National Research University Higher School of Economics Faculty of Computer Science

Programme ‘Master of Data Science’

MASTER’S THESIS

SAINT based architecture for banking data

Student Niyazi Garagashli

Supervisor: Msc, Middle Data Scientist, Eldar Ganbarov

Moscow, 2021

Abstract

SAINT (the Self-Attention and Intersample Attention Transformer) is a transformer-based deep learning model proposed by Somepalli et all, that strives to outperform classical machine learning algorithms such as XGBoost, LightGBM, etc. on tabular data.

In this work, we set out to test the feasibility of using SAINT with pretraining on a real-life banking dataset and compare its performance to de-facto industry-standard gradient boosting algorithms, such as XGBoost and LightGBM. We explore the advantages of Self-Supervised Learning with SAINT in predicting default for the bank client in a specific segment of bank clientele, as well as in a more general setting of limited availability of labeled samples.

Acknowledgment

I would like to express my gratitude to my eternally patient supervisor, Eldar Ganbarov, who have had continuously supported me through this whole ordeal. I would also like to thank Sber (formerly known as SberBank) for providing me with the data and the concept of data segmentation.

Contents

[Abstract 2](#_Toc90904448)

[Acknowledgment 2](#_Toc90904449)

[Contents 3](#_Toc90904450)

[Introduction 4](#_Toc90904451)

[Objectives 4](#_Toc90904452)

[Dataset 4](#_Toc90904453)

[Metrics 5](#_Toc90904454)

[Models 5](#_Toc90904455)

[Related work 6](#_Toc90904456)

[Other transformers for tabular data 6](#_Toc90904457)

[Self-supervised learning 6](#_Toc90904458)

[Non-segmented dataset experiment 8](#_Toc90904459)

[Segmented dataset experiment 9](#_Toc90904460)

[Pretraining on part of the segment 9](#_Toc90904461)

[Pretraining on other segments 9](#_Toc90904462)

[Contrastive pretraining vs. classical transfer learning 10](#_Toc90904463)

[Conclusion 11](#_Toc90904464)

[Limitations 12](#_Toc90904465)

[Bibliography 13](#_Toc90904466)

[Appendix A: dataset 15](#_Toc90904467)

[Data overview 16](#_Toc90904468)

[Data manipulation 16](#_Toc90904469)

[Appendix B: SAINT model 17](#_Toc90904470)

[Architecture overview 17](#_Toc90904471)

[Contrastive pretraining 17](#_Toc90904472)

[Modifications 18](#_Toc90904473)

[Annex C: Additional experiment 19](#_Toc90904474)

[Dataset 19](#_Toc90904475)

[Experiment 19](#_Toc90904476)

Introduction

Tabular data, despite being ubiquitous data representation in financial institutions, is the area where deep learning struggles the most.

SAINT architecture [[1]](#SAINT) is an attempt to change that. It uses attention over both rows and columns with enhanced embedding generation. SAINT also introduces a contrastive pretraining method to allow the model to learn a better representation even when labeled data is scarce. Pretraining, in general, is a powerful tool that can, in theory, be deciding factor in whether to use neural networks over classical machine learning models.

It is important to note that creating a labeled dataset can be hard to achieve. It can be cost-prohibitive, time-prohibitive, human resource prohibitive, or a combination of these and other factors. When the amount of labeled data is restrictively low, conventional machine learning methods may be underachieving in terms of metrics and lead to poor generalization properties of a model. This is where deep learning may have an upper hand, as it has been continuously shown in other domains of machine learning, such as NLP tasks or Computer Vision tasks, in which transfer learning is a standard procedure.

With SAINT, there are 2 ways to pretrain a model: contrastive pretraining introduced in the paper and model-agnostic approach where all layers except the head (-s) are frozen after initial training, and then only head (-s) finetuned on additional data.

We test both these approaches but make emphasis on the novel contrastive method.

Objectives

Our work aims to prove the benefits of SAINT application to the following problems:

1. Few labeled samples and a lot of unlabeled samples
2. Few labeled samples within a specific segment of the dataset and a lot of labeled samples from other segments of the same dataset

Dataset

This dataset was provided by Sber. It contains real-world data that has been obfuscated and de-personalized for this experiment. It is not publicly available and cannot be shared in its entirety. **Appendix A** contains a complete description of all the fields within the dataset and what features are used in the experiments.

The label is “*default\_12m*”, which corresponds to whether the bank client has defaulted in the last 12 months. It is binary and takes either 1 or 0 as a value. This is the target our experiment tries to predict.

This dataset is unbalanced with 94% of it having a target set as 0 and 6% as 1.

Metrics

Since the target variable is binary and the dataset is unbalanced, AUROC is the metric of choice, because it allows to better capture how well the model generalizes on highly skewed datasets. It is also considered to be an industry selection for metrics of assessing models predicting default scenarios in financial institutions. We also acknowledge that AUROC can be misleading for cases where label distribution is too skewed, e.g. having one class at 1% or less. That is, however, not the case for our dataset.

Models

Throughout the experiments following models were used: SAINT as a candidate, XGBoost [[2]](#XGBoost), and LightGBM [[3]](#LightGBM) as competitors.

SAINT was recreated from GitHub of the original paper’s author, no changes to the architecture were made on our part. The only changes done are to utility functions to allow for adaptation to our specific dataset. See **Appendix B** for details on modifications.

Both XGBoost and LightGBM were initialized as is, no hyperparameter tuning was attempted. XGBoost employed the “*gpu\_hist*” boosting method to benefit from GPU provided in the Google Colab environment.

Related work

Other transformers for tabular data

Applying transformer style of architecture has been done before, however, initially, this has been done to continue the work on natural language tasks. TABERT [[8]](#TABERT) was developed based on BERT with the intent of using it for such tasks.

Transformer for more general tabular data, TabTransformer [[9]](#TabTransformer), employed titular architecture to learn contextual embeddings of exclusively categorical features. Numerical features are then concatenated to that embedding and used as an input to MLP. Essentially, numerical features are not utilized in the transformer itself, as these features are not passed through the self-attention block. Thus, any possible information on correlations between two types of features is lost.

A more similar conceptually to SAINT are TABBIE [[10]](#Tabbie) and MSA transformer [[11]](#MSATransformer), both of which use axial attention in relation to tabular data – attending to data point within one feature and features to each other. The difference in SAINT’s approach is that the attention mechanism is hierarchical in nature. Intersample and self-attention happen in order, while in the aforementioned implementations it happens at the same time.

Self-supervised learning

Self-supervised learning is a division of unsupervised learning in that it does not require labeled data. However, unsupervised learning usually refers to tasks like clustering or dimensionality reduction, while self-supervised learning is about learning the pretext of a following downstream task. To the best of our knowledge, this idea was first proposed by Jürgen Schmidhuber in 1989 [[14]](#jurgen1989).

As noted by Yann LeCun, one of the most important people in the development of the Deep Learning area and author of Convolutional Neural Networks, during his keynote speech at AAAI 2020, “basically, it’s the idea of learning to represent the world before learning a task. This is what babies and animals do. We run about the world; we learn how it works before we learn any task. Once we have good representations of the world, learning a task requires few trials and few samples” [[12]](#AAAI2020).

Transformers have achieved great results at leveraging self-supervised learning. By masking words and trying to fill in the gaps models can reach an understanding of the language without human supervision. It is important to note that such “understanding” has little to do with how humans understand and use language, but it shows the ability of a model to build its own representation of language internal logic and structure. All without labels and human input.

Self-supervision allows for more stable and robust models [[13]](#robustness_ssl) and helps to overcome dataset restrictions to a degree, which makes it a very promising and evolving area of research. Most of such research, however, concentrates on image\video and natural language tasks.

Some of such methods like “denoising” first tried on MNIST visual dataset [[15]](#denoising) and replaced token detection as used by TabTransformer, are used in the SAINT model’s pretraining approach.

VIME [[18]](#VIME) uses 2 pretext tasks for Self-Supervised Learning for tabular data specifically: feature vector estimation and mask vector estimation. It simultaneously tries to recover the corrupted input sample and estimate the maks vector applied to it. In other words, it predicts what features have been corrupted and then predicts their original values. VIME can, therefore, construct a representation of the underlying data.

A different approach, the one not dependent on “denoising”, was recently proposed by Talip Uçar et al. in SubTab paper [[17]](#SubTab). Authors argue that instead of trying to recover information from corrupted input in an autoencoder setting, it can be far more beneficial to reconstruct the data from the subset of its features. Inspired by the success of “cropping” augmentation in the image domain, SubTab framework does idealistically the same trick. It creates multiple “views” consisting of a subset of features and maps them to a latent space. By recreating data from those views, a data representation is achieved. The work shows that authors have achieved SOTA benchmark performance on multiple datasets.

Non-segmented dataset experiment

To simulate a situation where there is a lot of data exists, but very little of it is labeled, the dataset has been separated into two: one contained 90% of all data, the other contained 10% that is left. Stratification ensured that both classes are in the same proportion, as in the original dataset.

This amounted to 29156 samples in the pretraining dataset and 3239 samples in finetuning dataset. Samples are not overlapping between datasets to ensure there is no leakage.

When pretrained on the majority of data, SAINT achieves an AUROC metric of 0.673 with finetuning on just 10% of initial data.

It shows slightly better results than straight training on the same 10% of data and it also outperforms competitors, as shown on results table below:

|  |  |
| --- | --- |
| **Model** | **AUROC** |
| SAINT pretrained on 90% of the dataset and finetuned on 10% of the dataset | **0.673** |
| SAINT trained on 10% of dataset | 0.665 |
| XGBoost on 10% of dataset | 0.669 |
| LightGBM on 10% of dataset | 0.651 |

Table : non-segmented dataset experiment results

Segmented dataset experiment

Segment here is defined as follows: all samples, where feature “*Capital (at the end of the last year)*” is equal or above 75th percentile. In other words, this is a segment of clients with the most capitalization. In general, Sber employs tactics of using a separate model for separate segments to ensure better insights in particular groups of clients.

It follows that such clients represent a small group of bank clients and won’t have a lot of data samples in the dataset.

Our segment consists of 4313 samples, which are then separated into train and test datasets in proportion 70/30, resulting in 3019 data points in train and 1294 in test.

Pretraining on part of the segment

Separating segment dataset further into 75\25 split, SAINT was first pretrained on 75% of the segment and then finetuned on the rest of the segment data. This allowed model to score 0.663 AUROC, which is already an improvement on the same architecture trained on just that 25%. Competing models scored slightly higher.

Pretraining on other segments

Other segments here are any data points that are not part of the selected segment, hard-capped at 10000 rows.

When the SAINT model gets finetuned on just 25% of the segment dataset following pretraining on other segments, it achieves a 0.761 AUROC score with an accuracy being 98.841. These are the highest results achieved during experimentation with any model.

Contrastive pretraining vs. classical transfer learning

All pretraining done above was with contrastive pretraining, as described in the original paper. It employs augmentation techniques such as “cut-mix” and tasks such as “contrastive” and “denoising”.

SAINT has 3 heads, MLP for each attention and MLP for classification. Freezing all 3 or just MLP for classification has little effect on final scores. However, the new method of pretraining outperforms traditional transfer learning, as shown below:

|  |  |
| --- | --- |
| **Model** | **AUROC** |
| SAINT pretrained on 75% of segment, finetuned on 25% of segment | 0.663 |
| SAINT pretrained on other segments, finetuned on 25% of segment (contrastive) | **0.761** |
| SAINT pretrained on other segments, finetuned on 25% of segment (transfer learning) | 0.698 |
| SAINT trained on 25% of segment, no pretraining | 0.600 |
| XGBoost on 25% of segment | 0.714 |
| LightGBM on 25% of segment | 0.720 |

Table : segmented dataset experiment results

Conclusion

Pretraining, and especially contrastive one, has been proven to allow our candidate model to outperform its competitors. Moreover, in some cases, pretraining helped to achieve almost the same results, as if the model was trained on the whole bulk of data. For example, SAINT trained on the whole segment has AUROC of 0.782, while SAINT pretrained on other segments and finetuned on only 25% has achieved AUROC of 0.761. The difference between these scores is not negligible, but it may be within an acceptable range depending on the business task.

Generally speaking, the value that pretraining can bring into the model depends on the amount of data available for training. The less labeled training samples there are, the more value can be extracted from pretraining on non-labeled samples. Conversely, if one has a significant amount of labeled data, pretraining does not improve the final model in a discernable way.

In most cases, SAINT does outperform vanilla, without hyperparameters optimization, gradient boosting models. But the supremacy of SAINT becomes even more radical when training data is small. Contrastive pretraining allows SAINT to have a deeper capacity for generalization than competing models since they are not capable of pretraining at all. Thus, when SAINT does benefit from learning internal relationships between features through pretraining, other models can only learn from labeled samples. Overall, our experiments showed that SAINT performs on par with competing models when there is an abundance of data, but outperforms them with contrastive pretraining applied when there is little data.

In an industry where obtaining a labeled dataset of a big enough size can be a challenging task, using contrastive pretraining of SAINT can be very beneficial. It holds true for situations where there is a lot of data, but very little of it is labeled. Even in cases where such a model cannot be deployed to critical production areas (see “Limitations” section), such an approach can be employed to test the hypothesis and assess the potential value of continuing to gather labeled datasets. It can also be used as a “placeholder” model when time is limited and results are required as soon as possible, all the while labeled data continues to be gathered for continuous training and retraining of SAINT or other models.

Furthermore, contrastive pretraining as a concept is not something confined exclusively to SAINT architecture. It can be applied to other deep learning models as well. With the continued development of neural networks in application to tabular data, such methods can be appealing competitive advantage over classical machine learning approaches in fields where obtaining enough labeled data can be cumbersome.

It is also worth pointing out that SAINT has a fair share of tunable hyperparameters. Due to time constraints, we haven’t been able to test all of them and their efficacy on a particular dataset. There are also more general components that can be substituted, such as loss function, optimizer, and preprocessing encoders.

Limitations

The banking industry works with highly sensitive data that influences the lives of many individual people, as well as companies. If any model is used to make a decision that can directly affect someone’s livelihood, this decision needs to be transparent and unbiased, because banking institution is held accountable for said decision.

Thus, a classical machine learning algorithm can sometimes be the only appropriate solution despite inferior performance. Gradient boosting methods, while not being strict “glass box” models per se, can still be easily explained through LIME [[5]](#LIME) or Shapley values [[6]](#Shapley).

Deep learning models, on the other hand, cannot be explained with the same ease or degree of certainty. Despite relatively recent advances in Explainable AI (XAI), most of the methods are only suitable for non-tabular data. [[4]](#XAI) While this field is also evolving, at this moment SAINT (or other DL models) are hard to explain, which limits their applicability in the banking sector.

As to the limitations of this work, the dataset we used is relatively small both in data point and in features. A vast dataset, such as the [LendingClub](https://www.kaggle.com/wordsforthewise/lending-club) one, could be a good candidate for further testing, as it has a lot of features, combines data for multiple years, and also has a prediction of the default as a target. Unfortunately, the LendingClub dataset is convoluted and noisy. Given the time constraints, it did not seem probable to perform necessary data exploration to begin the experiment itself. However, theoretically speaking, such datasets can be a good proving ground: classical machine learning techniques struggle with dimensionality this big, while deep learning models can generalize them relatively well.

Bibliography

[1] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, Tom Goldstein, “SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training”, *arXiv preprint, arXiv:2106.01342,* 2021

[2] Tianqi Chen and Carlos Guestrin, “Xgboost: A scalable tree boosting system”, *In Proceedings of the 22nd international conference on knowledge discovery and data mining*, pages 785–794, 2016

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, “Lightgbm: A highly efficient gradient boosting decision tree”, *Advances in neural information processing systems, 30:3146–3154,* 2017

[4] M. Sahakyan, Z. Aung and T. Rahwan, “Explainable Artificial Intelligence for Tabular Data: A Survey”, *IEEE Access, vol. 9, pp. 135392-135422*, *doi: 10.1109/ACCESS.2021.3116481*, 2021

[5] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, “Why Should I Trust You?": Explaining the Predictions of Any Classifier*”, arXiv:1602.04938*, 2016

[6] L. S. Shapley, "A value for n-person games", Contributions to Theory Games, Princeton, NJ, *USA:Princeton Univ. Press, vol. 2, pp. 307-317*, 1953

[7] Bjarte Mehus Sunde, “Early Stopping for PyTorch”, GitHub (<https://github.com/Bjarten/early-stopping-pytorch>), 2020

[8] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel, “Tabert: Pretraining for joint understanding of textual and tabular data”, *arXiv preprint arXiv:2005.08314*, 2020

[9] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin, “Tabtransformer: Tabular data modeling using contextual embeddings”, *arXiv preprint arXiv:2012.06678*, 2020

[10] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer, “Tabbie: Pretrained representations of tabular data”, *arXiv preprint arXiv:2105.02584*, 2021

[11] Roshan Rao, Jason Liu, Robert Verkuil, Joshua Meier, John F Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives, “Msa transformer”, *bioRxiv*, 2021

[12] Yann LeCun keynote speech, AAAI 2020, YouTube (<https://www.youtube.com/watch?v=UX8OubxsY8w>), 2020

[13] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, Dawn Song, “Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty”, *arXiv:1906.12340*, 2019

[14] Jürgen Schmidhuber, “Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments”, *Institut für Informatik Technische Universitat München FKI-126-90\_(revised)*, 1989

[15] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders”, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland*, 2008

[16] S. Moro, P. Cortez and P. Rita, “A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems”, *Elsevier, 62:22-31*, 2014

[17] Talip Ucar, Ehsan Hajiramezanali, Lindsay Edwards, “SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning”, *NeurIPS 2021*, 2021

[18] Jinsung Yoon, Yao Zhang, James Jordon, Mihaela van der Schaar, “VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain”, *Advances in Neural Information Processing Systems, 33*, 2020

Appendix A: dataset

|  |  |
| --- | --- |
| **Field name** | **Description** |
| ar\_revenue | Sales revenue (at the end of the last year) |
| ar\_total\_expenses | Total expenses for the last year |
| ar\_sale\_cost | Cost of sales (at the end of the last year) |
| ar\_selling\_expenses | Selling expenses (at the end of the last year) |
| ar\_management\_expenses | Administrative expenses (at the end of the last year) |
| ar\_sale\_profit | Profit from sales (at the end of the last year) |
| ar\_balance\_of\_rvns\_and\_expns | Balance of income and expenses (at the end of the last year) |
| ar\_profit\_before\_tax | Gross Profit (end of last year) |
| ar\_taxes | Current income tax (at the end of the last year) |
| ar\_other\_profit\_and\_losses | Other income and expenses (at the end of the last year) |
| ar\_net\_profit | Net profit (at the end of the last year) |
| ab\_immobilized\_assets | Fixed assets for the last year |
| ab\_mobile\_current\_assets | Current assets (at the end of the last year) |
| ab\_inventory | Stocks (at the end of the last year) |
| ab\_accounts\_receivable | Accounts receivable (at the end of the last year) |
| ab\_other\_current\_assets | Other current assets (at the end of the last year) |
| ab\_cash\_and\_securities | Cash and cash equivalents (at the end of the last year) |
| ab\_losses | Retained earnings (at the end of the last year) |
| ab\_own\_capital | Capital (at the end of the last year) |
| ab\_borrowed\_capital | Total debt (end of last year) |
| ab\_long\_term\_liabilities | Long-term liabilities (at the end of the last year) |
| ab\_short\_term\_borrowing | Short-term borrowed funds (at the end of the last year) |
| ab\_accounts\_payable | Accounts payable for the last year |
| ab\_other\_borrowings | Other liabilities (at the end of the last year) |
| bus\_age | Business duration |
| ogrn\_age | Term from the moment the PSRN was assigned |
| adr\_actual\_age | Term from the date of registration of the legal address |
| head\_actual\_age | Term since the appointment of the head |
| cap\_actual\_age | Term from the moment of capital installation |
| ul\_staff\_range | The number of employees |
| ul\_capital\_sum | Capital amount |
| ul\_founders\_cnt | Number of shareholders |
| ul\_branch\_cnt | Number of branches |
| ul\_strategic\_flg | A company of strategic importance |
| ul\_systematizing\_flg | Backbone company |

Table : dataset features

Data overview

For an overview of the dataset, please refer to this SweetViz report.



Data manipulation

Following features have been removed from the dataset due to high correlation (>0.8): 'ul\_systematizing\_flg', 'head\_actual\_age', 'cap\_actual\_age', 'ar\_net\_profit', 'ar\_sale\_cost'.

Additionally, the ‘record\_id’ feature was also eliminated since it is unique per record and contains no informational gain.

As part of the data preparation function employed during the training process, each numerical feature is scaled using the StandardScaler module of the Scikit-learn library, and each categorical feature is encoded using the LabelEncoder module of the Scikit-learn library.

Appendix B: SAINT model

Architecture overview

The Self-Attention and Intersample Attention Transformer architecture was proposed as an attempt to overcome the difficulties of training neural networks on tabular data. SAINT projects both categorical and continuous features onto the same dense vector space. This space is used as an input into a transformer encoder that uses attention in two separate ways. “Self-attention” attends to individual features within each data sample, while “intersample attention” is used to enhance the classification of the data sample by relating it to other rows in the table. This is somewhat similar to a nearest-neighbor classification, where distance is learned end-to-end and not fixed.

Diagram, schematic

Description automatically generated

Figure : SAINT architecture, taken from the original paper

Contrastive pretraining

The concept of contrastive pretraining comes from deep learning models for image data, it is used to augment the, often limited, a dataset using “flips” and cropping of initial images to enhance models’ robustness to inputs. To the best of our knowledge (as well as, to the knowledge of SAINT papers authors), such a technique was never implemented for tabular data.

It is a two-step process, in which the CutMix version of every sample in the batch is generated and then after the CutMix version has been embedded, a mixup on the embedding space is performed.

It also employs “denoising” and “replaced token detection” to enhance pretraining.

Modifications

The architecture of SAINT itself was not modified in any shape, way or form, and stays true to the code that was made available by the authors.

Only minor, quality of life changes were done to the data preparation utility function and training cycle.

The list is as follows:

* Additional data preparation function was added to serve pretraining function with slightly differently sliced data loaders
* SAINT\_pretrain function was changed to include validation dataset and early stopping criteria
* Early stopping was also added to the main training loop to combat overfitting

Early stopping implementation [[7]](#EarlyStopping) has shown to be effective when compared to papers original 100 epoch training cycle, as seen below:Chart, line chart

Description automatically generated

Figure : early stopping on 38 epochs out of 100

Annex C: Additional experiment

This section covers additional experiments made on non-primary datasets.

Dataset

The dataset contains data related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. It is available for public use [[16]](#banking_dataset). The description is available at [UCI datasets](https://archive.ics.uci.edu/ml/datasets/bank+marketing).

Experiment

Since there is no clear indication of possible segmentation options, and telemarketing for banking institution is not the same as banking itself, this experiment is for illustration purposes only.

We have trained:

1. SAINT model without pretraining
2. SAINT model with pretraining
3. CatBoost (no hyperparameters tuning) [17]

As expected, fully trained SAINT outperformed CatBoost by a significant margin: 0.924 AUROC score vs 0.892.

However, an important thing to note here is that SAINT pretrained on the whole data and then finetuned on the same data (but with labels) achieved a lower score (0.918) than the same model that has only been trained on labeled data.

In real-life scenarios, there is no practical need to pretrain your model on the same data points then used to finetune it. It can lead to overfitting and poorer generalization capabilities of the model.

In cases where labeled data are abundant, pretraining can be used to make a model more robust by separating datasets randomly in different sets, and pretrain the model on one, finetune it on the other. Unfortunately, due to time constraints, we have not been able to properly test that on this dataset.