FRAUD DETECTION IN NON-NETWORK KNOWLEDGE GRAPH

by

Xinyang Liu

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Carleton University
Ottawa, Ontario

© 2022

Xinyang Liu
Abstract

Fraud is becoming an increasingly severe problem in all sectors of finance, business, and government organizations. One of the most notable and widely-used techniques in the detection of such fraud is data mining, and numerous research have been conducted in order to mitigate this threat. However, most of these research revolve around frauds in credit cards, and studies on fraud detection in other fields, such as contract management, are still extremely limited. As a result, in this thesis, we develop an automated fraud detection system for both reporting and prediction purposes in the domain of contract management. We use the Construction Contract Management service data from Defence Construction Canada (DCC) to test and evaluate the approaches employed in our work. Due to the lack of training data in practical scenarios, we use a weak supervision approach to generate labels (legitimate vs. fraudulent) for the training data, and two machine learning models, namely Logistic Regression and Random Forest. We also propose a graph-based approach that transforms the contract management dataset into a graphical representation, which results in a non-network structured knowledge graph, and learns both the structural relationships and the statistical features of this graph to identify potential anomalies in the data. Results from both the weak-supervised machine learning approach and the graph-based approach reveal relatively high recall in detecting possible fraud cases in their evaluations.
Acknowledgements

I would like to thank my academic supervisor, Professor Ahmed El-Roby for his guidance throughout this thesis. And I would like to also thank DCC for providing the data used in the thesis and the support throughout.
# Table of Contents

Abstract ii

Acknowledgements iii

List of Tables vii

List of Figures viii

List of Appendices ix

1 Introduction 1
  1.1 Motivation ........................................... 1
  1.2 Technical Challenges ................................. 3
  1.3 Approach Overview .................................. 4
  1.4 Contributions ....................................... 5
  1.5 Organization ....................................... 5

2 Related Work 7
  2.1 Fraud detection in databases ....................... 7
    2.1.1 Statistical techniques .......................... 8
    2.1.2 Artificial intelligence techniques ............. 9
      2.1.2.1 Supervised ML ............................. 9
      2.1.2.2 Unsupervised ML .......................... 12
  2.2 Fraud detection in knowledge graphs ............... 13
  2.3 Fraud detection in data streams ................... 14

3 Methodology 16
  3.1 Data ............................................. 16
C Data preprocessing 72
D Fraud injecting functions 74
Bibliography 76
List of Tables

3.1 Number of Fraud vs. Non-Fraud Cases . . . . . . . . . . 29
3.2 Data % within $k$ STD from Mean, from [79] . . . . . . . 53

4.1 Confusion Matrix . . . . . . . . . . . . . . . . . . . . . . 58
4.2 Model Performance based on $K$-fold CV . . . . . . . . . 61
4.3 Graph-based Approach Performance . . . . . . . . . . . 64
List of Figures

1.1 Relationship between Coordinator, Change Orders, Contract, and Supplier ........................................ 3

3.1 Number of Contracts, Change Orders, Progress Claims by Year ......................................................... 18

3.2 Distribution of Contracts, Change Orders, and Progress Claims ............................................................ 20

3.3 Distribution of Change Orders by Amount and Date Comparison .......................................................... 21

3.4 Distribution of Contracts by Unplanned Cost Growth .............................................................................. 22

3.5 Overview of the Semi-Supervised ML Approach ..................................................................................... 23

3.6 Number of Possible Fraud Cases ............................................................................................................. 27

3.7 Diagram of Logistic Regression .............................................................................................................. 31

3.8 Diagram of A Random Forest ................................................................................................................ 33

3.9 Overview of the Graph-based Approach .................................................................................................. 34

3.10 Basic RDF Triple ..................................................................................................................................... 35

3.11 Two Types of Object Values in Example RDF Graphs ........................................................................... 36

3.12 Common Network Topology .................................................................................................................. 37

3.13 Sample Data in RDF Format ................................................................................................................ 38

3.14 1-Hop from Coordinator Query Sample Subgraphs ................................................................................ 41

3.15 2-Hops from Supplier Query Sample Subgraphs ..................................................................................... 43

3.16 Euclidean Distance in Different Dimensions, from [83] ....................................................................... 46

3.17 Standard Deviations in Normal Distribution .......................................................................................... 50


4.1 Number of Outliers by STD, 1.5-IQR, and 3-IQR .................................................................................... 57

4.2 Accuracy vs. Number of Trees in Random Forest .................................................................................... 60
List of Appendices

Appendix A 70
Appendix B 71
Appendix C 72
Appendix D 74
Chapter 1

Introduction

1.1 Motivation

Fraud is becoming a consistently growing threat with significant consequences in all modern day industries, and it can be defined as an illegal and deliberately deceptive action with the intention of acquiring unlawful and unethical financial profit at the expense of others. Across the globe, fraud is costing businesses and individuals more than $5 trillion each year, according to The Financial Cost of Fraud Report by Crowe [31]. In addition, the Association of Certified Fraud Examiners (ACFE) estimates that organizations lose 5% of revenues to fraud each year, with typical fraud cases last at least 14 months before being detected [3].

Among these fraudulent activities, real estate and construction fraud are ranked as the second and third most costly frauds in terms of median loss, with billing fraud as the number one scheme, based on a recent global study by ACFE [3]. The World Economic Forum also predicts annual loss to reach about $6 trillion in the Engineering and Construction (E&C) sector by 2030 due to corruption, mismanagement, and inefficiency [9]. In a similar study that focused on Canada, the average loss for all fraud cases is higher than the global average, and this also holds true in E&C, where it is estimated to be $628,500 [1].

With the rise in contract management and the outsourcing of goods
and services in the E&C sector, there has been an increase in contract and procurement fraud, which can occur at any stage of the process and by any level of personnel involved in the process. In many cases, rather than the stereotypical assumption of construction companies defrauding individuals or the public, it is the companies themselves losing money to fraud perpetrated by employees, contractors, subcontractors and venture partners. Therefore, for these businesses, appropriate controls, sound fraud detection and prevention strategies are indispensable in their fights against this fraud.

In order to prevent such losses, a great deal of research have been devoted to the detection of fraudulent transactions: from the traditional methods of manual detection, which are becoming more and more time consuming and inefficient with the advent of big data, to the more recent computational methodologies that many institutions began to focus more of their attentions on. Data mining and machine learning have been playing an imperative role and are among the most notable and widely-used techniques used in solving the problem of identifying fraud. They analyze and discover the underlying behaviour and trend in the data, and then categorize the transactions into two classes of either authentic or fraudulent. Some of the approaches that have been applied to detecting fraud are Support Vector Machine (SVM) [69], Naïve Bayes [25], Logistic Regression [7, 13], Random Forest [87, 43], and Artificial Neural Networks [54].

However, the vast majority of the existing studies and research on fraud detection revolve solely around credit card transactions, which can be quite different from contract fraud. In addition, E&C data tends to have more complicated relationships between the entities involved in a contract, such as the coordinator from the company, the supplier, the changes made to a contract, and the progress claims, etc. Figure 1.1 shows a relation with a coordinator issues different change orders for the same contract requested by the supplier that supplies the contract. And as we could not find any practical information related to fraud in contract management or how to identify and detect it, it becomes necessary to develop a fraud detection system specifically for transactions contract management.
For this thesis, the main objective is to design and implement an automated fraud detection system that can both report on possible fraudulent cases in the past and predict potential fraud in future transactions using real-world construction contract data from Defence Construction Canada (DCC). Due to the specific nature of DCC’s services in Construction Contract Management, it is critical to know how these projects and activities are conducted, understand the contract and financial data, and learn how to recognize some potential red flags that may lead to fraud in this sector. Then we can utilize various approaches to process the data and feed it into an appropriate classifier algorithm. In order to accomplish these tasks, a good understanding of data preprocessing, machine learning modelling, and knowledge graph query analysis is required. This fraud detection system can later help DCC increase transparency and find a solution for minimizing the vulnerabilities and risks associated with fraud.

1.2 Technical Challenges

One of the challenges in this thesis is actually the label-less training data, which is due to the fact that there were not enough existing re-
search in contract fraud and that DCC does not have any documented or identified fraud examples, which in turn made it difficult to build supervised machine learning models. So, the initial focus of this thesis was to apply a weak-supervision approach to generate psuedo labels by coding the fraud scenarios into the labeling functions then combining and leveraging these weak supervision sources through the agreements and disagreements of the labeling functions, and finally use the pseudo labels to implement two ML models, Logistic Regression and Random Forest, to address finding possible fraud cases.

However, this leads to another challenge of the thesis: one of the requirements in this problem is to explain why a case is believed to be a potential fraud, and the machine learning approach suffers with regard to this aspect. The process of generating the pseudo labels to be used to train the ML models is not explainable. Hence, we cannot justify the output candidate fraudulent cases to the user, which is a critical step in the fraud detection pipeline since these candidate fraud cases output by our work will need to be examined by a domain expert who will need some associated explanations.

1.3 Approach Overview

In addition to the weak-supervision machine learning approach, we introduce a graph-based approach, which transforms the contract management dataset into a graphical representation. The resulting knowledge graph does not take the same shape as a common network structure due to the complexity of the relationships between the entities involved, and the nodes in the knowledge graph can be of different types, such as coordinators, suppliers, contracts, etc. Then, we query the graph based on 1, 2, ..., n-hops from a given root node to obtain various subgraphs, and calculate the graph edit distance (GED) or a user-defined distance function to measure not only structural but also statistical differences between these subgraphs. Finally, based on some outlier detection thresholds, we are able to identify anomalies in the relationships between the entities of the same type in the graph.
1.4 Contributions

To address the issue of missing labels, we adopt the weak-supervision technique. And to better measure the relationships between the entities involved and to generalize to more types of fraud, we propose a graph-based approach for fraud detection. The main contributions of this thesis is summarized as follows:

- **Fraud Scenarios**: We come up with and define more than 20 hypothetical scenarios in the contract management industry that could be considered indicative of fraud.

- **ML model**: We leverage a weak-supervision machine learning method to obtain class labels (legitimate vs. fraudulent) for training data and build two ML models. The Logistic Regression and Random Forest models support the prediction of potential fraud cases on future transactions of contract management based on labeling functions coded from fraud scenarios.

- **Graph-based approach**: We develop a graph-based approach to query and analyze knowledge graphs constructed from the input contract data. The graph-based approach takes into account both structural and statistical differences between different node types in the graph, and enables us to identify possible fraud not covered in our fraud scenarios.

- **Output Explainability**: We are able to associate explanations to the output candidate fraud cases and justify the exact reason why they are returned as fraudulent.

- **Experiments on real datasets**: We conduct extensive experiments on real contract management data from DCC to evaluate the effectiveness and accuracy of our approach.

1.5 Organization

This thesis is organized in the following fashion. Chapter 1 (this chapter) provides an overview of the motivation, objectives and contributions of this thesis. Chapter 2 discusses background information and
prior research that are relevant to the thesis, which are mostly research related to fraud detection, machine learning, and knowledge graphs. Chapter 3 describes the techniques and approaches that were adopted to accomplish the goals of this thesis, including a description and exploration of the DCC data set, the data cleaning process, the creation of the input data and corresponding graph representation, the label generation process for weak-supervision ML, the development of ML models, and the structural- and numerical-level analysis of the graph. Chapter 4 explains the evaluation mechanism in the absence of ground truth to determine whether the objectives of this thesis have been met. And finally, chapter 5 summarizes the findings of the paper, points out the limitations of the methods, and suggests paths for possible further research.
Chapter 2

Related Work

This chapter contains the background information and related research on various techniques used for fraud detection in various fields and in various forms, such as in databases or knowledge graphs.

2.1 Fraud detection in databases

In recent years, the increment of new computer technologies has enabled most organizations, companies, and government agencies to perform financial transactions through the electronic commerce systems (ECS), which helped increase the productivity and efficiency of their services, while at the same time, providing yet further ways for criminals to commit fraud. Fraudulent activities in areas such as credit card, money laundry, telecommunication, and healthcare and automobile insurance, etc. are becoming easier and easier to perpetrate [19, 21, 22]. As a result, protection systems against fraud has been the topic of various surveys and review articles, and both prevention and detection mechanisms are used to combat fraud.

_Fraud prevention system_ (FPS) is the first layer of protection for the ECSs against fraud by stopping fraud from occurring in the first place [5]. It restricts, suppresses, destructs, destroys, controls, removes, or prevents the occurrence of attacks in hardwares, softwares, and networks. Some of the examples include elaborate designs such as watermarks and fluorescent fibers on banknotes, personal identification
numbers for bankcards, passwords for bank accounts, subscriber identity module (SIM) cards for mobile phones, and firewalls for computer systems or networks [47]. Nevertheless, these mechanisms alone are not strong enough to halt fraud, and oftentimes, one has to compromise between their expenses and their effectiveness.

The next layer of protection is the fraud detection system (FDS), which comes into play once fraud prevention has failed, and it tries to discover and identify fraudulent activities as quickly as possible once they enter the systems and reports them to a system administrator [15]. In practice, fraud detection must be used continuously, since we typically would not be aware of when fraud prevention has failed and we need to be able to recognize that the system is being breached as soon as possible. Also, earlier detection tools, as well as the latest developments, need to be applied in order to detect both newly entered fraudsters and those already adapt to earlier strategies [19].

The traditional method for detecting fraudulent transactions is the manual detection, such as the discovery sampling [70], which is extremely complicated, time-consuming, and inefficient. Thus, with the advent of big data, automated FDSs that utilize more recent computational methodologies such as data analytics are invented. These techniques used for fraud detection generally fall into two primary classes: statistical techniques and artificial intelligence.

2.1.1 Statistical techniques

Early data analysis techniques were oriented toward extracting quantitative and statistical data characteristics, which facilitate useful data interpretations and help to get better insights into the processes behind the data [80]. Statistical tools for fraud detection are many and varied. Some of the simpler and more straightforward examples are data preprocessing that detect, validate, correct, and fill in missing or incorrect data, and statistical parameters calculation, such as averages, quantiles, performance metrics, and probability distributions, that enables the comparison between norm and anomaly.
More complex techniques include data matching, which compares two sets of collected data by trying to match them against each other or comparing complex data types to remove duplicate records and identify links between two datasets [19]. Regression analysis examines the relationship between two or more variables of interest, independent or dependent, to help understand and identify relationships among variables. In addition, another statistical approach is to match algorithms to detect anomalies in the behavior of transactions or users as compared to previously known models and profiles.

2.1.2 Artificial intelligence techniques

Fraud detection is a knowledge-intensive activity. The traditional statistical techniques require human analysts to perform the analysis to reach an understanding about the knowledge and the data. However, with the advance of big data and modern technologies, the background knowledge necessary for the reasoning becomes substantial, and researchers begin to turn to artificial intelligence (AI) and machine learning (ML) to discover meaningful patterns and information for fraud detection.

One of the main AI techniques used for fraud detection falls into the category of machine learning and data mining, where rules and patterns are extracted and identified automatically from the recorded transaction data. ML is the most explored approach of effective fraud detection; it has been a popular topic of various FDS surveys and literatures, and is adopted by numerous authors to tackle fraud in different areas. The ML solutions can be classified into two categories: supervised and unsupervised learning.

2.1.2.1 Supervised ML

The supervised learning techniques use random samples manually labeled as “fraudulent” and “non-fraudulent” to construct models that should be able to classify new observations as either one of the two classes. These are the most common learning approaches in fraud detection.
Traditional ML methods, such as linear regression and logistic regression, have proved to be effective tools, and many research for credit card fraud involve this type of solutions [10, 52, 68, 66]. More sophisticated algorithms like decision trees and random forest are also applied in fraud detection in the area of credit card [68, 16].

Another supervised ML techniques used for fraud detection is the Rule-based expert system, which encodes expertise for detecting fraud in the form of rules. The expert system adopts a misuse detection approach, where fraudsters signatures are used to first define fraudulent behaviors and other behaviors are defined as normal. It then utilizes rule-based, statistics, or corresponding heuristic methods to reveal the occurrence of specific suspicious transaction [35]. Murad, Pinkas, and Rosset et al. [62] proposed the rule-based approach for telecommunications fraud, which studies the individual call profiles and the call patterns, etc. While similar approaches have also been developed for money laundering based on experience about transaction patterns, such as rates and amounts. One of the most elaborate money laundering detection systems, the U.S. Financial Crimes Enforcement Network AI system (FAIS), is also a rule-based system [67, 33]. This rule-based detection mechanism is considered to be simple and fast, but it has a major limitation on its capabilities because it fundamentally depends on predefined rules stated by experts and only looks for known patterns of misuse, hence is impossible to detect all different kinds of frauds [76].

Link analysis comprehends a different approach. It relates known fraudsters to other individuals, using record linkage and social network methods [75]. In telecommunications networks, security investigators have found that fraudsters seldom change their calling habits and rarely work in isolation from other fraudsters, which establishes the "communities of interest" [26]. A similar approach has been taken in money laundering, where link analysis is actually the primary analytic detection strategy used by law enforcement [41]. The FAIS identifies groups of participants involved in suspicious transactions and allows users to follow trails of linked transactions. [34, 67, 33].

A lot of works on fraud detection has been carried out with neural
networks. A neural network is a network or circuit of artificial neurons or nodes designed to imitate the functioning of the human brain [32]. An individual node receives a signal then processes it and signal nodes connected to it with the output computed by some non-linear function of the sum of its input. The connection between nodes are typically weighted, with the weight adjusted as learning proceeds. Compared to other ML techniques, neural networks can adapt more elegantly to the behavior of various users, due to its ability to calculate user profiles independently [76]. In credit card fraud detection, CARDWATCH [8] uses neural network to process spending patterns to detect possible anomalies; Brause and Langsdorf [20] proposes the rule-based association system combined with the neuro-adaptive approach; and HNC Software developed Falcon [37], a software package that relies on feed-forward neural networks. Similar feed forward neural networks with back propagation training algorithms is utilized in detecting fraud in the automobile insurance area [22]. In addition, telecommunication fraud can also be detected with neural networks. A project of the European Commission, ASPeCT, combines rule-based profiling with neural network [51]; and a similar combination of profiling and neural networks is also implemented by the main fraud detection software of Nortel Networks [24].

However, the supervised ML methods are not without their issues, and there are some apparent challenges faced by this type of approaches. The first challenge is that given a huge volume of input data, it is prohibitively expensive, if not impossible, to collect supervision or labels for all of them. Secondly, even when the labels are present, the dataset usually suffers from highly imbalanced classes, with much fewer samples of fraudulent instance than normal ones, which makes it difficult for classifiers to discover patterns in the minority class, hence affecting the performance of the classifiers. Also, this type of detection can only detect frauds of a type that has occurred previously. Therefore, unsupervised ML algorithms may be required to detect novel types of fraud and to overcome the other disadvantages that are obstructing the implementation of supervised ML approaches.
2.1.2.2 Unsupervised ML

In contrast, unsupervised methods are used when there are no prior labeled legitimate and fraudulent observations. They usually employ a combination of profiling and outlier detection approaches, model a baseline distribution for normal behavior, and seek activities that display the greatest departure from the norm, such as changes in behavior or unusual transactions. The returned observations can then be examined more closely by domain experts.

Some of the most crucial studies with respect to unsupervised fraud detection are the Peer Group Analysis and Break Point Analysis proposed by Bolton and Hand, both of which focus on the behavioral fraud detection [18]. Peer Group Analysis monitors behavior over time and flags individual accounts that begin to behave in a way distinct from their peer group of accounts that they has previously been the most similar to. Break Point Analysis, on the other hand, identifies changes in spending behavior based on the transaction information in a single account, where a break point is an observation or time where anomalous behavior is detected. It compares sequences of transactions for a change in behavior for a particular account and sees if recent transactions follow a different pattern from that of previous transactions. Features such as rapid spending and sudden increases in the amount of spending can be indicative of fraud. In recent years, other unsupervised learning techniques are gaining popularity, such as K-means clustering, auto-encoder, and isolation forest [71, 53, 49, 14].

Whether supervised or unsupervised methods are used, we can seldom be certain, by stand alone statistical analysis, that a fraud has been perpetrated, although it can be identified with relatively high degree of accuracy. Rather, the output should be regarded as an indication of fraud likelihood, alerting us that the observation is anomalous or more likely to be fraudulent than others. Therefore, the collaboration between ML models and human analysts is vital to a successful fraud detection applications.

There are very few fraud or anomaly detection surveys on contract management, and since fraud detection methods are very context de-
dependent, it is difficult and even ill-suited to apply existing approaches to this case. Also, since the outputs from ML algorithms are simply possible fraud and require inspection and confirmation from domain experts, an explanation of why specific instances are flagged is necessary, which is lacking in both supervised and unsupervised learning.

2.2 Fraud detection in knowledge graphs

Data scientists have developed rigorous machine learning models to detect fraud, and most traditional measures focus on discrete data points, such as specific individuals or accounts. However, today’s fraud typically involves multiple parties, with sophisticated fraudsters being a greater threat by working together. As a result, most existing ML algorithms overlook a critically important aspect of the data used to train them, the structure of the data, since tabular data is not designed to capture relationships. With graph data becoming ubiquitous, it is essential to look beyond individual data points to the connections that link them, and techniques for spotting anomalies and identifying fraud in structured graph data have been of focus recently.

Researchers have been adapting neural networks to operate on graphs and leverage the structure and properties of graphs; such neural networks are called Graph Neural Networks (GNN). GNNs are neural models that capture the dependence of graphs via message passing between the nodes of graphs. As the development of GNNs in recent years, many GNN-based fraud detectors have been proposed [46, 45, 73, 74, 90, 91, 77], where FdGars [74], GAS [45], and ASA [77] adopt graph convolutional networks (GCN), a variant of GNN, while SemiGNN [73] and Player2Vec [90] adopt graph attention network (GAT), another GNN variant. Both GNNs and their variants have demonstrated groundbreaking performances on fraud detection in graphs and yield outstanding results, due to their ability to represent and learn the relationships between graph nodes, edges, and subgraphs. However, just as the neural networks in statistical ML, GNNs can lead to overly complex and poorly interpretable models.

In order to strike a compromise between performance and inter-
pretability, an alternative of combing graph analytics with machine learning is proposed. This technique represents the data in a graphical form, computes some useful network features, such as the degree of each node, the closeness centrality coefficient, and PageRank, and finally enrich the original dataset with the additional structural information obtained. Incorporating graph analytics into the ML models can significantly boost their performances, since ML with graphs approach improves upon regular ML algorithms that learn from individual observations by detecting new patterns and adding valuable information on the connectivities between entities and the nature of their relationships, which are not captured by the initial feature set [29]. Neo4j provides the first enterprise-grade approach to data science that combines graph analytics with machine learning, harnessing the natural power of relationships and structures to infer behavior [38]. Zhan and Yin [88], Zhang et. al. [89], and Wen et. al. [78] also apply the same concept in their research on financial and insurance fraud detection.

However, unlike most graph-structured data in anomaly detection algorithms that have network topology, such as people and their connections in a social network, or buyer, seller, and product nodes connected via buy, sell, and customer edges, our data involves entities with much complicated relationships between each other, which renders the two above mentioned techniques inapplicable in our case. In addition, neither GNN nor ML with graphs solve the problem of no supervision in our data, or satisfy our need for explanation for each potential fraudulent case returned by the models.

2.3 Fraud detection in data streams

The development of digital devices and technologies has led to an increase in the generation and transmission of various, complex types of data in real time. In this era of information, anything transmitted over the internet is transmitted as a data stream, which is a sequence of data records ordered by explicit or implicit timestamps to convey information. In a formal way, a data stream is any ordered pair \((s, \Delta)\), where \(s\) is a sequence of tuples, and \(\Delta\) is a sequence of positive real time intervals [82].
Existing research on fraud detection in data streams are extremely limited. In the credit card fraud detection field, Ratnam [58] proposed a Stream Outlier Detection based on Reverse k Nearest Neighbors (SORNN) algorithm, and Arya et al. [11] introduced a predictive framework, Deep Ensemble ALgorithm (DEAL), to detect fraudulent transactions in real-time data stream. On the other hand, telecommunications fraud can also be detected using Data Stream Analytics [40]. However, although fraud detection in data streams is not extensively studied, outlier detection techniques are more widely researched and applied in the streaming environment, such as the DB-Outlier [39] and DBOD-DS [63] in distance-based outlier detection techniques, A-ODDS [64] in density-based, AMSD [57] in auto-regression-based, and AnyOut [12] in cluster-based, etc. Since our data is not of a stream format and does not place a high emphasis on real-time, we focus only on fraud detection in databases and knowledge graphs.
Chapter 3
Methodology

This chapter explains the procedures and methods used in this thesis to gather information, prepare data, train machine learning classifiers, convert database into graph representations, and analyze non-network knowledge graphs. It also describes some challenges that were encountered during the work and how these problems were solved.

3.1 Data

3.1.1 Terminology

**Contract:** A contract in construction industry is a mutual or legally enforceable agreement between two parties, which are generally one or more property owners and one or more contractors, that is based on policies and conditions recorded in document form and defines the scope of work, risks, obligations, and legal rights of both parties.

**Change Order:** A change order is an amendment to a construction contract that changes the original scope of work. Most change orders modify the works agreed in the contract, which, in turn, may alter the contract price or adjust the amount of time to complete the work, or both. Change orders are common to most projects, and very common with large project.

**Progress Claim:** A progress claim is a document sent from a contrac-
tor, subcontractor, or supplier to their customer requesting periodical payments or a complete payment, when payment is owed for work that has been performed to date.

**Coordinator:** A coordinator manages all activities and changes of an assigned contract and is responsible for ensuring communication and collaboration between the owners, clients, and contractors.

**Approver:** An approver approves changes that are within his/her assigned delegated authority limit (DSA limit) for contract management after confirming commitment of funding from the owners or clients. Changes that exceed the assigned coordinator’s delegated authority should be recommended to a higher authority.

**Supplier:** A supplier is an organization contracted as part of the delivery of a built asset, such as particular goods or services, either directly to the client or to the contractor, who is, in broad terms, an organization appointed by a client to complete construction works. Based on the data used in this thesis, the term Supplier is interchangeable with Contractor.

### 3.1.2 Data exploration

The data used in this thesis is extracted automatically from DCC’s JD Edwards EnterpriseOne (E1), and it is stored in 6 different tables about the contracts, their change orders, and their sets of progress claims. For the purpose of this thesis, we examine 3 years worth of data from 2019-01-01 to 2021-12-31, which totals to around 20,000 change orders and more than 40,000 payments from over 5,000 contracts. Figure 3.1 shows the number of contracts, change orders, and progress claims in each of the three years. And of these contracts, around 500 coordinators are assigned and over 1,100 suppliers are contracted.

Since the change orders and payments are not necessarily related to each other, we separate their information into two different datasets, resulting in 31 attributes in the change orders dataset and 18 for the payments dataset. For the change order dataset, the input features
Figure 3.1: Number of Contracts, Change Orders, Progress Claims by Year

range from 14 columns of contract related information such as contract number, contract type, responsible site, award amount, award date, completion date, and supplier, to 18 columns of change order related information like change order number, change order type, estimated, quoted, and negotiated amount and date; also included in these change order columns are the assigned coordinator, the approver with appropriate authority, and their respective DSA limit that indicates their delegated authorities. On the other hand, the payment related dataset also contains the 14 contract attributes, with an additional 4 columns about invoices: invoice number, amount, invoice date, and payment date.
For the purpose of this thesis, the distribution of the data needs to be tested, in order to confirm whether it is normally distributed and to determine the types of statistical methods to use. Both graphical methods, which plot the data and qualitatively evaluating whether it looks Gaussian, and statistical normality tests, which calculate statistics on the data and quantify how likely it is that the data was drawn from a Gaussian distribution, are employed.

One of the simplest and most commonly used graphical normality test is the Histogram Plot. Figure 3.2 shows the distribution of contracts, change orders, and payments based on some selected amounts and dates information. We can see that all six distribution plots are not normal distributions. The three amount plots are extremely left skewed and almost exponential like, with the majority of all three amounts concentrating on lower values: the contract award amounts are all positive, and although there are extremely expensive contracts, most are awarded at below $4,000,000; the change orders and payments are generally much less in value compared to those of contracts, and range mostly from -$20,000 to $50,000. The date plots, on the other hand, look more evenly distributed, with the contract award dates a rough approximation of a uniform-like distribution, and the change order and invoice dates a bit right skewed toward the end of the three-year duration, as there are more contracts to have changes and payments made to them.

The statistical test used to corroborate the findings from the histograms is the Shapiro-Wilk Test. It evaluates a data sample \( x_1, \ldots, x_n \) and tests the null hypothesis \( (H_0) \) that the sample came from a normally distributed population. The test calculates

\[
W = \frac{\left( \sum_{i=1}^{n} a_i x_{(i)} \right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2},
\]

where \( x_{(i)} \) is the \( i \)th order statistics, i.e. the \( i \)th smallest number in the sample, and \( \bar{x} = \frac{(x_1 + \ldots + x_n)}{n} \) is the sample mean. The coefficients \( a_i \) are given by \((a_1, \ldots, a_n) = \frac{m^\top V^{-1}}{C}\), where \( C \) is a vector norm, \( C = \|V^{-1}m\| = (m^\top V^{-1}V^{-1}m)^{1/2} \), the vector \( m = (m_1, \ldots, m_n)^\top \) is made of the expected values of the order statistics of \text{i.i.d.} random
Figure 3.2: Distribution of Contracts, Change Orders, and Progress Claims
variables sampled from the standard normal distribution, and $V$ is the covariance matrix of those normal order statistics. The test also returns the $p$-value that can be used to accurately interpret the W-statistic by comparing it to a threshold level called alpha ($\alpha$), which is typically 5% or 0.05. If $p \leq \alpha$, then we reject $H_0$ - the data sample is not drawn from a normal distribution; whereas if $p > \alpha$, we fail to reject $H_0$ and the data is Gaussian. In this case, the $p$-value from all six attributes are infinitely close to 0, thus not normal.

Additionally, rather than only exploring the distribution of single attribute from the data, the comparison between certain columns of the same type actually makes more sense, as they are related to each other both practically and semantically. As a result, we plot the differences between some variables to see if that would affect the distribution of the data. Figure 3.3 shows the histograms for the number of change orders by date difference between the quoted and negotiated dates, and by amount difference between the negotiated amount and the approver’s DSA limit, respectively. We can see that although neither are Gaussian, the distribution of change orders based on amount difference is no longer as left skewed as when using only one attribute; all change orders are dispersed across the amount axis, with a few spikes

(a) Negotiated Amount vs. DSA Limit  
(b) Quoted vs. Negotiated Date

Figure 3.3: Distribution of Change Orders by Amount and Date Comparison
along the way. However, the distribution by date difference becomes significantly skewed compared to those from a single column, which is understandable as negotiations to a change tend to occur immediately after receiving a quote. The results from the Shapiro-Wilk Test also confirms the non-normal distribution of the two tests.

![Distribution of Contracts by Unplanned Cost Growth](image)

**Figure 3.4: Distribution of Contracts by Unplanned Cost Growth**

Based on the data in this thesis, there are also some user defined functions specific to the construction industry, such as the \textit{Unplanned Cost Growth}:

$$UCG = \frac{(S + D + O)}{(Award \text{ Amount} + P)},$$ (3.2)

where $S$, $D$, and $O$ are the amounts of three types of unplanned change orders, \textit{Award Amount} is the award amount of the contract, and $P$ is the amount of the planned changes. This $UCG$ value reveals the percentage of unplanned change amounts compared to the value of
the contract and its already planned changes, which, if way too high, suggests that there could be an abnormally high amount of unplanned changes for a single contract. Figure 3.4 shows the distribution of contracts by UCG, which is also skewed left to a Gaussian distribution and concentrates mostly around 0 and 1, matching the common belief that changes to a contract typically would not exceed the value of the contract. Furthermore, the statistical test also ends up with a \( p \)-value of 0, rejecting the null hypothesis of normal distribution.

### 3.2 Semi-supervised ML

As the data is finished with exploration and preprocessing (explained in detail in Appendix C), it is time to build machine learning models in order to detect and predict future anomalies. Figure 3.5 is an overview of the semi-supervised ML approach employed in this thesis, where unlabeled data, together with some scenarios indicative of fraud, are input to a weak-supervision system Snorkel to generate probabilistic labels for training two ML models and finally outputting some potential fraud cases. The fraud scenarios will be discussed further in detail in Section 3.2.1, the architecture of Snorkel in Section 3.2.2, and the two ML models in Section 3.2.3.

![Figure 3.5: Overview of the Semi-Supervised ML Approach](image)

Traditionally, there has been two major distinctive categories of machine learning approaches: supervised and unsupervised learning [17]. In supervised learning, the computer is presented with a set of exam-
ple inputs and their corresponding desired outputs, and the goal is to learn a general function and construct a classifier from the labeled training data that maps inputs to outputs [84]. On the other hand, unsupervised learning tries to infer some underlying structures or hidden patterns from the input data with no specific output labels being provided. Falling between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data), semi-supervised learning aims to combine these two tasks by using a small amount of labeled data with a large amount of unlabeled data during training [23, 92]. It is believed that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. For this thesis, we choose to adopt weak supervision, of which semi-supervised learning is a special instance, due to the characteristics of the data.

3.2.1 Fraud scenarios

As there are no previously identified fraud cases within DCC’s Contract Management, some hypothetical situations in its contract management activities, especially those related to change orders and payments, that could be considered indicative of fraud need to be defined. Therefore, a workshop is held with professionals from different departments, service lines, and operational regions, and we come up with more than 20 hypothetical fraud scenarios that cover cases from three different aspects, such as extreme unplanned cost growth, cover bid, and payment for unfinished work. These fraud scenarios are to be used as inputs to the weak supervision approach that generates pseudo-labeled training data.

1. Cost growth related

   (a) Significant unplanned cost growth (UCG) by a coordinator compared to other coordinators.

   (b) UCG by a coordinator compared to other coordinators for each contract type.

   (c) Variation in cost growth between suppliers for a specific coordinator.
(d) Significant supplier contract cost growth compared to other suppliers.
(e) Unplanned Changes exceed 20% of the contract award amount.

2. Change order related

(a) Multiple changes at close to coordinator DSA limit (30% changes within 15% of DSA).
(b) Multiple changes at close to approver DSA limit (30% changes within 15% of DSA).
(c) Coordinator issues change orders within 5% of coordinator DSA limit to avoid directing to higher authorities.
(d) Coordinator issues change orders within 5% of approver DSA limit.
(e) Frequency of change orders: multiple change orders issued within a 1-week period on the same contract.
(f) Estimate consistently higher than quote - similar to cover bid with a possibly planted bidder that simply tries to drive the price higher.
(g) Estimate based on quote - difference between quoted and estimated amounts is minimal.
(h) Estimate based on quote for a specific coordinator - difference between quoted and estimated amounts is consistently close for a coordinator compared to others.
(i) No or limited negotiation occurring with supplier - difference between quoted and negotiated amounts is minimal.
(j) No or limited negotiation occurring with supplier for a specific coordinator - difference between quoted and negotiated amounts is consistently close for a coordinator compared to others.
(k) Date of estimate is later than the date of quote more than 20% of the time.
(l) Negotiated amount is greater than the quoted amount.
(m) For construction (CN) contracts, substantial completion date extensions without cost increase.

(n) For CN contracts, cost increase without substantial completion date extension.

(o) Excessive Time and Materials (T&M) changes by a specific coordinator.

(p) For CN contracts, significant use of “planned” changes to avoid detection by a specific coordinator.

3. Payment related

(a) Consecutive or very close invoice numbers that indicate frequent payments.

(b) Payments $\geq 50\%$ of contract award amount are made in the first $25\%$ of contract duration - payment for unfinished work.

(c) For CN and SC contracts, payment percentage of contract award amount exceeds contract duration by $25\%$.

(d) For CN contracts (contract award amount $> \$100,000$ and contract duration $\geq 365$ days), first progress claim value exceeds $\frac{1}{contract\ duration\ in\ months}$ of total contract value.

After these fraud scenarios are translated into Python code using attributes from the datasets, some preliminary possible fraud cases can be obtained based on each individual scenario. Figure 3.6 are three examples of the number of possible fraud cases given different parameters. Figure 3.6a shows the number of coordinators assigned to contracts with abnormally high unplanned cost growth (Scenario 1(a)) based on different standard deviations from mean for each contract type (choice of standard deviation as outlier criteria explained in Section 3.4). We can see that much more KN contracts tend to have out of range UCG compared to other types of contract, whereas FM contracts all have regular UCG values. Coordinators that issue change orders within certain percentages of approvers’ DSA limits (Scenario 2(d)), and coordinators that estimate changes higher than quotes consistently (Scenario 2(f)) with a frequency several standard deviations from mean are displayed.
in Figure 3.6b and Figure 3.6c respectively. Although these are not definitive frauds, they reveal the anomalies present in the dataset and offer some insights as to which values of parameters to choose later in the thesis. 

(a) UCG by Contract Type  
(b) Changes Close to DSA  
(c) Cover Bid

Figure 3.6: Number of Possible Fraud Cases

3.2.2 Snorkel

One of the immediate issues with the data is that it does not have any class labels, i.e., response variables, for its data points, which makes it difficult to build machine learning models on, so it is imperative to find a way to acquire those labels. However, hand labeling the data
is nearly impossible due to not only the massive amount of instances, but also the absolute needs for domain expertise. Fortunately, because labeling training data is increasingly the largest bottleneck in deploying machine learning systems, considerable studies have been done and the bulk of researchers resort to using weak supervision, where noisy, limited, or imprecise sources are used to provide supervision signal for labeling large amounts of training data in a supervised learning setting [50, 61, 48].

The approach we focus on is Snorkel [59], which improves upon a simple, single-sourced weak supervision to achieve increased accuracy in training labels and consequently improved predicative performance. The goal here is to learn a parameterized classification model $h_\theta$ that, given a data point $x_i \in \mathcal{X}$, where $\mathcal{X} = (x_1, ..., x_m)$ is a collection of unlabeled data points, predicts its label $y \in \mathcal{Y}$, where for simplicity, $\mathcal{Y}$ is assumed to be binary, i.e. $\mathcal{Y} = \{-1, 1\}$. And its workflow is designed to proceed in three main stages: writing labeling functions, learning a generative model, and training a discriminative model.

First, Snorkel lets users write their own labeling functions (LF) to express a wide range of weak supervision sources such as pattern, heuristics, and external knowledge bases, and then combines these sources. Suppose there are $n$ labeling functions $\lambda = (\lambda_1, ..., \lambda_n)$, where $\lambda_j : \mathcal{X} \rightarrow \mathcal{Y} \cup \{\mathcal{O}\}$ takes in a data point and outputs a label and $\{\mathcal{O}\}$ denotes an abstain vote, and $m$ unlabeled data points, Snorkel applies the LFs over the unlabeled data to produce a matrix of labels output $\Lambda \in (\mathcal{Y} \cup \{\mathcal{O}\})^{m \times n}$.

As these sources may have overlaps or even conflicts with each other, Snorkel resolves this issue by learning a generative model to estimate the accuracies, correlation structures, and other statistical dependencies of these sources, without access to ground truth [60]. The parameters $w$ of this model $P_w(\Lambda, \mathcal{Y})$ are estimated by maximizing the log marginal likelihood of the observed labels $\Lambda$:

$$\hat{w} = \arg \max_w \sum_{\mathcal{Y} \in \{-1, 1\}^m} P_w(\Lambda, \mathcal{Y})$$

(3.3)
And the label matrix $\Lambda$ is then re-weighted and synthesized into a single vector of probabilistic labels $\tilde{\mathcal{Y}} = (\tilde{y}_1, ..., \tilde{y}_m)$, where $\tilde{y}_i \in [0, 1]$.

Finally, these weakly labeled training data $(\mathcal{X}, \tilde{\mathcal{Y}})$ are used to train a discriminative classifier $h_\theta$ by minimizing the expected loss $l$ with respect to $\tilde{\mathcal{Y}}$:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{m} \mathbb{E}_{y \sim \tilde{y}_i}[l(h_\theta(\mathcal{X}_i), y)]$$

(3.4)

The discriminative machine learning model can generalize beyond the precise but low-coverage labeling functions, while retaining the precision and increasing coverage and robustness on unseen data at the same time.

In this case, as the scenarios and criteria for identifying potential frauds are identified in the workshop and ready for use, we follow this framework by coding those fraud scenarios as the labeling functions, then applying these LFs to the collection of data points extracted from E1, with cases meeting the criteria in the LFs set to be fraud and otherwise abstain, and finally the result it returns is single noise-aware probabilistic label per instance. Table 3.1 shows the number of cases labeled as “fraud” and “genuine” for the change order dataset and the payment dataset respectively. We can see that about 16% of all change orders are labeled as “fraud”, and around 19% of all invoices are considered anomaly, which are acceptable percentages of possible fraud for training data.
3.2.3 ML model

After the labels are generated for the training data, machine learning classifiers can be trained to predict whether a given transaction is genuine or fraudulent. The two models selected are Logistic Regression and Random Forest. These methods are chosen because they are some of the most widely used supervised learning models for classification purposes nowadays, and also because multiple existing research on credit card fraud detection all reach a similar conclusion that logistic regression and random forest have relatively better performances compared to other machine learning models [7, 28, 65, 72].

3.2.3.1 Logistic regression

Logistic regression uses both the logistic function by Raymond Pearl and Lowell Reed for the description of the growth of population in the United States [55], and a nonlinear function called the sigmoid to estimate the probability of a binary classification based on various features from the input data. The sigmoid function $\sigma$ is shown below:

$$\sigma(x) = \frac{1}{1 + e^{-f(x)}}$$  \hspace{1cm} (3.5)

and $f(x)$ is the log-odds,

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$  \hspace{1cm} (3.6)

where $\beta_0$ is an intercept term and $\beta_1, ..., \beta_n$ are coefficients associated with each variable $x_1, ..., x_n$. The best predicted weights $\beta_1, ..., \beta_n$ need to be determined such that the sigmoid function $\sigma(x)$ is as close as possible to all actual responses $y_i, i = 1, ..., n$, where $n$ is the number of observations. And they are obtained by maximizing the log-likelihood function (LLF) for all observations:

$$LLF = \sum_{i=1}^{n} (y_i \log(\sigma(x_i)) + (1 - y_i) \log(1 - \sigma(x_i)))$$  \hspace{1cm} (3.7)

Once the best weights that define $\sigma(x)$ are determined, if the value of the sigmoid is more than a threshold of 0.5, i.e. $\sigma(x_i) > 0.5$, the observation $i$ is considered to be in class 1 (in this case, fraudulent), and
any value smaller than 0.5 is automatically considered 0 (legitimate). Figure 3.7 demonstrates how logistic regression is fitted to some sample data points.

![Diagram of Logistic Regression](image)

Figure 3.7: Diagram of Logistic Regression

In order to train the logistic regression model on our dataset, all categorical variables are converted into dummy variables, which take only the value 0 or 1 to indicate the absence or presence of some categorical effect. For variables that have more than two categories, they can be represented by a set of dummy variables, with one variable for each category. Also, the date columns might introduce some complications when implementing the model, so they are converted into numeric values, which are the same as *Unix time* that counts the seconds since the epoch, 1970-01-01 00:00:00 UTC. Finally, after these processing, the logistic regression classifier can be trained with our data.
3.2.3.2 Random forest

Another model we choose is random forest. Random forest has been a very popular and effective model in machine learning due to its ease of use, accuracy, and robustness. Random forests are created by aggregating many individual decision trees as an ensemble. An individual decision tree is grown by using recursive binary splitting, and in the classification setting, classification error rate can be used as a criterion for making the binary splits, which is simply the fraction of the training observations in that region that do not belong to the most common class:

\[ E = 1 - \max_k (\hat{p}_{mk}) \]  

(3.8)

where \( \hat{p}_{mk} \) represents the proportion of training observations in the \( m \)th region that are from the \( k \)th class. However, classification error is not sufficiently sensitive for tree-growing, and in practice, the cross entropy measure, given by

\[ D = - \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \]  

(3.9)

is often preferred. Random forest fits a number of these decision tree classifiers on various sub-samples of the dataset, and each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model’s prediction. Figure 3.8 illustrates the technique and underlying workflow followed by random forest.

Random forests are considered to be highly accurate and robust due to the number of decision trees participating in the process, and also since it corrects for decision trees’ habit of overfitting to their training set by taking the average of all the predictions, which cancels out the biases and makes a forest more reliable. As a result, in this thesis, the number of trees in the model is set to be 500 after considering both the accuracy of the model and the computation time. The random forest classifier also adopts the data treatments of dummy variables and Unix time conversions that logistic regression requires. In addition, because random forest uses heuristics process, which allows for some randomness in model training, there will be variations in the tree structure each time a model is built. So we pass a specific seed to the random_state
value in order to ensure that the same result is generated in each run.

3.3 Graph-based approach

After the implementation of the two weak-supervised machine learning models, it is discovered that the interpretability and explainability of the models are quite low, which is problematic since one of the requirements in this thesis is to explain why a case is believed to be a potential fraud. What is more, there are some underlying relationships between the entities in the dataset, and the ML models fail to take those into consideration. As a result, a graph-based approach is proposed by converting the dataset into its corresponding knowledge graph representation and detecting anomalies on both structural-level, and statistical-level, as shown in the basic workflow in Figure 3.9. Section 3.3.1 explains in detail the graphical representations, and Section 3.3.2 and 3.3.3 examine the structural- and statistical-analysis respectively.
3.3.1 Knowledge graph

Knowledge graph (KG) is a knowledge base that uses a graph-structured data model or topology to integrate data, and it is often a collection of interlinked descriptions of entities, relationships, abstract concepts and events that also encodes the semantics underlying the used terminology [30]. A knowledge graph is a directed graph \( KG = \{V;E\} \) consisting of a set of vertices \( V \) that represent entities, classes, and literals, and a set of labeled edges \( E \) that connect these vertices. And the Resource Description Framework (RDF) [27] is a popular representation model for KGs.

RDF is a World Wide Web Consortium (W3C) recommended standard framework for describing information in the Web. It is based on the idea of creating a directed graph composed of a set of triple statements about some knowledge in the expression form \( \langle s;p;o \rangle \), where \( s \) refers to the subject, which denotes the resource and is either a uniform resource identifier (URI) or a blank node; \( p \) refers to the predicate, which is also a URI that indicates a trait of the resource and expresses a relationship between the subject and the object \( o \), which can be a URI, blank node or a Unicode string literal [85]. Here, \( s; o \in V \), and \( p \in E \). Figure 3.10 shows a basic RDF graph with two nodes (Subject, Object) and an edge connecting them (Predicate).

In addition, a collection of these RDF statements intrinsically represents a labeled, directed multi-graph, which makes the RDF data model better suited to certain kinds of knowledge representation than other relational or ontological models. Also, this simple and flexible data
model has a lot of expressive power to represent complex situations, relationships, and other things of interest, while also being appropriately abstract. Therefore, since the contract data we have has rather complicated relationships between the entities involved, we decide on RDF graphs, which, we think, should be able to capture those hidden relationships, while maintaining relatively acceptable computation time.

### 3.3.1.1 Ontology of KG

In order to convert the datasets into corresponding RDF graphs, Java’s Semantic Web and Linked Data framework Apache Jena was employed. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract “model”, and with this model, resources and properties can then be created and added to it [2].

In this case, a default empty memory-based model is created, then based on the characteristics of the data, six resources are identified and created by a URI:

- **Coordinator URI**: http://dcc.ca/c + coordinator ID number
- **Approver URI**: http://dcc.ca/a + approver ID number
- **Contract URI**: http://dcc.ca/ + contract ID number
- **Change Order URI**: Contract URI + co + change order ID number
- **Invoice URI**: Contract URI + inv + invoice ID number
- **Supplier URI**: http://dcc.ca/s + supplier ID number
Resources have properties. So for the six resources, we create six respective properties that are represented by arcs labeled with the names of these properties, which are also URIs:

- PREFIX co: http://data.linkedeodata.eu/ontology/CO#
- PREFIX coord: http://data.linkedeodata.eu/ontology/Coord#
- PREFIX apprv: http://data.linkedeodata.eu/ontology/Apprv#
- PREFIX cntrct: http://data.linkedeodata.eu/ontology/Cntrct#
- PREFIX inv: http://data.linkedeodata.eu/ontology/Inv#
- PREFIX suppl: http://data.linkedeodata.eu/ontology/Suppl#

These URIs are rather long and cumbersome, so usually, the properties are shorthanded and represented in the XML qname form for easier display in diagrams or texts. This shorthand for property URIs is expressed in a namespace concatenated with local name form, i.e., `nsprefix:localname`, where the part before “:” is called a namespace prefix and stands for a namespace, and the part after is called a local name and serves as a name in that namespace. The above six items are the namespace prefix part of the URIs, and together with some local names in their individual namespaces, they will generate the complete property URIs.

(a) RDF with Literal as Object

(b) RDF with URI as Object

Figure 3.11: Two Types of Object Values in Example RDF Graphs
Finally, a value is added to each property, which can be either another URI or a literal. Figure 3.11 shows example triples with two possible types of property values. In Figure 3.11a, the resource is a coordinator with “128623” as its ID number and is displayed as an ellipse; the property shown for the coordinator is the Request Number, which is the formal name for ID, and here “coord” is the namespace, while “Request Number” is the local name; and the value for the coordinator’s property is “128623”, which is a literal and is shown in a rectangle. On the other hand, the middle ellipse in Figure 3.11b is both the value for the “Issue_CO” property of coordinator “128623” and the resource for the property “co:Negotiated_Amount”, so it is represented using a change order URI in an ellipse, with “http://dcc.ca/68124” being the contract URI and “8” as its ID number, and the value for its property is finally a literal, the float amount “4110.47”.

Figure 3.12: Common Network Topology

After all data is extracted and stored as an RDF graph, it can be serialized as different formats to be used later on for analysis. Some common and now in use RDF serialization formats are Turtle, N-Triples, JSON-LD, and RDF/XML, etc. [85] In particular, RDF/XML [44] was historically the first W3C standard format for serializing RDF, and was first introduced as an XML serialization in the first recommended RDF specification, the Model and Syntax Specification (“RDF M&S”), in 1999 [56]. For this thesis, we also save the RDF graphs in the XML format, and Figure 3.13 shows an example RDF for some sample data extracted from the complete dataset. We can see that our graph looks significantly different from other common simple network topologies (shown in Figure 3.12), which is because of the complex relationships that exist between the various entities in our data.

37
Figure 3.13: Sample Data in RDF Format
3.3.1.2 SPARQL

In order to analyze the RDF graphs, we need to be able to query the graphs and retrieve the desired information. As a result, SPARQL [36], a SQL-like semantic query language for database, comes into play. It is the standard structured query language for querying RDF graphs and then extracting and manipulating the data stored. In addition, SPARQL provides a full set of analytic query operations such as JOIN, FILTER, SORT, and AGGREGATE; and apart from SELECT queries that are used to extract raw values and return results in a table format, it also supports ASK queries that provide simple True/False results, CONSTRUCT queries that transform extracted results into valid RDFs, and DESCRIBE queries that extract RDF graphs based on descriptions of query patterns and what the query processor deems as useful information.

In our case, listing 3.1 demonstrates a sample SPARQL query of our data that returns all change orders issued by coordinator “128623”, the contracts they are for, and their negotiated amounts and dates.

Listing 3.1: SPARQL Query Given Coordinator

```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX co: <http://data.linkeddata.eu/ontology/CO#>
PREFIX coord: <http://data.linkeddata.eu/ontology/Coord#>
PREFIX cntrct: <http://data.linkeddata.eu/ontology/Cntrct#>

SELECT ?coNum ?negAmt ?negDate ?orderNum 
WHERE {
  ?coord coord:Request_Number "128623" .
  ?co co:Change_Request ?coNum .
  ?co co:Negotiated_Amount ?negAmt .
  ?co co:Negotiated_Date ?negDate .
  ?cntrct cntrct:Order_Number ?orderNum .
}
```

39
Notice that the returned result does not have the coordinator number, hence if it is preferred to have such information included, the FILTER clause should be applied, and the SPARQL query would instead look like the following.

Listing 3.2: SPARQL Query with FILTER

```
SELECT ?coordNum ?coNum ?negAmt ?negDate ?orderNum
WHERE {
    ?co co :Change_Request ?coNum .
    ?co co :Negotiated_Amount ?negAmt .
    ?co co :Negotiated_Date ?negDate .
    ?cntrect cntrect :Order_Number ?orderNum .
    FILTER( ?coordNum == "128623")
}
```

3.3.2 Structural analysis

After the database is converted into an RDF graph, the next step would be to traverse and analyze the graph, and the approach we settle on is to query the graph based on 1, 2, ..., n-hops from a given entity of interest as root node to obtain various subgraphs. Then based on the relationships that might be found between various entities in the subgraph, the resulting outliers would have explanations corresponding to those relationships. The three entities that make sense to act as root nodes are the Coordinator, the Contract, and the Supplier, which can be changed in other usecases.

For the coordinators, 1-hop query returns the coordinator URIs, their ID numbers (request number), their individual DSA limits, and the URIs of all change orders that each coordinator issues. Combining all the resulting information, we actually end up with a network structured graph, with each subgraph as a star-shaped graph that has the coordinator as the central node, or the root node, and the ID, DSA
limit, and all the change orders that he/she issues as the leaves. Since analyzing the ID numbers and the DSA limits alone would not make much sense, the only logical analysis would be the coordinators and the number of change orders they each issue. Figure 3.14 displays a graph sample from the 1-hop from coordinators query, where each subgraph represents a coordinator and all change orders he/she issues.

Figure 3.14: 1-Hop from Coordinator Query Sample Subgraphs

In order to process the subgraphs and measure the similarities or the dissimilarities between each other, Graph Edit Distance (GED) algorithm is deployed. Its basic idea is to find the best set of transforma-
tions that can transform graph $g_1$ into graph $g_2$ by means of graph edit operations on $g_1$ [6], and the formal definition of graph edit distance between two graphs $g_1$ and $g_2$, which is written as $GED(g_1, g_2)$, is

$$GED(g_1, g_2) = \min_{(e_1, \ldots, e_k) \in \mathcal{P}(g_1, g_2)} \sum_{i=1}^{k} c(e_i)$$  \hspace{1cm} (3.10)$$

where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming $g_1$ into $g_2$ or a graph isomorphic to $g_2$, and $c(e) \geq 0$ is the cost of each graph edit operation $e$. And the set of graph edit operators allowed are insertion, deletion, and substitution of both vertices and their corresponding edges.

Without loss of generality, and for the sake of easier readability, our approach is explained using the coordinator as the type of interest in the analysis. However, the same concepts discussed next can be applied on all other entity types in the dataset. Applying the GED algorithm to our case, we calculate the pairwise GEDs between all of the subgraphs and end up with an $n \times n$ distance matrix $\mathcal{A} = (a_{ij})$, where $n$ is the total number of coordinators in the graph, and $i, j \in \{1, \ldots, n\}$. Then, $a_{ij}$ is the distance between the subgraph $i$ and the subgraph $j$, which represent two individual coordinator and their respective change orders.

In addition, as the different nodes - coordinator nodes vs. change order nodes - in the graph can be seen as of different types, it is important to be able to differentiate between them when analyzing the graph and calculating GED. As a result, the coordinators are set as the root node in each subgraph, and they are forced to be matched in the comparison to enable comparison between rooted graphs.

Examining the results from using the GED measure, an outlier in this distance matrix from the 1-hop from coordinators query could mean a coordinator issues abnormally high number of change orders. Similarly, 1-hop from contracts query returns all information about the contracts including the number of change orders issued for them, so the outcome of its fraud analysis would be contracts that have abnormally high amount of change orders. Finally, 1-hop from suppliers query gives us the supplier ID numbers and the number of invoices they write to DCC, and the pairwise GED matrix would reveal anomalous
suppliers that have written way too many invoices.

In addition to generating 1-hop subgraphs, we also generate 2-hops subgraphs utilizing graph edit distance to capture more complex relationships between the entities in our dataset; that is, 2-hops from coordinators, 2-hops from contracts, and 2-hops from suppliers. In both 2-hops from coordinators and 2-hops from contracts queries, detailed information about the change orders are included, which allows us to explore the number of change orders of a given type by either coordinator or contract, specifically planned changes. Planned changes are changes that are informed and reserved well in advance, so they would not be investigated or suspected, which might be taken advantage of, resulting in anomalous coordinators or contracts with a high number of planned changes. On the other hand, the 2-hops from suppliers query expands beyond the invoices to the contracts that those invoices are written for, as shown in figure 3.15. So, its GED outliers would identify suppliers with an extremely high volume of invoices per contract.

Figure 3.15: 2-Hops from Supplier Query Sample Subgraphs

43
3.3.3 Statistical analysis

Besides the graph edit distance technique, which investigates the structural relationships between different entities, various distance functions and Euclidean distances are defined and calculated in order to take into account the statistical properties of the graph when detecting fraud.

3.3.3.1 Distance function

As mentioned in the previous section, 2-hops from coordinators query retrieves all details about the change orders on top of the coordinator information from the 1-hop query. So, in addition to using GED to study planned changes by a coordinator, relationships between certain closely related variables or nodes can be highly valuable to explore as well. For example, the Negotiated Amount of a change order vs. the Quoted Amount of a change order. The quoted amount is the value on a quote from the contractor and is how much the contractor thinks the change is going to take, while the negotiated amount is the final amount that DCC and its contractor agree on after their negotiation. So generally speaking, the negotiated amount should be lower than the quoted amount, as that is the purpose of negotiation, and if the negotiated amount is very close to, the same as, or even higher than the quoted amount, it could be a potential fraud. Moreover, if the negotiated amount is much lower than the quoted amount, it could also be an anomaly. Therefore, we define a distance function between the two attributes that computes the difference between their amounts, and since we are detecting on a coordinator basis, the average of this difference over all change orders by a coordinator is taken in order to tell if it is a one-time extreme event or a consistent pattern. A pairwise distance matrix is also being created using this difference for all coordinators, and the outliers of this distance matrix would indicate outlying coordinators who have immensely different quoted vs. negotiated amount compared to other coordinators. The same analysis is done for estimated vs. quoted amount, estimated vs. quoted date, and negotiated vs. quoted date.

Another example from 2-hops from coordinators query is the Negotiated Amount vs. the coordinator’s DSA Limit. The reasoning be-
hind this analysis is that changes can only be approved by personnels with the appropriate authorities, which means that coordinators cannot approve change orders that exceed their DSA limits. So, in order to make a fraudulent activity seem legit, a coordinator would have to issue change orders with their negotiated amounts just below his/her DSA limit, so that these changes would not need to be directed to a higher authority. Hence, the distance function and the analysis for this comparison follows the same thought flow and steps as those for negotiated vs. quoted amount.

For this thesis, there are also some functions defined by DCC that are specific to the construction industry, like the Unplanned Cost Growth, so they are also incorporated as distance functions for the statistical studies. The UCG can be calculated for all three root entities - coordinators, contracts, and suppliers, and in order to acquire necessary information for the calculation, 3-hops from coordinators, 2-hops from contracts, and 4-hops from suppliers queries need to be executed. Then, with the pairwise distance matrix generated for these queries respectively based on their distance functions, their individual outliers would reveal questionable coordinators, contracts, and suppliers with extreme UCGs.

3.3.3.2 Euclidean distance

In the previous analysis, we are only considering at most two dimensions and one technique at a time, either the GED, or the difference function, or the user defined function. So in order to make the study more comprehensive, we turn to vector space and employ Euclidean Distance to discover possible fraud cases based on multiple attributes simultaneously, which can be helpful when these attributes have close interrelationships between each other, such as the estimated, quoted, and negotiated amounts and dates of a change order.

The Euclidean distance between two points in the Euclidean space is the length of a line segment between the two points, and it can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem [83]. The Cartesian coordinate of a point usually
Figure 3.16: Euclidean Distance in Different Dimensions, from [83]
denotes as \((x, y)\), with \((0, 0)\) as the origin of the coordinate system; here, the distances from the point to the X-axis and to the Y-axis are \(|y|\) and \(|x|\), respectively. And the Pythagorean theorem, which is fundamental in modern day Geometry, states that the area of the square whose side is the hypotenuse (the side opposite the right angle) is equal to the sum of the areas of the squares on the other two sides, i.e. \(a^2 + b^2 = c^2\), where \(a, b,\) and \(c\) are the three sides of a right triangle.

Diagrams in Figure 3.16 illustrate how Euclidean distance is computed using Pythagorean theorem in both two and three dimensions. So in two-dimensional space, let a point \(p\) have Cartesian coordinates \((p_1, p_2)\) and point \(q\) coordinates \((q_1, q_2)\), then the Euclidean distance between the two points \(p\) and \(q\) is
\[
d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2},
\]
by applying the Pythagorean theorem to a right triangle with horizontal and vertical sides, having the line segment from \(p\) to \(q\) as its hypotenuse, as explained in Figure 3.16a. Continuing to higher dimensions, in three dimensions (Figure 3.16b), the distance between two points with Cartesian coordinates is
\[
d(p, q) = \sqrt{\left(\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}\right)^2 + (p_3 - q_3)^2}
\]
\[
= \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.
\]
And in general, for two points with Cartesian coordinates in \(n\)-dimensional Euclidean space, the distance is
\[
d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.
\]

In this case, the Euclidean distance measure can be applied again to the \(2\)-hops from coordinators query, and the attributes used are also the estimated, quoted, and negotiated amounts and dates, the number of extend days of each change order, and the DSA limits of the coordinators, which excludes from the query result only the categorical variable.
change order *type* and some additional ID columns for the change orders. Since we also conduct the analysis based on coordinators, the distance functions involving these attributes from the previous technique are used, i.e., difference between estimated and quoted amounts and dates, difference between negotiated and quoted amounts and dates, and difference between negotiated amount and coordinator DSA limit. Then, together with the average days extended by each coordinator, the final data would be three amount difference variables and three number of days difference variables for each coordinator.

Before the Euclidean distance between these coordinators can be calculated pairwisely, the data needs to be normalized because of the different units used in each attribute. *Data normalization* often means dividing the data in a dataset by a norm of the data, the goal of which is to adjust the values of numeric features measured on different scales to a notionally common scale, without distorting differences in the ranges of values. This could, in turn, improve the performance and stability of any following techniques.

There are various approaches for normalizing data, such as the *Standard score*, the *Studentized residual*, and the *Min-max feature scaling*, that leverage different types of normalizations in statistics, like nondimensional ratios of errors, residuals, means, and standard deviations [81]. One of the most common and perhaps simplest method is *Min-max feature scaling*, which brings all values into the range [0, 1] using the formula:

$$X' = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}},$$

(3.14)

where $X$ is the original values, and $X'$ the normalized values. And in this case, we choose to feature scaling the data with min-max scaler and transform the attributes into the [0, 1] range.

Applying data normalization to our data ensures that no single particular attribute could significantly affect and govern the resulting Euclidean distance and cancel out the contributions from all other variables, so that each feature contributes approximately proportionately to the final distance. And after the normalization, pairwise Euclidean
distance is computed for all coordinators, and coordinators with abnor-
mal change order behaviours would be singled out as the outliers in the
pairwise Euclidean distance matrix.

3.4 Outlier detection metrics

To evaluate our approaches in line with the purpose of this thesis, i.e.
fraud detection or anomaly detection, techniques for identifying outliers
need to be examined. Two of the most widely used and effective criteria
are Z-Score and Interquartile Range.

3.4.1 Z-score

The principle of Z-score revolves around the Standard Deviation (STD),
which is a statistical measure of the amount of variation or dispersion of
a set of values. It is relative to the mean of the dataset and is calculated
as the square root of the variance. Let \( \mu \) be the expected value (mean)
of a finite set of random values \( X = [x_1, x_2, ..., x_N] \), with each value
having the same probability, \( x_i \) being the value of the \( i^{th} \) point in the
dataset, and \( N \) the number of data points in the set,

\[
\mu = \frac{1}{N} (x_1 + x_2 + ... + x_N)
\]

\[
= \frac{1}{N} \sum_{i=1}^{N} x_i
\]

(3.15)

so the standard deviation is calculated as

\[
\sigma = \sqrt{\frac{1}{N} [(x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (x_N - \mu)^2]}
\]

\[
= \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}
\]

(3.16)

Since standard deviation measures how dispersed a dataset is, a
low standard deviation indicates that the values tend to be clustered
closely around the mean of the data, while a high standard deviation indicates that the values are spread out far from the mean over a wider range [86]. This particular property of standard deviation makes it an especially useful tool for detecting outliers.

For Z-score, the mean and standard deviation of the data points are calculated and compared. Figure 3.17 indicates the percentage of values expected to lie within and outside the symmetric interval of different standard deviations from the mean of the dataset, assuming normal distribution. We can see that if the data distribution is assumed to be approximately normal, about 68% of the data are within one standard deviation from the mean \((\mu \pm \sigma)\), about 95% lie within two standard deviations \((\mu \pm 2\sigma)\), and about 99.7% are accounted for within three standard deviations \((\mu \pm 3\sigma)\). So, if a value is a certain number of standard deviations away from the mean, that data point is considered an outlier. And based on this 68-95-99.7 rule, it is safe to conform to a conventional heuristics that nearly all values are taken to lie within three standard deviations of the mean \((3\sigma \text{ interval})\). Therefore, that specified threshold would take as default value 3 standard deviations from the mean.

![Figure 3.17: Standard Deviations in Normal Distribution](image)

However, as seen from the data exploration process of this thesis,
the data we use is not normally distributed at all, which defies the underlying assumption of detecting outliers with standard deviation. As a result, we turn to the Interquartile Range technique. Nevertheless, thanks to the Chebyshev’s inequality, we learn that the 3σ interval would still be able to cover almost 90% of all cases even in non-Gaussian distributions, so we end up sticking to standard deviation as the metric for detecting outliers.

3.4.2 Interquartile range

The Interquartile Range (IQR) is also a measure of data dispersion, and it is defined as the spread difference between the 75th and 25th percentiles of the data. To calculate the IQR, the dataset is divided into quartiles, with \( Q_1 \) denotes the lower quartile, corresponding with the 25th percentile, \( Q_2 \) the median at the 50th percentile, and \( Q_3 \) the upper quartile that corresponds with the 75th percentile. So,

\[
IQR = Q_3 - Q_1.
\]  

(3.17)

The interquartile range is often used to find outliers in a dataset. To determine if a certain value is an outlier, the IQR is multiplied by 1.5, which defines a new range:

\[
Lower \ Bound : Q_1 - 1.5 \times IQR,
\]  

(3.18)

\[
Upper \ Bound : Q_3 + 1.5 \times IQR,
\]  

(3.19)

with any data values that are less than the Lower Bound or greater than the Upper Bound being considered as outliers. This IQR based outlier detection method is often referred to as the 1.5-IQR rule. Figure 3.18 compares the percentage of data accounted for by the 1.5-IQR rule vs. the 3σ interval, We can see that the 1.5-IQR rule actually cannot cover as much data points as when using standard deviation, so the outliers returned would not be as extreme or as concise.

However, such conclusion is based on the assumption of normal distribution by the data, and when a dataset has outliers that skew the data, the median actually becomes a better measurement of central tendency than the mean, and the IQR a better measurement of spread.
than standard deviation [42]. That is because mean and standard deviation take into account all points in the dataset, including the outliers, but median and IQR can ignore those outliers, resulting in more accurate measurements of the data. So if the data is fairly symmetrical or have no outliers, mean and standard deviation can be used for central tendency and spread, respectively. But if the data is skewed or if there are extreme outliers, use median and IQR instead.

As a result, since our data is not normally distributed and is often highly skewed, we resort to using 1.5-IQR to find outliers. However, after applying it to the dataset, the number of possible anomalies returned is way too much to be actual extremes. So an even stronger IQR-
based rule targeting extreme deviations from the rest of the dataset is introduced. The 3-IQR rule, which changes only the number that we multiply the IQR by is defined as:

\[
\text{Lower Bound} : Q_1 - 3 \times IQR, \quad (3.20)
\]

\[
\text{Upper Bound} : Q_3 + 3 \times IQR, \quad (3.21)
\]

and the points that fall beyond this range are considered extreme outliers. Employing this rule on our dataset significantly reduces the number of cases returned, but the number is still not optimal. Since the outputs of our system are to be verified by domain experts, overwhelming them with a large number of outliers to investigate is prohibitive.

<table>
<thead>
<tr>
<th>Distance from mean</th>
<th>Minimum population</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{2}\sigma$</td>
<td>50%</td>
</tr>
<tr>
<td>$2\sigma$</td>
<td>75%</td>
</tr>
<tr>
<td>$3\sigma$</td>
<td>89%</td>
</tr>
<tr>
<td>$4\sigma$</td>
<td>94%</td>
</tr>
<tr>
<td>$5\sigma$</td>
<td>96%</td>
</tr>
<tr>
<td>$6\sigma$</td>
<td>97%</td>
</tr>
<tr>
<td>$k\sigma$</td>
<td>$1 - \frac{1}{k^2}$</td>
</tr>
<tr>
<td>$\frac{1}{\sqrt{1-l}}\sigma$</td>
<td>$l$</td>
</tr>
</tbody>
</table>

Table 3.2: Data % within $k$ STD from Mean, from [79]

After encountering problems with using IQR in outlier detection, we turn our attention back to standard deviation, because we discover the Chebyshev’s inequality, which ensures that, for all probability distributions for which the standard deviation is defined, no more than a certain fraction of values can be more than a certain number of standard deviations from the mean [79]. Table 3.2 gives the corresponding distance
and the minimum amount of data within that distance from the mean. And since at least 88.8% of all cases would fall within properly calculated Z-score, even for non-normally distributed data, we utilize the standard deviation technique to detect outlying entities in our dataset.
Chapter 4

Experimental Evaluation

This chapter describes how the procedures and methods used in the thesis are implemented and evaluated, the different standards used and the directions taken. And it also demonstrates the performance of these approaches based on selected standards.

4.1 Implementation

This thesis is implemented in Python 3.8 and Java SE 16. Jupyter notebooks and Spyder are used for data extraction, exploration, preprocessing, and visualization; they are also responsible for executing code for building semi-supervised machine learning models, and querying and analyzing graph representations. On the other hand, Eclipse IDE is used for the intermediate step of statistical database to knowledge graph conversion.

NumPy (specific version in Appendix A) and Pandas are two of the most important core libraries in Python, and are heavily used throughout the thesis for their high performance scientific computing and easy-to-use data structures and data analysis tools. SciPy provides a large menu of mathematical computations for statistic data exploration, and Scikit-learn, a Python module built on top of SciPy, is used for advanced machine learning modeling. In addition, Matplotlib and Seaborn are used for data visualization.
Since the data for this thesis is expected to be extracted automatically from DCC’s database, PyODBC, an open-source module that implements the Python Database API Specification v2.0, is used in order to access the ODBC interface to interact with the database.

For the graph representations, Java provides a Semantic Web framework called Apache Jena to write data to RDF graphs. Then, in Python, RDFLib parses the RDF and poses SPARQL queries to the graphs, and NetworkX constructs subgraphs and studies their structures using standard graph algorithms.

Although analyzing graphs can be computationally expensive and the problem of computing graph edit distance is, in general, NP-hard, the volume of data in this thesis is considerably less than those with millions of nodes and edges. Therefore, it is feasible to do all analysis on a personal laptop using CPUs.

4.2 STD vs. IQR

One essential aspect of determining whether to use standard deviation or interquartile range for outlier detection is their number of outputs, as the amount of returned potential fraud needs to be manageable for domain expert to review.

Figure 4.1 presents the number of possible frauds return by the three approaches - Z-score, 1.5 IQR, and 3 IQR - based on two fraud scenarios. We can see that both lower and upper outliers from standard deviation are significantly less than those from the other two methods, and by inspecting the returned cases from all three approaches, we find out that the result from standard deviation is a subset of the other two results, i.e., all anomalies detected by standard deviation are also detected by the others. Also, the Z-score results are much more extreme or abnormal compared to those from the other techniques. Considering the purpose of this thesis, we believe that it is more effective for any future analysis to have a more concise set of anomalies that also appear more extreme and fraudulent. Therefore, we settle on standard deviation as our outlier detection method.
4.3 Performance of ML models

After the criteria for outlier detection are settled, they are applied to the scenarios identified in the workshop, in order to obtain anomalies and potential fraud cases both for labeling training data for the machine learning models and in the 1,2,..., n-hops query analysis on the graph representation. What remains is how to evaluate the performance of the weak-supervision ML models and the graph-based approach.

4.3.1 K-fold cross validation

For the ML models, we choose a non-exhaustive cross validation (CV) method, K-fold Cross Validation. Cross validation is a model validation technique that assess the generalizability of a result from a statistical analysis to an independent set of data. It resamples different portions of data to train and test on various iterations and is mainly used in prediction settings, where users want to estimated how accurately a predictive model will perform in practice.

K-fold cross validation specifies how the data is sampled. The original dataset is randomly partitioned into $k$ subsamples of equal size, and of the $k$ subsamples, one subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are treated
as the training data. The cross validation process is then repeated for $k$ times, with each of the $k$ subsamples used exactly once as the validation set. And finally, the $k$ results are averaged to produce a single estimate of prediction accuracy.

The next step is to decide on the scoring metrics for the $k$-fold CV. Since the thesis on hand is a classification problem, the Accuracy measure naturally comes to mind. It is the most commonly used metric for model selection, and based on the confusion matrix below (Table 4.1), the accuracy of a model is calculated as:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}}$$

$$= \frac{\text{True Positive} + \text{True Negative}}{\text{All Sample}}. \quad (4.1)$$

However, the goal of this thesis is actually fraud detection, and if a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence could be devastating - companies could lose millions of dollars due to this one missed fraud. As a result, the number of False Negatives needs to be minimized, and the metric Recall seems to be a better fit for selecting models:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$= \frac{\text{True Positive}}{\text{Total Actual Positive}}. \quad (4.2)$$
On the other hand, if a normal transaction (Actual Negative) is predicted as fraud (Predicted Negative), companies would only have allocated some resources to confirm it being genuine, which typically would not be as significant a loss as overlooking a fraudulent activity. So, the False Positives, thus the Precision metric, is not focused in this thesis.

### 4.3.2 ML models

Before evaluating the machine learning models, the number of trees fit in the random forest classifier needs to be decided. Figure 4.2 displays the accuracy of a random forest based on different numbers of trees used in the model, and we can see that the accuracy does increase with more trees. However, the rate of such improvement decreases as the number of trees increases, which means that at a certain point, the benefit in prediction performance from learning more trees will be lower than the cost in computation time for learning those additional trees. As a result, we settle on 500 trees after considering both the accuracy of the model and the execution time.

Now we can evaluate the ML models using the labels output by Snorkel as the basis. To assess the prediction performances of the two classifiers, 3, 5, 10-fold CV are operated, and both the accuracy of the recall of the models are reported. However, we focus more on the recall scores of the models, since for fraud detections, it is more important to have a minimized false negative, in order not to pass on any potential fraudulent activities. Table 4.2 shows both the accuracy and the recall for the logistic regression and the random forest classifiers based on different $k$ values in $k$-fold CV.

We can see that logistic regression has higher accuracies on the change order dataset compared to those on the payment dataset, whereas random forest performs better on the payment dataset regardless of whether it uses accuracy or recall as metric. Also, the accuracy and recall of logistic regression based on different $k$ values in $k$-fold CV stay the same throughout, and its recall of 0 indicates that none of the labeled fraud cases are successfully detected by the model. On the other hand, both the accuracies and the recalls of random forest improve as
Figure 4.2: Accuracy vs. Number of Trees in Random Forest

$k$ increases in $k$-fold CV, since a larger subset of data is used to train the model; and in terms of the recall scores, they are at an acceptable level for a preliminary ML model. To summarize, in either datasets and based on either metrics, random forest performs a lot better than logistic regression does.

However, although random forests seem to perform reasonably well based on the returned accuracy and recall scores, they lack on the explainability of why certain cases are predicted as fraud, which is actually one of the requirements and preferred feature of this thesis. In addition, since there are no identified frauds in DCC’s contract management activities, i.e. no ground truths, all labels used in the evaluation are generated by Snorkel and cannot be verified, so even if the random forest classifiers have relatively high accuracy and recall in predicting those labels, we cannot be one hundred percent sure about its performance under real circumstances.
### Table 4.2: Model Performance based on \(K\)-fold CV

<table>
<thead>
<tr>
<th></th>
<th>3-fold</th>
<th>5-fold</th>
<th>10-fold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logistic Regression</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.8445</td>
<td>0.8445</td>
<td>0.8445</td>
</tr>
<tr>
<td>Recall</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Random Forest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.9332</td>
<td>0.9368</td>
<td>0.9400</td>
</tr>
<tr>
<td>Recall</td>
<td>0.6093</td>
<td>0.6321</td>
<td>0.6518</td>
</tr>
</tbody>
</table>

(a) \(K\)-fold CV on Change Order Dataset

<table>
<thead>
<tr>
<th></th>
<th>3-fold</th>
<th>5-fold</th>
<th>10-fold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logistic Regression</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.8090</td>
<td>0.8090</td>
<td>0.8090</td>
</tr>
<tr>
<td>Recall</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Random Forest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.9507</td>
<td>0.9521</td>
<td>0.9538</td>
</tr>
<tr>
<td>Recall</td>
<td>0.8612</td>
<td>0.8695</td>
<td>0.8776</td>
</tr>
</tbody>
</table>

(b) \(K\)-fold CV on Payment Dataset

### 4.4 Performance of graph-based approach

The graph-based approach is more difficult to evaluate because there are neither any known fraud cases in our dataset, nor any weak labels that can be leveraged like those in the case of semi-supervision. As a result, a different technique is used to evaluate the graph-based approach.

#### 4.4.1 Outlier injection

A potential direction would be to inject fraud cases manually based on the scenarios developed by the domain experts, and see whether our approach can detect these cases.
For example, the first set of fraud scenarios deals with unplanned cost growth. So, if cases for “coordinators with extreme unplanned cost growth” are to be injected, the UCG by each coordinator needs to be calculated, and the potentially fraudulent coordinator would have a UCG more than 3 standard deviations (3σ) from the mean (μ) UCG of all coordinators. Thus, for the fraud case, a random UCG value is generated that lies beyond the μ ± 3σ interval, but is greater than the min of the UCG and smaller than the max. Since the UCG is computed as

\[ UCG = \frac{(S + D + O)}{(Award\ Amount + P)}, \]

i.e., the sum of unplanned changes divided by the sum of contract award amount and planned changes, two random numbers are generated with their quotient equal to the randomly generated UCG value, and two numbers would represent the sum of unplanned changes \((S + D + O)\) and the sum of contract award amount and planned changes \((Award\ Amount + P)\), respectively. Given the sum of unplanned changes, three additional numbers that sum to the given value are generated for the three specific types of unplanned changes. Then for each of those three values, a random number of change orders of that type are generated, with their negotiated amounts sum equals one of the given number; for example, \(Neg.Amt_1 + Neg.Amt_2 + ... + Neg.Amt_n = S\), where \(n\) is the randomly generated number of change orders of unplanned type “S”, and \(S\) is the randomized sum amount of all unplanned type “S” change orders. Similarly, given the sum of contract award amount and planned changes, two random numbers, one for contract award amount and the other for sum of all planned changes, are generated, and the value for sum of all planned changes is again divided into smaller negotiated amounts for multiple planned change orders, i.e., \(Neg.Amt_1 + Neg.Amt_2 + ... + Neg.Amt_m = P\).

So far, all information related to the calculation of one fraud case, in this case one coordinator, is completed, and the next steps would be adding complementary dummy values to irrelevant attributes to finish up the case and make it look the same as the already existed data. The frequently used ID numbers of the contract, the coordinator, and the supplier are each set to be a random value that looks like the original
values in the column, so that all changes created just now are issued by the same coordinator for the same contract with the same supplier. Another set of essential attributes are the estimated, quoted, and negotiated amounts and dates, and they are all set to be the same amount and the same date, respectively. Throughout this anomaly injection process, the most basic, yet crucial rule to keep in mind is that the injected data must be in the same forms as the existing data and that the variables must be in the appropriate ranges. For example, the contract types in the original dataset are “CN”, “KN”, “FM”, and “SC”, so the contract type of an injected case must also be one of those values; also, the contract award date of an injected cases must be within the range “2019-01-01” to “2021-12-31”, same as the existing data, etc. And once the case is complete, more cases under the same fraud scenario can be generated the same way, by repeating the above process the same number of times as the number of cases a user would like to inject.

In total, 12 functions for injecting fraud cases are created (shown in Appendix D). However, because of how the complementary information are generated during the case injection process, almost all of the fraud scenarios can be covered by these synthetic data. For example, when the estimated, quoted, and negotiated amounts and dates are not the main focus of an injection function, the amounts are set to be the same value and so are the dates, which fulfills several additional fraud scenarios (Scenario 2(g) - 2(j)) other than the one dealt by the injection function.

4.4.2 Graph-based approach

We use synthesized fraud data and whether or not they can be recognized to assess the performance of this approach. We obtain only the recall score for this approach, since we only have information about the actual positives (injected cases), which makes calculating accuracy impossible.

For the evaluation, we inject fraud cases according to the 12 functions that cover almost all fraud scenarios identified by the domain experts. The number of cases created for each function is based on
different percentages of the entire dataset, and we decide to test the graph-based approach by injecting fraud cases that make up 1%, 5%, and 10% of existing data. Once the fraud cases for one function are injected into the original datasets, the data is then converted into graph representations as explained in Section 3.3.1, and the graph is analyzed using the structural and statistical anomaly detection approaches discussed in Sections 3.3.2 and 3.3.3. Finally, we inspect how many injected cases are returned out of all injected cases, i.e. the recall score. Table 4.3 shows the performance of the graph based approach based on how well it can discover synthetic fraud cases that make up different percentages of the existing data.

<table>
<thead>
<tr>
<th>Function</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function 1</td>
<td>1</td>
<td>0.9444</td>
<td>0.9722</td>
</tr>
<tr>
<td>Function 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Function 3</td>
<td>1</td>
<td>0.8333</td>
<td>0.8889</td>
</tr>
<tr>
<td>Function 4</td>
<td>1</td>
<td>0.8889</td>
<td>1</td>
</tr>
<tr>
<td>Function 5</td>
<td>0.5000</td>
<td>0.8333</td>
<td>0.7222</td>
</tr>
<tr>
<td>Function 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Function 7</td>
<td>1</td>
<td>0.8889</td>
<td>1</td>
</tr>
<tr>
<td>Function 8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Function 9</td>
<td>1</td>
<td>0.7222</td>
<td>0.6667</td>
</tr>
<tr>
<td>Function 10</td>
<td>1</td>
<td>1</td>
<td>0.9167</td>
</tr>
<tr>
<td>Function 11</td>
<td>1</td>
<td>0.7778</td>
<td>0.8333</td>
</tr>
<tr>
<td>Function 12</td>
<td>0.7500</td>
<td>0.4444</td>
<td>0.4722</td>
</tr>
</tbody>
</table>

Table 4.3: Graph-based Approach Performance

We can see that most of the functions have relatively high recall
scores, which is due to the fact that although there are only 12 functions, they cover almost all of the fraud scenarios. So, even if they are not detected by the approach for the reason they are supposed to, there is still a chance for them to be returned by other distance function techniques. Also, the performance for injecting 1% of all data seems better than that of the 5% and 10%, with most recall scores being perfectly 1. This is because much less fraud cases are injected, so it is easier to recognize all of them. However, if one case is not returned by the approach, the recall would drop significantly as well.

On the other hand, some functions inject fraud cases that are not well-detected by our approach. The reason behind this could be because, all these cases are created based on either standard deviations or given parameters in some particular scenarios, but they are not detected accordingly; the graph-based approach detect outliers from a distance matrix, where these injected cases may not appear as extreme. In addition, there are two ends for finding possible outliers, \( \min \) to \( \mu - 3\sigma \) and \( \mu + 3\sigma \) to \( \max \). And if one of the interval is considerably shorter, and at the same time, there exists an extreme value on the other end of the spectrum, it might cloud the effects of these fraud cases on the matrix, thus later on the outlier detection.

Another observation from the table is that function 6 seems to have poor performances. This function creates cases with negotiated amounts that are extremely close to the coordinator DSA limits. However, similar to our previous standpoint, when shown in a distance matrix, these close differences may not have as much of an impact as a negotiated amount extremely apart from the DSA. Also, the DSA limits do not have huge values, so for the majority of normal cases, there actually is not much room for difference, and therefore, a really small negotiated vs. DSA value may not stand out and be as apparent as an outlier.

The results from function 12 also do not appear satisfactory. This function generates first progress claims that are more than \( \frac{1}{\text{contract duration in months}} \) of total contract value, which is a very specific rule defined by DCC. However, in the graph-based approach, it cannot
be detected exactly accordingly to this rule; instead, extremely large first payments compared to contract values are returned. As a result, if the duration of a contract is quite long, resulting in a relatively low percentage from the fraction, the actual value of the first invoice compared to the contract value would not be as extreme, thus may not be recognized as an outlier in the distance matrix.
Chapter 5

Conclusion

This thesis aims to design and implement an automated fraud detection system that is able to not only report possible fraudulent transactions in the past but also predict potential fraud in the future. There are extremely limited number of research on fraud detection in contract management, so we come up with our own method. Two different techniques are employed in order to detect fraud, the semi-supervised machine learning approach and the graph-based approach. The semi-supervised approach is adopted to address our unlabeled dataset due to no prior identified fraud cases, and the graph-based approach is proposed to measure the complicated relationships between the entities involved and to take into consideration both the structural and the statistical properties of these nodes.

The performances of both techniques are promising. For the semi-supervised ML models, although the results from logistic regression are not optimal, the accuracy and recall of random forest calculated based on the labels output from the weak-supervision system Snorkel are much better. However, since the labels cannot be easily verified due to no associated explanations, the performance of the ML algorithms under real circumstances remains doubtful.

On the other hand, the graph-based approach is able to return relatively high recall for almost all injection functions when discovering synthetic fraud cases that make up different percentages of the exist-
ing data. Therefore, we can see that this method is able to capture both the structural relationships and the statistical properties between entities pretty well. In addition, since the graph is traversed based on different hops from a given root node, the relationships found in these queried subgraphs provide the explanations for each of the detected fraud cases, compensating for the lack of explainability in the semi-supervised technique.

5.1 Impact & Limitation

In practice, the automated fraud detection system developed in this thesis can be applied to a contract management dataset, where it outputs some potential fraud cases or anomalies, along with explanations for each cases, which can then be directed to domain experts for review and verification. As a result, our approaches do not deliver any fraud promises; the resulting cases need to be examined before reaching a final decision, which can be a bit limiting.

Another potential limitation of this thesis lies within the evaluation of the approaches adopted. Although the results from both weak-supervised ML, especially random forest, and graph-based approach are quite promising, they utilize two different evaluation methods: the weak-supervision approach uses labels output by Snorkel as ground truth for cross validation, whereas the graph-based approach injects individual fraud cases that it would attempt to detect later on. As a result, the results from the two approaches cannot be compared with each other, hence, we cannot say which technique is better.

5.2 Future work

One possible future improvement upon our current approach is the semi-supervised models. Firstly, since we only employ two machine learning models in this thesis, we could include more models later on, such as Support Vector Machines (SVM) or Neural Networks, etc., so that the analysis would be more extensive and comprehensive. However, the issue of explainability remains with the ML technique, but
if the training labels output from the weak-supervision sources can be somehow explained, making it feasible for domain experts to review and determine whether they are actual frauds, the ML models built afterwards would be able to offer more definite and reliable performances. In addition, the current algorithms are still fairly preliminary, without addressing some potential problems within the dataset, such as the imbalanced classes, which is a very common problem in fraud detection in all fields. So although our data is not as extremely imbalanced as many other datasets, mitigating this issue would still benefit the performance of our models.

Another future work would be to generalize our approach to other fields and industries beyond contract management. There are many fields that have data with complex relationships between involved entities, so generalizing our methods to these areas can help them detect potential fraudulent cases by learning these relationships. What is more, our method includes some fraud scenarios and functions defined by DCC based on their contract management activities, so allowing users to define and input their own rules specific to their circumstances would make the generalized approach more applicable and appropriate to individual industries.
Appendix A

Software version

The specific versions of the Python libraries used in the implementation of the thesis are

- Numpy 1.22.2
- Pandas 1.4.1
- SciPy 1.8.0
- Scikit-learn 1.0.2
- Matplotlib 3.5.1
- Seaborn 0.11.2
- PyODBC 4.0.32
- RDFLib 6.1.1
- NetworkX 2.7.1
Appendix B

Data extraction

The data used in this thesis is extracted from the DCC’s JD Edwards (JDE) Enterprise Resource Planning (ERP) software EnterpriseOne (E1). E1 makes use of IBM Db2, which is a family of database products developed by IBM and a Relational Database Management System (RDBMS). The Db2 database is designed to deliver high performance, actionable insights, data availability and reliability for advanced analytics operations and transactional workloads.

In order to automatically access and retrieve the data for this thesis, the Python environment for SQL database server and an ODBC driver need to be set up first by installing the SQL Server ODBC Driver, then establish a connection to the server with the data source name (DSN). After that, SQL queries can be created and executed in order to select specific columns from multiple inner or left joined tables based on the conditions we set.
Appendix C

Data preprocessing

The data cleaning process is based on some particular knowledge acquired about construction contract management and on how these data would be used in the thesis.

All contracts of type “R0” are removed since they are not relevant to the purpose of the thesis. Also, all the date columns in the dataset are using *Ordinal Date*, which was formerly called the “Julian Date”. The ordinal dates are in the format of “Iyyddd”, where “yy” is the last two digits of the year, and “ddd” is the number of days from the beginning of the year, i.e. “01-01”, to the given date, so “ddd” ranges between 1 and 365 in a common year, and between 1 and 366 in a leap year. For this thesis, the ordinal dates need to be converted to the Standard date time format, so for example, an ordinal date “121060” would be converted to “2021-03-01 00:00:00” in standard date time, whereas “120060” would be “2020-02-29 00:00:00”.

In addition, the unit for all the amount columns is actually hundredth at the time of extraction, so in order to obtain the correct values, all amounts need to be divided by 100. Then for easier use, the amount columns, as well as all ID number columns for contracts, change orders, suppliers, coordinators, and approvers, are converted to either integer type or float type.

Another set of variables that needs to be treated is the delegated
authorities of both the coordinators and the approvers. The original *Job Step* from the database consists of 10 different levels, each has its own individual DSA limit associated with it. So based on the information provided by DCC, they are converted to a set of values ranging from the lowest of $25,000, which means that the coordinator or approver can authorize changes up to $25,000 and needs to hand over to a higher level personnel if exceed that value, to the highest of $4,000,000.
Appendix D

Fraud injecting functions

The 12 functions created for injecting fraud cases in the evaluation of graph-based approach are as follows:

- Function 1: Significant unplanned cost growth (UCG) by a coordinator compared to other coordinators.
- Function 2: Significant contract cost growth compared to other contracts.
- Function 3: Significant supplier contract UCG compared to other suppliers.
- Function 4: Unplanned Changes exceed 20% of the contract award amount.
- Function 5: Multiple changes at close to approver DSA limit (30% changes within 15% of DSA).
- Function 6: Coordinator issues change orders within 5% of approver DSA limit.
- Function 7: Date of estimate is later than the date of quote more than 20% of the time.
- Function 8: Negotiated amount is greater than the quoted amount.
- Function 9: For construction(CN) contracts, substantial completion date extensions without cost increase.
• Function 10: For CN contracts, significant use of “planned?” changes to avoid detection by a specific coordinator.

• Function 11: Payments $\geq 50\%$ of contract award amount are made in the first 25% of contract duration - payment for unfinished work.

• Function 12: For CN contracts, first progress claim value exceeds $\frac{1}{contract \ duration \ in \ months}$ of total contract value.
Bibliography


78


[65] Saravanan Sagadevan, Nurul Malim, and Ong Yee. Credit card fraud detection using machine learning as data mining technique. 08 2018.


[73] Daixin Wang, Yuan Qi, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, and Shuang Yang.


