**NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS**

**DEPARTMENT OF BUSINESS INFORMATICS**

**PROJECT**

**PREDICTING STUDENTS’ GRADE USING GRAPH NEURAL NETWORKS**

Iurii Gorbunov

Yokutkhon Naimkonova

Liliia Mohammad

Advisor:

Karpov Ilia,

Senior Lecturer of the Faculty of Computer Sciences

Moscow

2023

**Abstract**

The main idea of work is to improve educational progress at both institutional and individual levels. The knowledge tracing usage as a tool for advancing educational research and practice is promising. Predicting student grades and performance on exams could help the universities and students with insights to make changes towards a more productive and successful educational process. In addition, our proposal gives information on the importance of understanding which data most impacts student performance. Moreover, it is highlighted for universities to improve the quality of education and for students to pay attention to specific aspects of their studies.

**Introduction**

*Knowledge tracking* is the sphere where machine and deep learning models chase and evaluate students’ educational progress. The main goal of this sphere is to implement algorithms for modeling students’ knowledge and to improve educational progress not only at the level of the institution but also of the individual, facilitating more rapid changes in the educational process (Piech, C, etc., 2015). Thus, knowledge tracing represents a promising tool for advancing educational research and practice.

On the one hand, universities and educational platforms are interested in ensuring that students' academic performance is at a high level. On the other hand, the motivation of a student to have good grades is much more obvious, but also remains relevant. Predicting that the student's grades or their performance on the exam will be low, universities can offer him help, and the student himself can change his attitude to the educational process towards a more productive, and, ultimately, more successful one.

Moreover, it is worth considering how machine learning models work: based on a variety of different data, the final result is predicted, but not all data are equally influential on the result. In other words, which data have the greatest impact on student performance is also an important question, the answer to which will help universities improve the quality of education, and students pay attention to certain aspects of their studies.

**Literature Review**

The structure of students’ course progress data can be represented similarly to the graph structure: G = (V, E, A). The necessary knowledge for the successful mastering of the program are decomposed into N educational concepts, known as vertices V = {v1 , · · · , vN }, and these concepts have dependency relationships - edges E ⊆ V × V. The degree of dependency between edges is defined by the adjacency matrix A ∈ RN ×N (Nakagawa, H. etc., 2019).

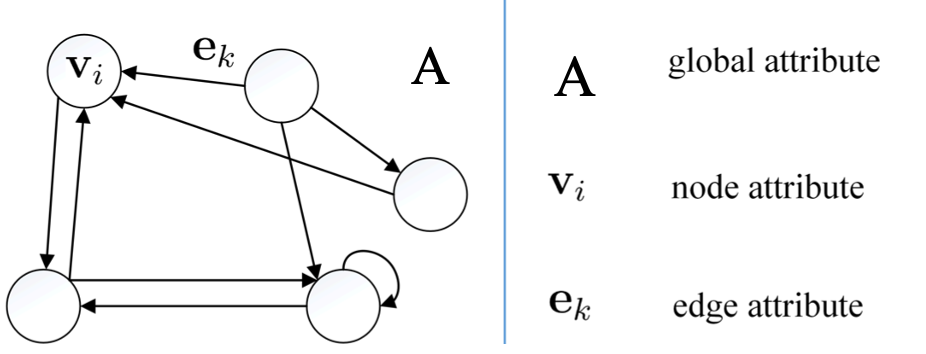


Figure 1. Graph network block

One of the approaches of implementing graph structure in knowledge tracking is building graph neural networks (GNN) model. It was widely used in different spheres: from classification to prediction and recommendation systems (graph attention multi-layer). By storing layers, GNNs are able to learn node representations with the help of utilizing information from the node’s neighborhoods in the layer (Yu, Y., etc., 2022).

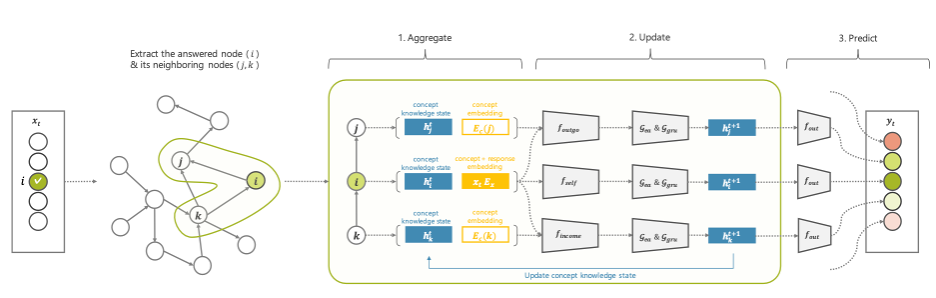


Figure 2. The architecture of graph knowledge tracking with multi-layer perceptron.

The process of implementing neural network to graph structured data is presented in figure 2. The first step of the model – to *aggregate* knowledge states and any additional embeddings for knowledge concept *i* and concepts *j ∈ Ni* in the neighborhood*.* The second one is to *update* knowledge states based on aggregated features and the structure of the graph, where *fself* states for multi-layer perceptron (MLP), G*ea* states for erase-add gate, and G*gru* – for gated recurrent unit gate. Finally, the model estimates the probability of a student answering concepts correctly at the next time steps (Nakagawa, H. etc., 2019).

The study of College of Electronic and Information Engineering, Tongji University, Shanghai on topic “Graph Neural Network for Senior High Student’s Grade Prediction” used GNNs to predict students' academic performance in online courses. The study used data from 10,765 students who completed 36 online courses on the edX platform. The data included various demographic and academic variables, such as students' age, gender, and previous academic achievements. The review used a dataset containing data on understudies' socioeconomics, earlier course execution, and informal community cooperation inside a specific course.

The specialists originally analyzed the interpersonal organization information and found that understudies who had more associations with their friends would in general perform better scholastically. They then, at that point, prepared a GNN on this dataset, utilizing it to foresee understudies' last course grades.The results showed that the GNN had the option to precisely anticipate understudies' grades with a typical mistake pace of just 2.63 brings up of 100. The specialists likewise observed that the GNN had the option to recognize which highlights of the dataset were generally significant for anticipating execution, which included interpersonal organization centrality, earlier course grades, and orientation.

Generally, the review features under topic “Study-GNN: A Novel Pipeline for Student Performance”. Prediction Based on Multi-Topology Graph Neural Networks the potential for GNNs to be used as a device for anticipating understudies' scholarly performance, especially while thinking about interpersonal organization collaborations as an element. This could have significant ramifications for instructive foundations hoping to recognize understudies who might be in danger of falling behind and give designated mediations to work on their presentation.

The pipeline takes into account the complex relationships between different educational entities, such as students, courses, and instructors, and builds a graph-based representation of this information. The multi-topology aspect of the approach allows for the incorporation of different graph topologies to better capture these relationships.

The model is trained using a combination of supervised and unsupervised learning techniques, which enables it to learn from both labeled and unlabeled data. This approach allows for improved accuracy in predicting student performance across a variety of educational settings.

These studies demonstrate the potential of GNNs for predicting students' grades based on their academic and social interactions. They show that by modeling the relationships between students and courses as a graph, GNNs can capture complex patterns of interactions and provide accurate predictions of academic performance. However, further research is needed to explore how GNNs can be used to support personalized learning and early interventions for students at risk of falling behind.

Finally, the article “Online Academic Course Performance Prediction usingRelational Graph Convolutional Neural Network” by authors Hamid Karimi, Tyler Derr, Jiangtao Huang, Jiliang Tang gives analysis of using the GNN to Deep Online Performance Evaluation (DOPE), which firstly models the student course relations in an online system as a knowledge graph, then utilizes an advanced graph neural network to extract course and student embeddings, harnesses a recurrent neural network to encode the system's temporal student behavioral data, and ultimately predicts a student's performance in a given course. Comprehensive experiments on six online courses verify the effectiveness of DOPE across multiple settings against representative baseline methods. Furthermore, we perform ablation feature analysis on the student behavioral features to better understand the inner workings of DOPE.

**Preprocessing and Data Loading.**

The next main step of the work is data collection. Nevertheless, collection of data from various sources will simply be fundamentally wrong. Therefore, among the options remains either finding a suitable existing dataset, or compiling your own using a questionnaire or other collection of information. The choice was made in favor of the first option, because the second one is very time-consuming and does not guarantee an acceptable final data quality. The dataset used: <https://www.kaggle.com/datasets/thedevastator/higher-education-predictors-of-student-retention>

There are many other popular datasets that have high ratings on Kaggle that are not real data, but artificially generated, which in total may not give the correct picture on the topic. The selected dataset is created from real data from higher education institutions and related to students enrolled in different undergraduate degrees. DOI of the article based on the selected dataset: 10.5281/zenodo.5777340. Thus, the next step is to load and prepare selected data. JupyterNotebook was chosen for this task.

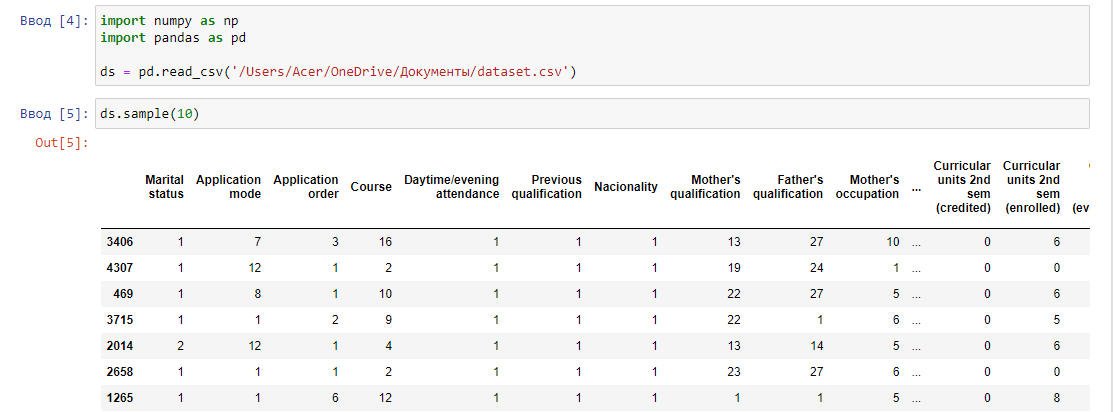


Table 1. EDA

At the moment, the source dataset has 35 columns, which is too many, because most of them most likely have a negligible impact on the target. In addition, the target variable itself is a categorical value. The primary task for now is to find out the correlation between variables and the target, but in order to find out the correlation, the target must be a numeric value.So, we convert the target to a numeric value and check the correlation with the rest of the variables.

*ds['Target'] = ds['Target'].map({ 'Dropout':0, 'Enrolled':1, 'Graduate':2})*

*ds.corr()['Target']*

The GNN model, which we are going to build as part of the work, accepts specific data as input: they should include information about the nodes and edges of the proposed graph. In other words, before further work, the data needs to be processed. To begin with, we will leave in the dataframe only those columns that will be used in the model itself (“Target”, “ Previous education” and “Course”).

In our model *the nodes are the grades we got for the exam*. And the *edges of the nodes are information about the current course and previous education.*

The data on the current course contains the following courses:

* Biofuel Production Technologies,
* Animation and Multimedia Design,
* Social Service (evening attendance),
* Agronomy,
* Communication Design,
* Veterinary Nursing,
* Informatics Engineering,
* Equiniculture,
* Management,
* Social Service,
* Tourism,
* Nursing,
* Oral Hygiene,
* Advertising and Marketing Management,
* Journalism and Communication,
* Basic Education,
* Management (evening attendance)

The data on previous education contains following categorical variables:

* Secondary education,
* Higher education—bachelors degree,
* Higher education—degree,
* Higher education—masters degree,
* Higher education—doctorate,
* Frequency of higher education,
* 12th year of schooling—not completed,
* 11th year of schooling—not completed,
* Other—11th year of schooling,
* 10th year of schooling,
* 10th year of schooling—not completed,
* Basic education 3rd cycle (9th/10th/11th year) or equivalent,
* Basic education 2nd cycle (6th/7th/8th year) or equivalent,
* Technological specialization course,
* Higher education—degree (1st cycle),
* Professional higher technical course,
* Higher education—masters degree (2nd cycle)

The data has been preprocessed with CountVectorizer and StandardScaler before building the model. The data were divided into a test and a train with test size 20% and random state 42. To fill "x" with two variables, the function “Numpy.concatenate” was used.

**Practical Part.**

For implementing our practical part we will use 2 libraries: PyTorch and PyTorch Geometric (PyG), which is incredibly fast. It's several times faster than GNN's most famous environment - DGL (Kipf&Welling, 2016). The architecture of both libraries is based on the graph convolutional layers (see figure 3).

The PyTorch Geometric Graph layers have a similar API to PyTorch layers, but they require the graph edges from the edge\_index in the Batch class. The Batch class represents one or more graphs combined into a single graph with some gaps. To achieve weight-sharing for graph convolutions, the batches use matrix-multiplication and a combined adjacency matrix. Additionally, the Batch object has a variable called batch that tracks which node belongs to which graph.

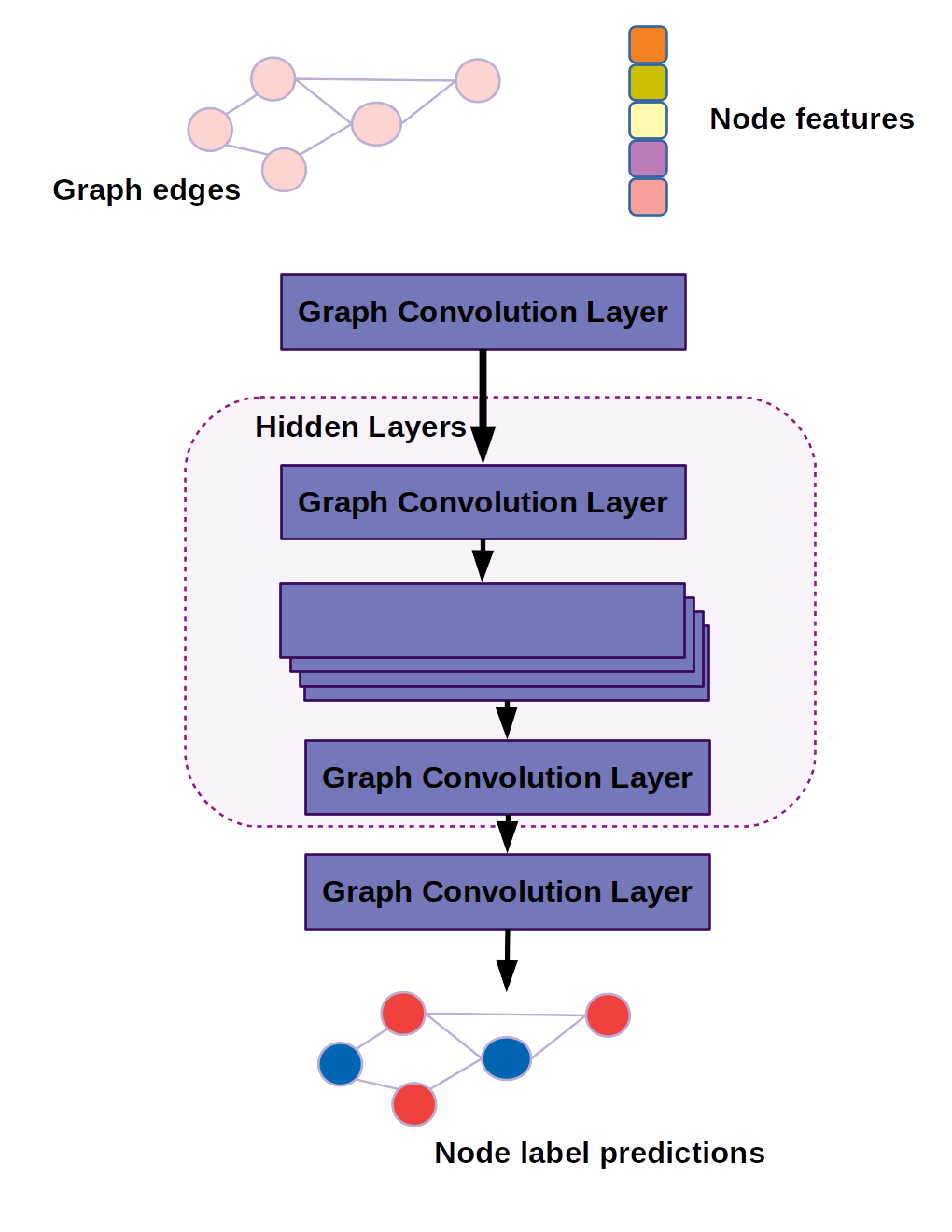
****

Figure 3. The architecture of the graph convolutional network used to benchmark DGL  
 and PyTorch Geometri.

After data is prepared, we come to building the model, using PyTorch and PyG. Firstly, we initialize the model using super() function to call the constructor of Graph Neural Network. After we initialize ReLU, Rectified Linear Unit. To convert the summed weighted input from a node into its activation or output, ReLU is used. It is a piecewise linear function that outputs the input directly if it is positive, otherwise, it outputs zero. ReLU has gained popularity as the go-to activation function for neural networks due to its ease of training and superior performance. Additionally, we have used Cuda function to increase computational performance through the use of graphics processors.

For criterion we have used Negative Log-Likelihood Loss (NLLLoss) - the metric which is used extensively in the process of training model , in the majority, in cases where the training set is unbalanced. NLLLoss works the following: log probabilities are created while adding a new layer of Log soft max as the last layer of a neural network.

As the optimizer we have used Adam. In contrast to stochastic gradient descent, Adam includes 2 benefits which expands advantages. The first one is adaptive gradient algorithms which maintains per-parameter learning for better performance. The second one - root mean square propagation which helps to adapt learning rates based on average of the gradients for their weights. It helps well in case of having non-stationary problems.

Such a model setup gave us accuracy of 79,91%. Work was also carried out with GNNExplainer [5] - it is an optimization task that maximizes the mutual information between a GNN’s predic-tion and distribution of possible subgraph structures. However, the best results obtained with GNN are *Feature importance,* which at the same time are relatively small - less than 0.15. This can be explained by the quality of the initial data and their predisposition to use in GNN, but these are the only attributes that potentially have connections that apply to GNN. The full code is provided in the accompanying file.

**References**

1. Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction, 4, 253-278.
2. Nakagawa, H., Iwasawa, Y., & Matsuo, Y. (2019, October). Graph-based knowledge tracing: modeling student proficiency using graph neural network. In IEEE/WIC/ACM International Conference on Web Intelligence (pp. 156-163).
3. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. Advances in neural information processing systems, 28.
4. Yu, Y., Fan, J., Xian, Y., & Wang, Z. (2022). Graph Neural Network for Senior High Student’s Grade Prediction. Applied Sciences, 12(8), 3881.
5. Ying Z. et al. Gnnexplainer: Generating explanations for graph neural networks //Advances in neural information processing systems. – 2019. – Т. 32.
6. Ming Li , Xiangru Wang, YiWang, Yuting Chen and Yixuan Chen. Study-GNN: A Novel Pipeline for Student Performance Prediction Based on Multi-Topology Graph Neural NetworksмKey Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua 321004, China; (2022)
7. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
8. Jakob Nikolas Kather, Johannes Krisam, Pornpimol CharoentongNiels Halama. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study (2019)
9. Chen, L., Bai, Y., Huang, S., & Zhou, Z. (2022). Making Your First Choice: To Address Cold Start Problem in Vision Active Learning. ResearchGate. <https://www.researchgate.net/publication/364193983_Making_Your_First_Choice_To_Address_Cold_Start_Problem_in_Vision_Active_Learning>
10. Delip Rao, B. M. (2019). *Natural Language Processing with PyTorch: Build Intelligent Language Using the Deep Learning.* OREILLY.
11. He, Y. (2022, February 22). PyTorch Geometric Signed Directed: A Software Package on Graph Neural Networks for Signed and Directed Graphs. arXiv.org. <https://arxiv.org/abs/2202.10793>
12. I. Stančin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019, pp. 977-982, doi: 10.23919/MIPRO.2019.8757088.
13. Papa, J. (2021). *PyTorch Pocket Reference.* O'REILLY.
14. Lau, R. (2022, March 12). Cross-Entropy, Negative Log-Likelihood, and All That Jazz. Medium. <https://towardsdatascience.com/cross-entropy-negative-log-likelihood-and-all-that-jazz-47a95bd2e81>