By building out a user profile that includes frequent locations and businesses the user goes to, a detection system can detect if a person is shopping at a store that is out of the norm and calculate the likelihood that this is fraud. Lastly, there are essentially no repeated users in this dataset, eliminating the possibility of creating features based on user profiles.

Table 4.1: Summary Statistics

Transaction Measure	Mean	Min	Max
Legitimate Transaction			
Amount	178,197	0.01	92,445,516
Count = 6,354,407			
Fraudulent Transaction			
Amount	1,467,967	0	10,000,000
Count = 8,213			
Cash In (Deposit)			
Amount	168,920	0.04	1,915,267
Count = 1,399,284			
Cash Out (Withdrawal)			
Amount	176,273	0	10,000,000
Count = 2,237,500			
Debit			
Amount	5,483	0.55	569,077
Count = 41,432	17.54		
Payment			
Amount	13,057	0.02	238,637
Count = 2,151,495			
Transfer			
Amount	910,647	2.60	92,445,516
Count = 532,909			

Table 4.2: Example Transaction in PaySim

Transaction Number	Category
Hour	
Transaction 1	1
Type	
Transaction 1	Payment
Amount	
Transaction 1	$9839.64
Name Origin	
Transaction 1	Alice
Old Balance Origin	
Transaction 1	$170,136.00
New Balance Origin	
Transaction 1	$160,296.36
Name Destination	
Transaction 1	Bob
Old Balance Destination	
Transaction 1	$0.00
New Balance Destination	
Transaction 1	$ 0.0
Is Fraud	
Transaction 1	0

4.2.3 Methods

In order to implement my models, I leveraged the Python package Sci-kit learn. Sci-kit learn has readily available implementations of major classifiers that I am interested in testing, as well as the infrastructure for predicting and analyzing the results of the models. I use the Python package Pandas to read in and manipulate my data in a data-frame format. For experiments where I am changing the distribution of fraud to non-fraud transactions, I extract the two transaction types from the total dataset. I then randomly sample a portion of the non-fraudulent transactions based on that experiment, and recombine that dataset with the entirety of the fraudulent dataset. This random sampling attempts to avoid potential selection bias in my results.

Once my data is cleaned, I randomly split the data into training sets and test sets. I reserve one seventh of my dataset for testing, about 14.2%. I do this to ensure my model does not learn the patterns of the tests, so I can test the accuracy of the model when exposed to new information. With my data prepared, I then run my series of tests with different classifiers and different architectures. The results are shown below.

For each model, I used a random sample of 20% of my data and altered the parameters and hyperparameters. Examples of this include changing the learning rate for the neural network model or choosing the max depth of the decision tree classifier. One of the main parameters I was interested in was the learner formula, which is the actual equation for which weights are calculated. For example, I found neural networks to be most accurate with the stochastic gradient descent solver and a ReLU activation function. With logistic regression, I found the LBFGS solver to be much better than SAG or the other solver options. This sort of guess and check method may seem inefficient, but is one of the only ways to tune many machine learning algorithms.

Testing the neural network model is slightly more complicated, as results can drastically differ based on the architecture. Neural networks are constructed of an input layer, output layer, and as many hidden layers in between as desired. Both the number of hidden layers as well as the number of nodes within each of the layers can be specified, growing or shrinking the model in complexity. Other than the architecture, I also tested the epoch or max iterations parameter. An epoch is one forward propagation of data and one backward propagation. Each time data cycles through the network, the weights adjust and tune to the previously seen examples. In addition to the other experiments, I analyzed the best number of epochs for the fraud classification problem. The results shown in the results section will be the metrics from the best architecture, as there were many trials with similar results.

While Gaussian probability distributions are effective at anomaly detection, much of the related work in this field have leveraged standard classifiers as their most successful models. In order for these models to work correctly, they must see many examples of all classes of data in order to make accurate predictions. Since fraud detection is a binary problem, it is logical to assume a classifier will do best in an environment where it is learning with 50% fraud and 50% legitimate transactions. However, this is not always attainable in the real world, so I test with multiple data proportions, with fraud making up between 50% of the transactions to less than 1% using the unaltered dataset.

In addition to testing the models, I also test whether normalizing the data increases the accuracy of the classifiers. Normalizing takes away bias due to magnitude in features, and also allows for direct comparison between features. The normalization equation can be found in equation 4.1.

$$Normalized(transaction) = \frac{(transaction - min(transaction))}{(max(transaction) - min(transaction))} \tag{4.1}$$

The most common method of measuring whether a model is improving or worsening is accuracy. As discussed above, this is helpful in more balanced class situations, but is biased when working on anomaly detection. Because of this, I include precision, recall and an F1 score of each model in addition to the accuracy. The equations for precision, recall and F1 are shown in equations 4.2, 4.3, and 4.4, respectively.

$$Precision = \frac{TruePositives}{(TruePositives + FalsePositives)} \tag{4.2}$$

$$Recall = \frac{TruePositives}{(TruePositives + FalseNegatives)} \tag{4.3}$$

$$F1 = \frac{2(Precision * Recall)}{(Precision + Recall)} \tag{4.4}$$

As seen in the equations above, these measures are focused more on the proportion of true positives, false positives, true negatives, and false negatives. Precision describes a ratio of correctly identified fraud to all transactions identified as fraud. Recall is the proportion of correctly identified fraud to all identified as fraud. The F1 score is a measure that incorporates both precision and recall, and is a helpful tool for determining a system's bias towards precision or recall. As discussed in Chapter 1, depending on the industry, false positives are often more expensive than false negatives. This is because it is sometimes less expensive to pay out for small levels of fraud rather than lose a potential client because of suspected fraud. I do not analyze my results using a cost model, but do use both the accuracy and F1 score.

4.2.4 Results

The tuned models and their metrics are shown in the following tables. Table 4.3 shows the results using non-normalized data. Table 4.4 shows the results using normalized data.

The results shown in Table 4.3 show that decision trees had the highest F1 score and accuracy in most trials. This shows that my hypothesis was incorrect, as the decision

Table 4.3: Results: Non-normalized Data

Method	Accuracy	Precision	Recall	F1 Score
Original data distribution				
Logistic Regression	0.999	0.3735	0.493	0.425
Neural Network	0.999	0	0	0
Decision Tree	0.9997	0.8974	0.903	0.899
SVM	0.004	1	.0013	0.003
Gaussian	0.952	0.532	0.015	0.029
50/50 data distribution				
Logistic Regression	0.910	0.002	0.902	0.004
Neural Network	0.0002	0.0002	1	0.0004
Decision Tree	0.989	0.017	0.995	0.033
SVM	0.002	0.0002	1	0.0004
Gaussian	0.491	0.0003	0.856	0.0006
66.6/33.3 data distribution				
Logistic Regression	0.910	0.854	0.885	0.869
Neural Network	0.999	0	0	0
Decision Tree	0.993	0.029	0.994	0.055
SVM	0.002	0.0002	1	0.0004
Gaussian	0.509	0.0003	0.857	0.0007
75/25 data distribution				
Logistic Regression	0.956	0.004	0.841	0.007
Neural Network	0.999	0	0	0
Decision Tree	0.995	0.036	0.997	0.070
SVM	0.002	0.0002	1	0.0004
Gaussian	0.505	0.0003	0.882	0.0007
80/20 data distribution				
Logistic Regression	0.971	0.005	0.809	0.011
Neural Network	0.999	0	0	0
Decision Tree	0.996	0.048	0.981	0.091
SVM	0.002	0.0002	1	0.0004
Gaussian	0.738	0.0005	0.617	0.001

tree with the original dataset fraudulent proportion performed best. It seems that when the training data contained more fraud proportionally, the decision tree, as well as the other classifiers became heavily skewed towards predicting fraud. While the altered training set did allow other classifiers, like the neural network, to be more sensitive to fraud, it overall increased the false positive rates.

In contrast, SVM's performed poorly, averaging less than a 1% accuracy. The low accuracy, combined with the high recall, implies the SVM only predicted fraud for all transactions. This may be because of the difference between the training dataset and the test dataset. The SVM may be able to find a division between fraudulent and non-fraudulent transactions during training, but this may not apply well to the whole dataset.

Table 4.4 shows the results of experiments performed on normalized data. During the experiments, it seemed that normalizing the data made improvements to most model's behaviors. However, in these experiments, normalizing the data led to lower results for almost every model, as well as heavy biases in the system. These biases can be seen in the very low precision, where the classification model is either predicting no fraudulent transactions or all fraudulent transactions. Based on the precision and recall, it seemed that normalizing the data led the models to either skew towards predicting only fraudulent transactions or skew towards predicting all fraudulent transactions. The only model to improve was the SVM's, which became less biased towards predicting only fraud.

Figure 4.3 shows the true positive rates as the fraud data proportion increases in the non-normalized run. Each of the models had increases in true positive rate as the data proportion increased, though by different magnitudes. The most drastic increase was in the neural network, where the model biased towards predicting fraud. As seen in Figure 4.4 and shown in Table 4.5, the false positive rates also tend to increase as the true positive rates increase. The only model that decreases in false positive rate with

Table 4.4: Results: Normalized Data

Method	Accuracy	Precision	Recall	F1 Score
Original data distribution				
Logistic Regression	0.999	0	0	0
Neural Network	0.999	0	0	0
Decision Tree	0.999	0	0	0
SVM	0.999	0	0	0
Gaussian	0.999	0	0	0
50/50 data distribution				
Logistic Regression	0.501	0	0	0
Neural Network	0.501	0	0	0
Decision Tree	0.499	0.499	1	0.666
SVM	0.534	0.536	0.499	0.516
Gaussian	0.499	0.499	1	0.666
66.6/33.3 data distribution				
Logistic Regression	0.670	0	0	0
Neural Network	0.670	0	0	0
Decision Tree	0.330	0.330	1	0.497
SVM	0.330	0.330	1	0.497
Gaussian	0.330	0.330	1	0.497
75/25				
Logistic Regression	0.750	0	0	0
Neural Network	0.750	0	0	0
Decision Tree	0.250	0.250	1	0.400
SVM	0.250	0.250	1	0.400
Gaussian	0.250	0.250	1	0.400
80/20				
Logistic Regression	0.801	0	0	0
Neural Network	0.801	0	0	0
Decision Tree	0.199	0.199	1	0.332
SVM	0.801	0	0	0
Gaussian	0.167	0.120	0.500	0.192

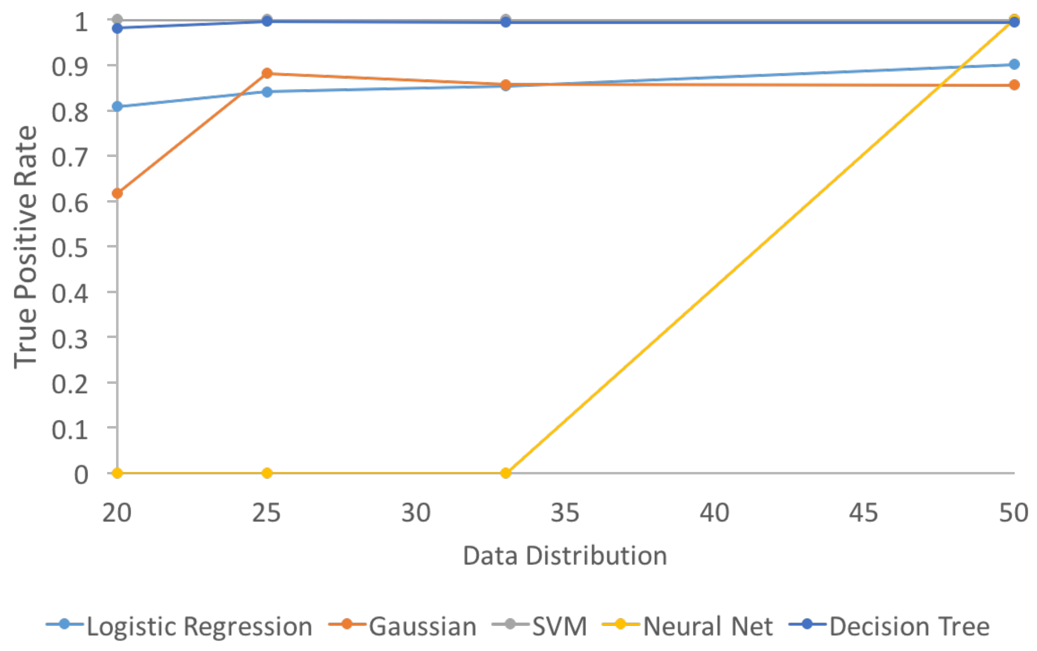


Figure 4.3: True Positive Rates by Data Proportions

more fraudulent data examples is the SVM.

While most of these false positive rates are increasing only slightly in magnitude, that small increase multiplied by 6 million transactions results in a large absolute number of false positives. For the increase of 0.06 in false positive rate from the logistic regression model, there would be an additional 360,000 false positives. To return to related work, if a company has low overhead, increased false positives are not as expensive as if a company has very high overhead [13]. Therefore, the increased sensitivity to fraud might be beneficial.

Table 4.5: Results: Non-normalized Data False Positive Rates by Model

Data	Logistic Regression	Gaussian	SVM	Neural Network	Decision Tree
50/50	0.090	0.509	0.995	1	0.011
60/33	0.057	0.491	0.998	0	0.007
75/25	0.044	0.495	0.998	0	0.005
80/20	0.029	0.262	0.998	0	0.004

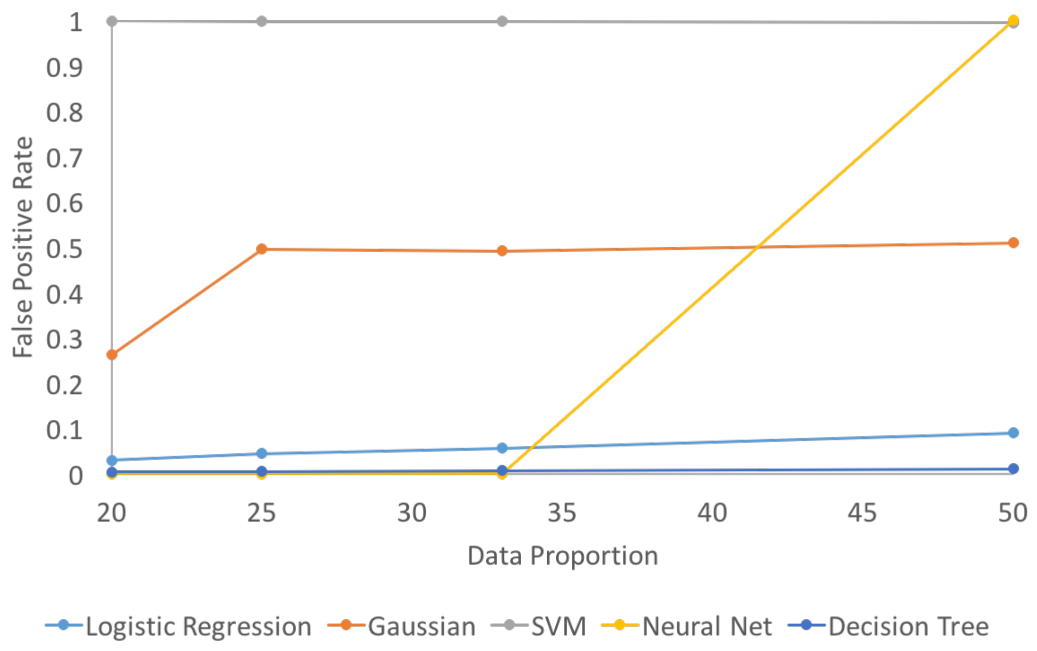


Figure 4.4: False Positive Rates by Data Proportions

4.2.5 Discussion

It is possible that the data divisibility impacted the results of the paper. Classifiers perform best when the data is easily seperable between the data classes. When the data in separate classes are overlapping, then it is more difficult to fit a classifier that is accurate while also not overfitting to the data. If data is not separable, a classifier may have difficulty distinguishing classes. Based on where the classifier creates a division between classes, it will likely predict many examples incorrectly because of the lack of pattern in the data. Figure 4.5 is a visualization of data that is not easily separable. Here, it is easy to tell that there is no line that can be drawn to divide the green dots from the red dots.

In Figure 4.6, two principal components of the data are shown, with fraudulent transactions depicted in red. This figure shows that the classifiers may have had difficulty identifying fraud because of the lack of separability in this data. There seems to be a grouping of fraudulent transactions on the left section of this graph, but these are mixed

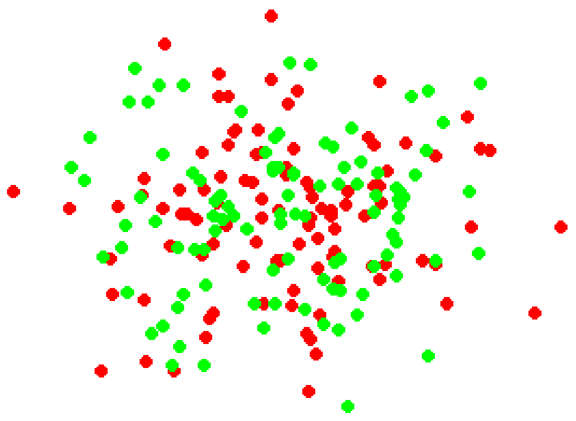


Figure 4.5: Inseparable Data Example [6]

in with legitimate transactions. This means that there is not a clear line a classifier could draw to divide fraudulent and legitimate transactions.

Based on the results in Table 4.3, decision trees perform the best in almost all the scenarios. Using the non-normalized data and an unskewed dataset, training with a 1/3 proportion of fraudulent data has created the best results, with a 50/50 data proportion being a close second. In many of the interviews discussed in Chapter 3, practitioners discussed rule based systems as popular fraud detection systems. Rule based systems are very similar to decision trees, and can be implemented as decision trees. These results are consistent with interviews, and demonstrate that rule based systems are likely effective, contrary to my hypothesis.

While most models performed satisfactorily in these trials, the SVM's consistently performed worse than the other models in all trials. I hypothesize this may be because of the poor separability of the data. SVM's attempt to divide the data with the largest margin of error possible. If there is no clear division, the SVM cannot make an accurate division. In addition, when changing the data proportions, the SVM may have found a division between fraud and legitimate transactions that was present in the training data, but not in the testing data. This is further evidenced by the high recall of most of the

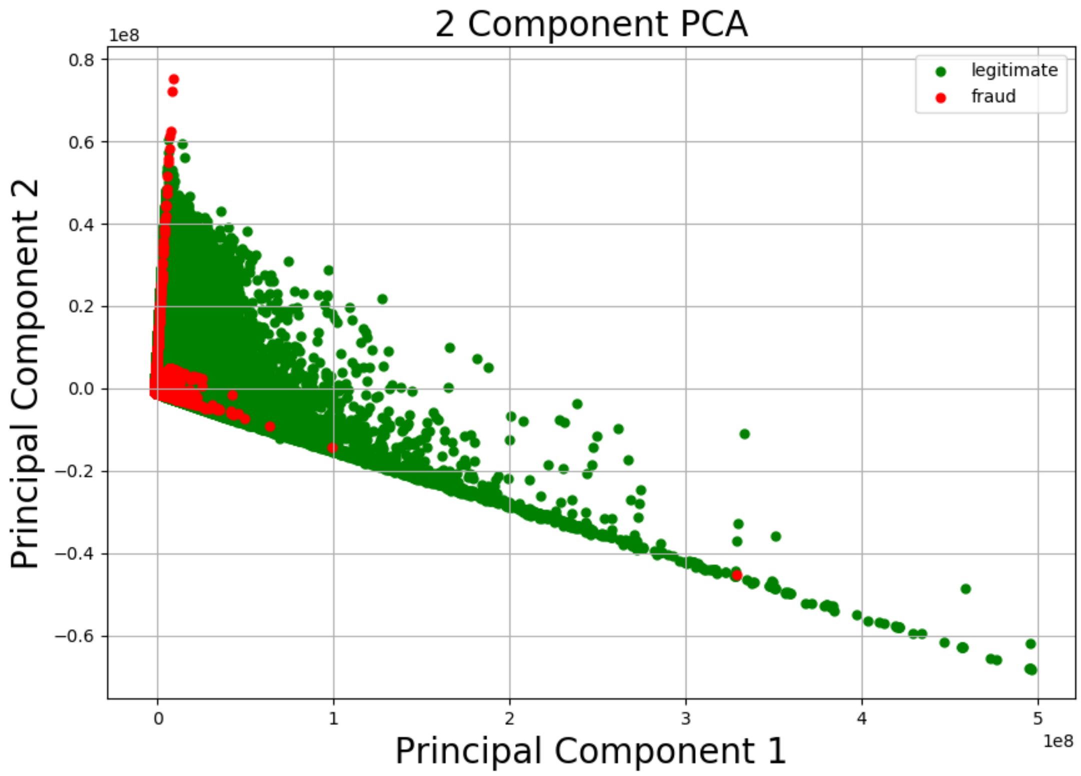


Figure 4.6: Principal Components Analysis of Data

SVM runs, showing that there were no false negatives for any of the SVM runs with training data manipulation. SVM's in all of these cases were predicting all fraudulent, biasing heavily towards the minority class. When the data was normalized, the SVM's became less skewed, but most runs still biased towards predicting fraud.

Through these experiments, I have found decision trees to be the most successful in classifying fraudulent transactions in this setting. My initial hypothesis was incorrect; neural networks did not perform well in this setting, and easily biased towards predicting one class or another, regardless of the architecture. A lesson from this is to always choose a classifier based on the problem at hand, not by the supposed complexity of a model.

CHAPTER 5

FURTHER APPLICATIONS

The challenge in credit card fraud detection comes from the lack of examples of fraudulent behavior; this is because fraud is an anomaly. The work done in this thesis is framed by previous anomaly detection research as well as credit card fraud-specific research. Because of this, the implementations of credit card fraud detection systems can be applied to other fields of anomaly detection with slight alterations. In this chapter, I describe other fields in which anomaly detection techniques are applied.

5.1 Anomaly-Based Intrusion Detection

Anomaly detection techniques can be used to find instances of intrusion attempts into a system, and have been leveraged in the network space. Intrusion detection systems monitor a network or system for malicious activity, and often use machine learning to detect this malicious traffic from good or policy following traffic. Wang and Stolfo analyze payloads in order to detect anomalies [27].

Their system used unsupervised learning to determine the distribution of byte frequency of a single application payload between a single host and port. This distribution and standard deviation is learned during a training phase. They then use Mahalanobis distance to calculate simiarlity of new data against the profiles computed during the training period. If a new input exceeds the threshold they set, then an anomaly is detected. While testing on 1999 DARPA IDS data, the system achieves an almost 100% accuracy with a 0.1% false positive rate [27].

5.1.1 Biometric Anomaly Detection

In a study by Ahmed and Traore, biometrics are used to detect system intrusion at the user level [9]. This system uses behavioral biometrics, which include patterns of a person's physical movements, to identify a user. This is different than physiological biometrics, in which a user is identified by a feature such as a fingerprint or iris.

In the enrollment phase of this model, the client machine runs software that monitors a user's mouse movement and keystroke data. This information is then feed to a server, which computes a profile that is used for behavior comparison in the verification stage. Using the mouse and keystroke data, Ahmed and Traore create dynamic signatures using neural networks, which capture features that define a user uniquely. During the verification stage, the sum of the absolute difference between the expected user's profile signature and the current user, and a decision is made based on a predefined threshold.

5.2 Hardware Health Monitoring

One common use for anomaly detection is in monitoring health of a system, including the health of hardware. In the Aerospace field, anomaly detection has been use to monitor the health of liquid-fueled rocket propulsion testbeds. Schwabacher et al. used four different algorithms to test on historical data from the Space Shuttle Main Engine and an experimental rocket engine at NASA Stennis Space Center [20]. The four approaches used are as follows.

The first approach defines an anomaly as a point that is far from its closest neighbor, using a nearest neighbor to calculate each point's nearest neighbor. In the next algorithm called the Inductive Monitoring System, system clusters are learned from training data, then distance to the nearest cluster is used to determine to measure its anomalousness. The third system they use is similar to a decision tree classifier, as it learns rules from the training data and marks a point as anomalous if they violate these rules. The last algorithm the authors implement is an SVM. These methods all correspond to those used in the credit card fraud detection, and have similar benefits of reducing the number of human experts needed to analyze spacecraft sensor data.

5.3 Bioinformatics

Bioinformatics is a field of study concerned with analyzing complex biological data, in particular genetic codes. Existing research covers detecting anomalies in discrete and symbolic sequences, as surveyed by Chandola et al. [14]. There are 2 different methods that are commonly used for this bioinformatic anomaly detection problem. The first method is unsupervised, where anomalous sequences are detected from an unlabeled database of sequences.

The other version is called semi-supervised. This involves training a model on a database that is assumed to be all normal sequences, then testing to see how different the example sequence is from the training examples. This model is semi-supervised because all examples are assumed to be normal sequences, but no labels are used. Here, a popular method is also to use k-nearest neighbor and clustering, similar to the methods discussed above.

5.4 Cloud Computing

Cloud computing systems have become more popular in the recent years. Because of this increased use, dependability assurance has become a very important issue in system design and management. Song Fu creates a framework to automatically detect anomalies in cloud computing [21].

In this framework, principal component analysis is applied to a number of performance related metrics or features. This allows for a reduced metric dimension, but keeps the variance in the health related data. This reduced space is then fed through a semi-supervised decision tree classifier, which further reduces the dimensionality and identifies the anomalies [21]. This is a very similar method to the decision tree classifier built in Chapter 4. Even though cloud computing and credit card fraud are very different topics, almost identical frameworks can be used to solve both problems.

CHAPTER 6

CONCLUSIONS

Credit card fraud is growing with the advent of new payment methods and online transactions. While not everyone falls victim to these attacks, there are negative impacts to all from fraudulent transactions. These transactions directly impact users through losses passed on to customers. Increased fraud can also decrease trust in the overall financial system. There is also a negative societal impact, as money gained from fraudulent transactions are often used for illicit purposes. Because of these reasons, it is worth building effective fraud detection systems.

Fraud is particularly hard to detect because it is an anomaly, and only makes up about 1-3% of transactions. Thus, there are not many examples of the fraudulent class in order to train classifiers with. There are several approaches to deal with this challenge, such as fitting probability distributions to the data or changing the proportion of the training dataset. As described in related works and shown in the results in Chapter 4, this method helps build classifiers that are more sensitive to fraudulent transactions, but may also increase false positive rates. With low overhead costs, this trade off is worthwhile, and will decrease overall costs from fraud. If overhead is high, costs may not decrease and could possibly increase due to higher sensitivity to fraud.

After performing a series of experiments using synthetic mobile money data, my results indicate that decision trees perform the best for fraud detection. The decision tree model performed the best in all trials in the non-normalized data experiments, and tied with the Gaussian model in the normalized experiments. This implies that rule based systems, which can be implemented as decision trees, are probably better suited for the fraud detection problem than I hypothesized. In my second set of experiments, I found normalizing the data mostly decreased the results from almost all models except the SVM. Normalizing seemed to cause heavy bias towards predicting fraud, leading to high recall but low accuracy. In other cases, normalizing the data led to predicting no fraud, with zero true positives detected in the data.

6.1 Future Work

In this thesis, I was limited by the data I had access to. I would have liked to use more realistic credit card transaction data that included repeated users as well as location information. As future work, I would have liked to try some feature engineering based on historical information about a user. For example, I would have created a feature based on whether or not a transaction was above a user's average transaction size. This would have emphasized when transactions were abnormally high for a user, indicating fraud.

In addition, I would have liked to create features based on time between transactions of a user. If two transactions are too close together, it is unlikely that a user made both of these transactions. This paired with location data could indicate whether a series of transactions was even possible to be made by the same user. Lastly, I would create a feature indicating whether a user had ever transacted with the destination user before. It might be an indication of fraud if someone is transacting with another user or store that is outside of their normal behavior. Through these features, the classifiers would have had information about how far the transaction is straying from past behaviors of that user in addition to how different the transaction is to other legitimate transactions.

Based on the results from this experiment, an extension of my fraud detection system could involve combining a rule based system with a more sensitive classifier. Since decision trees were the best suited for this problem, I would have liked to try building out a rule based system based on some of the rules reported in the previous literature. In this architecture, all transactions would filter through the rule based system first, potentially eliminating transactions that are certain to be legitimate and certain to be fraud. This would leave the uncertain cases, which a classifier would then train on. This architecture is similar to Panigrahi et al., which combines the results from the different elements using Dempster-Shafer Theory [17].

With more realistic credit card data, I also could have used some of the network analysis approaches for feature engineering discussed in the related work. There has been much research into feature engineering using deep feature analysis or even social network analysis. These approaches mainly look at patterns in the behavior of users in relation to their own past transactions and other users. Social network analysis looks at how a user is acting in comparison to other members in their community or immediate circle of acquaintances [10]. Future work on this topic would involve implementing some of these systems, and incorporating them into the experiments ran in this thesis.

BIBLIOGRAPHY

[1] Experian and 41st Parameter: Preventing fraud and enabling sales. Available at http://www.experian.com/decision-analytics/41st-parameter.html.

[2] FDIC Law, Regulations, Related Acts - Consumer Financial Protection Bureau. Available at https://www.fdic.gov/regulations/laws/rules/6500-300.html.

[3] FICO® Application Fraud Manager — FICO®. Available at https://www.fico.com/en/products/fico-application-fraud-manager.

[4] Fraud - Definition, Types, Examples and Processes. Available at https://legaldictionary.net/fraud/.

[5] Retail Decisions, Inc.: Private Company Information - Bloomberg. Available at https://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=4527922.

[6] Vision, Learning and Robotics: Feature Normalization for Learning Classifier. Available at http://surenkum.blogspot.com/2013/03/feature-normalization-for-learning.html.

[7] Most popular payment methods in the U.S. 2017 — Statistic. Available at https://www.statista.com/statistics/568523/preferred-payment-methods-usa/.

[8] U.S. payment card fraud losses by type 2018 — Statistic. Available at https://www.statista.com/statistics/419628/payment-card-fraud-losses-usa-by-type/.

[9] Ahmed Awad E. Ahmed and Issa Traore. Anomaly Intrusion Detection based on Biometrics. Technical report, IEEE, 2005.

[10] Bart Baesens, Veronique Van Vlasselaer, Wouter Verbeke, and Veronique van Vlasselaer. *Fraud analytics using descriptive, predictive, and social network techniques : a guide to data science for fraud detection.*

[11] Stefano Boccaletti. Complex Network Theory: introduction and applications. Available at https://www.slideshare.net/CIGTR/complex-network-theory-in.

[12] Jason Brownlee. Support Vector Machines for Machine Learning. Available at https://machinelearningmastery.com/support-vector-machines-for-machine-learning/.

[13] Philip K Chan and Salvatore J Stolfo. Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. Technical report.

[14] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, may 2012.

[15] Tom Fawcett and Foster Provost. Combining Data Mining and Machine Learning for Effective User Profiling. Technical report, 1996.

[16] Prashant Gupta. Decision Trees in Machine Learning Towards Data Science. Available at https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[17] Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and A.K. Majumdar. Credit card fraud detection: A fusion approach using DempsterShafer theory and Bayesian learning. *Information Fusion*, 10(4):354–363, oct 2009.

[18] Paysim. Synthetic Financial Datasets For Fraud Detection — Kaggle. Available at https://www.kaggle.com/ntnu-testimon/paysim1.

[19] Johan Perols. Financial Statement Fraud Detection: An Analysis of Statistical and Machine Learning Algorithms. *AUDITING: A Journal of Practice & Theory*, 30(2):19–50, may 2011.

[20] Mark Schwabacher, Nikunj Oza, and Bryan Matthews. Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring. *Journal of Aerospace Computing, Information, and Communication*, 6(7):464–482, jul 2009.

[21] Song Fu. Performance Metric Selection for Autonomic Anomaly Detection on Cloud Computing Systems. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–5. IEEE, dec 2011.

[22] Salvatore J Stolfo, David W Fan, Wenke Lee, Andreas L Prodromidis, and Philip K Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results 1. Technical report, 1997.

[23] S. Suryansh. Neural Networks: All YOU Need to Know Towards Data Science. Available at https://towardsdatascience.com/nns-aynk-c34efe37f15a.

[24] Saishruthi Swaminathan. Logistic Regression Detailed Overview Towards Data

Science. Available at https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[25] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and Tong Zhang. Crowd Fraud Detection in Internet Advertising.

[26] Vishal Vatsa, Shamik Sural, and A. K. Majumdar. A Rule-Based and Game-Theoretic Approach to Online Credit Card Fraud Detection. *International Journal of Information Security and Privacy*, 1(3):26–46, jul 2007.

[27] Ke Wang and Salvatore J. Stolfo. Anomalous Payload-Based Network Intrusion Detection. pages 203–222. Springer, Berlin, Heidelberg, 2004.

[28] Mac Wang. How to protect your business from online payments fraud - CSO — The Resource for Data Security Executives, 2018. Available at https://www.cso.com.au/article/646962/how-protect-your-business-from-online-payments-fraud/.

[29] Roy Wedge, James Max Kanter, Kalyan Veeramachaneni, Santiago Moral Rubio, Sergio Iglesias Perez, Banco Bilbao, Vizcaya Argentaria, and Spain Madrid. Solving the ”false positives” problem in fraud prediction Automated Data Science at an Industrial Scale. Technical report.

[30] T. Jeffrey Wilks and Mark F. Zimbelman. Using Game Theory and Strategic Reasoning Concepts to Prevent and Detect Fraud. *Accounting Horizons*, 18(3):173–184, sep 2004.

[31] Massimiliano Zanin, Miguel Romance, Santiago Moral, and Regino Criado. Credit Card Fraud Detection through Parenclitic Network Analysis. *Complexity*, 2018:1–9, may 2018.

[32] Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen. Discovery of Ranking Fraud for Mobile Apps. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):74–87, jan 2015.