Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/45/20241627

Predicting game popularity through graphic modeling

Siyuan Wu[1,3], **Tingting Yang**[2,4,5]
[1]Pomfret School,Pomfret,United States of America
[2]Technology Innovation Center at TAIKANG LIFE INSURANCE Co., Ltd

[3]name-eman@163.com
[4]Mobphyspde100@outlook.com
[5]corresponding author

Abstract. The gaming industry has been a profitable area. But recently given the saturation of the market where there are more than 100,000 games on steam, each game must under go careful planning prior to its release. The planning often involves making sales predictions given various features of the game. Our model incorporates a graphical neural network in making binary classification using data provided by the Steampower APl. There exists prior methods such as deep factorization machine, random forest, and graph convolutional network in targeting similar issues but none of them have been used to analyze game popularity. Referring to Tab GNN, we first implemented a multiplex graph embedding using the features obtained from Steam power then applied GCN conv to enhance the learning of graphical embedding. These embeddings were then fed into a 3-layer-DNN to make binary predictions. Experiments with other baseline models were conducted showing that our model is possible in making relatively more accurate results in predicting game popularity.

Keywords: Gaming industry, Sales predictions, Graphical neural network, Steampower API, Multiplex graph embedding

1. Introduction

Tabular data refers to information organized in tables, where each column denotes an attribute and each row represents a specific data point.[1] The prediction of tabular data plays a crucial role in data analysis across various industries. Numerous methods have been proposed to address tabular data prediction (TDP).

Steam currently stands as the predominant digital gaming platform for PC gaming, boasting over 100,000 games and a user base exceeding 90 million. Consequently, the success of a new game release hinges on surpassing competitors within its genre in certain aspects, capturing a significant market share, and generating profit. Therefore, it is imperative to conduct a thorough market analysis before launching or even producing a new game[2,3] .

Existing machine learning models for regression or classification fall short of providing accurate predictions for high-dimensional datasets. Our proposed model, which combines a tabular graphical neural network and a deep neural network, tackles this issue by examining and forecasting the popularity of a forthcoming game based on various features. This analysis is conducted prior to the scheduled release, utilizing information obtained from the Steam API.

By constructing a graphical representation that depicts relationships among samples and their distinct features, our model can predict the values in a target column more accurately. This approach outperforms

DOI: 10.54254/2755-2721/45/20241627

other models addressing the same problem, offering enhanced precision in forecasting the success of a new game.

2. Prior work

We explored graphical data modeling for game samples and chose to reference [4] and [5], which proposed graph neural networks for tabular data.

The paper [6] predicts click-through rate for displayed advertisements, where each sample contains features that are categorized by the arthur into multiple feature fields. The model, Feature lnter action Graph Neural Network, consists of the feature embedding stage and the graphic attention stage. For an advertisement sample, each feature field will go through a field-aware embedding and a multi-head self-attention layer to be converted into a dense hidden embedding. A fully connected graph for the sample will be constructed, with nodes being the hidden embeddings and edges being weights representing the strength of relations between features. The graph will be passed into a Gated GNN structure to learn the edge weights, and the produced hidden states will be used by an attention scoring layer to output the probability of the user clicking the advertisement sample. Fi-GNN achieves an AUC-ROC score of 0.8062 for Criteo dataset [6], which shows improvement in comparison with the baselines such factorization machine (0.7836) and xDeepFM [7] (0.8009). Although Fi-GNN is used for classification in the paper, its effectiveness in modeling feature interaction makes it adequate for uncovering relations behind the features of the game samples, potentially resulting in more robust predictions.

While Fi-GNN obtains the graphical representation of data through modeling feature interactions, the paper for TabGNN [8] takes a different approach by modeling sample interactions. TabGNN is a model for higher-order feature extraction. Its effectiveness is evaluated on multiple datasets for both classification tasks and regression tasks with around 2% performance improvement in general against a baseline that only uses the original features. For graph representation, all samples are projected into a multi-layer graph (multiplex graph) where the nodes are the samples. Each layer is dedicated to a feature by connecting sample nodes on the layer based on their similarity in the feature. For each sample, the TabGNN will use a graph aggregating method similar to [5] to obtain a hidden embedding of the graph by scanning neighbours around the sample to a certain depth. These hidden embeddings will be concatenated and sent to a predictor (e.g., DeepFM) for final predictions.

Decision tree is another helpful concept for this project. A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered and the model is trained and tested on a set of data that contains the desired categorization. The base of the tree is the root node. From the root node flows a series of decision nodes that depict decisions to be made. From the decision nodes are leaf nodes that represent the consequences of those decisions. Each decision node represents a split point, and the leaf nodes that stem from a decision node represent the possible answers.

3. Data Retrieving and Processing

3.1. Data Retrieving

Our goal is to predict the popularity of newly released games based on their various features using a graphical model. However, Steam [Steam] does not publish much of the information we need such as the number of downloads of the games nor does it have a field to reflect their ratings. As a result, instead of crawling on Steam, we chose to crawl on the Steampower API offered by Valve, the company who developed Steam. lt not only has information on number of downloads and number of recommendations but also information on many other aspects in an organized fashion so that we do not have to deal with grabbing details from html code directly.

Referring to the code published by a blogger online Steam Crawler, we wrote our own crawler using different modules including the Request and Urllib module. We have used json and pandas in manipulating the data and csv module to store the data into csv files instead of using relational

DOI: 10.54254/2755-2721/45/20241627

expressions (the Re modules) and Excel worksheets (Xlwt module) mentioned in the proposal. We managed to retrieve more than 50,000games with these following attributes

('type', 'name', 'steam_appid', 'required_age', 'is_free', 'controller_support', 'dlc', 'detailed_description', 'about_the_game', 'short_description', 'fullgame', 'supported_languages', 'header_image', 'website', 'pc_requirements', 'mac_requirements', 'linux_requirements', 'legal_notice', 'drm_notice', 'ext_user_account_notice', 'developers', 'publishers', 'demos', 'price_overview', 'packages', 'package_groups', 'platforms', 'metacritic', 'reviews', 'categories', 'genres', 'screenshots', 'movies', 'recommendations', 'achievements', 'release_date', 'support_info', 'background', 'content_descriptors')

3.2. Data Preprocessing

With these aforementioned attributes, we build a dataset with more than 10,000 samples with 12 non-null features, namely, {required age, contoller_support, number of DLCs, number of supported languages, RAM requirements, memory requirements, number of supported platforms, price(USD), release date, developers, categories, genres} With the number of Recommendations being the label. For simplicity and fair data distributions, we performed binary classification tasks based on the number of recommendations of games using a threshold of 300 and 600. The distribution of the number recommendations is shown below.

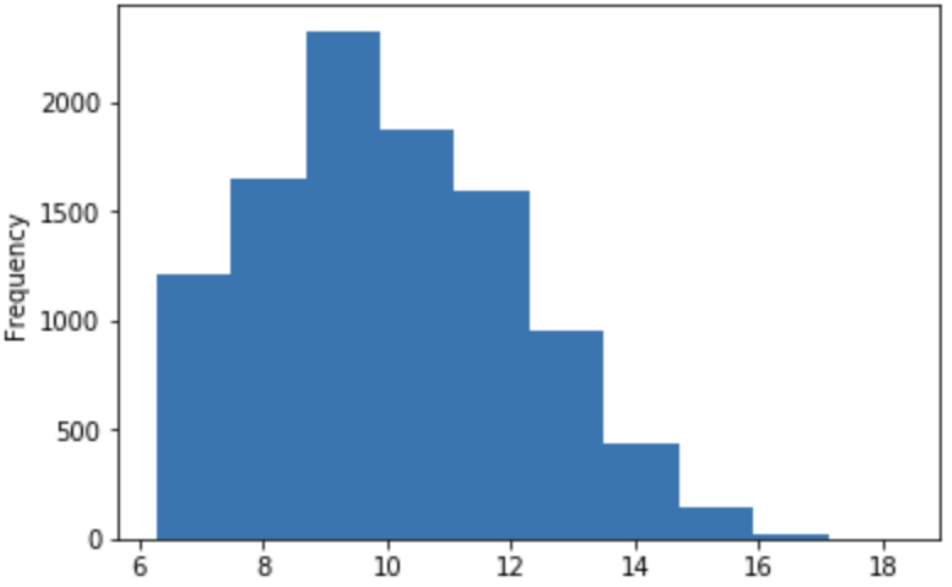

Figure 1: Label (number of recommendations) distribution of dataset

It is clear that the majority of our samples have less than 500 recommendations and should a threshold be set at 500. But to prevent data obscurity (where a game with 499 recommendations will be set to class 0 and a game with 501 recommendations will beset to class 1) and to ensure parity of positive and negative samples, we left a gap between 300 and 600. Hence, games with less than 300 recommendations will be in Class 0 and games with more than 600 recommendations should belong to Class 1.

Among the 12 features, 7 of them are numerical features or simple binary features {required age, contoller_support, number of DLCs, number of supported languages, number of supported platforms, price(USD), release date}

and 5 of them are categorical features

{RAM requirements, memory requirements, developers, categories, genres}.

DOI: 10.54254/2755-2721/45/20241627

We have experimented with various encoding mechanisms: label (integer) encoding and multi-hot encoding. Similar to one-hot encoding, where each feature is represented as a list of ones and zeros, with a one indicating that the feature belongs to a particular state, multi-hot encoding allows multiple ones to occur in such an array, indicating that the feature can be in multiple states. This is particularly relevant for our features 'category' and 'genres' as a game can fall into multiple categories and genres.

In order to ensure consistency of data, our numerical features are also converted into multi-hot encoding, where each encoding will start with a sequence of ones with length equal to the ranking of its original numerical value amongst all values of the same feature in the dataset. For instance, in release year, the game sample with the earliest year will be set to rank 0 with all entries of the encoding be zeros, whereas the game sample with the latest year will have rank rmax with an encoding of all ones, where rmax = yearearliest - yearlatest. Any other samples released n years later than the earliest game will have n ones at the front with the rest rmax - n entries being all zeros. Multi-hot encoding will facilitate the construction of the graph, which will be discussed in the following section.

For the Steam game dataset, multi-hot encoding will not affect the performance of models since most numerical features have integer values only, and we consider rounding price values to the nearest integer should still maintain most information of this feature. For a decision tree, the integer value of a numerical feature that best separates the data is now the index of this feature serving the same purpose. As for neural networks, they are capable of reconstructing the original numerical value if during training they find it useful.

Here is a sample of how features are process.

Table 1: Sample Feature Encoding

	Price	Platforms	Release Year	Categories(ID)
Integer Encoding	2.9	3 (largest)	1997 (smallest)	$[1, 8]$
Pure Multi Hot	$\begin{bmatrix} 1 & 1 & 0 & 0\dots \\ \dots & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0\dots 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0\dots 0 & 1 \\ 0\dots 0 & 0 \end{bmatrix}$

3.3. Graph Processing

To examine the performance of graph neural networks on the Steam game data, the dataset needs to be converted into a graph. We have considered 2 ways of building the graph representation: a single graph and a multiplex graph. The representation of the multiplex graph is adopted from TabGNN[8]. Each layer corresponds to one of the features and uses all game samples as its nodes. A pair of nodes will be connected only if their multi-hot encoding vectors have unequal entries less than a threshold number, which is a hyperparameter to be tuned. With this method, the neighbors of a node share similar features with it, contributing structural information to the graph. As for the single graph, it can be considered as a special multiplex graph with only one layer, where all the features are concatenated into one for applying the criterion of node connection.

The use of multi-hot encoding facilitates the building of graphs along with applying the connection criterion. Let $U \in R^{n\times c}$ denote the feature matrix for n game samples with respect to one of their feature which has multi-hot encoding with c entries. The i^{th} row, U_i, is the multi-hot encoding vector for the i^{th} game. The product $A^+ = UU^T \in R^{n\times n}$ is the "weighted" adjacency matrix in between games, where $A^+_{ij}$is the number of overlapping ones between the i^{th} and the j^{th} game.

Similarly, for $U^\star = J - U \in R^{n\times c}$ where J is the matrix of all ones, the matrix $A^- = U^\star U^{\star T} \in R^{n\times n}$ is the "weighted" adjacency matrix counting the overlapping zeros between each pair of game samples. Together with a threshold $\delta \in [0, n]$, the adjacency matrix that denotes the connections between nodes for this particular features is given by $A_\delta = (A^+ + A^-) \geq (n - a)$, where $n - \delta$ is the minimum number of overlapping entries that two connected nodes must have. By computing A_δ

DOI: 10.54254/2755-2721/45/20241627

for the concatenated features and for each feature, a single graph and a multiplex graph is obtained respectively.

4. Baseline Models

One of the most popular tabular datasets originates from the prediction of click through rate, where each pair of data represents a preference of user item relation [9]. In such click through rate prediction tasks where information is organized in tabular form, methods like decision tree or deep factorization machine are often applied.

Since the prediction of tabular data is often rooted into answering simple questions, like which melanges of ages the game intends to appeal to or what is the impact of a games price on its popularity, the structure of such algorithms can be made into trees. With recursion, the ability of making predictions in spliting nodes into subsets based on previous observations makes the Decision Tree Algorithm accurate and popular. [10]

Deep Factorization Machines combine the power of both factorization machines in modeling linear feature interactions and deep neural networks in modeling high order transformations parallelly into one algorithm. This allows such algorithms to model complex and nonlinear relationships which sometimes happens in tabular datasets. The two components of DeepFM: MLP and FM share the same embeddings and their outputs are integrated together as the final prediction. The structure of such an algorithm is shown below. [7]

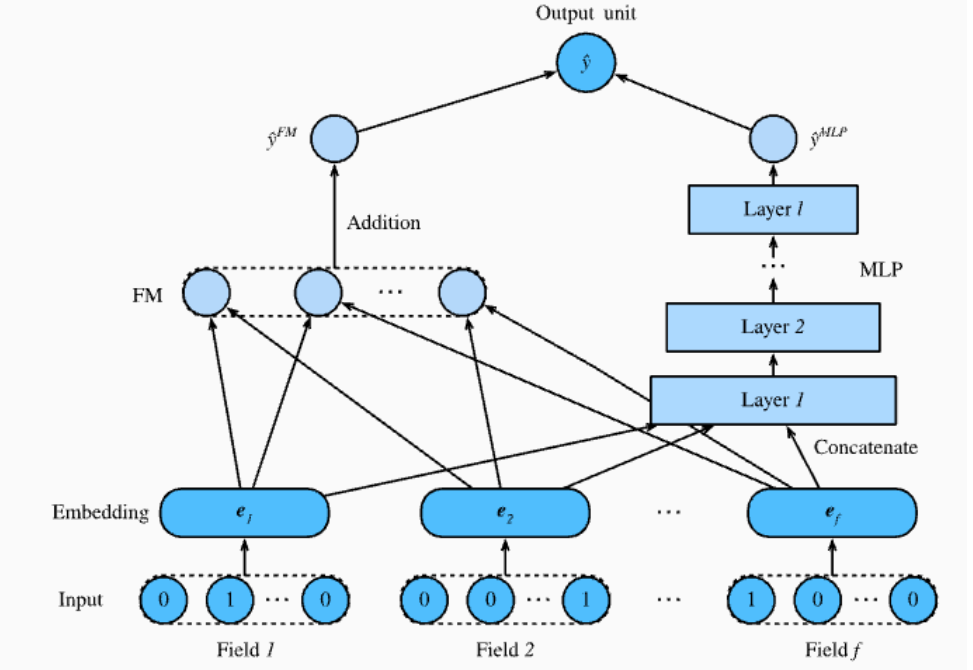

Figure 2: Architecture of Deep Factorization Machine.

In addition to the aforementioned baselines, we also included a simple dense neural network (DNN) of 3 fully connected layers. This provides a comparison between Decision Tree and a neural network of similar capacity.

5. Graph Models

For the design of our model, since we focused on examining the effectiveness of graphical representation and graph convolutional networks (GCN) on Steam game data, we did not attempt to design a more

DOI: 10.54254/2755-2721/45/20241627

complicated model to outperform TabGNN. Instead, we built our model quite similar to our baseline with the exception of an additional GCN block to clearly demonstrate how much improvement can graph embedding bring to feature extraction. We designed two models for the single graph and the multiplex graph.

5.1. GCNconv Layer Math Expression

Instead of writing our own graph convolutional layer, we chose to use the pre-written GCNconv layer offered by torch_geometrics.

The GCNconv layer utilizes the convolutional operator from the paper [5]. It has the format as follows

$$X' = \widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}X\Theta \tag{1}$$

Where $\widehat{A} = A + I$ denotes the adjacency matrix with self connection added and $\widehat{D}_{ii} = \sum_{j=0}\widehat{A}_{ij}$ denotes its diagonal degree matrix. Essentially, the D matrix normalizes the adjacency matrix based on the degree of the node itself and the degree of its neighbors. Hence, it node-wise formulation is given by the following equation

$$x_i' = \Theta^T \sum_{j\in N(u)\cup\{i\}} \frac{e_{j,i}}{\sqrt{\widehat{d}_j\widehat{d}_i}} x_j \tag{2}$$

With d_i denotes the summation of edge weight with its neighbors plus 1 to incorporate a self loop[10].

5.2. Single Graph GCN Model

The architecture of the single graph GCN consists of 3 blocks: the initial embedding, the graph embedding and the classifier. The initial embedding takes as input the concatenated multi-hot encoding vector of a sample, passes it through 2 fully connected layers with an activation in between. After the first block, each node on the graph obtains an initial feature vector, which will be used by the following graph embedding block for convolutional aggregations. There are N sequential graph convolutional layers with residual connections inside the block. Activations are applied to the output embeddings of each layer just before making the residual connection. The graph convolutional layers are stacked for increasing its perceptive field in the graph for each node, whereas the residual connection is applied for sake of convergence and reducing the influence of potential over smoothing due to the stacking. The dense vectors obtained through the graph embedding block will be passed to a block of 3 fully connected layers to output the binary classification probability vectors for each sample.

As a comparison, the TabGNN paper uses the entire DeepFM just for the initial embeddings and does not have an extra block of fully connected layers to learn from the graph embedding outputs. To better demonstrate and analyze the effect of graph embedding with respect to the baseline DNN, we used a much simpler initial embedding and retained the 3 fully connected layers from the DNN for classification. Moreover, for a comparison of similar complexity, a 6-layer DNN is also trained as an additional baseline.

5.3. Multiplex Graph GCN Model

The multiplex GCN uses the same 3 blocks in a single graph GCN, except in a slightly different way for dealing with multiple layers. The multi-hot encoding of each different feature now has its individual initial embedding block using the same setup of the single graph GCN, while the graph embedding block is also individualized to each feature and uses only the corresponding feature layer of the multiplex graph for convolutional aggregation. For each sample, the dense vectors output by the graph embedding blocks belonging to each feature will be aggregated into a single dense vector by taking the average, which will be sent to the classifier block and make predictions just like in the single graph GCN.

DOI: 10.54254/2755-2721/45/20241627

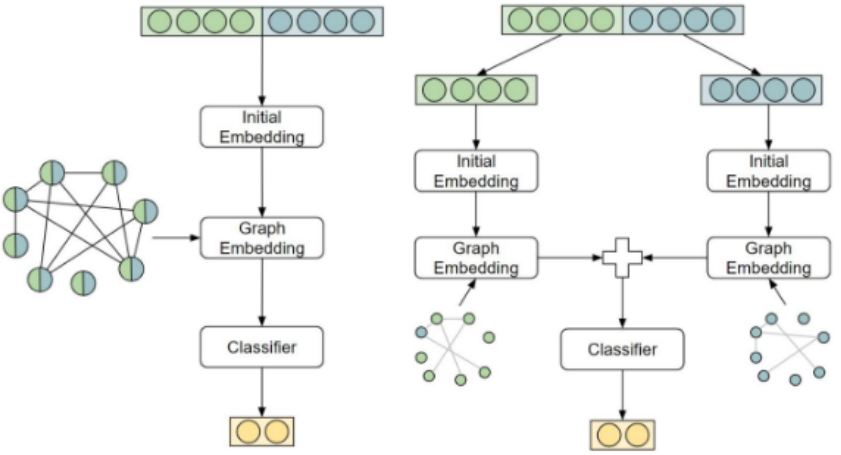

Figure 3: Architectures for the single graph model(left) and multiplex graph model(right)

5.4. Experiment Methods and Result

All models are trained using $7122(80\%)$ game samples and tested on $1781(20\%)$ game samples drawn randomly from the available 8903 samples for 5 times using 5 different random seeds. The hyperparameters for neural networks are tuned based on 20% of the training data used for validation. During training of GCN models, a batch size of 2048 is used in order to fit the graphs and the feature encoding data into the GPU (Nvidia P100 16GB from Google Colab). This means only the part of the graph corresponding to the nodes in the batch was used for each model update, which empirically did not affect the convergence of the model and theoretically increased stochasticity for the gradient descent process. Cross entropy loss was used for all neural network based models with an Adam optimizer. A descent learning rate is tested to be 0.001 , and we applied $l2$ regularization by setting the weight decay (which is $l2$) parameter of Adam to 0.005 to deal with overfitting. The effect of setting this parameter was that the model stayed longer with high validation accuracies before it overfit. The following are the test results obtained by using the best parameters of the models trained to the validation best accuracy in each experiment setup.

5.5. Baseline Results

Table 2: Baseline Model Accuracy using One-hot Encoding

Baseline Models	Random Forest	Deep FM	DNN (3 layers)
Accuracy	76.51%	68.26%	74.34%

Table 3: GCN Accuracy on Single Graphs built by different Thresholds

Unequal Feature Values	20	16	10	3
Single Graph GCN	79.79%	$\mathbf{82.59}\%$	81.30%	73.67%

5.5.1. Threshold comparisons To test the effect of different connection criterions, a single graph GCN model is trained on single graphs built by different thresholds. There are 3 residually connected GCN

DOI: 10.54254/2755-2721/45/20241627

layers in the model, with each having 512 hidden units, which are determined through hyperparameter tuning. In general, the model takes 300 epochs to converge, and the accuracies are summarized in Table 3. In the table, the threshold is presented as the maximum number of unequal feature values that are allowed in the multi-hot encodings of two connected nodes. A larger unequal feature value means more nodes are allowed to be connected, resulting in a more connective graph while increasing the chance of having more cliques in the graph. Cliques are harmful to GCNs because they make the structures in the graph less abundant and induce over-smoothing. Each node in the clique aggregates information from all other nodes in just one GCN layer, making it harder to distinguish between different nodes to obtain useful hidden representations. Starting from the threshold of 3 , the performance of GCN is growing as the threshold grows, but starts to decline once too many nodes are allowed to connect to each other. While a loose threshold of 20 can smooth out the graph's structural information, not enough information is included in a graph of too few connections made by a tight threshold of 3 , harming the performance even more. Empirically, a threshold of 16 is adequate for keeping enough information for each node and not yielding too many cliques, reaching the highest performance in the GCN model. Since the multi-hot encoding input of each sample in the single graph setting is the concatenation of all of 10 features' encodings, as an interpretation, a threshold of 16 for 2 connected nodes allows some of the features to be very different but requires the rest to be similar. The following figures show the effect of different thresholds on the distribution of the node degrees over all nodes.

The provided data illustrates the distribution of games or applications across various categories. With the majority falling under the 'CategorySinglePlayer' (9856) and 'CategoryMultiplayer' (2993) classifications, it suggests a prevalence of both single-player and multiplayer experiences. The 'CategoryCoop' (1126) category indicates a subset designed for cooperative play, while 'CategoryMMO' (283) represents a smaller number of Massively Multiplayer Online (MMO) games. The 'CategoryInAppPurchase' (152) points to items featuring in-app purchases, emphasizing a revenue-generating aspect. The 'CategoryIncludeSrcSDK' (24) and 'CategoryIncludeLevel'.

5.5.2. Residual Connection For the single graph of threshold equal to 16 , we trained a single graph model with and without residual connections in the graph embedding block of $3\mathrm{GCN}$ layers. As shown in Figure 5, the GCN trained without residual not only converged slower, but also the features it learnt didn't generalize well into the validation set, yielding poor train accuracy (79%) and validation accuracies (68%) in the first 4 epochs. With residual connections, both the train and validation reach a higher accuracy (81%) together, demonstrating the robustness and effectiveness of the model. Hence, we decided to have residual connections for the graph embedding block of models.

Table 4: Comparison of General Results among Single and Multiplex Graph Convolutional Networks (GCN) and Other Baselines

Graph Layers	Unequal Feature Values	GCN (1 Layer)	GCN (2 Layers)	GCN (3 Layers)	GCN (4 Layers)	DNN (3 Layers)	DNN (6 Layers)	Random Forest
1	16	76.03	79.94	81.81	82.59	74.34	75.51	76.51
5	1	78.44	81.58	84.17	**84.45**	N/A	N/A	N/A

5.5.3. Multi vs Single vs Others For a general comparison between single graph GCN, multiplex graph GCN and other baseline models, we trained GCN models for the single graph of threshold 16 and for the multiplex graph of 5 feature layers and threshold 1, repeated using 1 to 4 convolutional layers inside the graph embedding blocks and obtained the results in Table 4. Note that due to the limitation of our GPU, having more than 5 feature layers in the multiplex graph will require the batch size to be smaller than 2048 , causing the model to perceive less graphical information than in the case of the single graph.

For the sake of comparison with the single graph model, we chose 5 of the most important features to build the multiplex graph, whose importance will be analyzed in the next section. From the results, the

DOI: 10.54254/2755-2721/45/20241627

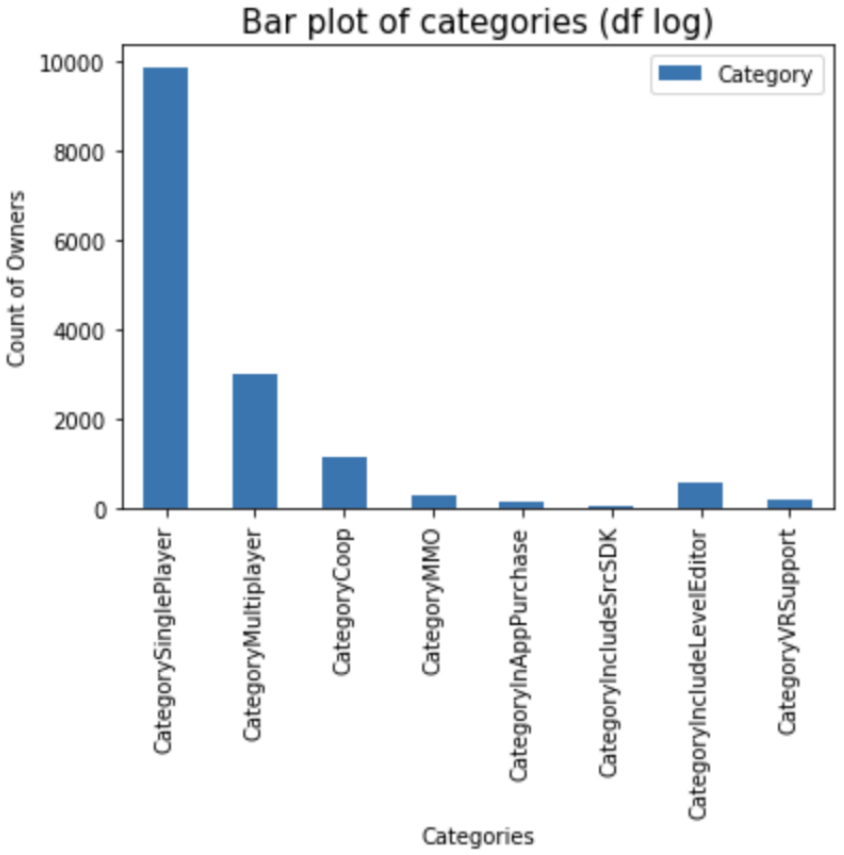

Figure 4: Bar plot of categories

performances of GCN models are better than the baselines, while the multiplex graph GCNs dominate the single graph GCNs when they both have the same amount of convolutional layers. The superiority of GCN models is arised from the extra graphical information the model perceived and the sample interactions the model performed based on the graph. Although sample relations always exist in the data, only the model that could utilize it effectively can gain more insights and learn better representations of the data. Even by feeding the adjacency matrix into the Random Forest classifier, we found that its performance got worse as they are just redundant information to the model. As for the better performance of GCN using the multiplex graph, it is because the single graph can be considered as a compressed version of the multiplex graphs. Two nodes can be both connected and disconnected at the same time in different feature layers of the multiplex graph, but they can only be in one relation in a single graph. With this advantage, even with only 5 features, the multiplex graph GCN is robust enough to achieve the highest test accuracy.

5.5.4. Feature Importance We have analysed feature importance through the Decision Tree model from Sklearn and tested the difference of results in using integer and multi hot encoding. Several interesting observations were seen.

Table 5. Results of Different Encoding on Random Forest Algorithm

Using different types of encodings gave us different accuracies as well as different rankings of feature importance. The first graph comes from applying multi hot encoding where price is given the highest priority with number of supported languages and release coming as second and third. While the second graph entails information when utilizing integer encoding and the priorities are somewhat different,

DOI: 10.54254/2755-2721/45/20241627

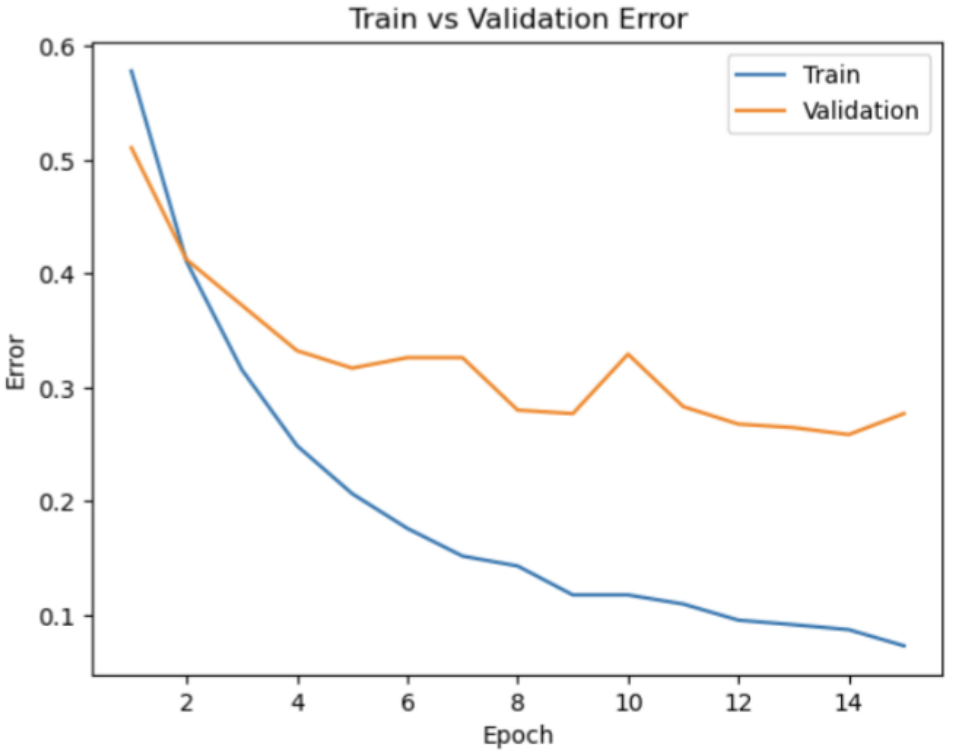

Figure 5: Convergence plot of the model

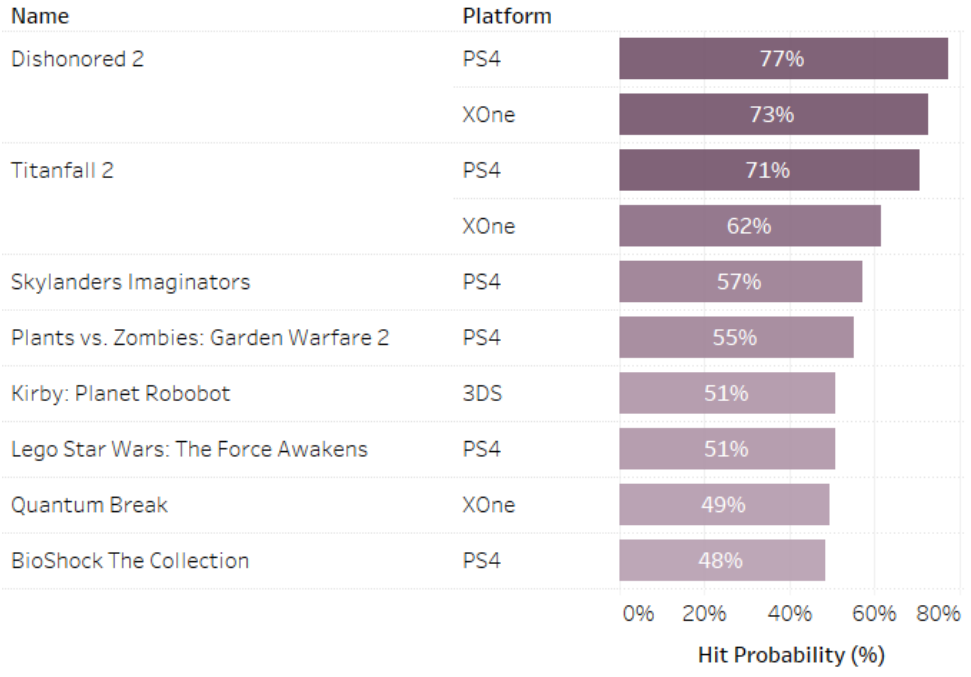

Figure 6: Architectures for the single graph model(left) and multiplex graph model(right)

DOI: 10.54254/2755-2721/45/20241627

Table 5: Results of Different Encoding on Random Forest Algorithm.

Feature ”Developers”	Multi hot encoding	Integer Encoding Directly	Randomized integer encoding	Sorted integer encoding
Accuracy	0.7651	0.7852	0.7606	0.8569

developers being the top rank with price and number of supported languages coming after.

We further analyzed the reason why developers will become the top priority when encoding into integers. Different integer encoding methods were used including simply applying integers to the developers in their natural order, randomly assigning integers, and applying integers in a sorted fashion. This gave interesting results where sorted integer encoding produces the highest accuracy. The frequency of the appearance of the developers were plotted and it is noticeable that developers with many games tend to lay on the left side of the graph. This leads to the fact that simply applying integer encoding generates a higher accuracy than multi hot encoding as the Random Forest algorithm makes decisions based on the comparison of numerical values and a sorted encoding implies to the algorithm that games with larger developers' labels are generally more erupted which is not fair to other developers. So in order to prevent our model from directly comparing numerical values instead of actually 'learning', we chose not to use integer encoding in making our models.

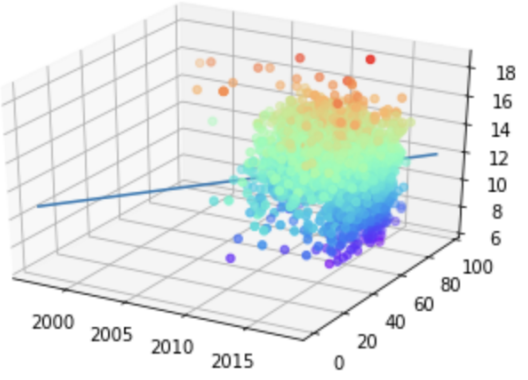

Figure 7: Scatter plot of the feature regression

6. Ethical, Social Impact, and Fairness

This project involves the use of several baseline models, such as decision tree and deep factorization machine. In particular, this project contributes to the practice of Tabular GNN and shows the potential of using TabGNN to explore new fields, such as predicting game sales as in this project.

Multiplex Graph is a key component of the project and its main idea is that each sample has multiple graphs and, for each graph, sample nodes are connected with other samples with respect to the currently selected feature. The practice of Multiplex Graph in this project shows the potential of ML to handle the problem where it involves a lot of features and allowing these features to interact with each other to generate reasonable predictions.

Beside the goal of this project, predicting the popularity of a pre-released game, it's also possible to implement this project's idea to other problems, such as predicting loans for banks, predicting music

DOI: 10.54254/2755-2721/45/20241627

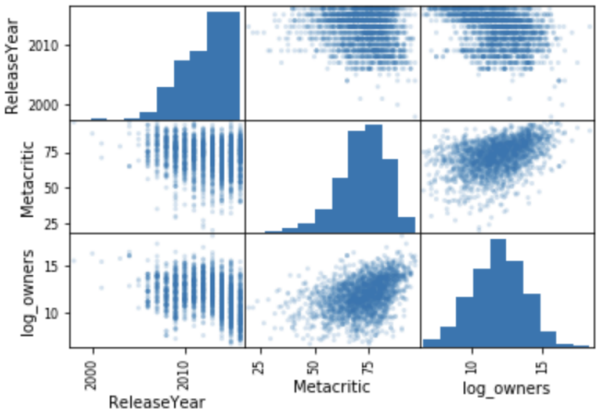

Figure 8: Correlation plot of release year, metacritic, log_owner

sales for record companies, or box office in the movie industry etc. To perform a reliable prediction, numerous data in tabular format is required, hence, the performance may be limited by privacy issues.

Fairness issues can also be seen in this project. From the generated feature importance, it shows that a game that charges more tends to be given a higher popularity whereas a game with lower cost would be given a low popularity given our bias from the data. However, even though price could impact on the popularity of games, it is not the determining factor. Instead, the pictorial and animation quality, the storyline, the design of the game itself also shines on a game's popularity. Our dataset is only able to reflect the market view of a game rather than its quality view.

7. Conclusion

In this report, we analysed the effectiveness of multiple models in predicting game popularity. Of all these existing methods, Tabular GNN gives the most promising results in making binary classification using various features crawled from the Steampower API. Following the paper, we constructed a multiplex graph to capture the multifacet feature interactions and then apply GCN to aggregate the neighbor information. Finally, these embeddings were fed into a 3-layer-DNN for final prediction.

For future work and a possible way to reduce the downside, we thought of applying a transformer to analyze the descriptions of the game and then add inter-layer-attention to weight the significance of each layer of the graph. For example, the developer group size and the number of games released can be weighted lighter to result in a more fair prediction, when compared to the small developer group.

8. References

[1] O. D. I Jeni Tennison, K. A. Gregg Kelogg, and W. C. Ivan Herman, "Model for Tabular Data and Metadata on the Web," *Model for tabular data and metadata on the web*. [Online]. Available: https://www.w3.org/TR/tabular-data-model/#:~:text=Tabular%20data%20is%20data%20that,thing%20described%20by%20the%20row. [Accessed: 13-Apr-2022].

[2] X. Guo, Y. Quan, H. Zhao, Q. Yao, Y. Li, and W. Tu, "Tabgnn: Multiplex graph neural network for tabular data prediction," *arXiv.org*, 20-Aug-2021. [Online]. Available: https://arxiv.org/abs/2108.09127. [Accessed: 13-Apr-2022].

[3] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction." [Online]. Available: https://arxiv.org/pdf/1910.05552v1.pdf. [Accessed: 13-Apr-2022].

[4] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "XDeepFM: Combining explicit and implicit feature interactions for recommender systems," *arXiv.org*, 14-Mar-2018. [Online]. Available: https://arxiv.org/abs/1803.05170v1. [Accessed: 13-Apr-2022].

[5] P. Velickovic, G. Cucurul, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv.org*, 04-Feb-2018. [Online]. Available: https://arxiv.org/abs/1710.10903. [Accessed: 13-Apr-2022].

DOI: 10.54254/2755-2721/45/20241627

[6] A. Kolan, D. Moukthika, K. S. S. Sreevani, and H. Jayasree, "Click-through rate prediction using decision tree," Mar-2020. [Online]. Available: https://www.researchgate.net/publication/339992279_Click-Through_Rate_Prediction_Using_Decision_Tree. [Accessed: 14-Apr-2022].

[7] "16.10. deep factorization machines— colab notebook ni SageMaker Studio Lab," 16.10. Deep Factorization Machines - Dive into Deep Learning 0.17.5 documentation.

[8] Sulistyowati, D. N., Yunita, N., Fauziah, S., & Pratiwi, R. L. (2020). "Implementation of Data Mining Algorithm For Predicting Popularity of Playstore Games In The Pandemic Period of COVID-19." *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer)*, 6(1), 95-100.

[9] Suwindra, I. N. P., Putra, I. K. G. D., Sudarma, M., & Sastra, N. P. (2021). "Effectiveness of neuro-fuzzy inference system for predicting player character in the Balinese serious game model." *International Journal of Fuzzy Logic and Intelligent Systems*, 21(3), 293-309.

[10] S. Yan, L. Qi, Y. Zhou, M. Peng, and GM. Shafiqur Rahman, "Joint user access mode selection and content popularity prediction in non-orthogonal multiple access-based F-RANs," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 654–666, 2019. Publisher: IEEE.