Self-supervised learning for Event Sequences on synthetic task of next item prediction (GPT-approach based module)

Egor Fadeev [1] **Alexander Ganibaev** [1] **Matvey Lukyanov** [1] **Aleksandr Yugay** [1]

Abstract

Self-supervised learning is a powerful technique for leveraging large amounts of unlabeled data to improve the performance of machine learning models, particularly in domains where labeled data is scarce or expensive to obtain. In this project, we focus on self-supervised learning applied to event sequences, specifically transaction data, and explore the use of two different pre-training approaches for obtaining embeddings: classical representations and contrastive representations. We demonstrate that both embedding models are viable for downstream tasks, specifically in predicting the next merchant category code (MCC) of a transaction. Our experiments show that the pre-trained contrastive embeddings perform better on less stable data, while the pre-trained representation embeddings suit better for homogeneous transaction data. These findings can help guide the selection of pre-training approaches for transactional data, and our work opens up opportunities for further exploration of self-supervised learning in other domains.

Github repo: link

1. Introduction

Self-supervised learning is a type of machine learning which is applicable in situations where there is an abundance of unlabeled data and a relatively small amount of labeled data. This is because self-supervised learning algorithms can use unlabeled data to create their own supervision signals, rather than relying on labeled data. Overall, it is a powerful technique for leveraging large amounts of unlabeled data to improve the performance of machine learning models, particularly in domains where labeled data is scarce or expensive to obtain. Self-supervised learning is particularly useful in natural language processing and computer vision tasks, where large amounts of unannotated data are readily available. Another area in which this approach is applicable is the banking sphere and transaction data. Having large amounts of data collected from their customers' activities, banks can use them to study the patterns of the customer behavior in order to enrich the existing labeled datasets and to use them for further downstream tasks.

One specific ML paradigm that can be used with self-supervised learning is representation learning. In this approach, one may use embeddings that represent the relevant patterns of the data points or sequences and map them to a low-dimensional space of fixed-length vectors. Moreover, pre-trained embeddings are often used as comprehensive input features for ML models. Such approach allows to avoid extensive feature engineering and makes the usage of such data possible without deep knowledge of the specific domain. In certain cases, the usage of pre-trained embeddings may also make the training of the downstream models more efficient in terms of time and memory usage.

One of the sub-approaches of representation learning is contrastive learning. In this method, models are trained to differentiate between similar and dissimilar pairs of objects. In other words, the goal of contrastive learning is to learn a representation $x \mapsto M(x)$ which brings similar objects, known as positive pairs, closer to each other in the embedding space, and dissimilar object (negative pairs) — farther, by the means of a specific construction of a loss function that penalizes the model for assigning high similarity scores to negative pairs and vice versa. Specifically, this type of models is perfectly applicable to transaction data, as such datasets usually include a large amount of structured data such as time stamps, transaction amounts, merchant categories, etc. Thus, such type of learning the embeddings is convenient for identifying the patterns for further usage in such downstream tasks as fraud detection or customer segmentation. Thus, in the first case, this approach can be used to obtain the representations of normal transaction patterns, and then to identify the ones that significantly deviate from these patterns as potential fraud. In the case of

[1]Skolkovo Institute of Science and Technology, Data Science, Moscow, Russia. Correspondence to: Aleksandr Yugay <Aleksandr.Yugay@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

customer segmentation, one may group bank customers by their similarity according to transaction patterns. Further, it can be useful for targeted marketing, personalized product recommendations, and customer retention tasks.

In our project, we have obtained the transactional embeddings from two publicly available datasets by using both classical representations and contrastive representations. Thus, we have addressed a downstream task of the prediction of the next merchant category code (MCC) using two different pretraining approaches. The training procedures of both consist of two stages:

1. **Unsupervised stage**: pre-training the embeddings
2. **Supervised stage**: performing a certain downstream prediction task using the obtained representations.

The main contributions of this project are as follows:

- We implement next transaction prediction on transaction datasets by solving the representation loss minimization problem.
- We implement next transaction embedding prediction on transaction datasets by solving the contrastive loss minimization problem.
- We demonstrate that both embedding models are viable for the downstream tasks.

2. Related work

Self-supervised learning for event sequences in machine learning as an emerging area of research has shown potential in enabling machines to learn from unlabeled datasets, reducing the complexity and cost of model development. Recent papers have proposed innovative approaches, exploring different techniques for extracting meaningful representations and suggesting novel architectures that capture the temporal dependencies present in event sequences. This field is diverse and promising, with many approaches to be explored, and in this section, we review some of the most relevant and promising papers in this area.

One promising approach based on contrastive learning is described in the paper (Babaev et al., 2022). They propose CoLES, a self-supervised method based on contrastive learning for embedding discrete event sequences. CoLES uses a novel data augmentation algorithm to generate subsequences of observed event sequences, which are used as different high-dimensional views of the sequence for contrastive learning. CoLES representations can be used in supervised domain-related tasks, such as fraud detection, or fine-tuned for out-of-domain tasks. The authors demonstrate the effectiveness of CoLES on several public event sequence datasets, including financial transactions, and show that CoLES outperforms other methods on different downstream tasks.

The authors of (Oord et al., 2018) propose a universal unsupervised learning approach, Contrastive Predictive Coding (CPC), to extract useful representations from high-dimensional data. CPC uses a probabilistic contrastive loss to induce the latent space to capture information that is most useful to predict future samples. The authors demonstrate that CPC achieves strong performance on four distinct domains: speech, images, text, and reinforcement learning in 3D environments.

In (Babaev et al., 2019), the authors propose a deep learning approach, specifically an Embedding-Transactional Recurrent Neural Network (ET-RNN), to compute credit scores based on the customer's credit and debit card transaction history. The proposed method outperforms traditional models and has several advantages, including the ability to make credit loan decisions quickly and without requiring any additional input from the client. The paper highlights potential disruptions to the retail banking loan industry and raises the issue of interpretability in black-box models.

Another relevant paper is (Falcon and Cho, 2020), which presents a conceptual framework for contrastive self-supervised learning (CSL) in computer vision. The authors characterize CSL approaches in five aspects: data augmentation pipeline, encoder selection, representation extraction, similarity measure, and loss function. They analyze three leading CSL approaches in computer vision: AMDIM, CPC, and SimCLR, and show that despite different motivations, these approaches are special cases under their framework. The authors design a new CSL approach called Yet Another DIM (YADIM), which achieves competitive results on CIFAR-10, STL-10, and ImageNet and is more robust to the choice of encoder and representation extraction strategy.

These papers demonstrate the significance and potential of deep learning approaches on fine-grained transactional data for credit scoring applications in the banking industry, while also highlighting the effectiveness of contrastive learning-based methods and the potential of unsupervised learning to improve the performance of self-supervised learning methods.

3. Algorithms and Models

Let $h_{1:n_i} = \{x_{ij}\}_{j=1}^{n_i}$ denote a sequence of i-th user's transactions, where $x_{ij} \in \mathrm{R}^d$ is a feature vector representing j-th transaction of i-th user (x_{ij1} — MCC of the transaction), n_i — number of transactions in i-th user's history. The downstream task that we are solving is the prediction of the MCC of the next user's transaction, i.e. we want to build a classification model:

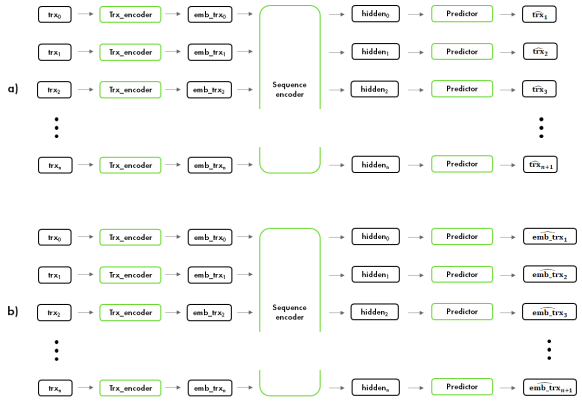

Figure 1. Self-supervised pretraining scheme: a) representation approach; b) contrastive approach

$$f(h_{1:n_i}) = \hat{x}_{i(n_i+1)1}$$

Such model consists of three parts: transaction encoder ϕ, sequence encoder ψ and predictor g. Transaction encoder ϕ maps each transaction to its latent representation $z_{ij} = \phi(x_{ij})$, then the sequence of latent representations is passed to sequence encoder ψ, yielding latent representation of the whole sequence $q_i = \psi\left(\{z_{ij}\}_{j=1}^{n_i}\right)$. The latter is used for final prediction $\hat{x}_{i(n_i+1)1} = g(q_i)$.

The idea of self-supervised learning is to pretrain transaction encoder ϕ on some intermediate task and then apply it to solve the downstream task. In our reasearch we compare two different approaches to pretrain embeddings: 1) "representation" — train model to predict the next transaction; 2) "contrastive" — train model to predict the embedding of the next transaction.

$$g_{repr}\left(\psi_{repr}\left(\{\phi_{repr}(x_{ij})\}_{j=1}^{n_i}\right)\right) = \hat{x}_{i(n_i+1)} \quad (1)$$

$$g_{contr}\left(\psi_{contr}\left(\{\phi_{contr}(x_{ij})\}_{j=1}^{n_i}\right)\right) = \widehat{\phi_{contr}(x_{i(n_i+1)})} \quad (2)$$

To compare these approaches for embedding learning we first train two different models on the corresponding self-supervised task, then freeze weights of obtained ϕ_{repr} and ϕ_{contr}, and subsequently use them $\left(\overline{\phi_{repr}}\right.$ and $\left.\overline{\phi_{contr}}\right)$ to solve the downstream task of next transaction's MCC prediction.

$$f_1(h_{1:n_i}) = g_1\left(\psi_1\left(\{\overline{\phi_{repr}}(x_{ij})\}_{j=1}^{n_i}\right)\right) = \hat{x}^{(1)}_{i(n_i+1)1} \quad (3)$$

$$f_2(h_{1:n_i}) = g_2\left(\psi_2\left(\{\overline{\phi_{contr}}(x_{ij})\}_{j=1}^{n_i}\right)\right) = \hat{x}^{(2)}_{i(n_i+1)1} \quad (4)$$

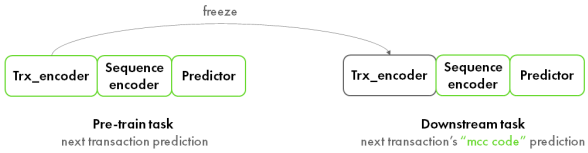

Figure 2. Transfer of the pretrained transaction encoder to solve the downstream task

Finally, we measure the quality of classification models with weighted f1-score due to high class imbalance for MCC in transaction datasets which we used.

In our framework all transactions are represented by two categorical features: MCC and the index of the current amount bin. Transaction encoder ϕ consists of two embedding modules and one linear layer (Fig. 3). In the general case, however, transaction encoder can be extended to take any tabular representation of transaction with categorical and continuous features.

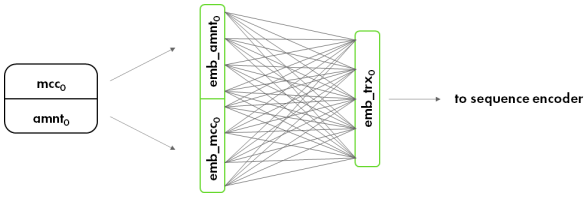

Figure 3. Scheme of transaction encoder

As for sequence encoder ϕ, RNN or Transformer architectures can be used. In particular, in our experiments we used 1-layer GRU, where the last hidden state of each sequence was passed to head predictor to solve the corresponding task.

Head predictor g is a simple MLP, its architecture is depicted in the Fig. 5. The difference between g_{repr} and g_{contr} is that the former actually has multiple heads, each predicts category of the corresponding feature of the next transaction (number of neurons in the output layer is equal to number of categories in the feature), while the latter should predict the embedding of the next transaction, so the number of neurons in the output layer is equal to the dimension of transaction encoder's output. In addition, output of g_{contr} is L2 normalized. Predictors g_1 and g_2 are similar, they have only one head to predict the MCC of the next transaction, so output dimension is equal to the number of unique MCC's.

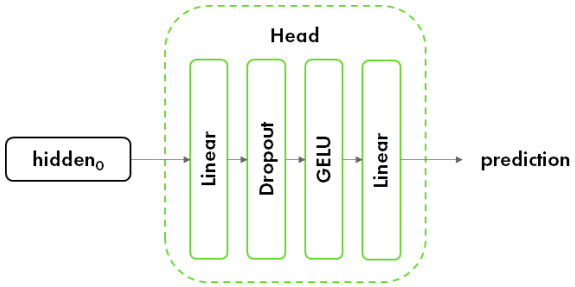

Figure 4. The architecture of head MLP

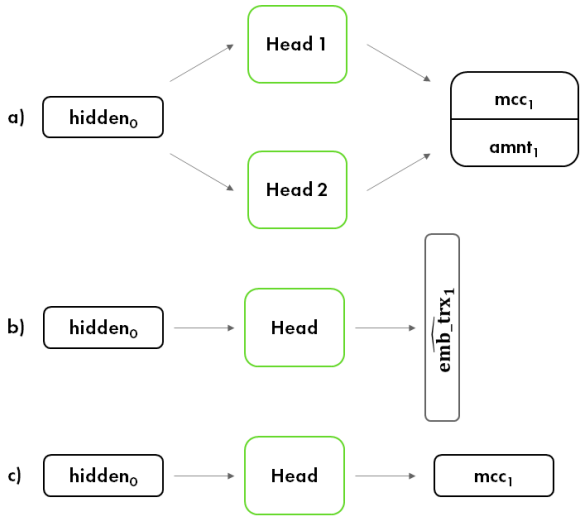

Figure 5. Different outputs of models: a) g_{repr} has two heads, each predicts class of the corresponding feature; b) g_{contr} predicts embedding of the next transaction; c) g_1 and g_2 (downstream models) predict next transaction's MCC

Predictors g_1, g_2 and each head of g_{repr} solve classification task, therefore Cross Entropy Loss (5) is used to train these models. As for g_{repr}, total loss is just a sum of individual losses corresponding to each classification task.

$$\mathcal{L}_{CE}(X) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{\exp(l_{ic})}{\sum_{j=1}^{C}\exp(l_{ij})}\right) \tag{5}$$

where C — number of classes, $l_{ij} \in \mathbb{R}$ — logit of class j predicted by the model for the observation i, c — index of true class, m —- batch size.

For g_{contr} contrastive loss is used, because it prevents the model from trivial solutions, when all transactions are mapped to the same embedding vector. Query Softmax Loss (6) makes embedding of a transaction closer to its prediction and contrasts it with predictions for other transactions.

$$\mathcal{L}_{QS}(X) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{\exp\left(\tau z_i^T l_{i1}\right)}{\sum_{j=1}^{k+1}\exp(\tau z_i^T l_{ij})}\right) \tag{6}$$

where $z_i \in \mathbb{R}^d$ — embedding of the i-th target transaction; $l_{i1} \in \mathbb{R}^d$ — prediction of the i-th target transaction's embedding (positive example); $l_{ij} \in \mathbb{R}^d, j \in \{2, \ldots, k+1\}$ predictions of other target transactions' embeddings (negative samples); k — number of negative samples used for each target; τ — temperature hyperparameter; m — batch size. Negative samples for each target transaction embedding are sampled from predicted embeddings in the batch, excluding prediction for itself.

4. Experiments and Results

4.1. Datasets

Age group prediction competition[1] Each transaction includes the client id, date, MCC, and amount being charged. For simplicity reasons this dataset will be marked as **SBER**. Amount feature is discretized on 10 parts. Originally, the dataset of 44M anonymized credit card transactions representing 50K individuals was used to predict the age group of a person. However, this target is not a good fit for the aim of our research, since age is quite stable and it is almost meaningless to take it as a target in the next token prediction task. Hence, MCC is used as a target: in both pretraining and main part the next MCC code of each client MCC sequence is predicted. Additionally, amount is predicted in pretraining part.

The MCC feature has 202 classes, which are highly unbalanced, which can be seen in the figure below.

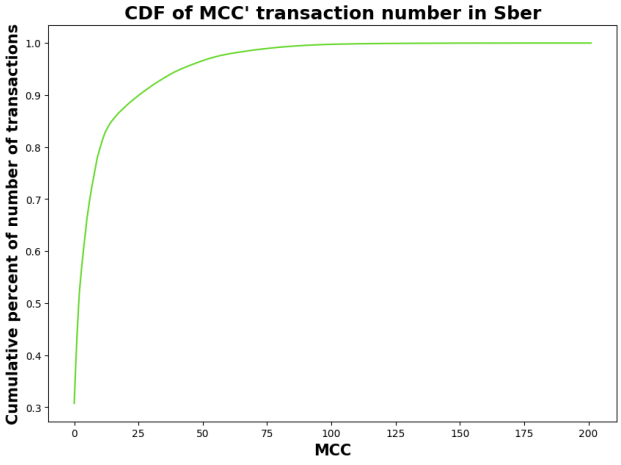

Figure 6.

[1] https://ods.ai/competitions/sberbank-sirius-lesson

Churn prediction competition [2]. Each transaction is characterized by date, MCC, client id, amount, currency, channel and category, but only 4 first features are used by us. For simplicity reasons this dataset will be marked as **ROSBANK**. Amount feature is discretized on 10 parts. Originally, the dataset of 1M anonymized card transactions representing 10K clients was used to predict churn probability. But again, the target was changed for the same reason as in Sber dataset. So, as before, MCC and amount is used as a target in pretraining part; and MCC is predicted in main part.

The MCC feature has 344 classes, which are highly unbalanced, which can be seen in the figure below.

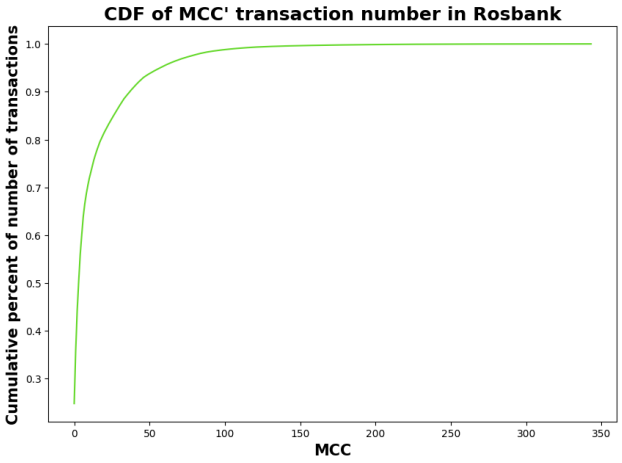

Figure 7.

4.2. Dataset split

In both datasets the scheme to splitting is the same: the split is done by clients, 80% of clients are in the train, 10% are in validation and 10% are in testing parts. For each client for each token the next token is predicted.

4.3. Hyperparameters

For correct model comparison all models have the same architecture:

In tables below we show model hyperparameters, which were chosen for representation and contrastive learning. Both models are highly dependent on hyperparameters, so careful manual tuning was done, so that both models work as intended.

[2]https://boosters.pro/championship/rosbank1/overview

Table 1. Architecture

HYPERPARAMETER	VALUE
MCC EMBEDDING SIZE	16
AMOUNT BIN EMBEDDING SIZE	16
LINEAR PROJECTION (TRANSACTION ENCODER OUTPUT SIZE)	32
RNN HIDDEN SIZE	32
HEAD HIDDEN SIZE	64
HEAD DROPOUT	0.1

Table 2. Hyper-parameters for Representation Learning

DATASET	SBER	ROSBANK
TOTAL STEPS	10000	20000
MAX EPOCHS	50	100
LEARNING RATE (PRETRAIN)	0.1	0.1
LOSS (PRETRAIN)	CE	CE
PATIENCE (PRETRAIN)	5	5
LEARNING RATE	0.01	0.01
LOSS	CE	CE
PATIENCE	5	5

Table 3. Hyper-parameters for Contrastive Learning

DATASET	SBER	ROSBANK
TOTAL STEPS	10000	20000
MAX EPOCHS	50	100
LEARNING RATE (PRETRAIN)	0.1	0.1
LOSS (PRETRAIN)	QS	QS
PATIENCE (PRETRAIN)	3	3
LEARNING RATE	0.01	0.01
LOSS	CE	CE
PATIENCE	5	5
LOSS TEMPERATURE	10	10
NEGATIVE COUNT	10	10

4.4. Training performance

All neural networks were trained on a single NVIDIA GeForce RTX 3080 Laptop GPU card. We ran each model 50 times for the accurate comparison of model performance. On Sber datasets each run took about 5 minutes, and about 1 minute on Rosbank.

4.5. Baselines

As baselines two naive next tokens predictors were used: take most popular token from the transaction sequence of each user **POP** and take the last known token **PREV**

4.6. Results

As was already mentioned, weighted f1-score is used for model performance comparison because of class imbalance.

In the table below the results of baselines and our target models can be seen. We provide the source code for all the experiments on the datasets described in this report.

Table 4. Results

DATASET	SBER	ROSBANK
POP	0.143	0.091
PREV	0.144	0.234
NO PRETRAINING	0.262	0.231
	(0.002)	(0.010)
REPRESENTATION LEARNING	0.200	**0.236**
	(0.017)	(0.006)
CONTRASTIVE LEARNING	**0.264**	0.234
	(0.002)	(0.006)

Mean and standard deviation of
weighted f1-scores on 50 runs is shown

The Prev naive baseline performed reasonably well in Rosbank and bad in Sber, from which it can be concluded that Rosbank has much less variability in MCC within short range of times. Speaking about main results, it was observed that different types of pre-training yield varying levels of performance on different datasets. Models that utilize pre-trained contrastive embeddings exhibit slightly better performance on less stable Sber dataset (0.264), while models that use pre-trained representation embeddings are unable to outperform non-pretrained predictions. On the other hand, the pretrained model with the representation loss marginally outperforms prediction results on more stable Rosbank dataset (0.236). For the Rosbank dataset in particular, our results lie in one standard deviation from the results of the no pretraining model.

5. Conclusion

In this project, we focus on self-supervised learning applied to event sequences, where we aim to capture complex information from transaction data by mapping it into a fixed-length vector space using embedding pre-training. We accomplish this through optimizing both representation loss and contrastive loss. We conclude that both algorithms can be used to produce embeddings in an efficient way that can further be used for various downstream tasks.

We experimentally demonstrate how these embeddings can be utilized in downstream tasks, specifically in predicting the next MCC (item). We noticed that each type of pre-training outperforms the baselines by a fairly insignificant margin on a certain dataset. The model with pre-trained contrastive embeddings shows better performance on less stable data, whereas the model with pre-trained representation embeddings cannot even beat the prediction without pre-training. At the same time, the representation loss shows better prediction quality on a more stable dataset. Judging by it, we can conclude that the usage of pre-trained representation embeddings is a fair choice for homogeneous transaction data, whereas the pre-trained contrastive embeddings suit better for the data with more vague and complex patterns.

Finally, in our further work we will test even more hypothesis within this approach. Specifically, we want to train the models on more datasets and for larger amount of downstream tasks.

References

[1] Babaev, D., Ovsov, N., Kireev, I., Ivanova, M., Gusev, G., Nazarov, I., & Tuzhilin, A. (2022, June). CoLES: Contrastive Learning for Event Sequences with Self-Supervision. *In Proceedings of the 2022 International Conference on Management of Data* (pp. 1190-1199).

[2] Babaev, D., Savchenko, M., Tuzhilin, A., & Umerenkov, D. (2019, July). Et-rnn: Applying deep learning to credit loan applications. *In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2183-2190).

[3] Falcon, W., & Cho, K. (2020). A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104.*

[4] Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748.*

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Egor Fadeev (20% of work)

- Preparing the GitHub Repo
- Preparing the Section 1 and 5 of this report

Alexander Ganibaev (20% of work)

- Reviewing literature on the topic (4 papers)
- Preparing the GitHub Repo
- Preparing the Section 2 of this report

Matvey Lukyanov (30% of work)

- Conducting experiments
- Preparing EDA
- Preparing the GitHub Repo
- Preparing the Section 4 of this report

Aleksandr Yugay (30% of work)

- Coding all models and preparing pipelines
- Preparing the GitHub Repo
- Preparing the Section 3 of this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: 10% in models.py are our modifications of classes taken from Pytorch-LifeStream, and dataset.py contains minor modifications of a Pytorch-LifeStream class. All other code in notebooks and main.py we consider our own.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 3

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look abstract

4. A complete description of the data collection process, including sample size, is included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 4.1

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 4.1

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 4.2

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 4.3

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: look section 4.4

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: look section 4.4

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: look section 3

12. Clearly defined error bars are included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: look section 4.6

13. A description of the computing infrastructure used is included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: look section 4.4