Embedding for Anomaly Detection on Health Insurance Claims

Jiaqi Lu  
School of Computer Science  
McGill University  
Montreal, Canada  
jiaqi.lu2@mail.mcgill.ca

Benjamin C. M. Fung  
School of Information Studies  
McGill University  
Montreal, Canada  
ben.fung@mcgill.ca

William K. Cheung  
Department of Computer Science  
Hong Kong Baptist University  
Kowloon Tong, Hong Kong  
william@comp.hkbu.edu.hk

Abstract—Properly analyzing health insurance claims data could lead to significant business insights and benefits for health service providers and insurance companies. Yet, health insurance data is often high dimensional and contains complex interleave sequences of claims. Instead of conducting machine learning tasks directly on the raw data, a better approach is performing the tasks on high-quality embeddings of the raw data. Driven by the real business need of Solution Segic Inc., a Canadian technology company in the group insurance industry, we extract health insurance claims embeddings with neural networks in the context of anomaly detection. We propose and thoroughly examine six embedding components that are customized based on different possible assumptions made on the data. One of our proposed embedding components, EC-ReStepRec, significantly outperforms other candidates on two anomaly detection tasks. This is the first embedding study done on health insurance claims for anomaly detection.

Index Terms—embedding, representation learning, machine learning, health insurance claims

I. INTRODUCTION

Health insurance claims data are the bills between health service providers and insurance companies for the services obtained by a patient. A typical claiming process begins with a patient receiving health services from a provider. Next, the service provider submits a claim directly to an insurance company. The claim goes through validation checks followed by rules based on the patient’s plan for pricing. Then, the insurance company pays the service provider [1]. Health insurance claims could be generally categorized into medical, pharmaceutical, and dental based on the services and service providers under request. As shown in Figure 1, each claim record generally contains information about the patient, the service provider, and the service. The exact attributes included in a claim depend on its category. For pharmaceutical claims, typical patient attributes include name, date of birth, address, etc. Typical service provider attributes include pharmacy code, pharmacist code, etc. Typical service attributes include medication code, quantity, date of service, etc.

Health insurance claims have been increasingly studied, resulting in many analytical insights that contribute to healthcare applications. Koh et al. [2] summarize the applications into four categories: evaluation of treatment effectiveness, healthcare management, customer relationship management, and anomaly detection.

Among the aforementioned applications, anomaly detection deserves special attention from insurance companies and governments. In the context of health insurance claims, there are three types of anomalies: frauds, abuses, and errors. Frauds indicate intentional acts of deception, misrepresentation, or concealment in order to get payment. Abuses indicate excessive or improper use of services that are inconsistent with acceptable business or medical practice and that result in unnecessary costs. Errors are unintentional mistakes made in processing claims. The boundaries between the three categories are not always clear. Frequent errors could suggest an abuse. Besides, intention is hard to be reflected in a claim itself. All three types of anomaly deserve special attention. Further manual examinations are required to determine the actions followed. The general goal is to accurately identify the anomalies.

In reality, however, it is hard to conduct analyses or perform machine learning tasks directly on health insurance claims data, which are often high dimensional and in the form of interleaving sequences. Prevailing data analytical techniques are typically applied to datasets where the records are relatively small in dimension [3]. The same analytical dilemma also appears in other domains such as accounting and banking [4]–[10].

Traditionally, feature engineering plays an important role in addressing the issue of high dimensionality. Based on the knowledge for the target dataset, a relatively small set of indicators would be selected as the input of the detection models. Recently, through the development of deep learning techniques, embedding has been widely studied as a solution to tackle the curse of dimensionality.

Driven by the business requirements of our industrial partner, Solution Segic Inc., a Canadian technology company in the group insurance industry, we have been working on their health insurance claims data. We aim for embeddings that can effectively represent the health insurance claims data.
in low dimensional space but still be descriptive, and thus the embeddings could be effectively applied in the analytical scenario of anomaly detection.

To obtain an effective embedding, we propose six embedding components for health insurance claims data. The embedding components that we present are carefully designed based on different assumptions made on the nature of the data. Each embedding component has a clear but very different learning preference. By training each embedding component as part of a deep learning model, respectively, we obtained the corresponding embeddings and evaluated them on two anomaly detection tasks of different granularity. Our main contributions are summarized as follows:

- This paper is the first embedding study on health insurance claims for anomaly detection. With embedding, we effectively address the curse of dimensionality without heavily relying on domain knowledge for feature selection.
- We propose six embedding components to perform health insurance claims embedding. Working closely with health insurance practitioners, we thoroughly consider the possible assumptions on health insurance claims. Based on different assumptions, we design the embedding components so that each has a distinct learning preference. Our implementation of the embedding components is available online.\(^1\)
- We conducted extensive experiments on real-life health insurance claim data provided by our industrial partner. Results suggest that the embedding obtained by our proposed embedding component, EC-ReStepRec, is of outstanding quality and significantly outperforms other embeddings under comparison.

Section II describes the works related to health insurance claims embedding. Section III formally defines the research problem. Section IV presents each proposed embedding component in detail. Section V shows the experiment on a real-life health insurance claims dataset and the evaluation of the embeddings by two anomaly detection tasks with visualization. Section VI concludes the paper.

II. RELATED WORK

A. Anomaly Detection

Existing machine learning methods for anomaly detection in health insurance claims can be generally categorized into supervised [11] and unsupervised learning methods such as customized scoring models [12], [13]. Both learning methods face the same challenge of high dimensionality in real-life data. Most existing works address this issue based on preliminary knowledge, for example, by computing metrics or aggregated features on the raw data and then using those advanced indicators in the detection model [14]. The knowledge required to figure out the appropriate indicators mostly comes from in-depth case studies and literature reviews with the help of experienced domain experts. Given the advancement

\(^1\)(the link will be updated once the paper is accepted.)
Figure 3 shows an example.

are likely to be suspicious but are generally hard to capture.

represent persistent behavioral patterns or ordered patterns that

insurance claim anomaly detection. For example, they can
generally categorized into two genres of relations:

• Independent relation: the relation among the attributes
within the same claim.

• Dependent relation: the relation among the attributes
cross multiple claims in the same sequence.

Dependent relations are important in the context of health
insurance claim anomaly detection. For example, they can
represent persistent behavioral patterns or ordered patterns that
are likely to be suspicious but are generally hard to capture.
Figure 3 shows an example.

Figure 3: An example of dependent relation: services 1001,
1002 and 1003 are usually requested in order. A patient with
misordered service records is flagged suspicious.

Dimensionality. The dimension for an encoded claim could
be extremely large. The challenge amplifies if the data are in
sequence, where multiple claims are assembled as one input.
This is a challenge because the curse of dimensionality renders
many traditional machine learning algorithms ineffective on
many machine learning tasks. Therefore, a compact but still
informative embedding is important.

In order to resolve those challenges, we resort to embed-
dings. An embedding is a relatively low-dimensional space
into which high-dimensional vectors are transformed. Embed-
dings are helpful because they reduce the dimensionality of
data while still effectively representing the relations within the
original data in the mapping space. Good embeddings could
well serve for various purposes. For example, they could be
the input for a specific target task or be directly visualized
in order to intuitively illustrate the distribution of the original
data.

Here we formally define our research problem. A claim
is defined as \( T = \{ x_1, x_2, \ldots, x_m \} \), where \( x_i \) is an attribute
or a feature. Given a set of sequences, \( D = \{ S_1, S_2, \ldots, S_n \} \)
where each sequence \( S_j \) is constituted by varying length of
claims, \( S_j = (T_1, T_2, \ldots, T_k) \), our problem is to find a mapping
function \( f : D \rightarrow R^d \) and thus every sequence \( S_j \) is mapped to
a continuous vector of length \( d \), \( E = \{ e_1, e_2, \ldots, e_d \} \), where \( m, n, k, \) and \( d \) are all positive integers. \( d \) should be significantly
smaller than \( m \times k \). The mapping should be of high quality so
that the mapping space can effectively represent the original
data.

IV. MODEL: EMBEDDING COMPONENT DESIGN

Figure 4 provides an overview of the architecture.
The objective of this paper is to propose a model to generate an em-
bedding component of health insurance claims to facilities the
subsequent classification task, which is anomaly detection in
our case. In this paper, the classifier is a small fully-connected
neural network responsible to classify the embedded sequences
into classes depending on the user-defined customized task.
In the training phase, both the embedding component and the
classifiers are trained as a whole. In the evaluation phase, only
the embedding components are evaluated. The whole model
takes a sequence of claims \( S_j \) as input. \( T_i \) is the \( i \)th claim in
the sequence, denoted by \( T_i = \{ x_{i1}, x_{i2}, \ldots, x_{im} \} \).

We have explored, proposed, and evaluated different em-
bedding components that are developed based on different
assumptions that can be imposed on health insurance claim
data. Each embedding component is customized for one type
of assumption and thus is endowed with a specific learning
preference, enabling the embedding component to explore cer-
tain relationships effectively. Here we discuss six architectures
of embedding components.

A. EC-Flatten

In EC-Flatten, there is no explicit assumption made in terms
of the relationship between attributes. As we want to grant the
model maximal flexibility, the claims in a sequence are con-
catenated into a one-dimensional vector. Therefore, attributes
that come from the same claim and the attributes that come
from different claims are treated equally. Figure 5 illustrates
the architecture of EC-Flatten, where \( \{ h_1, h_2, \ldots, h_{m+k} \} \) is an
intermediate output with \( m \times k \) dimensions.
B. EC-Recurrent

In EC-Recurrent we assume that the inter-claim relationship in sequential context is important. Thus, each claim is fed into the model as one step. An abstraction persists and is updated from one step to the next. Finally, the output embedding is a global abstraction of the whole sequence. Figure 6 illustrates the architecture of EC-Recurrent, where \( \{h_{i,1}, h_{i,2}, ..., h_{i,p}\} \) is the \( p \)-dimensional global abstraction at step \( i \).

C. EC-Step

In EC-Step we assume that claims in the same sequence do not closely rely on each other. Instead, the inter-attribute relationship within each single claim is more important. As shown in Figure 7, each claim is fed into the model as one step. However, instead of allowing the information to evolve along the steps as EC-Recurrent, at each step the information exposed to the model is isolated. Abstraction is made step by step. \( \{h_{i,1}^1, h_{i,2}^1, ..., h_{i,p}^1\} \) is the \( p \)-dimensional abstraction on the input of step \( i \). Next, the step-wise abstractions are concatenated into an intermediate output with \( p \times k \) dimensions, which is \( \{h_{i,1}^2, h_{i,2}^2, ..., h_{i,q}^2\} \). The intermediate output is further mapped to a continuous space.

D. EC-FlaRec

EC-FlaRec is a hybrid architecture of EC-Flatten and EC-Recurrent. Therefore, while assuming the existence of inter-claim relationship, EC-FlaRec also benefits from certain flexibility. After concatenating the \( q \)-dimensional abstraction \( \{h_1^2, h_2^2, ..., h_q^2\} \) produced by EC-Flatten with one intermediate layer, and the \( p \)-dimensional global abstraction \( \{h_{i,1}^1, h_{i,2}^1, ..., h_{i,p}^1\} \) produced by EC-Recurrent, the concatenated vector is mapped to a continuous space. Figure 8 illustrates the architecture of EC-FlaRec.

E. EC-StepRec

In EC-StepRec we still assume that the inter-claim relationship in sequential context is critical. Yet, in addition to the global abstraction, the partial abstractions obtained during the intermediate steps are also informative. EC-StepRec is similar to EC-Recurrent, where a piece of information persists and is updated among the steps. Instead of outputting the last step abstraction only, here the abstractions obtained at each step are outputted, further abstracted, concatenated and mapped to a continuous space. Figure 9 illustrates the architecture of EC-StepRec. The first layer abstraction on the input of step \( i \) is represented as \( \{h_{i,1}^1, h_{i,2}^1, ..., h_{i,p}^1\} \), where \( p \) is the dimension. The first layer outputs are further abstracted into \( \{h_{i,1}^2, h_{i,2}^2, ..., h_{i,q}^2\} \), where \( q \) is the dimension. The second layer abstractions are concatenated into a \( (q \times k) \)-dimensional intermediate output \( \{h_{i,1}^3, h_{i,2}^3, ..., h_{i,q \times k}^3\} \), which is then mapped to the embedding space.
In EC-ReStepRec we assume that the sequence-wise inter-attribute relationship is important. By introducing a reshape trick, the unit per step for the recurrent layer is no longer a claim, but the values for one attribute across all claims in sequence. Instead of capturing the inter-claim relationship, here the recurrent layer captures the sequence-wise inter-attribute relationship. Next, similar to EC-StepRec, step-wise abstractions are outputted, further abstracted, concatenated, and mapped to a continuous space. Figure 10 illustrates the architecture of EC-ReStepRec. Due to the reshape trick, the input of step $i$ is $\{x_1^i, x_2^i, \ldots, x_k^i\}$. The rest of the symbols used in Figure 10 are in line with Figure 9.

### Table 1: Attribute description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>medication code</td>
<td>The identifier of the medication ordered.</td>
</tr>
<tr>
<td>quantity</td>
<td>The number of unit of the medication ordered.</td>
</tr>
<tr>
<td>age</td>
<td>The age of the patient when the claim is submitted.</td>
</tr>
<tr>
<td>claim amount</td>
<td>The total cost in dollar, including prescription drug cost, and pharmacist’s professional fee.</td>
</tr>
<tr>
<td>transaction day of year</td>
<td>The day when the claim is submitted in the year. It is a number between 1 and 365 (366 if it is a leap year).</td>
</tr>
<tr>
<td>day of supply</td>
<td>The number of days the supply of dispensed medication will last.</td>
</tr>
</tbody>
</table>

### V. Experiments

#### A. Dataset

The experiments were performed on a pharmaceutical claims dataset provided by our industrial partner. We assembled a labeled dataset with the help of domain experts. The dataset consists of both anomalous and benign samples. The anomalous class can be further divided into two types of anomalies: $T_1$: exaggeration of claim amount and $T_2$: persistent early-refill behaviors on narcotics. Each sample is a sequence of claims of different length.

The labeling process simulates the traditional rule-based anomaly detection method. The domain experts of our industrial partners first explain the anomalous patterns. Based on the patterns both the domain experts and our machine learning team carefully design the validation rules accordingly. All claims go through the validation rules for $T_1$ anomaly detection individually. All claims are grouped by medicine code and patient identifier first, then go through the validation rules for $T_2$ anomaly detection as part of a sequence of claims. Upon the accepted pharmaceutical claims ranging from April 2015 through October 2018, we obtained 1,908 $T_1$ anomalies, 7 $T_2$ anomalies, and 8,760 benign cases. It is clear that the dataset is highly imbalanced. To mitigate the problem, 5,000 $T_1$ anomalies and 2,500 $T_2$ anomalies are simulated by utilizing the validation rules reversely. The simulated anomalies are once again verified the domain experts, so they are high-quality synthetic data. Therefore, we finally assembled a dataset with 6,908 $T_1$ anomalies, 2,507 $T_2$ anomalies, and 8,760 benign cases.

Real-life health insurance claim datasets are difficult to obtain, as the data is highly-sensitive and noisy. Many attributes have to be excluded because of two main reasons: missing value: this happens frequently for non-mandatory fields of the claims and unreliable filling: this happens frequently for fields whose format is ambiguous.

After consulting the domain experts, we only use mandatory and reliable attributes in our study. Additionally, we also apply necessary transformations which intuitively can help the model to learn faster and easier. The involved attributes and their descriptions are listed in Table I.
B. Baselines

As we mentioned in Section II, we chose the following baseline methods, which have been employed in similar application scenarios.

- Schreyer et al. [17] employed deep autoencoder to detect anomalies in accounting data. Here we compare with their best two deep autoencoders, \textit{AE8} and \textit{AE9}, as two baselines. Since our focus is the embeddings, we have to adapt the models and fix the dimension of the latent representation as 128.
- Baldassini et al. [9] obtained client embeddings on current account transactions with a \textit{marginalized stacked denoising autoencoder (mSDA)} [16]. Yet, mSDA does not reduce dimension. To be computationally efficient and to also guarantee a fair comparison we first use \textit{principal component analysis (PCA)} to compact the inputs into 128 dimensions and then stream the data into mSDA [15].

C. Experiment Setting

Table IV: Parameter settings for the embedding components

<table>
<thead>
<tr>
<th>m</th>
<th>k</th>
<th>p</th>
<th>q</th>
<th>d</th>
</tr>
</thead>
</table>
| EC-Flatten | 563 | 15 | 128 | /
| EC-Recurrent | 128 | / | / | /
| EC-Step | 128 | 128 | / | /
| EC-FlaRec | 128 | 16 | 128 | /
| EC-StepRec | 128 | 128 | / | /
| EC-ReStepRec | 128 | 128 | / | /

The data go through a standard preprocessing procedure, including one-hot encoding the categorical attributes and normalization of the numeric attributes. Each claim is encoded into a 563-dimensional vector. Since the claim sequences are of varying length, before a sample, whether an anomaly or a benign case, goes into the model, it is either truncated or zero-padded into a sequence of 15. In order to illustrate the capacity of embedding components, we intentionally only use simple preprocessing steps here.

To implement the framework described in Figure 4 we train the full model for a binary classification task that differentiates the anomalous class and the benign class. The default main classifier is a three-layer fully connected neural network, with 64 neurons, 8 neurons, and 1 sigmoid neuron in order. The output is a value between 0 and 1 which we interpret as the probability of being an anomaly.

We randomly split the full dataset into a training set and a testing set. The training set accounts for 80% of the full dataset. 10% of the training set is reserved as a validation set. After training we collect the embedding components EC-Flatten, EC-Recurrent, EC-Step, EC-FlaRec, EC-StepRec, EC-ReStepRec, the encoders of AE8 and AE9, and the trained mapping of mSDA. Each of these maps the original dataset into a $R^{128}$ embedding space.

The parameter settings for the embedding component are shown in Table IV. Each embedding component is regularized by dropout with 0.6 drop out rate and by batch normalization. The models are implemented using TensorFlow in Python and are trained until convergence or reaching a running time limit. Our implementation is available online.

D. Evaluation

Following the convention in [5], [6], we evaluate nine embedding devices by two anomaly detection tasks of different
granularity. Essentially, the tasks could be regarded as a binary classification task and a three-class classification task. Also, we present their t-distributed Stochastic Neighbor Embedding (t-SNE) visualization as an intuitive evaluation [26].

1) Binary Classification Task: For each embedding device, we use it to transform the original data into the $R^{128}$ embedding space and then use the embeddings as the input of 10 traditional machine learning classifiers, including K-nearest neighbors (KNN) where $K = 5$, support vector machine with the linear kernel (L-SVM), support vector machine with the radial basis function kernel (R-SVM), decision tree (DT), random forest (RF), adaboost (Ada), naïve bayes (NB), logistic regression (LR), linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA). Those classifiers are trained to discriminate between the anomalous class and the benign class.

Table II reports the micro-average F1-scores on the testing set for each classifier. A detailed table with F1-scores per class is provided in Appendix A. We evaluate the quality of a specific embedding in two perspectives.

- **Superiority**: This is evaluated by the best micro-average F1-score achieved by any classifier with that embedding. The best micro-average F1-scores for a embedding are in bold.
- **Robustness**: This is evaluated by the average micro-average F1-scores achieved by all classifiers with that embedding.

It is clear that the embedding obtained by EC-ReStepRec outperforms the others in both perspectives. The best micro-average F1-score is 0.93 and the average micro-average F1-score is 0.923. We highlight the best values with square boxes.

2) Three-Class Classification Task: The same sets of classifiers are also trained to discriminate between the T1 anomalous class, the T2 anomalous class, and the benign class. The way we evaluate the embedding quality is the same as in Section V-D1. Table III summarizes the results. A detailed table with F1-scores per class is provided in Appendix B. Again, EC-ReStepRec achieves the best performance in terms of both superiority and robustness. The best micro-average F1-score is 0.88 and the average micro-average F1-score is 0.794.

3) t-SNE Visualization: Figures 11 and 12 illustrate the t-SNE visualization on the embeddings with different levels of granularity. We highlight the EC-ReStepRec embedding with a rectangular yellow box. Clearly, in both scenarios, the EC-ReStepRec embedding maps the samples of the same class closely while mapping the samples from different classes to different regions. Different classes are grouped and have clear boundaries. This indicates that EC-ReStepRec embeddings are of high quality and can meaningfully represent the original input.

4) Result Discussion: Our experimental results suggest that EC-ReStepRec yields the best embedding for anomaly detec-
The outstanding performance of EC-ReStepRec indicates that the learning preference of EC-ReStepRec has the best fit of the health insurance claims, which implicitly means that the assumptions corresponding to EC-ReStepRec describe the health insurance claims well.

VI. CONCLUSION AND LESSON LEARNED

In this paper, we present a method for learning health insurance claims embedding. We discuss six embedding components that are designed based on different assumptions. Our experiments on health insurance claims show that one of our proposed embedding components, namely EC-ReStepRec, achieves the best embedding for anomaly detection. Furthermore, due to the assumptions we made on the target data are quite general, it is possible that our work could also be applied to other similar datasets, for example, other transactional datasets with the characteristics of high dimensionality and sequentiality.

Finally, we would like to share the lesson learned from this university-industry collaboration. Both the deep learning domain and the health insurance industry are complex. There was a steep learning curve for both parties at the early stage of the project. In addition to tackling technical challenges, a lot of effort was spent on gathering and labelling the data with the consideration of privacy, security, and ethical issues. Given our encouraging research results, all these efforts pay off.

ACKNOWLEDGMENT

This research is supported by the Engage Grants (EGP 529904-18) from the Natural Sciences and Engineering Research Council of Canada (NSERC) with McGill REB file number: 146-0818.

REFERENCES


### Appendix A

**Binary Classification Task Performance**

Table V: Binary classification performance on test set (0, 1 indicate the F1-score on the benign class and the anomalous class respectively. \( m \) indicates the micro F1-score.)

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>F-score</th>
<th>0-F1Score</th>
<th>0-Recall</th>
<th>0-Precision</th>
<th>0-F1Score</th>
<th>1-F1Score</th>
<th>1-Recall</th>
<th>1-Precision</th>
<th>1-F1Score</th>
<th>0-Recall</th>
<th>0-Precision</th>
<th>0-F1Score</th>
<th>1-Recall</th>
<th>1-Precision</th>
<th>1-F1Score</th>
<th>Average</th>
<th>Average</th>
<th>Average</th>
<th>MDAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN(s)</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>L-SVM</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>R-SVM</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>Ada</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>NB</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>QDA</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

### Appendix B

**Three-class Classification Task Performance**

Table VI: Three-Class classification performance on test set (0, T1, T2, indicate the F1-score on the benign class, T1 anomalous class, and T2 anomalous class respectively. \( m \) indicates the micro F1-score.)

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>F-score</th>
<th>0-F1Score</th>
<th>0-Recall</th>
<th>0-Precision</th>
<th>0-F1Score</th>
<th>1-F1Score</th>
<th>1-Recall</th>
<th>1-Precision</th>
<th>1-F1Score</th>
<th>0-Recall</th>
<th>0-Precision</th>
<th>0-F1Score</th>
<th>1-Recall</th>
<th>1-Precision</th>
<th>1-F1Score</th>
<th>Average</th>
<th>Average</th>
<th>Average</th>
<th>MDAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN(s)</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>L-SVM</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>R-SVM</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>Ada</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>NB</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>QDA</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>