# Graph-based entity resolution and its application in debtor payment prediction

Master Thesis

Tianrui Mao

# Graph-based entity resolution and its application in debtor payment prediction

## Master Thesis

by

## Tianrui Mao

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday September 15, 2022 at 10:00.

| | | |
|---|---|---|
| Student number: | 5236983 | |
| Project duration: | December 1, 2021 – September 1, 2022 | |
| Thesis committee: | Dr. Huijuan Wang, | TU Delft, chair, supervisor |
| | Dr. Asterios Katsifodimos, | TU Delft |
| | Ir. Bastijn Kostense, | Exact, supervisor |

An electronic version of this thesis is available at http://repository.tudelft.nl/.

# Preface

When I write these lines, it also means that my master's career is coming to an end. There are joys, hardships, anxiety and growth in this journey. Firstly, I am very grateful to the instructions, encouragement, and patience given by Prof. Dr. Huijuan Wang and Ir. Bastijn Kostense. Without their help, it would be difficult for me to finish this research by myself. Then, I would like to thank my parents, they have accompanied me through countless days and nights, and have been supporting my studies both spiritually and materially. Finally, my thanks also go to Exact data scientist group. We chat together, exercise together, have meals together, and solve experimental difficulties together. This is an unforgettable memory and friendship.

*Tianrui Mao*
*Delft, September 2022*

# Disclaimer

The information made available by Exact for this research is provided for use of this research only and under strict confidentiality.

# Abstract

In real life, the phenomenon of delayed payment of invoices often occurs. Malicious delayed payment may even lead to the break of capital flow of companies. Debtor payment prediction according to the historical payment behaviour of debtors can help give companies more insight into the risk of late payment. The debtor payment prediction is important for financial risk analysis because companies can better manage their cash flow to avoid the risk of capital chain breakage if they know when the invoice they present will be paid in advance. However, real-world data is far from perfect. With the inflow of various free-form data streams, it is challenging to integrate invoice payment records regarding debtors' historical payment behaviour. As a result, the debtor payment prediction model sometimes performs sub-optimal because sufficient historical payment data cannot be retrieved for debtors. To improve debtor payment predictions, the data sparseness problem is studied to be addressed through entity resolution technology. Instead of fine-tuning a more complex model or adding new financial features, the prediction performance is studied to be improved by mining the information hidden in the data. More specifically, an end-to-end entity resolution workflow is proposed for invoice payment data to match debtors representing the same entity so that more historical payment data is available for the debtor that is known to be matched with other debtors. With the incorporation of entity resolution, the debtor payment prediction is verified to be improved in this project.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Entity resolution (ER) is the process of determining whether two references to real-world objects are referring to the same or different objects. The term entity describes the real-world object, a person, place, or thing, and the term resolution is used because ER is fundamentally a decision process to answer (resolve) the question, are the references to the same or to different entities? Entity resolution has a wide range of application scenarios. For instance, entity resolution technology is often applied to government census, welfare and tax projects. The government usually collects a large amount of population and enterprise information for centralized processing. In order to obtain accurate statistical data, it is important for governments to know whether two records collected by different forms or at different times represent the same entity.

Electronic commerce (e-commerce) is a hot topic in the business field in recent years, which refers to the business mode of trading and message transmission through electronic technology (usually the Internet). One embodiment of e-commerce in inter-company business is the use of electronic money transfer, such as electronic invoices. The use of electronic invoices brings convenience to inter-company transactions, but it also brings certain risks. For instance, deliberately delaying payment of invoices by customers may bring economic losses to sellers. To cope with this risk, e-invoice payment prediction, which utilizes the customer's historical payment data and the invoice's profile data to predict when the customer will pay the invoice, is proposed to help companies know their financial status so they can plan their financial affairs more efficiently. However, real-word data is usually far from perfect. Existing invoice payment prediction model could lack enough data to make an ideal prediction for some enterprises, which is mainly caused by the sparse distribution of enterprises' data. For instance, in Exact, there is no format requirement for profile data of the company, and the attributes are not mandatory to fill in except name, which sometimes makes it difficult for us to obtain the required information for prediction(also known as the cold-start problem for the model). Previous studies mainly focused on improving the performance of the debtor payment prediction model, but do not make enough exploration in improving the input data quality for prediction. In order to solve data sparseness problem for model, this research is proposed to establish a novel entity resolution algorithm for matching Exact accounts data and provide corresponding evaluation methods. On top of this, extensive experiments will be implemented to verify if the performance of debtor payment prediction model can be improved with the incorporation of entity resolution.

## 1.1. Introduction to Exact

This research is performed based on the Exact transaction data. Exact is an accounting company that is specialized in providing accounting management software for Small to Medium Enterprises (SME). As of writing this essay, it is estimated that approximately 20% of Dutch SMEs use Exact software[1]. Exact strives to improve its products by exploring state-of-the-art technologies that are conducive to the operation of the customer company.

The business volume of exact has increased year by year. An overview of the number of transactions over ten years in exact is shown in figure 1.1

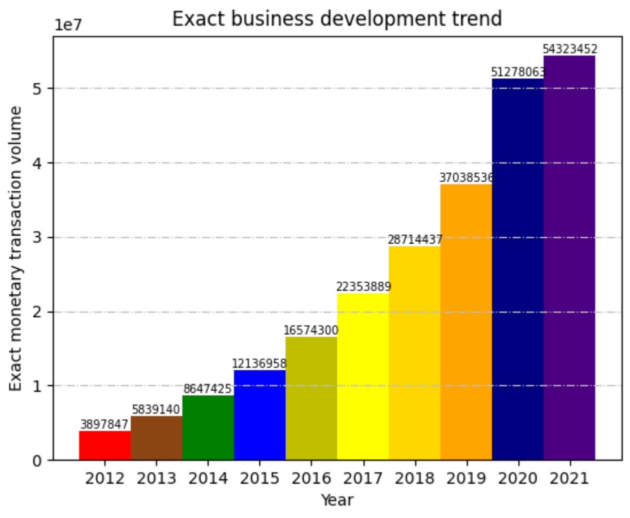


Figure 1.1: Exact business development trend

Large-scale and free-form data makes it increasingly difficult for Exact data scientists to directly extract high-quality data for analysis from the raw data set. Therefore, Exact funded this research project, aiming at improving data quality and obtaining better data analysis results.

## 1.2. Problem Definition and Research Questions

In this research, we aim to construct a novel entity resolution algorithm to match accounts data representing a same enterprise and explore how incorporating the output of entity resolution could improve the performance of debtor payment prediction model. The final goal of this research is to provide an end-to-end workflow regarding how to build a entity resolution algorithm to eliminate data sparseness, how to evaluate the performance of entity resolution algorithm, and how to apply the output of entity resolution algorithm to the debtor payment data to improve debtor payment prediction(also known as late payment prediction). To work towards the aforementioned goals,we define the following research question:

- RQ: Can entity resolution be utilized to improve the debtor payment prediction?

The research question can be split up into the following three sub-questions:

- SQ1: How can we design an entity resolution algorithm suitable for the scenario of large-scale and free-form data?
- SQ2: How can we evaluate the performance of entity resolution algorithm?
- SQ3: How can we utilize the output of the entity resolution algorithm to improve the debtor payment prediction?

In this project, we will design experiments to find the answers to the above sub-questions one by one. The details of the experiments and which sub-question they try to answer can be discovered in chapter 3 and chapter 4.

[1] `https://www.exact.com/`

## 1.3. Methodologies

To design an entity resolution algorithm suitable for real-world data, firstly a statistical analysis is needed to have a overview of how the real-world data is shaped, which attribute could be considered for representing an entity, and which attribute could be taken for the ground truth (label in machine learning). Then, text similarity measurement is studied to convert the attributes in the form of strings into numerical values. Next, using the similarity values from different attributes as the input features, classical and state-of-the-art machine learning models are studied to find the most accurate and stable classifiers for determining whether two entities match or not. Finally, merging strategies based on the output of the classifier for integrating the possible matching accounts data will be explored. We will assess the performance of different merging strategies by calculating the precision and recall of the detected matching record pairs. In this case, the most promising merging strategy for the real-world data can be selected out.

On the premise that the entity resolution workflow is well defined, we will see how we could utilize the proposed entity resolution workflow to improve debtor payment prediction. It is hypothesized that because of sparsity and incompleteness of raw data, the historical payment data that can be searched for some companies could be quite few. In this case, enriching the historical payment data with matching entities is supposed to increase the debtor payment prediction performance as the model can better capture the payment tendency of users with more data.

## 1.4. Contributions

In this work, we try to utilize entity resolution technology to improve the debtor payment prediction. The debtor payment predictions are significant for SMEs since knowing the future state of their liquidity can help them better manage their company trade decisions, such as when to start investing in the next commercial project. Specifically, the contributions of this thesis can be summarized as follows:

- This research can provide an experience of how to deploy entity resolution algorithm on large-scale, unlabeled and free-form industrial data.
- This research can bring practical value to the company's financial risk analysis as it introduces the basic process of how to deploy a debtor payment prediction model so that companies can use the predictions to better manage their finance flow status.
- This research can provide a new idea for how to improve the debtor prediction performance of the model. Instead of fine-tuning a more complex model or adding new financial features, we tap the potential of existing data through entity resolution to improve the debtor payment prediction .

This research covers the exploration of graphics, natural language processing and finance knowledge. It presents an interdisciplinary and diversified manner to solve practical problems in the real-life.

## 1.5. Thesis Structure

The outline of this essay is organised as follows: In chapter 1, the background and the target of this research will be introduced. In chapter 2, a thorough literatures review surrounding the topics of entity resolution, machine learning, and graph representation learning will be carried out. Then, in chapter 3, we will discuss our proposed entity resolution algorithms and apply the algorithms on the data set provided by Exact, and give the corresponding algorithm evaluation method. Next, in chapter 4, we will apply the output of entity resolution algorithm to debtor payment data to see if entity resolution could contribute to improving the prediction performance. In chapter 5, the findings of experiments in chapter 3 and chapter 4 will be summarized to answer the research questions given in section 1.2. Finally, in chapter 6, the limitations and corresponding potential extensions of this research will be discussed in the hope that a better result can be reached in the future.

# 2 Related Work

In order to further understand the research problem and find possible solutions, a preliminary literature review is conducted in chapter 2. At the same time, it will also be analyzed that whether the methods proposed in the literature are applicable for our case. This chapter is structured as follows: In section 2.1, we will focus on how to calculate word similarity metrics and whether the metrics are applicable to Exact data. In section 2.2, machine learning classifiers and corresponding model evaluation methods, which is related to the matching/nonmatching classification when implementing data pairwise comparison, will be discussed. Then, in section 2.3, we will discuss different ideas of realizing entity resolution. These ideas are not all practiced in this research, but they provided reference and ideas for possible extension of this research in the future. Next, in section 2.4, graph embedding and its related graph metrics will be studied because we try to use them to improve payment predictions. At last, in section 2.5, we will take a look at the background and implementation schemes of debtor payment prediction in Exact.

## 2.1. Text similarity

In order to find the references representing the same entity from the data, it is necessary to compare different attributes' values of the references. However, in our research, the value of the attribute is not a numerical value but a text. So we need to find a way to quantify the similarity of text. [39] describes that text similarity measurement can be divided into two aspects: text distance and text representation. There are three ways of measuring similarity according to the distance of length, distribution, and semantics of the text:

- **Length distance:** Length distance utilizes numerical characteristics of text to calculate the similarity of vector text. The most commonly used distance metric is Euclidean distance: d(La, Lb) = $\sqrt{(La - Lb)^2}$, where La and Lb represents the length of text a and text b.

- **Distribution distance:** Distribution distance judges similarity based on whether texts originates from same or similar distribution. One popular method for calculating distribution distance is KL divergence: d(p||q) = $\sum p(x) \log \frac{p(x)}{q(x)}$, where p (x) and q (x) be two probability distributions with values of x.

- **Semantic distance:** Semantic distance is usually used when there is no common word in the text. In this case, the similarity obtained by length or distribution is unable to express real similarity, so we can consider calculating the distance at the semantic level. Word's distance is the main method used in representing the semantic distance. It measures the similarity between two texts through minimum distance required for one text to reach another text in the semantic meaning.

For text representation, the author of [39] also mentions there are three kinds of methods to compute the text similarity:

- **String-based:** String-based similarity is based on the similarity between characters in the text. The most commonly used distance parameter is Edit Distance (Levenshtein Distance). It counts the minimum number of edits needed to transform the text from a to b.

- **Corpus-based:** The corpus-based method uses the information obtained from the corpus to calculate the text similarity. This information can be either a textual feature or a co-occurrence probability. One representative corpus-based similarity metric is TF-IDF. It uses the frequency of words in corpus to calculate the similarity: tf-idf (w, d, D) = tf (w, d) × idf (w, D), where tf (w, d) indicates how often a word occurs in a document and idf (w, D) represents the opposite trend of how a word appear in all documents.

- **Semantic text matching:** Semantic similarity determines the similarity between text based on their meaning instead of shape. In most cases, the texts are encoded thus generating a new expression through a machine learning or embedding model to predict the semantic similarity between two texts.

Furthermore, the authors of [10][22][1] point out that knowledge-graph representation learning could perform better than text representation methods because it takes into account both local and global features of the text. Knowledge-graph representation learning projects the entities and relationships in the knowledge graph into a vector space. The generated entity and relation vector are essentially a more semantic representation, which can more efficiently express semantic meaning.

## 2.2. Machine learning and evaluation approaches

### 2.2.1. Supervised machine learning

On the basis of data features are available, the problem now becomes how to implement the binary classification of matching and mismatching according to the features values. Generally, classification tasks can be accomplished by machine learning (ML) algorithms. In this section, we will explore various approaches of using ML to conduct classification tasks and how to access the performance of ML algorithms. [37] presents us a good review of various of classical machine learning algorithms designed for classification tasks. In this essay, machine learning algorithms are divided into supervised and unsupervised categories. The main difference between supervised algorithms and unsupervised algorithms is if the ground truth of the data is available. Since we obtain the label for the Exact data through ground truth statistics, we will focus on studying the supervised machine learning algorithms.

[36] tells us the commonly used classical supervised machine learning algorithms for classification tasks in industry are Bayesian Network (BN), Decision Trees (DT), k-Nearest Neighbors (k-NN), Logistic Regression (LR), Random Forests (RF), and Support Vector Machine (SVM). Moreover, latest research shows that artificial neural network[1] and XGboost[3] has become relatively competitive to conventional machine learning algorithms. These supervised machine learning algorithms categorize data from the prior information. To be specific, Firstly, a classification algorithm is applied to the training data set, in this process the model tries to fit the output it gives to the given prior as much as possible. Then the trained model is applied to the test data set and the output of trained model is compared with the label of test data to verify whether the performance of the model is good enough to complete the classification task. A flow chart of supervised machine learning algorithm is shown in figure .2.1.

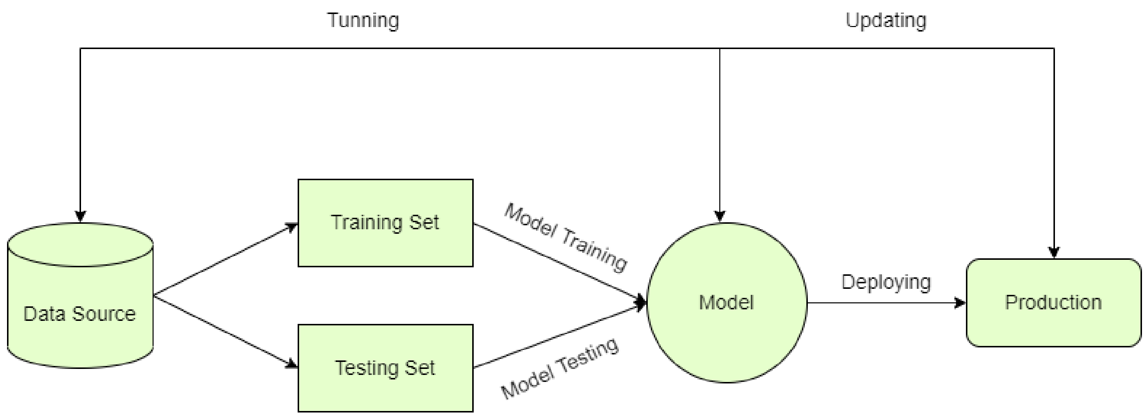


Figure 2.1: Flow chart of supervised machine learning algorithm

In order to achieve a good classification performance, it is important to select good data features. Feature selection leverages data to explore the distinctive attributes so that classifiers are easier to produce good performance based on these distinctive attributes. [7] summarizes the general and representative measures for feature selection. Some of them are potentially useful to our research project:

- **Correlation coefficient:** r(a,b) = $\frac{Cov(a,b)}{\sqrt{Var(a)}\sqrt{Var(b)}}$, where Cov(a, b) is the covariance of a and b, and Var(.) is the variance.
- **Pearson correlation coefficient:** r(a,b) = $\frac{N\sum a_i,b_i}{\sqrt{N\sum a_i^2-(\sum a_i)^2}\sqrt{N\sum b_i^2-(\sum b_i)^2}}$, where N is the number of instances of a and b.
- **Mutual information:** $I(a;b) = \sum_a \sum_b p(ab) \log \frac{p(ab)}{p(a)p(b)}$, where p(.) is the probability density function.
- **Symmetric uncertainty (SU):** SU(a,b) = $\frac{2I(a;b)}{H(a)+H(b)}$, where H(.) is the entropy of a feature.
- **Information distance:** d(a,b) = $\frac{H(a|b)+H(b|a)}{2}$, where $H(a \mid b)$ is the conditional entropy of a given b.
- **Euclidean distance:** d(a,b) = $\sqrt{\sum (a_i - b_i)^2}$.

### 2.2.2. Machine learning evaluation

The performance of the match classifier is related to whether we can correctly identify the same entities, so it is important to guarantee the good performance of the classifier. With regard to classification problems, the evaluation of the classifiers can be defined based on confusion matrix. Confusion matrix for binary classification is shown in Table 2.1.

Table 2.1: Confusion matrix for binary classification and the corresponding array representation

| | **Actual Positive Class** | **Actual Negative Class** |
|---|---|---|
| **Predicted Positive Class** | True positive (tp) | False negative (fn) |
| **Predicted Negative Class** | False positive (fp) | True negative (tn) |

As is shown in previous study [20], the accuracy is the most commonly used evaluation metrics in classification problems. By accuracy the quality of produced output given by classifiers is evaluated based on percentage of correct predictions over all instances. However, as many studies have pointed out, the accuracy metric is unfair in evaluating discrimination processes. Taking cancer detection as an example, because the proportion of people suffering from cancer in the crowd is very small, the classifier can achieve a correct rate higher than 99% through judging all the people as non-cancer people. But this is obviously unfair, because we expect a high-performance classifier should be able to identify cancer patients in the crowd.

In order to evaluate the model fairly in different scenarios, [20] counts the following commonly used metrics for evaluating machine learning algorithms from different aspects in Table 2.2.

## 2.3. Entity resolution

Entity resolution is a technology used to detect data records that refer to the same real-world entity in a single data source or multiple data sources, and integrate these records together. For our research, entity resolution is valuable as it helps improving data quality and increasing the reliability of data thus lays a good foundation for the subsequent data analysis. In terms of entity resolution based on structured data, various approaches have been come up with to realize and optimize this technology. For instance, entity resolution workflow shown in figureure 2.2 is applied to deal with real-world objects identification systematically[24]. On basis of this workflow, MapReduce paradigm is applied to improve the computational efficiency of entity resolution through parallel computation[2]; different kinds of natural language processing(NLP) metrics are explored to digitize entity attributes [21]; [16] describes that deep neural network is promising for entity classification.

### 2.3.1. Matching strategy

Traditionally, entities are resolved by using pairwise similarity on reference attributes. Pairwise matching is based on the similarity between the values of corresponding identity attributes to determine the equivalence between two references. There are five basic methods for determining the similarity of attribute values.

Table 2.2: Metrics for Classification Evaluations

| Metrics | Formula | Evaluation Focus |
|---|---|---|
| Accuracy (acc) | $\frac{tp+tn}{tp+fp+tn+fn}$ | The accuracy metric measures the ratio of correct predictions over the total number of instances. |
| Error Rate (err) | $\frac{fp+fn}{tp+fp+tn+fn}$ | Misclassification error measures the ratio of incorrect predictions over the total number of instances evaluated. |
| Sensitivity (sn) | $\frac{tp}{tp+fn}$ | This metric is used to measure the fraction of positive patterns that are correctly classified. |
| Specificity (sp) | $\frac{tn}{tn+fp}$ | This metric is used to measure the fraction of negative patterns that are correctly classified. |
| Precision (p) | $\frac{tp}{tp+fp}$ | Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class. |
| Recall (r) | $\frac{tp}{tp+tn}$ | Recall is used to measure the fraction of positive patterns that are correctly classified. |
| F-Measure (FM) | $\frac{2*p*r}{p+r}$ | This metric represents the harmonic mean between recall and precision values. |
| Geometric-mean (GM) | $\sqrt[2]{tp*tn}$ | This metric is used to maximize the tp rate and tn rate, and simultaneously keeping both rates relatively balanced. |

- **Exact match:** Attributes' values of two references must be the same.
- **Numerical difference:** Attributes' values are allowed to only be similar rather than same.
- **Approximate syntactic match:** Attributes' values can be considered as similar in format.
- **Approximate semantic match:** Attributes' values can be considered as similar in meaning.
- **Derived match codes:** The attributes' values is mapped to a new latent space, and whether two references are the same entity can be judged by the attributes' values of reference in the new latent space.

In fact, besides attributes' values, other relational information might be available in the data. The author of [5] puts forward that it is possible to use both attribute and relational information for determining the underlying matched entities. It is suggested that entity resolution algorithm based on graph theory has the potential to further improve the identification performance through transitive equivalence and association analysis using relational information. The meaning of transitive equivalence for entity resolution is shown in figure 2.3. Through attributes' similarity analysis, we know that Ref1 and Ref2 represent the same entity and Ref1 and Ref3 represent the same entity, so can we infer that Ref2 and Ref3 are also the same entity? If it is, relational information can be used as an additional feature to discover more matching data. Actually, This requires a case by case analysis, which is also an interesting problem that our research will explore in the chapter 3.

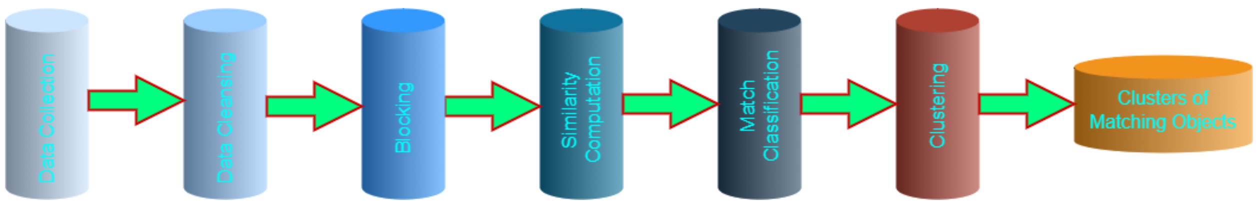


Figure 2.2: Workflow of entity resolution

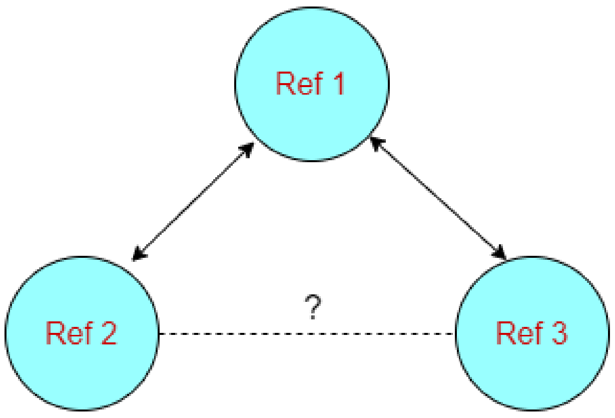


Figure 2.3: Transitive equivalence

### 2.3.2. Evaluation of entity resolution

An ER algorithm takes as input a set of records I and groups together records that represent the same real world entity. To evaluate an ER algorithm, we should compare its result R to the ground truth S, which is also a partition of the input records I. [31] gives a brief introduction to the most commonly used evaluation measures in practice.

- **pF1:** Combines the pairwise precision (fraction of record pairs from R that are also in S) and pairwise recall (fraction of S pairs found in R).
- **cF1:** Combines the fraction of clusters from R that are also in S and the fraction of clusters from S in R.
- **K:** Combines the average similarity of clusters in R with respect to those in S and the average similarity of clusters in S with respect to those in R.
- **ccF1:** Similar to cF1, but the clusters are now compared by their similarities instead of whether or not they are identical.
- **VI:** Measures the "information" lost and gained while converting R to S.

## 2.4. Link Prediction

In network theory, link prediction is the problem of predicting the existence of a relation between two entities in a network. Since we will study how to find the matching relation between entities in the entity network, link prediction techniques are related to our research. A comprehensive overview of commonly used link prediction methods is provided in [28]. The application of link prediction can be divided into two categories: static network link prediction and dynamic network prediction. The purpose of first category is to discover links in a static graph while the second category aims at predicting possible links in the future. For this research, we focus at the static network link prediction as we are going to explore the similarity relationship between entities, which has little correlation with temporal characteristics. Moreover, [28] classifies the link prediction methods into three categories, which are similarity-based link prediction methods, probabilistic and maximum likelihood-based link prediction methods, and dimensionality reduction-based link prediction methods. As [28] also mentions that probabilistic and maximum likelihood-based link prediction methods usually need quite a lot computational resource and additional information other than merely the structural information of the network, which is not satisfactory to our requirements, we will mainly focus on the similarity-based methods and dimensionality reduction-based methods in the following subsections.

### 2.4.1. Similarity-based approaches

The first category of methods that [28] introduces is similarity-based approaches. One of the advantages of similarity-based approaches is low computational complexity and strong interpretability. For our research, we are interested in finding matching relation between two entities. Similarity-based link prediction indicators could provide us with the pairwise relation of the two entities so that we can utilize them to decide whether there exists a matching relation between two entities. Some of the similarity-based indices that are potentially useful to our use case are:

- **Common neighbors (CN):** This index indicates the amount of common neighbors of two nodes and is calculated as: CN(x, y) = $|\Gamma(x) \cap \Gamma(y)|$, where $\Gamma(x)$ represents the set of neighbors of x.
- **Jaccard's index (JC):** This index is a normalized measure of the overlap of neighbors between two nodes. Its calculation formula is JC(x, y) = $\frac{\Gamma(x)\cap\Gamma(y)}{\Gamma(x)\cup\Gamma(y)}$.
- **Adamic-Adar index (AA):** This index is similar to the Jaccard's index but assigns more weight to the less-connected neighbors. It is calculated as AA(x,y) = $\sum_{s\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log(ks)}$, where ks is the degree of node s.
- **Resource allocation index (RA):** This index is quite similar to the Adamic-Adar index. What is different is that resource allocation index punishes the nodes with high degree more heavily. Its calculation formula is RA(x,y) = $\sum_{s\in\Gamma(x)\cap\Gamma(y)} \frac{1}{ks}$.
- **Cannistraci resource allocation index (CRA):** This index is similar to the resource allocation index. But it take into account the common neighbors of two nodes. Cannistraci resource allocation index regards if two nodes have a large local common community, then there is a high possibility that two nodes to be linked and also the value is normalized by the degree of node to protect the less-connected neighbors. Its calculation formula is CRA(x,y) = $\sum_{s\in\Gamma(x)\cap\Gamma(y)} \frac{\gamma(s)}{ks}$, where $\gamma(s)$ represents the subset of all $\Gamma(s)$ that are also in CN(x, y).

### 2.4.2. Dimensionality reduction-based link prediction approaches

For dimensionality reduction-based link prediction approaches, there are two schemes to implement it. The first one is matrix factorization-based approach and the second one is embedding-based approach.

Matrix factorization-based method proposed by [32] extracts latent features and optional explicit features for each node to obtain vectors that contain representations of each node. After that, the matrix constructed by the representation vector of each node can be used for classification through matrix factorization technology. The author of [32] also mentions that the performance of matrix factorization-based link prediction methods empirically performs better compared with similarity-based approaches. However, the interpretability of matrix factorization is relatively weak. At the same time, matrix factorization-based method is expensive for large scale data set as the computational complexity of matrix factorization is fairly high[8].

Embedding-based approach maps the nodes of graph to a latent vector space which keeps the original features of the graph. The main goal of graph embedding method is to pack the attributes of each node into a vector with a smaller dimension. Therefore, the similarity of nodes in the original complex irregular space can be easily quantified through standard metrics in the embedded vector space[40]. The most classic embedding method is called DeepWalk based embedding proposed by the author of [33]. DeepWalk simulates the random walks by choosing a starting node and randomly follows an edge to another one node until the pre-defined walk length is reached. In this case, the probability of two nodes in graph can be defined as the probability of visiting node $V_j$ on a random walk of a specified length T starting from $V_i$. Then after mapping, the similarity feature is kept in embedding space through a softmax function parametrized by the dot product of the node embedding vectors and a matrix containing vector representations for all nodes in G will be generated by DeepWalk. Another one commenly used node embedding method is called Node2Vec[19]. This embedding method employs two extreme ways of exploring a network: Deep-first search and breadth-first search. Node2vec deploys two parameters p and q which allow users to explore both global community structure and local roles through adjusting the value of p and q.

### 2.4.3. Learning-based link prediction approaches

Besides the methods mentioned above, learning-based link prediction method is also a hot research direction. It utilizes the node profile information or link structure information as input to carry out classification task or prediction task. For instance, [27] proposed a clustering-based method using node features to realize social community classification. In addition, [9] put up with a reinforcement learning model using both path information and entity information to realize link prediction. The author mentions that the RL-based method is more competitive as it guides agent to search for relation paths with high contributions to link prediction. Meanwhile, the author of [43] proposed GNN-based link prediction model which also considers the causality between node features. Complex models use their powerful parameter fitting ability to capture more information in complex networks. However, a more complex model also means greater computational complexity. For our research, our application is based on large-scale industrial data, and we hope to propose a lightweight algorithm with relatively low computational cost. Therefore, this study won't consider using complex learning models for link prediction.

## 2.5. Debtor payment prediction

Debtor payment prediction means predicting if the customers will pay the debtor on times to the suppliers. Using the information from invoices or attributes of debtors, we could predict whether an invoice transaction could bring risks for the company's working capital. Debtor payment prediction is closely related to the credit risks analysis of companies. It is important to evaluate the possible payment time as deliberate late payment of debtor could lead to capital flow fracture for companies.

Debtor payment prediction is a widely studied problem and has been applied in industry. According to the authors of[14] and , logit model is most commonly used in the field of debtor prediction as it performs well and easy to deploy. Meanwhile, [35] develops different kinds of regression models the predict the debtor risks. In addition, [17] and [34] state that artificial neural networks shows a better performance compared with conventional models due to its strong parameter fitting ability. Besides, it is shown that bayesian network shows a good prediction effect and robustness regarding debtor collection prediction[25].

How to choose the data characteristics for predicting debtor payment is also a hot topic. The author of [21] classifies the available data features from Exact into five levels:

- **User level:** Past payment behavior records of users to debtor.
- **Debtor level:** Attributes of debtor's record, such as debtor amount and debtor's repayment deadline.
- **Node level:** Using the data of payment transactions between companies, we could build a monetary transaction network, which can provide us with some useful information for prediction, such as the 1 hop neighbors of the company.
- **Temporal level:** Some statistical values in time dimension can also be used as the input features of debtor payment prediction, such as mean time between late payments.
- **Window level:** Considering the possible structural and personnel changes in the company, using the latest company-related data may bring better forecasting results. How to determine the length of time window of data to achieve better prediction effect is also a problem worth exploring.

Moreover, the author of [26] points out dynamic graph features and node embeddings could help improve the KPI prediction in Exact. Although there are some differences between KPI prediction and debtor payment prediction, the research experience regarding using graph features to improve KPI prediction could also be used as reference for debtor payment prediction.

# 3 Entity Resolution

In this chapter, we will explore the first and second subquestions of how can we design an entity resolution algorithm for large-scale and free-form data and how can we evaluate it. As discussed in the Chapter 2.3, the goal of entity resolution is to integrate records representing the same entity to eliminate data sparseness to make it easier for scientists to dig up patterns behind data. The way to implement entity resolution is to first explore the profile information of enterprises' account data to select the suitable attributes for account data comparison. On top of this, we will try to use graph knowledge to capture the relational information hidden in the entity graph to see if the relational information can help increase the performance of entity resolution. In the next section, we will take a closer look into how the output of entity resolution is utilized to ameliorate the debtor payment prediction.

## 3.1. Introduction to Exact data

As discussed above, we will use the profile information of enterprises' account data to find the matching account clusters, where each cluster represents a collection of accounts describing the same enterprise. Therefore, before implementing entity resolution algorithm, we are ought to take a look at how the Exact data is organized.

An overview of Exact data structure can be is shown in figure 3.1. In this figure, the yellow circles represent the initiators of transactions. These initiators are the registered customers in Exact and they are defined as type D accounts. In principle, each company has one and only one type D account. Meanwhile, the blue rectangles indicate recipients of transactions. These recipients are entities that Exact registered customers create for the transaction and they are defined as type A accounts. One company can have multiple type A accounts because one company can have invoice transactions with multiple other companies. This research will be based on the invoice transaction data from Exact in 2021. The proportion of type D accounts and type A accounts in our data set is shown in Table 3.1. As our accounts are divided into two categories: type D and type A, our account matching is also divided into two categories, namely DA matching and AA matching. A DA matching, which is indicated as the dashed arrow in figure 3.1, means one type D account record and one type A account record are regarded as representing one same entity. A AA matching, which is indicated as the solid arrow in figure 3.1, means one type A account record and another one type A account record are regarded as representing one same entity. For this research, we only take into account AA matching because in the case of debtor payment prediction, it makes the most sense to predict which debtors (type A accounts) are the same. If it is possible to match the debtors to each other, we can merge the historical payment behaviour data of those debtors, which could result in the performance gain for the prediction model as more data is available for the feature calculation.

Table 3.1: Distribution of typeA and typeD accounts in Exact data in 2021

| | **TypeA** | **TypeD** | **Total** |
|---|---|---|---|
| **Proportion** | 0.985 | 0.015 | 1 |

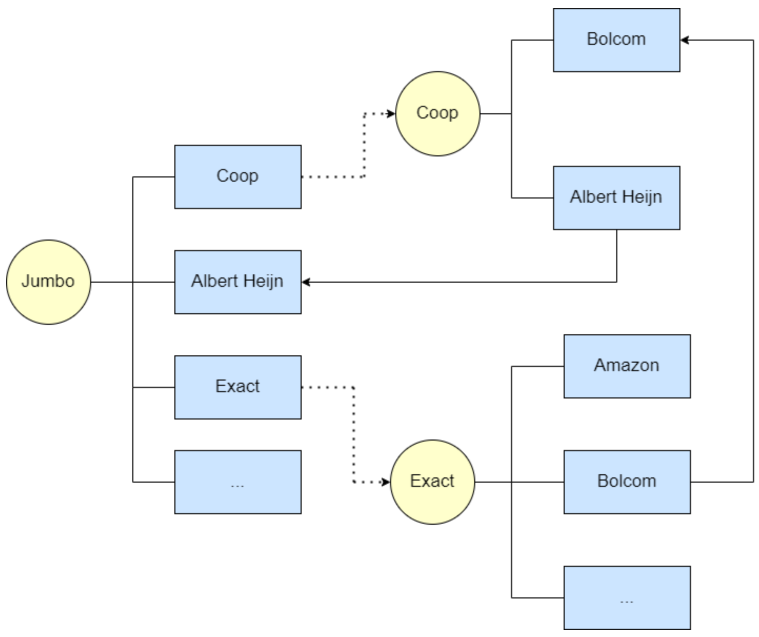


Figure 3.1: Exact data structure

To know which attributes of type A account could utilized by us to find the AA matching in the data set, we take a closer look at the profile information of type A accounts. The available attributes of type A accounts for entity resolution is shown as follows:

- **Name:** This is the name of the company. Name column is mandatory for users to fill in so this column will never be null.
- **Postcode:** This is the zip code of the company.
- **Email:** This is the Email of the company and it can be either a enterprise email or a personal email.
- **Phone number:** This is the phone number of the company and the phone number can be Dutch, non-Dutch or landline.
- **Website:** This is the website of the company.
- **AddressLine1:** This is the main address of the company. Usually it consists of a place name and a room number.
- **AddressLine2:** This column indicates some supplements to the company's main address information.
- **AddressLine3:** Similar to AddressLine2. This column also indicates some supplements to the company's main address information.
- **VATNumber:** This is a number given by governments to indicate what type of the tax should the company pay. With this number, the government can track which tax payment belong to which company.
- **ChamberOfCommerce:** This is a number provided by the Dutch administrative body for businesses and this number is related to the ownership and organizational form of the company.

Multi-attribute-based entity recognition algorithm is proved to have better recognition effect empirically by the author of [4]. However, the condition for our case is somehow more complicated as the values of most of attributes are not always available. An overview of attributes supplied with information for type A accounts is shown in figure 3.2. Considering website, addressLine2, and addressLine3 are not available for most of the type A accounts, in the following experiment we will try to utilize one or multiple attributes from name, postcode, email, phone number, addressLine1, VATNumber, and ChamberOfCommerce to design the entity resolution algorithm.

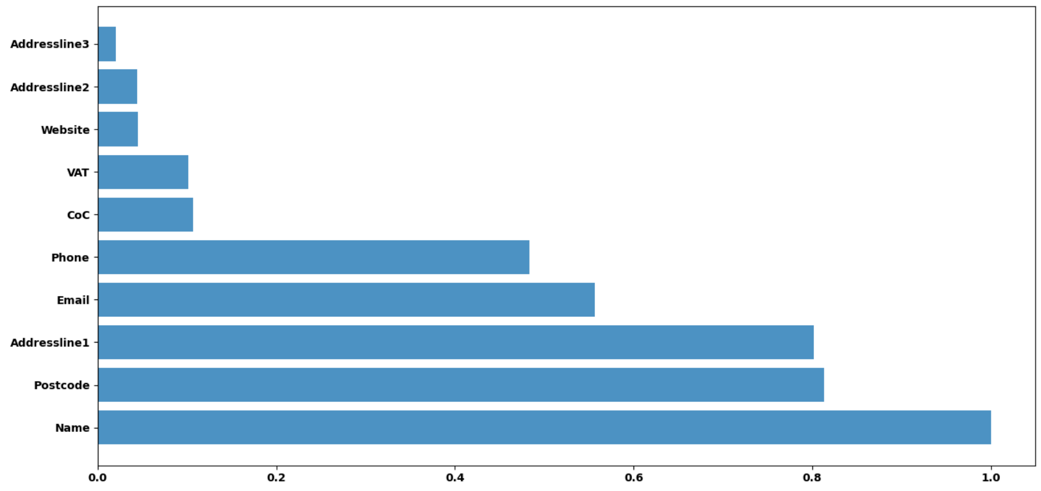


Figure 3.2: Overview of columns supplied with information for typeA accounts

## 3.2. Entity resolution algorithm

As is shown in Fig2.2, the Stanford Info lab put up forward the general workflow of entity resolution. Inspired by this idea, we also design our entity resolution algorithm following this structure and make innovations in merging strategy. More specifically, the workflow of our proposed entity resolution algorithm is shown in Fig3.3 and we will introduce how do we realize entity resolution algorithm step by step in this section.

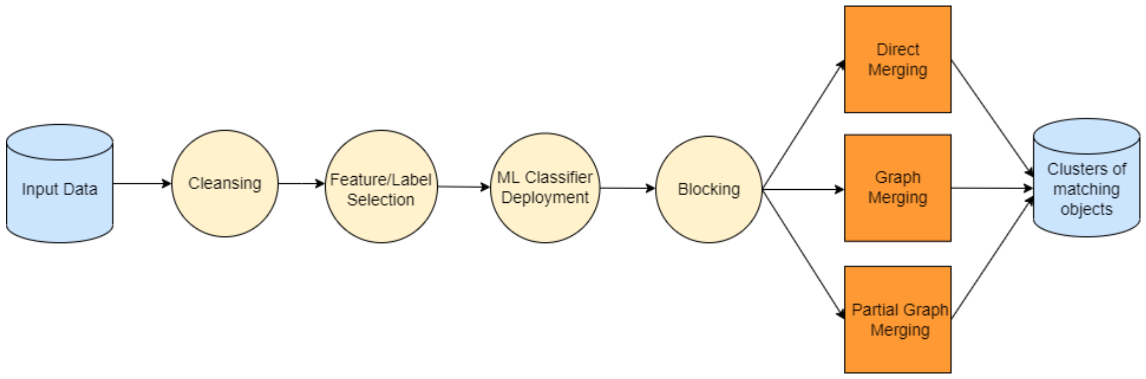


Figure 3.3: proposed ER workflow

### 3.2.1. Data preprocessing

Exact has no requirements for the form of each column of accounts, which will hinder our analysis of the data. In order to ensure that the entity resolution algorithm can achieve good results in the end, it is wise to preprocess the data as detailed as possible to remove outliers before deploying the algorithm. In view of the data preprocessing experience of Exact data scientists[26], our data cleaning strategies are given as follows:

- **Name:** Firstly, name strings are converted to lowercase letters. Then, Dutch business structure indicators like "bv", and "eenmanszaak" are removed.

- **Postcode:** A postcode starts with the four digits and ends with two uppercase characters, like "2534ZL" and "2628AE". For this condition, we first convert the postcode to uppercase and remove white space.

Then we detect if it follows the structure starting with the four digits and ending with two uppercase characters. If it follows, we include it. Otherwise, we insert the None value.

- **Email:** We first convert the email address to lowercase. We then check if it starts with at least 1 character, followed by the "@" sign, followed by at least 1 character again, followed by the "." sign, followed by 2 or 3 characters. If this is the case, we include it. Otherwise, we insert the None value.
- **Phone number:** As initial digits could be in different forms, like "0031" and "+31". In this case, we could keep the last nine digits of the phone number. If the length of the phone number is shorter than nine, we insert the None value.
- **AddressLine1:** We convert the address strings to lowercase.
- **VATNumber(VAT):** A VAT number starts with the two uppercase characters, like "NL" and "BE", followed by 9 digits, followed by the character "B", followed by 2 digits. To cleanse VATNumber, firstly we convert all characters to uppercase. Then, if the data matches the description above, we include it. Otherwise, we insert the None value.
- **ChamberOfCommerce(CoC):** Dutch Chamber of Commerce numbers consist of 8-12 digits. In order to cleanse ChamberOfCommerce, we first remove all non-digit characters and white space. Then we detect if the length of data equals length 8 or 12. If so we include it. Otherwise, we insert the None value.

### 3.2.2. Feature & Label choice

To classify if two data records representing the same entity and evaluate the classification performance, we need to find suitable attributes as input features and ground truth respectively. For input feature selection, name should be given the highest priority as the values of name are available for all data records. To keep the conciseness and low-consumption of the entity resolution algorithm, we firstly try only using name as input feature for machine learning classifier and then see if we can improve the performance of classifier further with the incorporation of other attributes that are not always available. Another one problem comes with feature selection is how to digitize the value of each attribute as the original values of typeA accounts' attributes are provided in string form thus cannot be directly utilized as the input of classifier. For name, address, email, phone number, and postcode, we use levenshtein similarity[41] to digitize the similarity between two strings as the computation cost of levenshtein similarity is low thus is suitable for large-scale data. Levenshtein similarity is calculated as $score(word1, word2) = \frac{cost}{maxsize(word1, word2)}$, where cost represents the minimum number of characters required to change from word1 to word2. For CoC and VAT, we don't take into account using them as input features as they are not often available.

However, based on the scientists' experience from Exact and our observation, if the CoC and VAT of two records are the same, then it is most likely that this two records representing the same entity. There do exist some corner cases like one company may use bogus CoC or VAT and one company can have multiple CoC or VAT, which makes it complicated to justify whether two records really represent the same entity under some situations. In most cases, CoC and VAT are qualified enough to find a large chunk of matching pairs. So we take CoC & VAT as our ground truth label and use the data whose values of CoC and VAT are available for the following experiments. More specifically, if the CoC and VAT of two data record are the same, we will regard this data record pair as a matching pair and mark it as 1. Otherwise we will regard this data record pair as a nonmatching pair and mark it as 0. Although CoC and VAT must remain available for model training and testing, for model application, we do not need to keep these two attributes, but only need to ensure that input features of model are available.

### 3.2.3. Classifier deployment

To identify if two data records with different attribute values represents the same enterprise, we need to rely on some mathematical methods to find the threshold between matching data record pair and nonmatching data record pair. In our research, we choose to realize this binary classification task through machine learning algorithm because the threshold can be obtained from learning thus requires no additional domain knowledge. To reach a good classification performance, we need to know which model is most promising and how much data is required for model deployment. In addition, we should also ensure that the classifier performs enough well in detecting matching data record pair as only the matching pair could bring additional useful information for the debtor payment prediction. To reach this goal, we randomly sample equal number of matching records and non-matching records to make a balanced training set. The distribution of the levenshtein similarity of sampled data is shown in figure 3.4. We can see that the value of levenshtein similarity for matching record pairs is relatively higher while the value of levenshtein similarity for non-matching record pairs is relatively lower. At the same time, when looking at the performance evaluation metric of classifier, we select precision, recall, and F1-score alongside the accuracy as we cares about the performance of model in identifying matching pair. The detailed steps regarding classifier selection and training is shown as follows:

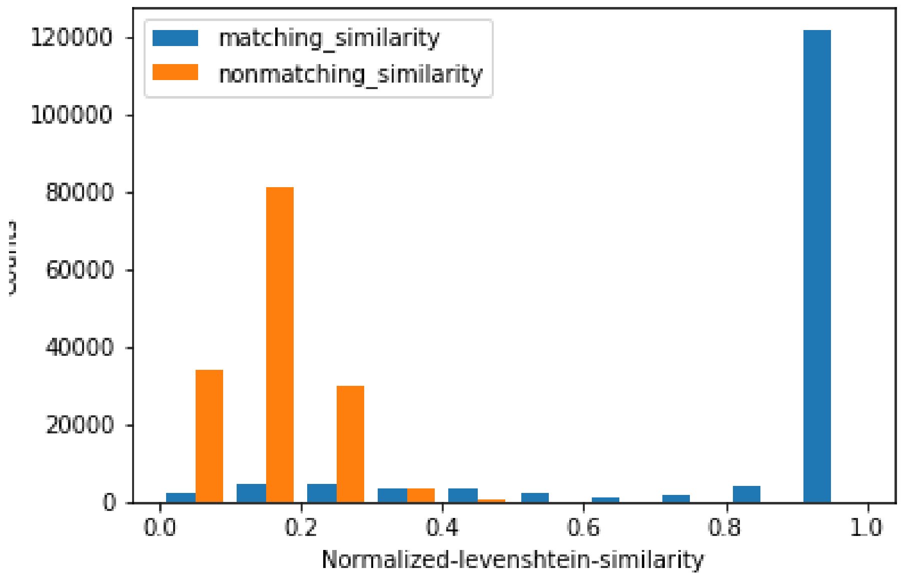
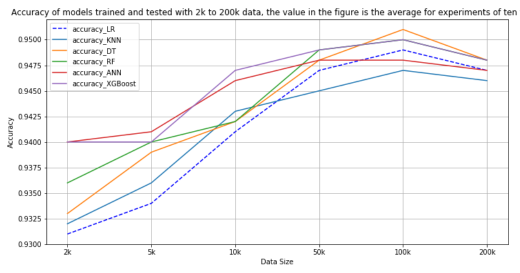
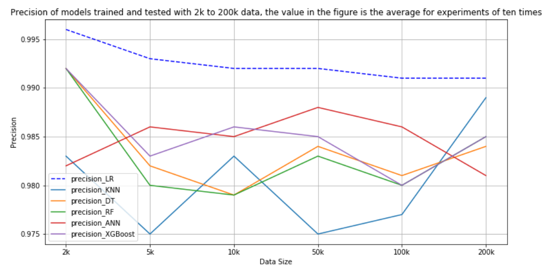


Figure 3.4: The result of class balance, the size of each class is 150k.

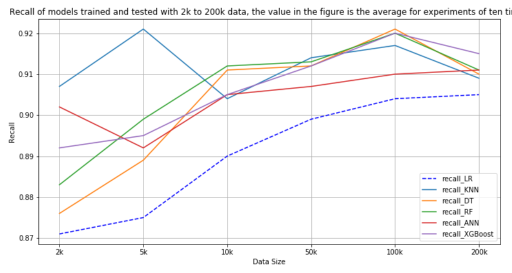
- **Model selection:** As suggested in chapter 2.2, we choose commonly used classical machine learning classifiers, which include logistic regression, knn, decision tree, and random forest. Besides, we also take into account some latest classifiers, which include applied neural network and Xgboost, to see if complex model can bring more promising classification performance for our case.
- **Dataset consistency:** For one epoch, we keep the same data set for the training and testing of different models.
- **Data split:** We set the proportion between training set and testing set as 7:3. This is a commonly used training & testing set splitting proportion in the field of machine learning.
- **Data shuffle:** The training set is scrambled to ensure that the performance score of model is not dependent on the ordering of the data.
- **Parameter optimization:** The performance of most machine learning models depends on how well the hyper-parameters are optimized. To make sure that the hyper-parameters are well optimized, we use the greedy grid search method to find the optimal value of hyper-parameters. Meanwhile, in order to reduce the probability of overfitting, we introduce K-Fold cross-validation for hyper-parameter tuning. K-fold cross validation means splitting data set into k folds. One single fold is used as a validation set and the remaining folds are used to train the model. At the end of all runs, the performance scores for k-time runs are gathered and the hyper-parameters with the highest average of performance scores are selected for the model. In all experiments, k is set to 5.
- **Multi-time experiment:** To ensure the reliability of our experiment results, we do model training and model testing for ten times. Then the average value of performance metrics for ten times is selected as our final result.



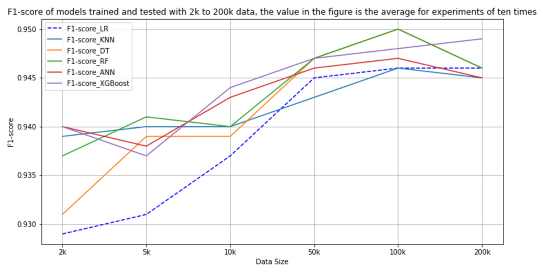
(a) Accuracy of models trained and tested with 2k to 200k data, the value in the figure is the average for experiments of ten times



(b) Precision of models trained and tested with 2k to 200k data, the value in the figure is the average for experiments of ten times



(c) Recall of models trained and tested with 2k to 200k data, the value in the figure is the average for experiments of ten times



(d) F1-score of models trained and tested with 2k to 200k data, the value in the figure is the average for experiments of ten times

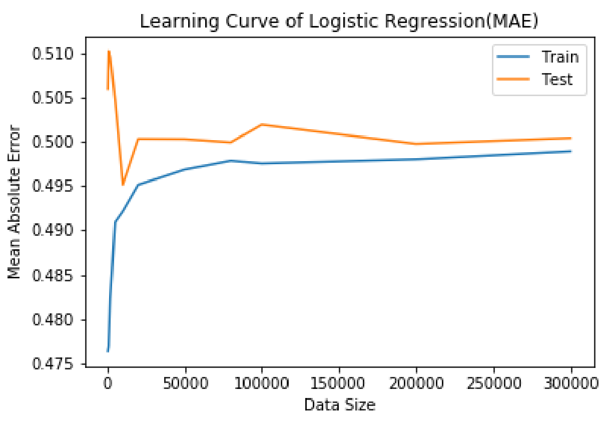
Figure 3.5: Accuracy, precision, recall, and f1-score of models trained and tested with 2k to 200k data, the value in the figure is the average for experiments of ten times

After training models on the balanced data set, we obtain the values of accuracy, precision, recall, and F1-score of different models, which are shown in figure 3.5. It can be seen that the performances of the different models are fairly high. This is reasonable as we can see a obvious difference between the similarity of matching class and nonmatching class in figure 3.4. The result also shows that with the increase of data set size, the performance of different models gradually increases and the performance of simple model such as logistic regression is similar to the complex model such as Xgboost. Considering the training cost and transparency of models, logistic regression is finally taken as the classifier for matching/nonmatching classification. In addition, to explore how much data is needed to deploy the logistic regression model, we draw the learning curves with MAE and RMSE as objective indicators shown in figure 3.6. The reason why we choose these two indicators is that they can evaluate the bias and variance between predicted value and the actual value, so that we can fully understand how the generalization ability of our model is. It could be seen from the figure 3.6 that, the difference between MAE and RMSE of training curve and testing curve get smaller with the increase of data size, which means the logistic model generalizes and fits well with the data. Moreover, the difference between training curve and testing curve tends to be stable after 100k data is utilized. To ensure that the logistic regression model fit enough well with the data, we finally take 200k data for model deployment.

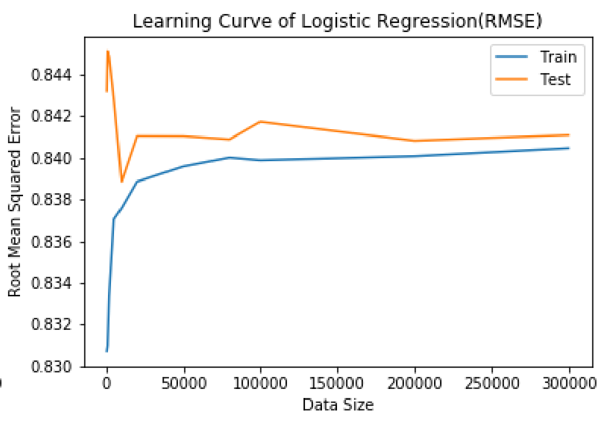
### 3.2.4. Feature space exploration

Besides merely using name attribute to decide if two records represent the same entity, we are also interested in if other attributes could help increase the classification performance further. On account of the values of address, postcode, email and phone number(we call these four features as incomplete features for short) are not always available, we select the data when one of the incomplete feature together with target label that are available to verify if incomplete feature could perform better than only using name similarity. For the steps regarding the experiment, we use merely name similarity, merely incomplete feature similarity, and the combination of name similarity and incomplete feature similarity to train and test the model to see if the performance changes or not. Similar to chapter 3.2.3, we still use logistic regression and two hundred thousand data to implement the experiment. Meanwhile, the metric to calculate the string similarity for address, postcode, email and phone number is still levenshtein similarity. The results are shown in Table.3.2 ,3.3,3.4, and 3.5, which represent the average value for ten-time experiments.

Based on the score shown in the above tables, it can be seen that using name similarity as input feature



(a) The learning curve of logistic regression. We take MAE as objective metric to evaluate the bias between predicted value and actual value.



(b) The learning curve of logistic regression. We take RMSE as objective metric to evaluate the variance between predicted value and actual value.

Figure 3.6: The learning curve of logistic regression with MAE and RMSE being taken as the objective metrics

Table 3.2: The performance of model when using name similarity, address similarity or name similarity and address similarity as input attributes

| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Name | 0.945 | 0.992 | 0.900 | 0.943 |
| Address | 0.864 | 0.952 | 0.765 | 0.849 |
| Name & Address | 0.947 | 0.991 | 0.903 | 0.944 |

Table 3.3: The performance of model when using name similarity, email similarity or name similarity and email similarity as input attributes

| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Name | 0.993 | 0.998 | 0.987 | 0.993 |
| Email | 0.985 | 0.994 | 0.974 | 0.984 |
| Name & Email | 0.995 | 0.999 | 0.990 | 0.995 |

Table 3.4: The performance of model when using name similarity, phone similarity or name similarity and phone similarity as input attributes

| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Name | 0.992 | 0.997 | 0.985 | 0.991 |
| Phone | 0.983 | 0.994 | 0.971 | 0.982 |
| Name & Phone | 0.994 | 0.998 | 0.990 | 0.994 |

Table 3.5: The performance of model when using name similarity, postcode similarity or name similarity and postcode similarity as input attributes

| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Name | 0.943 | 0.990 | 0.896 | 0.939 |
| Postcode | 0.934 | 0.985 | 0.882 | 0.931 |
| Name & Postcode | 0.948 | 0.991 | 0.900 | 0.942 |

is more promising than using the similarity of address, postcode, email and phone number if they are available. However, if we use the similarity of one of incomplete features together with name similarity as input features, the performance of the classifier will get a slight improvement. To understand why this happens, we calculate the pearson correlation coefficients of these features shown in Table3.6. It can be seen that the

Table 3.6: The pearson correlation coefficient between the similarity of name and similarity of incomplete features

| | Name Similarity |
|---|---|
| Address Similarity | 0.77 |
| Postcode Similarity | 0.80 |
| Email Similarity | 0.95 |
| Phone Similarity | 0.95 |

name similarity shows a strong correlation to the similarity of incomplete features, which means the addition of incomplete features can only bring a bit useful information. Consequently, considering the cost of training and deploying multiple classifiers, we finally decide not to use address, postcode, email and phone number but only to keep name for classification task.

### 3.2.5. Blocking

Unless the data set is small, it is impractical to treat all possible pairs as a potential candidate for merging. Apart from the scaling issue, all record pairs checked by an O($n^2$) approach will be rejected since usually only about 1% of all pairs are true matches[5]. To keep the scalability of proposed algorithm to large-scale data, the general solution is blocking. Blocking means pre-grouping records in blocks or clusters according to one or multiple blocking key. Once blocking is done, records inside the block are then cross-compared. The benefit of blocking is that the comparison is only done on records that already have something in common, which greatly reduces the number of comparisons needed to find a matching pair. For our research, postal code is chosen as the blocking key to pre-group the records as this attribute is the attribute with the highest occurrence frequency besides the name.

### 3.2.6. Merging strategy

After determining which model and blocking key to use and how much training data is needed, we continue to explore the merging strategy based on the results of the classifier. The records that are classified as the matching pairs are merged directly. This is the simplest method for implementing data record merging and we call it direct merging in our research. However, due to the limitation of the classifier, direct merging could not capture all the matching pairs in the data set. Besides the record profile information, the relational information among records could also be utilized for detecting matching pairs. To be specific, we take the each record as one node and add a link between two nodes if they are classified as the matching record pair by the classifier. After that, we could obtain a graph showing the relational information of matching between records. Assuming a transitive equivalence relationship, which means if the record A is matched with the record B and record B is matched with the record C, then record A can be inferred matched with the record C knowing the above conditions, existing in our data, we detect the connected component in the graph and manually add link(links) to these components to make them fully connected. In this case, we can utilize the relational information hidden in the graph to find more matching pairs that cannot be found by the classifier. The method uses both record profile information and relational information is called graph merging in our research.

At last, considering transitive equivalence relationship may not always be correct. Instead of fully connecting all the components, we try to add links between node pairs of within the connected components having some statistical characteristics to make the link addition more precisely. This method is called partial graph merging in our research. Partial graph merging also utilizes both classifier and graph relation to find matching record pairs. The difference between partial graph merging and graph merging is that partial graph merging only adds links that meet some statistical requirements. More details about the implementation of merging strategies mentioned above will be introduced in Chapter 3.3.

## 3.3. Evaluation of entity resolution algorithm

### 3.3.1. Pre-analysis

To see if graph merging has the potential to further improve our entity resolution performance, we are interested in the structure of graph created by ground-truth. The specific steps are randomly sampling three hundred thousand records available with target labels, pre-grouping records according to the blocking key, and finally making records within the same block cross-compare with each other. We take each record as one data node, and a link is drawn between them if the their CoC and VAT are the same. After this we obtain a graph constructed based on the ground-truth and we call this graph as ground-truth graph in the following experiments. Then, we extract the connected components with no less than three nodes(this is because transitive equivalence only happens when there are not less than three data nodes) in the ground-truth graph and find all the connected components in the ground-truth graph are fully connected. The reason why the connected components in the ground-truth graph are all cliques is that each record only have one CoC and VAT number. In this case, if the we know record A matches record B and record B matches record C, then record A matches record C is a must as both record A and record C have only one CoC and VAT and their CoC and VAT are known to be the same as record B. The distribution of the size of all the connected components in the ground-truth graph is shown in figure 3.7, where the size means how many nodes are there in the connected component. It can be seen that the size of most of the connected components in the ground-truth graph is small, which indicates that the distribution of matching accounts in the ground-truth graph is relatively sparse.

Generally speaking, manual link addition within the connected components seems to have the potential to improve the performance of entity resolution for our data as the connected components in the ground-truth graph are all fully connected. So we will go on with verifying if graph merging can improve the entity resolution performance.

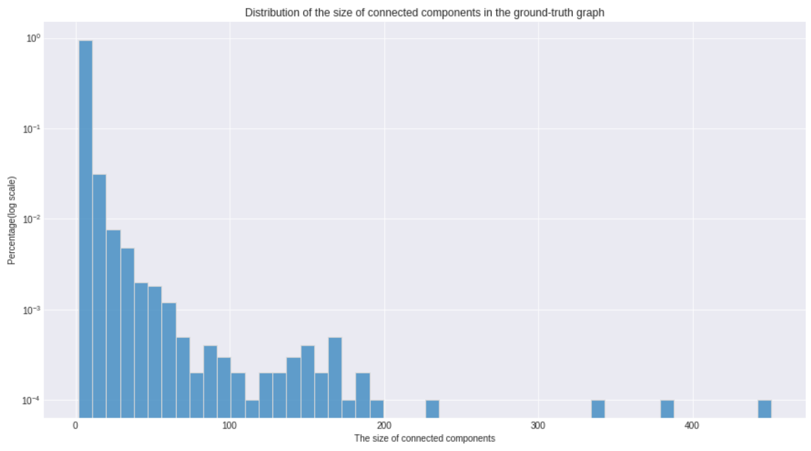


Figure 3.7: Distribution of the size of connected components in the ground-truth graph

### 3.3.2. Evaluate entity resolution results with pF$_1$ method

As for the evaluation metric of entity resolution, we choose pF$_1$ [30] method mentioned in Chapter 2.3.2. PF$_1$ method utilizes pairwise precision and recall to evaluate the performance of entity resolution. Similar to the pre-analysis, we randomly sample three hundred thousand records available with target labels and do a pairwise comparison for the records in the same block. Then, for direct merging, we draw a line between two nodes if they are classified as matching pair by the classifier. Next, based on the results of direct merging and the assumption of transitive equivalence is sound for our dataset, we add links in the connected components with more than three nodes(transitive equivalence cannot happen for a connected component with less than three nodes) if they are not fully connected. Finally, we compare the matching links obtained by direct merging and graph merging to our ground-truth links. To benchmark whether proposed method performs better than simple grouping using postcode, we also compare the matching links obtained by connecting all nodes in the same block to to our ground-truth links. The result is shown in Table.3.7, which represents the average value of five-time experiments. In this table, we can firstly see that the proposed entity resolution algorithm(direct merging and graph merging) performs much better in precision while performs relatively worse in recall than simple grouping. The harmonic means of precision and recall for direct merging, graph merging, and simple grouping are 0.927, 0.930, and 0.827 respectively.

Considering the obvious difference in the harmonic means of precision and recall between simple grouping and proposed entity resolution algorithm(direct merging and graph merging), we don't treat simple grouping as a suitable merging strategy in this project. Then, the direct merging can reach a high precision and relatively lower recall, which corresponds to the performance of our pre-trained classifier. Next, we find that the graph merging reaches a higher recall but sacrifices precision. And the precision value it sacrificed is close to the recall value it gained. This can be attributed to link addition brings accurate matching links for some components while also brings false matching links for some components and the number of false matching links and accurate matching links is approximately equal. Based on this result, we cannot say graph merging is definitely better than direct merging or verse visa. Which merging strategy is better depends on the requirements of application. After entity resolution, we will obtain multiple clusters of records, where each cluster of records represents the same entity.

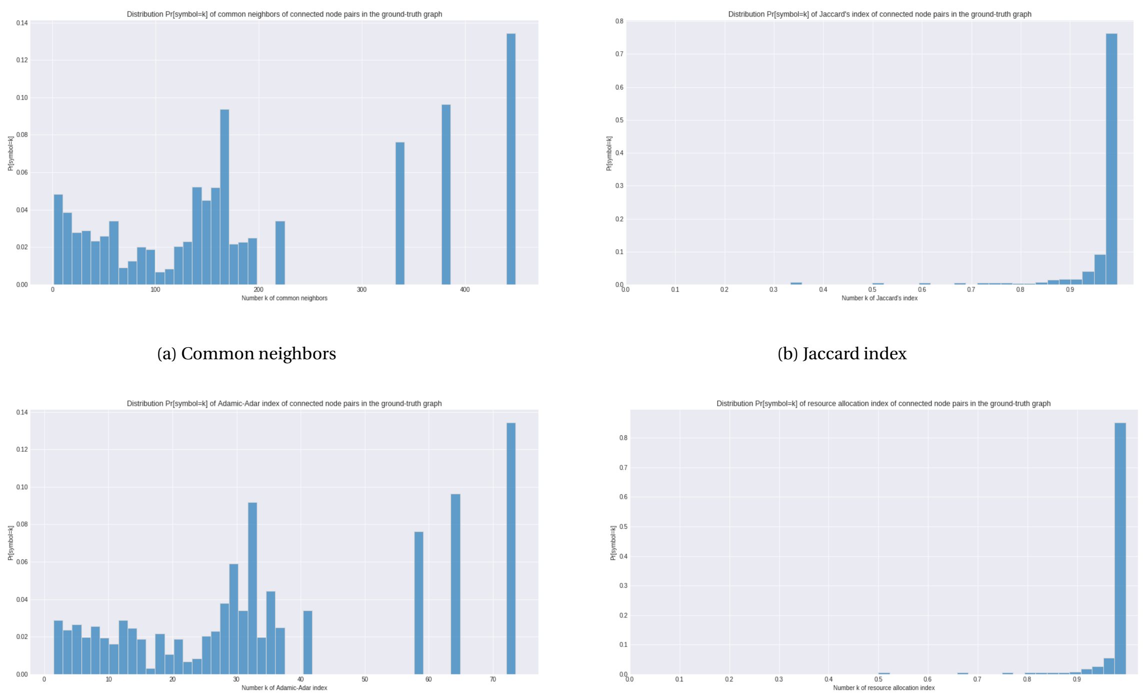
Table 3.7: Average recall and precision of direct & graph merging for five times of experiment.

| | Precision | Recall |
|---|---|---|
| Direct merging | 0.941 | 0.914 |
| Graph merging | 0.921 | 0.940 |
| Simple grouping(postcode) | 0.705 | 1.0 |

### 3.3.3. Partial graph merging

Direct merging reaches a high precision in detecting the existing matching links but cannot cover enough links due to the limitations of the classifier. Graph merging beats the direct merging in recall because it utilizes the relational information. However, graph merging fails in precision compared with direct merging as it introduces false matching links along with correct matching links. In view of this situation, instead of fully connecting all the connected components, we are interested in if we can do the link addition more precisely. That is, we try to explore if we can sacrifice less precision but gain more recall by connecting partial nodes with specific characters in connected components. Note that the connected components mentioned here represents the connected components with no less than three nodes as the transitive equivalence equation can only happen in the connected components with no less than three nodes. To reach this goal, firstly we set a control parameter m, which equals the number of links to be added divided by the difference of number of links of graph merging and direct merging. Through the control parameter m, we can dynamically adjust the number of links we'd like to add to achieve the desired entity resolution effect. This strategy is called partial graph merging in our research, whose core idea is sacrificing less precision to gain more recall. Partial graph merging reaches a middle state between graph merging and direct merging when $0<m<1$.

Inspired by the idea of link prediction problem, we are interested in if we can find some interesting patterns through link prediction indicators so that we can add links more precisely according to the instruction by link prediction indicators. As for the similarity-based indicators suggested in Chapter 2.4, we use common neighbors, Jaccard index, Adamic-Adar index and resource allocation index. To explore if there are some patterns in the matching links between entities which can give us instructions on how to add links more precisely, we calculate the distribution of common neighbors, Jaccard index, Adamic-Adar index, and resource allocation index of the connected node pairs(links) in the connected components of ground-truth graph respectively. The result is shown in figure 3.8. For Jaccard and resource allocation index, the links' distribution shows an obvious trend to have a large value. So we infer that starting to manually add links from the node pairs with relatively higher Jaccard index in the connected components could be more promising than randomly adding links within all node pairs in the connected components. For common neighbors and Adamic-Adar index, the index distribution for links is relatively uniform. However, within the range of relatively higher common neighbors and Adamic-Adar index, the links' distribution is more dense. So by the same token, we infer that starting to manually add links from the node pairs with relatively higher common neighbors and Adamic-Adar index in the connected components could be also more promising than randomly adding links within all node pairs in the connected components. To explore if this inference really works for



(a) Common neighbors

(b) Jaccard index

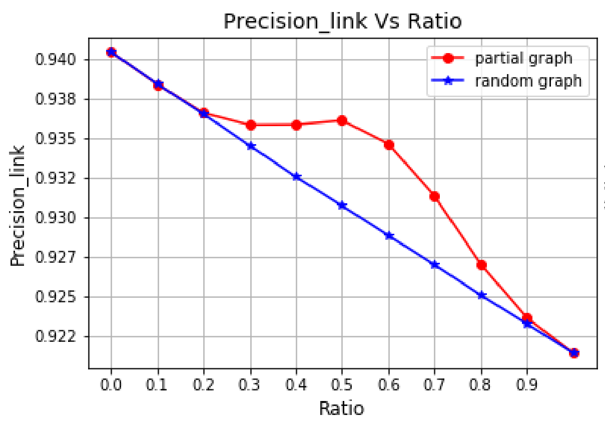
(c) Adamic-Adar index

(d) Resource allocation index

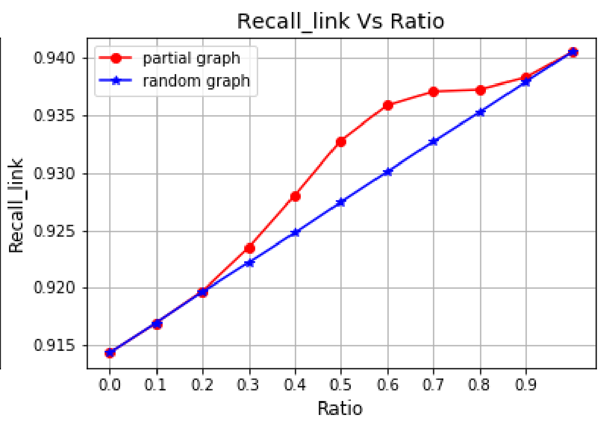
Figure 3.8: The distribution of common neighbors, Jaccard index, Adamic-Adar index, and resource allocation index

our data, we carry out experiments by always adding link for the node pair with higher common neighbors, Jaccard index or Adamic-Adar index firstly and then turn to the lower one within the connected components. Then, to benchmark whether we have done the link addition more precisely, we compare the performance of these partial graph merging strategies to the performance of randomly adding the same number of links within components without any limitation. The random link addition strategy mentioned above is called random graph merging and it is taken as the baseline in the following experiments. The results of the comparison between partial graph merging with common neighbors, Jaccard index, Adamic-Adar, or resource allocation index and the benchmark random graph merging is shown in figure 3.9, figure 3.10, figure 3.11, and figure 3.12 respectively. It can be seen that partial graph merging(no matter common neighbor-based, Jaccard index-based, Adamic-Adar index-based or resource allocation index-based one) shows a higher performance compared with random graph merging, especially when m equals 0.5 or 0.6, which means it is wise to do link addition for the node pairs with higher common neighbors, Jaccard index, Adamic-Adar index, or resource allocation index first in the connected components and take m as 0.5 or 0.6 to reach a relatively high precision along with a relatively high recall.

Another type of network derived feature is graph embedding, which is commonly used for solving link prediction and node classification problem. Network embedding is intended to represent the network by mapping the nodes of graph to the low-dimensional vector space which keeps the original structure of the graph. After network embedding, each node will represented by a unique vector so it can be utilized for a ML classifier or mathematical calculation. As for the network embedding model, we choose node2vec, a random walk-based method to derive the embedding vectors of the nodes and is well-known for its scalability to large-scale data. Specifically, the following configuration is considered: dimension = 12, p = 2, q = 0.25, number of walks = 20, walk length = 40. The value of hyper-parameter is selected according to the survey did by the author of [21]. It is found that parameters do not make a large difference on a similar dataset by Exact and we select the parameter which leads to the best performance as mentioned in [21]. After node2vec embedding, the nodes of entity graph are represented by low-dimensional vectors and we use cosine similarity to evaluate the similarity score between nodes. Similar to similarity-based link prediction method, we calculate the distribution of cosine similarity of embedding vectors for connected node pairs in the connected components of ground-truth graph. The result is shown in figure 3.13. It can be seen that links in the connected

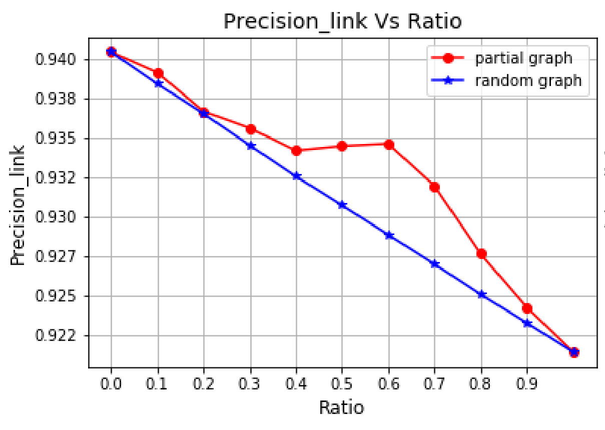


(a) The precision of partial graph merging(common neighbors-based) and random graph merging , x-axis represents the ratio of added links

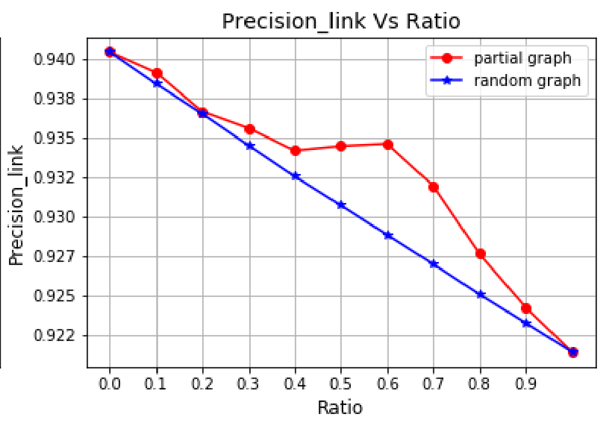


(b) The recall of partial graph merging(common neighbors-based) and random graph merging, x-axis represents the ratio of added links

Figure 3.9: The development of precision and recall of partial graph merging(common neighbors-based) and random graph merging vs the development of the value of control parameter m(Ratio)

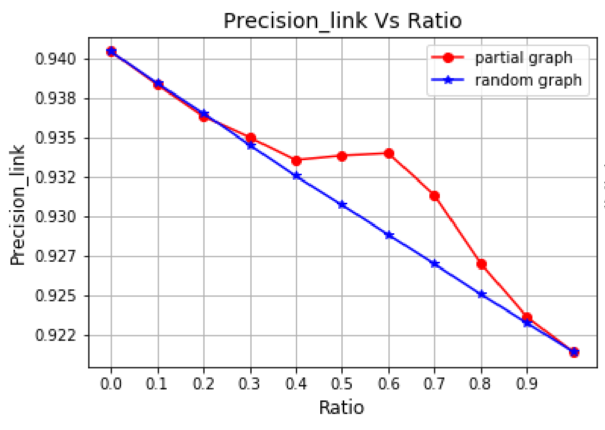


(a) The precision of partial graph merging(Jaccard index-based) and random graph merging, x-axis represents the ratio of added links

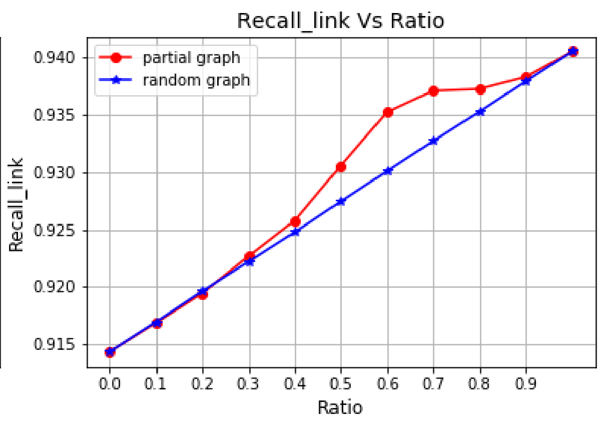


(b) The recall of partial graph merging(Jaccard index-based) and random graph merging, x-axis represents the ratio of added links

Figure 3.10: The development of precision and recall of partial graph merging(Jaccard index-based) and random graph merging vs the development of the value of control parameter m(Ratio)

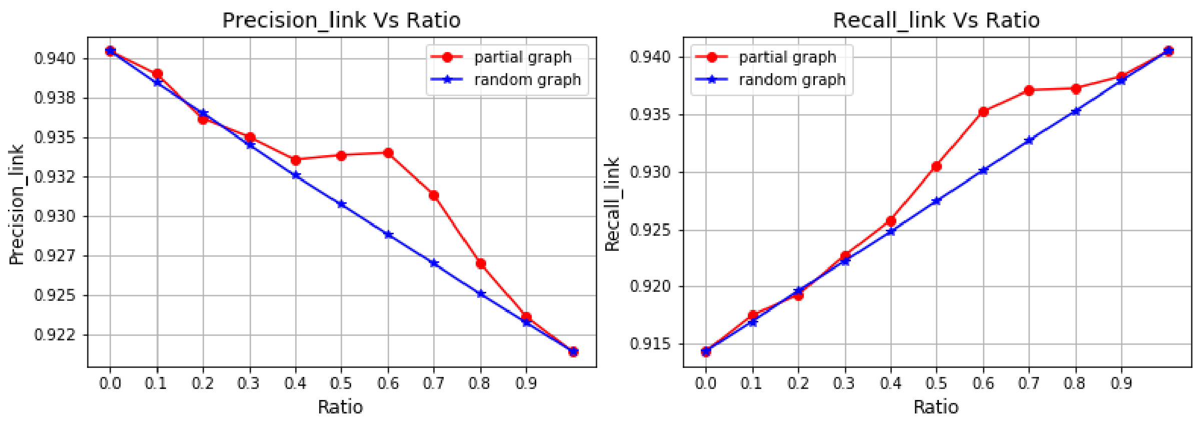


(a) The precision of partial graph merging(Adamic-Adar index-based) and random graph merging, x-axis represents the ratio of added links



(b) The recall of partial graph merging(Adamic-Adar index-based) and random graph merging, x-axis represents the ratio of added links

Figure 3.11: The development of precision and recall of partial graph merging(Adamic-Adar index-based) and random graph merging vs the development of the value of control parameter m(Ratio)



(a) The precision of partial graph merging(resource allocation index-based) and random graph merging, x-axis represents the ratio of added links

(b) The recall of partial graph merging(resource allocation index-based) and random graph merging, x-axis represents the ratio of added links

Figure 3.12: The development of precision and recall of partial graph merging(resource allocation index-based) and random graph merging vs the development of the value of control parameter m(Ratio)

components have a fairly high value of cosine similarity score for embedding vectors. So we infer that giving high priority to node pairs with high value of cosine similarity score for embedding vectors for link addition would be promising in the prediction graph. The results of the comparison between partial graph merging with node2vec embedding and benchmark random graph merging is shown in figure 3.14. It can be seen that partial graph merging shows a similar performance compared to random graph merging when m<0.5 and becomes better after m >=0.6.

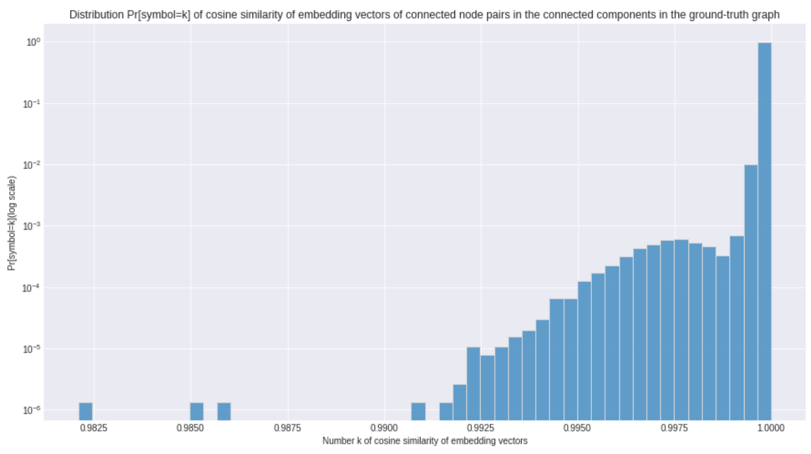
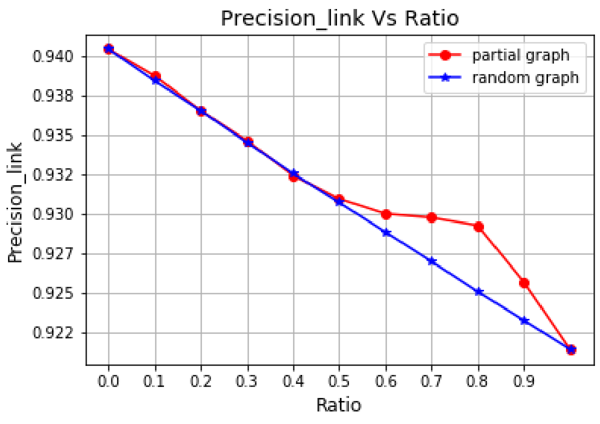
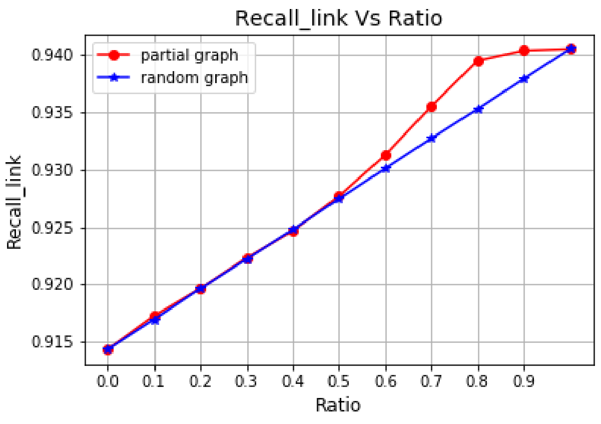


Figure 3.13: Distribution of cosine similarity of embedding vectors of connected node pairs in the connected components in the ground-truth graph

To select the suitable merging strategy for late payment prediction, we accumulate the results of different merging strategy together and make a comparison. The result is shown in figure 3.15. For late payment payment prediction, we have no absolute inclination that precision is more important than recall or verse visa. So finally we take direct merging, graph merging, and common neighbor-based partial graph merging when m=0.5 where we can reach the highest precision, the highest recall, and relatively high precision and recall respectively.

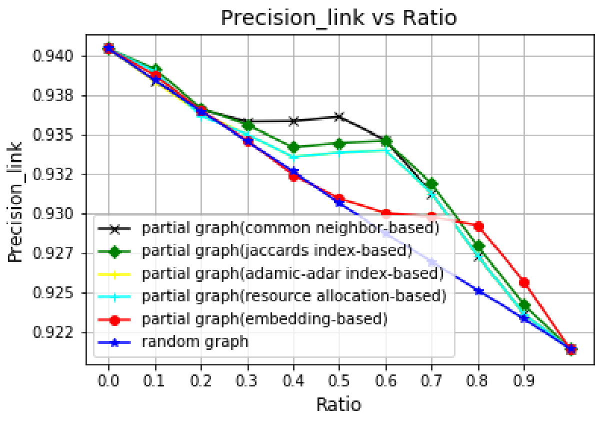


(a) The precision of partial graph merging(graph embedding-based) and random graph merging, x-axis represents the ratio of added links

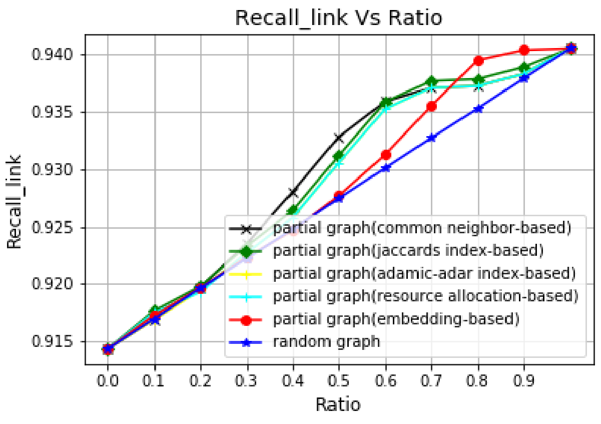


(b) The recall of partial graph merging(graph embedding-based) and random graph merging, x-axis represents the ratio of added links

Figure 3.14: The development of precision and recall of partial graph merging(graph embedding-based) and random graph merging vs the development of the value of control parameter m(Ratio)



(a) The precision of partial graph merging and random graph merging, x-axis represents the ratio of added links



(b) The recall of partial graph merging and random graph merging, x-axis represents the ratio of added links

Figure 3.15: The development of precision and recall of partial graph merging and random graph merging vs the development of the value of control parameter m(Ratio)

# 4 Debtor Payment Prediction

In chapter 4, we will go on with the experiment regarding using the output of entity resolution to improve debtor payment prediction(also known as late payment prediction). In previous chapters we have discussed how the entity resolution is carried out to merge the records representing the same entity. In the following stage, we will explore how the debtor payment prediction model perform with & without the resolved entities by proposed entity resolution algorithms. The background and method for debtor payment prediction and the corresponding experiments will be introduced in details in the following sections.

## 4.1. Background of debtor payment prediction

Invoice is a document issued by the seller of goods (or services) to the buyer. This document contains information about the purchase, such as the amount owed, the payment deadline, and sometimes a short description of goods (or services). The buyer who receives the invoice is called debtor and to predict when the buyer will pay invoice is exactly what the debtor payment prediction means. The mode of a debtor payment is shown in figure 4.1. When there is a trade between two users, the user who sells goods will send a invoice to the user who buys goods with a a deadline that is expected to receive money. After receiving the invoice, the user who buys goods can decide the time to pay the invoice. The exact payment time can be either later than deadline or earlier than deadline. For this project, instead of predicting the time of invoice payment, we try to predict whether a debtor receiver will pay the invoice on time(before the deadline given by the invoice sender) according to its historical payment behaviours and profile information of the invoice.

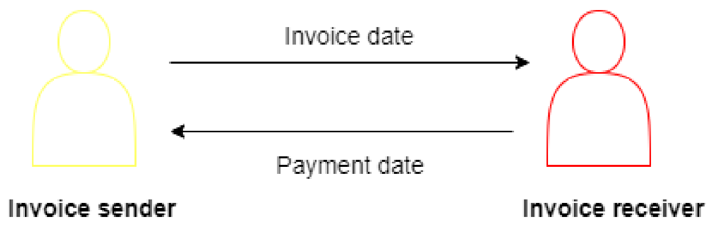


Figure 4.1: The mode of a debtor payment

## 4.2. Feature space

In this section, we will have a look into the features currently used for the late payment prediction in Exact. Meanwhile, in previous chapters we have discussed how the entity resolution of debtor accounts is done. In this chapter we will explore how the entity resolution affect the prediction result of debtor payment prediction through affecting the calculation way of financial features(also called as debtor payment features). The descriptions of available financial features for debtor payment prediction is shown as follows:

- **Invoice ID:** The id of the invoice.

- **Account ID:** The id of the debtor.
- **Division ID:** The division of the invoice belongs to.
- **Custom ID:** The id based on the results of entity resolution. The records that are identified as the same entity by proposed ER algorithm will have the same custom id.
- **Payment_days:** The syscreated date of when the invoice was paid.
- **Tenure_acc:** The number of days that the debtor is registered.
- **Amount_dc:** Amount of the invoice.
- **Invoice_date:** Invoice launch date.
- **Med_d2c_acc_2000:** The median of the number of days before the invoice was paid, of the last 2000 invoices of the debtor.
- **Std_d2c_acc_2000:** The standard deviation of the number of days before the invoice was paid, of the last 2000 invoices of the debtor.
- **Count_late_paid_acc_2000:** The number of time that an invoice was paid late, of the last 2000 invoices of the debtor.
- **Count_nolate_paid_acc_2000:** The number of time that an invoice was not paid late, of the last 2000 invoices of the debtor.
- **Count_inv_acc_2000:** The number of invoices of the debtor.
- **Count_late_outstanding_acc_2000:** The number of time that an invoice was paid late, that is currently still outstanding, of the last 2000 invoices of the debtor.
- **Count_nolate_outstanding_acc_2000:** The number of time that an invoice was not paid late (yet), that is currently still outstanding, of the last 2000 invoices of the debtor.
- **Payment_term_bucket:** The payment term.
- **Due_date:** The due date of the invoice.
- **Most_use_payment_term:** The most used payment term of a debtor.
- **Outstanding_amount:** Outstanding invoice amount.
- **Count_nolate_paid_div_300:** The ratio of late paid invoices of latest 300 invoices for the accounts in the same division of the debtor.
- **Count_late_paid_div_300:** The ratio of invoices that are not paid late of latest 300 invoices for the accounts in the same division of the debtor.
- **Count_late_outstanding_div_300:** The number of time that an invoice was paid late, that is currently still outstanding, of the latest 300 invoices for the accounts in the same division of the debtor.
- **Count_nolate_outstanding_div_300:** The number of time that an invoice was not paid late, that is currently still outstanding, of the latest 300 invoices for the accounts in the same division of the debtor.
- **Med_d2c_div_300:** The median of the number of days before the invoice was paid, of the last 300 invoices for the accounts in the same division of the debtor.
- **Std_d2c_div_300:** The standard deviation of the number of days before the invoice was paid, of the last 300 invoices for the accounts in the same division of the debtor.
- **Med_d2c_overall_ly:** the median of the number of days before the invoice was paid, in the same month as the invoice date, last year, across all divisions.
- **Med_d2c_overall_u:** the median of the number of days before the invoice was paid, across all divisions.

- **Med_d2c_sector_ly:** The median of the number of days before the invoice was paid, in the same month as the invoice date, last year, in the same sector as the debtor.
- **Med_d2c_sector_u:** the median of the number of days before the invoice was paid, in the same sector as the debtor.
- **Ratio_paid_late:** The ratio of late paid invoices.

In Exact, the financial features mentioned above are divided into three categories: sector level features, division level features, and account level features. The relationship among sector level(e.g. Med_d2c_sector_u), division level(e.g. Med_d2c_div_300), and account level(e.g. Count_late_paid_acc_2000) features is shown in figure 4.2. Account level features show the debtor's historical payment behavior, such as the number of times the debtor has delayed paying invoices. Sector and division represent a larger business scope based on account. The reason why section and division level feature are selected here is that whether a debtor will pay its invoice on time or not could be also related to which section and division the debtor account is in. For our research, the entity resolution mainly affects the account level features. Previously, a debtor's account level features will be counted based on the account id. After entity resolution, a debtor's account level features will be counted based on the custom id(As explained earlier, the records that are identified as the same entity by proposed ER algorithm will have the same custom id) and the debtors who are identified as the same entity will share the information about feature value calculation. The thinking behind this is our debtor payment predictor may not perform enough well due to the sparse distribution of raw accounts data. With the incorporation of entity resolution, a debtor account will be available with more historical payment data so that the predictor could perform better with the consideration of more data. Instead of trying to improve the model prediction performance by incorporating new features or deploying a more complex model, we aim to improve the prediction by comprehensively mining information inside the existing data.

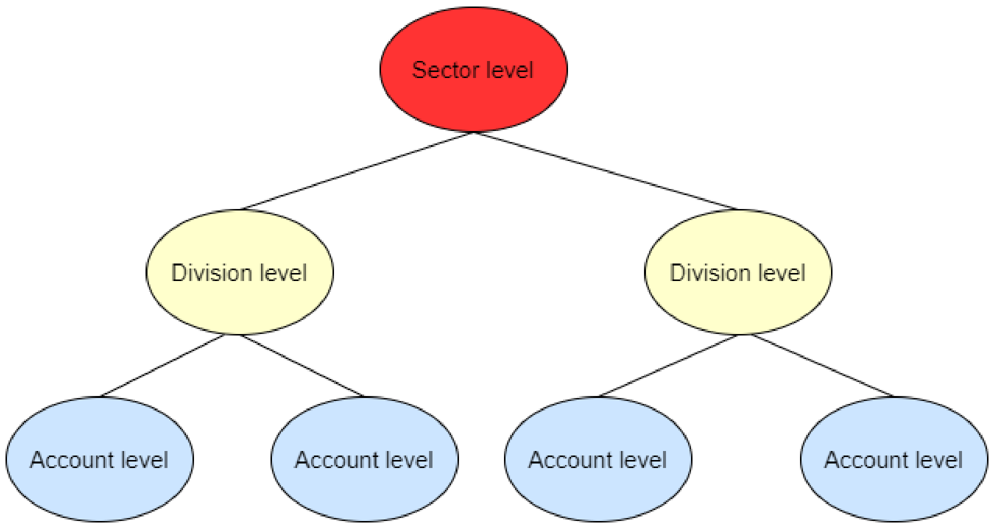


Figure 4.2: Relationship between sector level, division level features, and account level features

To have a deep insight into the correlation between each financial feature, a heatmap is drawn showing the Pearson correlation coefficient among features, which is shown in figure 4.3. An interesting finding is that features in the same level show a generally stronger correlation to each other. For instance, the correlation coefficient between std_d2c_div_300 and count_late_paid_div_300 is 0.68 while the correlation coefficient between std_d2c_div_300 and and count_late_paid_acc_2000 is 0.44. In general, the financial features current available do not show a strong correlation to each other, which means each feature can bring a certain amount of useful information to the prediction results thus should be kept in the following experiment.

## 4.3. Experiment setup for late payment prediction

Once the features are settled they can be utilized to train and test model for late payment prediction. The experiments will be implemented based on debtor payment features without entity resolution and debtor

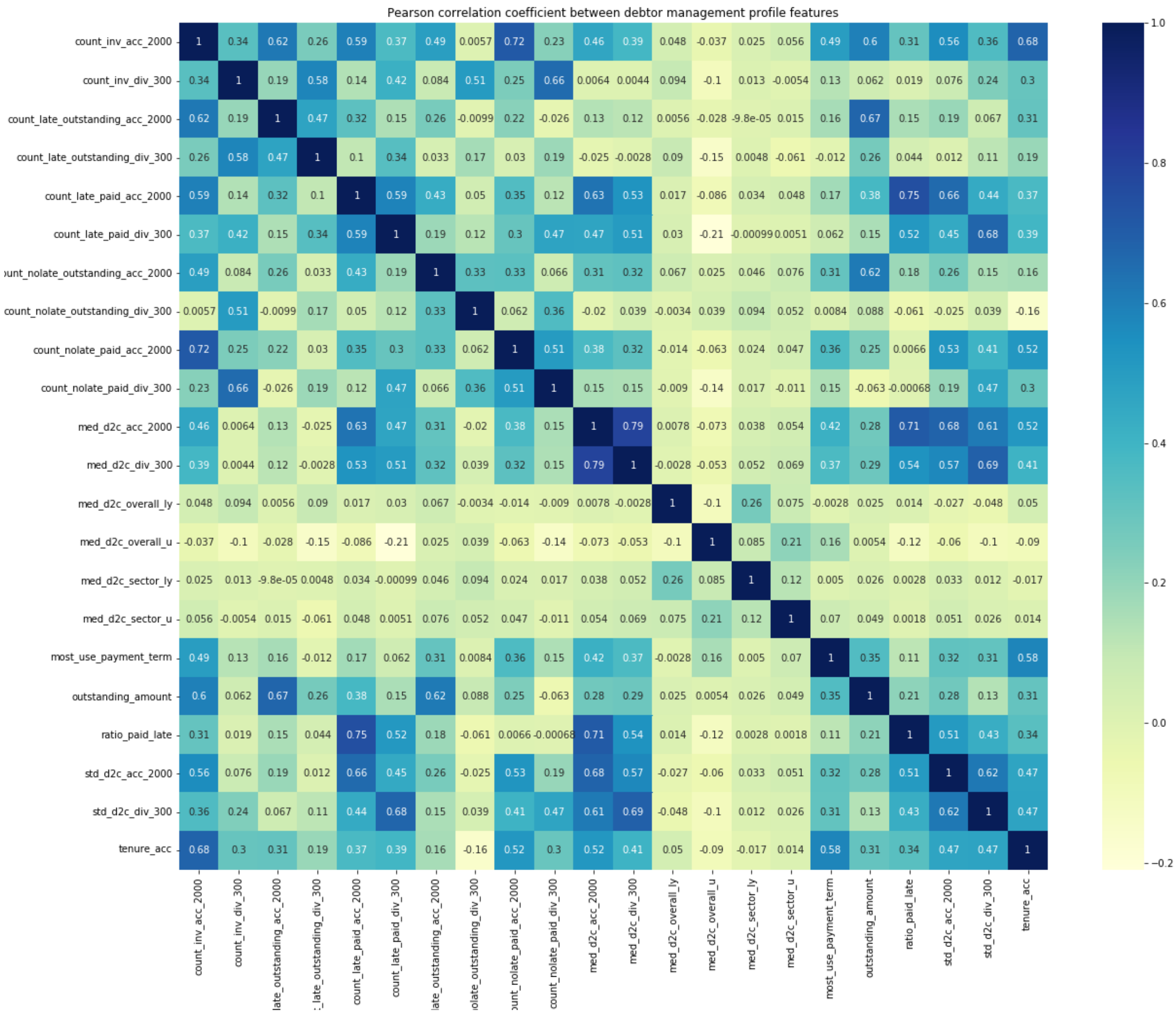


Figure 4.3: Pearson correlation coefficient between debtor management profile features

payment features with entity resolution respectively. For this experiment, we will try to predict if debtors will pay their invoice on time for October 2021. Note that we don't only select the data of typeA accounts with CoC and VAT, but all the data of typeA accounts in per month, because the algorithm we proposed doesn't need target label in application. In this section the configuration of the experiment will be introduced in details.

### 4.3.1. Preprocessing

On accounts that the experiment is based on a real-world data-set, there is the situation that missing data occurs in terms of some financial features as not all of the invoice profile columns are mandatory for users to fill in. To solve this problem, we fill up financial features for which the data is missing are 0. Meanwhile, different financial features have different numerical distribution ranges, which will affect the results of data analysis. In order to eliminate the affect of numerical distribution, data normalization is needed to solve the comparability between financial features. In view of the large differentiation of our feature values, we use the non-linear normalization method, log normalization to map the original data. After the original data is normalized, the value of each feature is in the same order of magnitude, which is suitable for comprehensive comparative evaluation.

### 4.3.2. Modelling

The models that we choose for this experiment are logistic regression and decision tree. These classifiers are currently treated as the candidates of target classifier for late payment prediction in Exact as these models can usually generate a good performance in binary classification problems, light-weight, and transparent.

- **Dataset consistency:** For one epoch, we keep the same data set for the training and testing of different models.
- **Data split:** The data splitting is not taken in a 7:3 general manner but in a "leave-one-out" manner. The thinking is if one debtor has multiple invoice payment data(n for instance), we will take the latest one as the testing set and the previous n-1 as training set. In this case, we can simulate the situation that using history to predict the future.
- **Data shuffle:** The training set is scrambled to ensure that the performance score of model is not dependent on the ordering of the data.
- **Parameter optimization:** The parameters of the classifier are tuned with greedy-grid search. And the value of parameters for logistic regression and decision tree used for this experiment is shown in Table.4.1 and Table.4.2 respectively

Table 4.1: Parameter setting for logistic regression

| | solver | penalty | C | n_jobs | tol | max_iter | verbose |
|---|---|---|---|---|---|---|---|
| Parameter value | lbfgs | l2 | 1.0 | -1 | 1e-3 | 500 | True |

Table 4.2: Parameter setting for decision tree

| | criterion | splitter | random_state | max_features | min_samples_split | min_samples_leaf |
|---|---|---|---|---|---|---|
| Parameter value | gini | best | 1 | log2 | 16 | 8 |

### 4.3.3. Class distribution

To know if class balance is needed before the classifier training, we calculate the distribution of invoices that are paid on time or late for October 2021. The result is shown in figure 4.4. The number of invoice paid on time is approximately 30 percent more than the number of invoice paid late. Considering the class distribution is not extremely imbalanced, we will not carry out class balance measure and the data is input into the tuned models directly. In this case, we can see the effect of with/without entity resolution for the model more clearly.

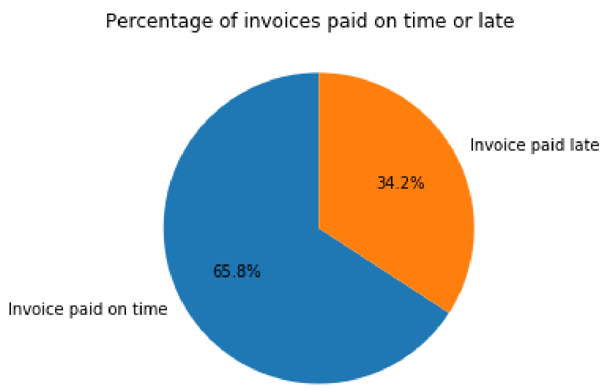


Figure 4.4: Percentage of invoices paid on time or late in October 2021

### 4.3.4. Evaluation

In order to select the most suitable classifier and entity resolution strategy according to the experimental results, it is important to choose a suitable metric to evaluate the proposed experiments of debtor payment prediction. Accuracy is generally the most popular evaluation method for binary classification problem. However, we are also interested in if the classifier performs enough well in detecting the late payment cases. Based on this requirements, precision and recall are incorporated to calculate the fraction of positive samples that are successfully found and the fraction of the successfully found positive samples from all positively samples. In addition, F1-score is provided to give a fair evaluation of the balance between precision and recall of late payment cases. The calculation equation of F1-score is shown as follows:

$$\text{F1 score} = \frac{2 * precision * recall}{precision + recall} \tag{4.1}$$

### 4.3.5. Pipeline

For our research, we try to verify whether the performance of debtor payment prediction model could be improved through providing the model with more profile information data. And the more profile information data is offered through entity resolution. To reach this goal, we design the experiment pipeline based on the model configuration mentioned above, which is shown in figure 4.5. In step1, the invoice payment data for each debtor account(typeA account) is extracted. The invoice payment data, account id, and entity id obtained by entity resolution will be treated as input features while the payment timeline of the invoice is treated as target label. In step2, we will use the available data to conduct two groups of experiments. That is, firstly we only use invoice payment data as input features to train and test the performance of the classifier. This experiment will be treated as baseline experiment. Then, the entity id for each debtor account obtained by entity resolution will be introduced and account level features of invoice payment data will be calculated based on entity id. As mentioned in section 3.3.3, by taking direct merging, graph merging, and common neighbor-based partial graph merging when m=0.5, we can reach the highest precision, the highest recall, and relatively high precision and recall simultaneously. We will try entity resolution with direct merging, graph merging and partial graph merging(common neighbor-based) and select out the most suitable merging strategy for debtor payment prediction. In step3, the input data for these experiments will be pre-processed and then input into the logistic regression and decision tress model. On account that there is a slight degree of imbalance between classes of input data and we'd like to show the effect of entity resolution more clearly, we will not do additional class balance measure for the model. Finally, in step 4, we will evaluate the results of the experiments with/without entity resolution using accuracy, precision, recall, and f1-score to make a final decision that if entity matching of debtor accounts can help improve the debtor payment prediction. The results of the experiments will be presented in section 4.4.

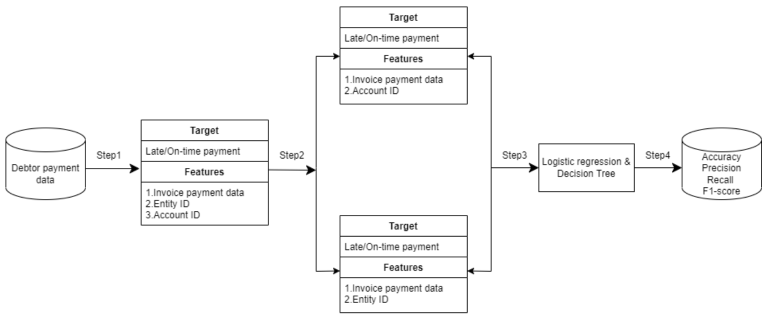


Figure 4.5: Pipeline of debtor payment prediction experiments

## 4.4. Evaluation of debtor payment prediction experiment

In this section, the potential of entity resolution will be explored in details. we will analyze the result of the experiments introduced in the section 4.3 and select out the most suitable classifier and merging strategy for debtor payment prediction.

### 4.4.1. Baseline experiment

To begin with, we only use invoice payment data as input features and classifiers are trained without class balancing. The results of logistic regression and decision tree are shown in Table.4.3. It can be seen that decision tree model shows a similar performance compared with logistic regression model. Logistic regression model presents a better performance in overall accuracy while decision tree model shows a better performance in F1-score. Considering that late payment prediction values more in detecting late payment cases, decision tree is finally taken as our target model and our following experiments will also use this model.

Table 4.3: The performance of logistic regression and decision tree for baseline experiment

| | LR | DT |
|---|---|---|
| Accuracy | 76.08% | 74.89% |
| Precision | 64.10% | 59.23% |
| Recall | 42.93% | 47.61% |
| F1_score | 51.42% | 52.79% |

To get a better understanding of the impact of the each financial feature, we take a look at the feature importance from the decision tree model. The feature importance score of decision tree is computed as the (normalized) total reduction of the criterion brought by that feature. The criterion is also known as the Gini importance[18]. Figure 4.6 presents the feature importance score of decision tree model. From the figure it can be seen account level features play an important role in classifier decision making. For instance, the most important two account level features are "ratio_paid_late" and "count_late_paid_acc_2000", which represents the ratio of late paid invoices of the debtor account and the number of time that the debtor account paying late in its latest 2000 invoice payment transaction. This result shows that the incorporation of entity resolution has the potential to improve debtor payment prediction as the entity resolution could provide more data for account level features so that the classifier can consider more comprehensively for account level features in its decision making process.

### 4.4.2. ER-based experiment

In this experiment, the entity id for each invoice payment data is incorporated and the account level features will be recalculated based on the entity id. On accounts of the baseline experiment, decision tress model

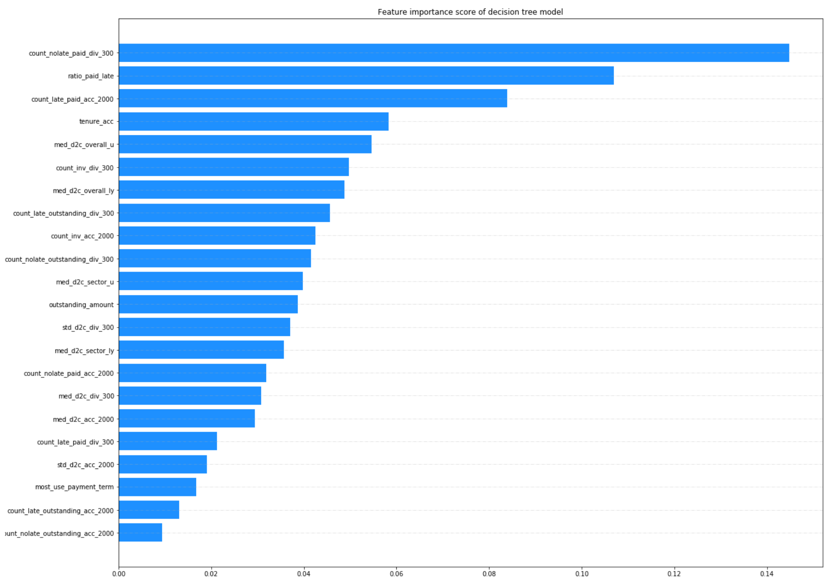


Figure 4.6: Feature importance score of decision tree model

with class balancing measure is finally taken as the target classifier as it is more promising in identifying the late payment class. Meanwhile, we use direct merging, graph merging, and common neighbor-based(m=0.5) partial graph merging to generate entity id respectively. The performance of decision tree model with the incorporation of different entity resolution strategies are shown in Table.4.4. It can be seen that the prediction performance of decision tree increases after the incorporation of entity resolution. In general, decision tree with common neighbor-based partial graph merging strategy reaches the best performance as it has the highest accuracy and f1-score.

Table 4.4: The performance of decision tree model with the incorporation of entity resolution

| | DT(No ER) | DT(Graph Merging) | DT(Direct Merging) | DT(Partial Graph Merging) |
|---|---|---|---|---|
| Accuracy | 74.89% | 74.88% | 75.42% | 75.51% |
| Precision | 59.23% | 59.16% | 60.25% | 60.37% |
| Recall | 47.61% | 47.85% | 48.91% | 49.35% |
| F1_score | 52.79% | 52.91% | 53.99% | 54.31% |

While the proposed entity resolution algorithm has an effect in improving the performance of debtor payment prediction. We are also interested in if we can observe a more obvious effect by limiting the range of data set. That is, only the accounts that has been matched by entity resolution algorithm by at least one time will be taken as the training and testing set. The goal of this experiment is to see the potential of entity resolution so we can decide if it is worthwhile to do more in-depth research for entity resolution in the future. We will still use decision tree with common-neighbor based partial graph merging entity resolution for this experiment as it is known to have the best performance in the previous experiment. To benchmark the effect of entity resolution, we take decision tree without entity resolution measure as baseline model. Both of these two models will use the same dataset for training and testing. The result is shown in Table.4.5. Table.4.5. shows that the incorporation of the entity resolution brings an obvious improvement for the debtor payment prediction of matching accounts. To have a further look at if the improvement for debtor payment prediction of matching accounts still exists under a general scenario, we take all the accounts data as training set but only the accounts data that are known to be matched as test set. The configuration of model is kept as the same with the previous one experiment. Table.4.6. shows that the prediction performance of matching accounts still increases obviously with a classifier trained under generic scenario. This result proves the potential of entity resolution. In the future, further research can be done to find more matching accounts, which could bring more obvious performance improvement of the model on the whole data.

Table 4.5: The performance of decision tree with/without entity resolution. Only the accounts that are known to be matched are chosen for model training and testing

| | DT(No ER) | DT(Partial Graph Merging) |
|---|---|---|
| Accuracy | 72.97% | 84.34% |
| Precision | 58.65% | 77.46% |
| Recall | 47.80% | 70.81% |
| F1_score | 52.67% | 73.98% |

Table 4.6: The performance of decision tree with/without entity resolution. All the accounts are chosen for model training and only the accounts that are known to be matched are chosen for model testing

| | DT(No ER) | DT(Partial Graph Merging) |
|---|---|---|
| Accuracy | 74.05% | 85.71% |
| Precision | 61.25% | 81.24% |
| Recall | 47.73% | 70.96% |
| F1_score | 53.65% | 75.75% |

# 5 Conclusion

In chapter 5, we will summarize the experiments and results which are studied and explored in this research. In the first stage of this research, we propose a generic entity resolution workflow and we use Exact data to illustrate our method. Machine learning combined with graph link prediction technologies are utilized to find the matching links between debtor accounts. Then, in the second stage of this research, decision tree with class balancing is found to be most promising for debtor payment prediction and entity resolution is incorporated for debtor management data to improve the prediction performance of the decision tree further. The utility of entity resolution in this research can be generalized as data integration, which means providing more data for debtor accounts' historical payment statistics calculation.

In section 1.2, a series of questions are put forward to lead this research and define the research targets of this project. The core research problem is: **Can entity resolution be utilized to improve the debtor payment prediction?** Based on this core research problem, three sub-questions are given and we will answer these three sub-questions one by one according to the findings of experiments mentioned in the report.

**SQ1:** How can we design an entity resolution algorithm suitable for the scenario of large-scale and free-form data?

There are three common challenges when matching accounts data in the real life, and the same is true of Exact, which are free-form, unlabeled and large-scale. To create required entity resolution algorithm in view of these challenges, we firstly check the distribution of attributes that could be applied to entity comparison and identification, and test using them as input features of a machine learning classifier one by one. The more frequently the attribute appears, the more we give priority to using it as a classification feature. Considering the performance and simplicity of the method, name attribute and logistic regression model are finally taken as the attribute and classifier for matching/non-matching classification. To find the matching pairs between data records, theoretically we need to cross-compare all the data records. However, pairwise comparison is impractical for a large scale data so we firstly use postcode, which is available for most of the accounts, to initially group the records and then make records within the same group cross-compare with each other. Finally, it is found that the classifier can't capture all the matching links, so we construct a entity graph and utilize the relational information hidden in the graph to find more matching links. The proposed algorithm can realize a dynamic link addition according to requirement of the project. It is verified that the proposed entity resolution algorithm can generate a good performance for detecting matching account pairs.

**SQ2:** How can we evaluate the performance of entity resolution algorithm?

To evaluate the performance of entity resolution algorithm, firstly we need to know how do we define if two account data records representing the same entity. In this project, the account profile attributes CoC and VAT are numbers used by government agencies to track and identify companies. Two records are defined to embody the same entity if they have the same CoC and VAT. Similarly, two records are defined to embody not the same entity if they have different CoC or VAT. Under this definition, we select the data available with CoC and VAT attributes for entity resolution experiments. In the experiment, the matching record pairs obtained by prediction is compared to the ground-truth matching record pairs so we can know how the algorithm performs in the experiment data set. The specific metrics for evaluation are pairwise precision and pairwise recall introduced in Chapter 3. These two metrics are utilized to evaluate whether the matching pairs captured by the entity resolution algorithm are precise enough and whether entity resolution algorithm captures enough matching pairs.

Non-matching record pairs are not considered in the evaluation of entity resolution as only matching record pairs can bring useful information for data integration. Unfortunately, as the attributes CoC and VAT are only available for partial of accounts data, the proposed evaluation method can only be utilized on the accounts data available with CoC and VAT. To evaluate the entity resolution algorithm on the whole data set, manual labelling or ground-truth-free methods are needed to be carried out in the future.

**SQ3:** How can we utilized the output of entity resolution algorithm to improve the debtor payment prediction?
The way to utilize the output of proposed entity resolution algorithm to improve the debtor payment prediction is integrating the account level features of matching accounts. The accounts that are matched by proposed entity resolution will share their information for account level features computation. For instance, account A pays its invoices late for 4 times in its 10 invoice payment records while account B pay its invoices late for 8 times in its 10 invoice payment records. In addition, account A is known to matched with account B. Then after feature integration, account A and account B will be both regarded as paying their invoices late for 12 times in its 20 invoice payment records and we use this value for the debtor payment prediction of account A and account B. It is proved that using entity resolution to do account data integration can promote the debtor payment prediction. This strategy could bring obvious effect for the accounts with a few historical payment data if they are found to be matched with the accounts with sufficient historical payment data.

# 6 Future work

In this section, we will discuss the limitations and future extension for the described method and workflow. For some aspects of the experiment, we think that the method proposed in the current experiment is not comprehensive enough, and it still has an improvement direction. However, due to the limited time and scope of this thesis, these aspects were not investigated and we list them here for further study.

## 6.1. Entity resolution

In this research, a learning-driven entity resolution workflow is proposed for free-form and large-scale data. With the incorporation of entity resolution, enterprise account data can be integrated to eliminate data sparseness problem. From the existing experiments, it can be seen that the existing entity resolution algorithm still has room for improvement.

### 6.1.1. Manual labelling

In this research, we use the attributes CoC and VAT to justify if two account records represent the same entity(enterprise). However, this definition could incorporate bias for some cases. For instance, when a company operates a variety of businesses, then it may have multiple VAT numbers. This will prevent us from matching two records representing the same company. At the same time, some subsidiaries affiliated with the parent company will use the same CoC and VAT as the parent company. This will lead us to regard these subsidiaries as the same company as their parent company. But is it really fair? In order to achieve a more accurate entity definition so that we can classify account records more fairly, we should consider inviting experts in the field to do manual labelling to determine whether two records represent the same entity.

### 6.1.2. Generalization of matching/non-matching classification

In this project, the pre-trained classifier performs relatively well, which could be attributed to the obvious difference between matching record pairs and non-matching record pairs in our data set. However, we haven't done extended experiments on other public data sets for entity resolution. It is assumed that the real-world data could be more complicated so that we can't classify it well by a classic machine learning model with a simple similarity metric. In this case, the state-of-the-art NLP model is worth exploring. For instance, word2vec[11] uses a neural network model to learn word associations from a large corpus of text. The learned word representation could be used for classification task. [15] shows word2vec can generally perform better than classic word representation techniques and it is widely used in industry because of the low computational cost. With the word embedding representation as input, convolutional neural network is verified to have the potential to perform better than classic machine learning classifier due to its advantages in feature extraction[6].

### 6.1.3. Blocking

In this project, the proposed entity resolution algorithm uses a single attribute as blocking key to do initial pre-grouping. People could try to expand the blocking key with more attributes such as the e-mail and postcode of the company if they have more time to run an entity resolution algorithm. In addition, the evaluation of the time efficiency for proposed entity resolution algorithm could be provided to show the scalability of the algorithm to big data more intuitively.

### 6.1.4. Link prediction

Link prediction technique is utilized in this research to predict the possible matching links inside the entity graph. We didn't try all kinds of link prediction methods so the potential of link prediction is still worth exploring. Link prediction indicators such as Katz index[29] and SimRank[12] are different from the common neighbors or Jaccard's index used in the experiments because they consider the global graph structure, which means that they could bring a better prediction performance compared with the indicators merely considering local structure of the network. The author of [42] mentions that using the combination of several indicators could be quite beneficial for the performance of link prediction so it is also a good idea to study whether the combination of multiple link prediction indicators could bring us a more precise matching link addition effect for our data set. On top of this, it is shown that graph neural network(GNN) can generate convincing high performance in many real-world graphs for link prediction[23]. On account of the complexity of GNN architectures, GNN was not implemented in this project but left for future work.

## 6.2. Debtor payment prediction

The current debtor payment prediction experiment only involves the simple decision tree model and the traditional financial features. There are still many types of models and attributes based on entity resolution that are worth trying to improve the performance of debtor payment prediction in the future.

### 6.2.1. Classifier deployment

In this research, we test whether the incorporation of the output of entity resolution could bring a performance improvement on decision tree model and the result shows that such potential does exist. To verify whether the proposed method could bring a general improvement effect, we should test the results of more types of classifiers after incorporating entity resolution. For instance, AdaBoost is a boosting algorithm widely used in the filed of financial prediction and can reach a high performance because it can aggregate the classification results of multiple weak classifiers and flexibly adjust the weights of different weak classifiers[38]. Empirically, AdaBoost with decision trees as week classifiers is supposed to perform better than a single decision tree. It's valuable to explore whether the incorporation of the output of entity resolution can still bring an improvement to the performance of a complex model.

### 6.2.2. Graph features

Besides integrating the financial features, the output of entity resolution can be utilized in another style to bring bonus information for debtor payment prediction, that is, constructing entity graphs. After the integration of accounts through entity resolution, we could obtain the entity graph showing the relationship information between debtor accounts. It is assumed that same accounts detected by entity resolution will have similar payment behaviours, so the payment behaviour information inside the entity network could be mined to further improve the debtor payment prediction. In addition, it is shown that the graph features of monetary transaction network based on resolved entities also have the potential to improve the payment prediction[26][13], which is also a direction worthwhile to try to further improve the debtor payment prediction in the future.

# A
# Appendix

## A.1. Extensive experiments for debtor payment prediction

Due to the limited time of this project, we only take the data of October 2021 to verify whether the proposed entity resolution algorithm could be utilized to improve debtor payment prediction in chapter 4. To know whether the proposed entity resolution algorithm could bring an improvement to prediction performance under generic scenarios, more experiments regarding the effect of incorporating entity resolution for debtor payment prediction should be implemented in the future. Here, an extensive experiment based on the data of September 2021 for testing the effect of entity resolution for debtor payment prediction is conducted. The baseline classifier for debtor payment prediction is still decision tree and people can choose different classifiers according to the application background of the project.

The class distribution of invoices paid on time and late for September 2021 is shown in figure A.1. It can be seen that the class is still slightly imbalanced. Therefore, we also do not take further class balance measures. For the steps of model deployment and incorporation of entity resolution, we keep the same strategy mentioned in chapter 4. The feature importance score for the baseline decision tree model based on the data of September 2021 is shown in figure A.2. Similar to October 2021, account-level features related to debtor's historical payment behavior still play an important role in the final decision of the classifier.

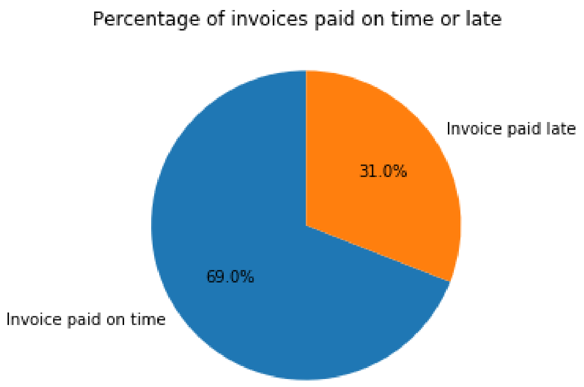


Figure A.1: Percentage of invoices paid on time or late in September 2021

In the same way as chapter 4, we still use direct merging, graph merging, and common neighbor-based(m=0.5) partial graph merging to integrate debtor accounts data. Partial graph merging brings the most obvious performance improvement for decision tree and we select out the accounts that are matched by partial graph merging to see if we can still observe a more obvious model performance improvement on matched account data. Table A.2 shows a similar tendency as we observed in chapter 4. The performance of the model improved significantly when we selected all accounts as the training set, but only selected matched account as

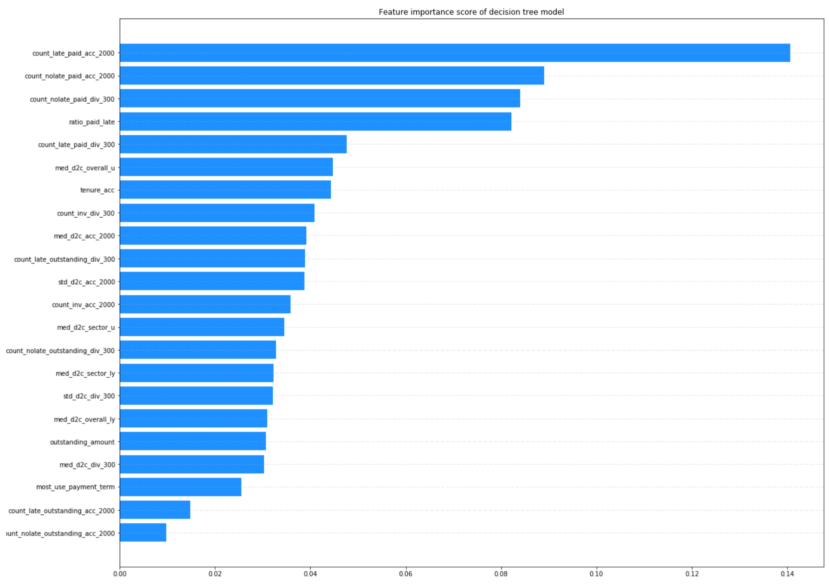


Figure A.2: Feature importance score of decision tree in September 2021

testing set. This result further supports our point of view that it is worth further studying entity resolution to find more matching accounts as entity resolution has brought telling performance improvement for the debtor payment prediction of matched accounts.

Table A.1: The performance of decision tree model with the incorporation of entity resolution in September 2021

| | DT(No ER) | DT(Direct Merging) | DT(Graph Merging) | DT(Partial Graph Merging) |
|---|---|---|---|---|
| Accuracy | 75.80% | 76.05% | 75.65% | 76.23% |
| Precision | 59.11% | 59.50% | 58.81% | 59.99% |
| Recall | 48.69% | 49.72% | 48.40% | 49.65% |
| F1_score | 53.39% | 54.17% | 53.10% | 54.33% |

Table A.2: The performance of decision tree with/without entity resolution in September 2021. All the accounts are chosen for model training and only the accounts that are known to be matched are chosen for and model testing

| | DT(No ER) | DT(Partial Graph Merging) |
|---|---|---|
| Accuracy | 77.09% | 84.55% |
| Precision | 61.45% | 77.14% |
| Recall | 49.0% | 63.83% |
| F1_score | 54.52% | 69.86% |

## A.2. Extensive experiments for the scenario of class imbalance

Another situation often encountered in industrial production for debtor payment prediction is class imbalance. The ratio of invoices paid late and paid on time may not be close to 1:1 so additional class balancing measures are required to ensure the performance of the classifier. Although the proportion of classes is only slightly unbalanced for the invoice payment data of September 2021, we still use it to test whether incorporating the output of entity resolution can improve debtor payment prediction when class balancing measures are introduced. We assign more weight to the minor class through setting the class_weight of the classifier as balanced. "Balanced" mode automatically adjust weights inversely proportional to class occurrence frequency for each class in the input data. The feature importance score for the decision tree model with class balancing measures is shown in figure.A.3. It can be seen that some changes have taken place in the importance order of features after the introduction of class balancing measures. However, account level features still have relatively higher feature importance score, which means that entity resolution is still promising for improving debtor payment prediction.

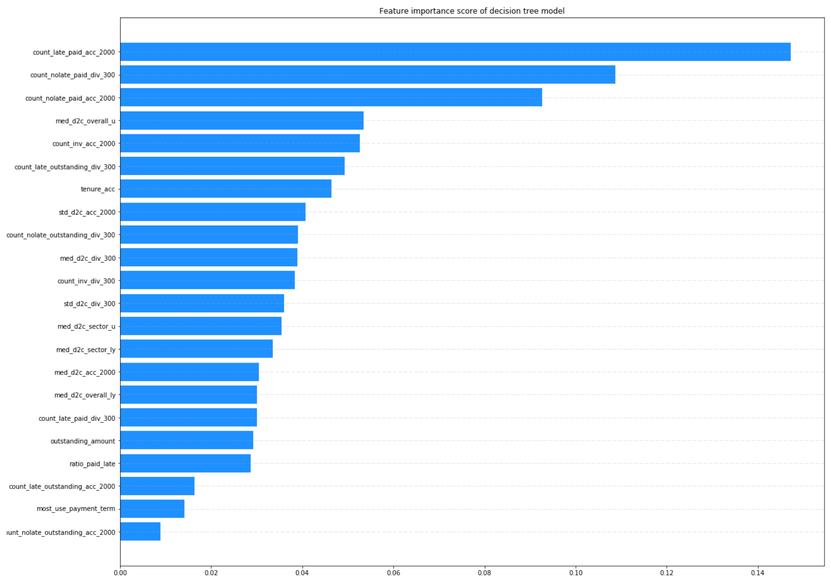


Figure A.3: Feature importance score of decision tree with class balancing measures in September 2021

Table A.3 shows the comparison of prediction performance between baseline decision tree(class balanced) and decision tree(class balanced) with different entity resolution strategies. It can be seen that direct merging and partial graph merging can still bring improvements to the prediction performance and the improvements they bring are similar. In the same way mentioned above, we select out the accounts that are matched by partial graph merging as testing set to see if entity resolution has a more obvious improvement effect for matched accounts. The result is shown in Table A.4. A significant improvement is still observed after incorporating class balancing measures for the model, which means the proposed entity resolution algorithm could also be utilized to ameliorate debtor payment prediction under the scenario of class imbalance.

Table A.3: The performance of decision tree models(class balanced) with the incorporation of entity resolution in September 2021

| | DT(No ER) | DT(Direct Merging) | DT(Graph Merging) | DT(Partial Graph Merging) |
|---|---|---|---|---|
| Accuracy | 71.77% | 71.98% | 71.73% | 71.61% |
| Precision | 50.33% | 50.58% | 50.27% | 50.11% |
| Recall | 67.3% | 69.26% | 67.37% | 70.26% |
| F1_score | 57.59% | 58.46% | 57.58% | 58.5% |

Table A.4: The performance of decision trees(class balanced) with/without entity resolution in September 2021. All the accounts are chosen for model training and only the accounts that are known to be matched are chosen for and model testing

| | DT(No ER) | DT(Partial Graph Merging) |
|---|---|---|
| Accuracy | 73.53% | 82.77% |
| Precision | 52.16% | 64.73% |
| Recall | 67.13% | 84.73% |
| F1_score | 58.71% | 73.39% |

# Bibliography

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[2] Yasser Altowim and Sharad Mehrotra. Parallel progressive approach to entity resolution using mapreduce. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 909–920. IEEE, 2017.

[3] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, 2021.

[4] Indrajit Bhattacharya and Lise Getoor. Relational clustering for multi-type entity resolution. In *Proceedings of the 4th International workshop on Multi-relational Mining*, pages 3–12, 2005.

[5] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5–es, 2007.

[6] Denny Britz. Understanding convolutional neural networks for nlp. *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)*, 2015.

[7] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[8] Bolun Chen, Fenfen Li, Senbo Chen, Ronglin Hu, and Ling Chen. Link prediction based on non-negative matrix factorization. *PloS one*, 12(8):e0182968, 2017.

[9] Ling Chen, Jun Cui, Xing Tang, Yuntao Qian, Yansheng Li, and Yongjun Zhang. Rlpath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and representation learning. *Applied Intelligence*, 52(4):4715–4726, 2022.

[10] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[11] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[12] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *The 2011 International joint conference on neural networks*, pages 1237–1244. IEEE, 2011.

[13] Wensheng Dai. Application of improved convolution neural network in financial forecasting. *Journal of Organizational and End User Computing (JOEUC)*, 34(3):1–16, 2022.

[14] Mary Dawood, Nicholas Horsewood, and Frank Strobel. Predicting sovereign debt crises: an early warning system approach. *Journal of Financial Stability*, 28:16–28, 2017.

[15] Giovanni Di Gennaro, Amedeo Buonanno, and Francesco AN Palmieri. Considerations about learning word2vec. *The Journal of Supercomputing*, 77(11):12320–12335, 2021.

[16] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Deeper–deep entity resolution. *arXiv preprint arXiv:1710.00597*, 2017.

[17] Marco Fioramanti. Predicting sovereign debt crises using artificial neural networks: A comparative approach. *Journal of Financial Stability*, 4(2):149–164, 2008.

[18] Mohammad M Ghiasi and Sohrab Zendehboudi. Decision tree-based methodology to select a proper approach for wart treatment. *Computers in biology and medicine*, 108:400–409, 2019.

[19] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[20] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[21] Arthur Hovanesyan. Late payment prediction of invoices through graph features. 2019.

[22] Linmei Hu, Mengmei Zhang, Shaohua Li, Jinghan Shi, Chuan Shi, Cheng Yang, and Zhiyuan Liu. Text-graph enhanced knowledge graph representation learning. *Frontiers in Artificial Intelligence*, 4, 2021.

[23] Md Kamrul Islam, Sabeur Aridhi, and Malika Smail-Tabbone. A comparative study of similarity-based and gnn-based link prediction approaches. *arXiv preprint arXiv:2008.08879*, 2020.

[24] Hideki Kawai, Hector Garcia-Molina, Omar Benjelloun, David Menestrina, Euijong Whang, and Heng Gong. P-swoosh: Parallel algorithm for generic entity resolution. Technical Report 2006-19, Stanford InfoLab, September 2006. URL `http://ilpubs.stanford.edu:8090/784/`.

[25] Benedikt Köhler and Hansjoerg Fromm. Analysis of a debt collection process using bayesian networks. In *Proceedings of the 55th Hawaii International Conference on System Sciences*, 2022.

[26] Bastijn Kostense. Assessing covid-19 impact on dutch smes using dynamic network analysis. 2021.

[27] Fenhua Li, Jing He, Guangyan Huang, Yanchun Zhang, and Yong Shi. Retracted: A clustering-based link prediction method in social networks, 2014.

[28] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[29] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.

[30] Christopher D Manning and Prabhakar Raghavan. utze, introduction to information retrieval, 2008.

[31] David Menestrina, Steven Euijong Whang, and Hector Garcia-Molina. Evaluating entity resolution results. *Proceedings of the VLDB Endowment*, 3(1-2):208–219, 2010.

[32] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[34] Peter Sarlin. Sovereign debt monitor: A visual self-organizing maps approach. In *2011 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, pages 1–8. IEEE, 2011.

[35] Roberto Savona and Marika Vezzoli. Fitting and forecasting sovereign defaults using multiple risk signals. *Oxford Bulletin of Economics and Statistics*, 77(1):66–92, 2015.

[36] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. Ieee, 2016.

[37] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. *Journal of Basic and Applied Sciences*, 13:459–465, 2017.

[38] Jie Sun, Ming-yue Jia, and Hui Li. Adaboost ensemble for financial distress prediction: An empirical comparison with data from chinese listed companies. *Expert systems with applications*, 38(8):9305–9312, 2011.

[39] Jiapeng Wang and Yihong Dong. Measurement of text similarity: a survey. *Information*, 11(9):421, 2020.

[40] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4): 825–853, 2021.

[41] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

[42] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[43] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. Counterfactual graph learning for link prediction. *arXiv preprint arXiv:2106.02172*, 2021.