FPT UNIVERSITY

-------*-------**

FINAL REPORT

Data Science - Capstone Project - DSP391m

Topic:

Movies Recommendation System

Team Members
Pham Duc Toan

Do Thanh Dat

Do Huy Hoang

Nguyen Quang Hai

Class: AI1703-ADS

Advisor: Do Thai Giang

Hanoi, Nov 2024

TABLE OF CONTENTS

1. Team Member and Project Timeline.....1
1.1. Role and Responsibilities.....1
1.2. Project Activities and Timeline.....1
2. Introduction and Background.....4
2.1. Background.....4
2.2. Project Summary.....5
3. Data Description.....6
3.1. Source.....7
3.2. Size and Format.....7
3.3. Features.....7
4. Data Cleaning and Preprocessing.....11
4.1. Cleaning Steps.....11
4.2. Challenges and solutions.....13
5. Exploratory Data Analysis.....15
5.1. Creating master table.....15
5.2. EDA and Visualization.....16
6. Methodology.....20
6.1. Model Selection.....20
6.2. Data Splitting Strategy.....20
6.3. Feature Engineering and Selection.....21
7. Model Development.....24
7.1. Model Architecture.....24
7.2. Training Procedure.....25
8. Model Evaluation and Fine-Tuning.....29
8.1. Evaluation Metrics.....29
8.2. Hyperparameter Tuning.....30
8.3. Cross-Validation Techniques.....31
9. Results Interpretation and Visualization.....33
10. Conclusion and Recommendation.....35
10.1. Evaluation Metrics.....35
10.2. Recommendation.....36
10.3. Future Works.....36

REFERENCES.....38

1. Team Member and Project Timeline

1.1. Role and Responsibilities

Name	Project Role	Responsibilities
Phạm Đức Toàn	AI Engineer	+ Scheduling the timeline and pushing the team to complete the project on time. + Writing reports. + Defining the valid metrics.
Đỗ Thành Đạt	Data Analytic Engineer	+ Analyzing the business problem. + Collecting the data and EDA data.
Đỗ Huy Hoàng	AI Engineer	+ Building model machine learning. + Evaluating the machine learning model results.
Nguyễn Quang Hải	AI Engineer	+ Writing reports. + Creating slides for presentation. + Building model machine learning.

1.2. Project Activities and Timeline

Project Phase	Project Activity	Start Date	Completion Date

Design Thinking and Data Collection	- Brainstorm ideas and research, read articles related to the chosen topic. - Identify the most suitable model, learn and compare them through articles. - Search and collect relevant data for the project.	04/09/2024	13/09/2024
Data Cleaning	- Proceed to remove duplicate data. - Double check to ensure that all required and correct data fields have been retrieved. - Use mean, median, mode to handle missing values for each column.	14/09/2024	20/09/2024
Exploratory Data Analysis	- Based on the data, re-evaluate the necessary data fields to address the issue of incomplete data due to potential missing values. - Create a master table based on the data, which will be aggregated to serve machine learning models. - Visualize the important data fields and the relationships between those data fields.	21/09/2024	02/10/2024
Model Development, Model Evaluation and Fine-Tuning	- Using statistical probability, we can confidently determine the labels of the two main customer groups (Main and Non-main).	03/10/2024	16/10/2024

	- Employing the lazy-predict library to find the best model for the problem. - Utilizing random search for fine-tuning the selected model from the previous step. Selecting the most suitable model for the problem (one that can be easily explained to the business, and its accuracy).		
Results Interpretation and Visualization	- Running feature importance analysis to identify significant features influencing the labels. - Using the features identified earlier to cluster customers without clear labels. Employing K-means and the elbow curve method to determine the number of clusters. - Summarizing the results, indicating how many clusters there are and how many customers are in each cluster.	16/10/2024	19/10/2024
Conclusion and Recommendations	- Sitting down with the business to discuss various methods for campaigns tailored to each customer cluster and evaluation techniques such as A/B testing.	20/10/2024	25/10/2024

2. Introduction and Background

2.1. Background

In an increasingly digital world, streaming platforms and online movie services rely on personalized recommendations to enhance user experience and keep viewers engaged. Movie recommendation systems have become a cornerstone of these platforms, guiding users toward content they will likely enjoy based on their preferences and interactions. We use Light Graph Convolutional Network (LightGCN) is a graph-based model optimized for collaborative filtering, particularly well-suited for recommendation tasks. Unlike traditional Graph Convolutional Networks (GCNs), LightGCN focuses solely on message passing and embedding propagation between user and item nodes, enabling it to capture complex relationships more effectively and efficiently. With its lightweight structure, LightGCN bypasses non-linear transformations and feature transformations to focus exclusively on embedding propagation, resulting in a model that is both computationally efficient and capable of high-performance recommendation tasks.

2.2. Project Summary

Objectives

- Data Collection and Preprocessing: Gathering user-movie interaction data, constructing a user-item interaction graph, and preparing it for LightGCN's input format.

- LightGCN Model Implementation: Configuring LightGCN to generate embeddings for users and movies, focusing on layer-wise propagation without non-linear transformations. This structure allows the model to capture user-item relationships efficiently by learning user and item embeddings that can predict interactions.

- Training and Optimization: Training the model using Bayesian Personalized Ranking (BPR) loss to improve the quality of ranked recommendations. We will adjust hyperparameters, such as embedding dimensions, number of layers, and regularization, to enhance model performance.

- Evaluation and Metrics: Assessing the model's recommendation quality using precision, recall, and Normalized Discounted Cumulative Gain (NDCG)

metrics, which are standard for evaluating the ranking accuracy of recommendation systems.

- Recommendation Generation and Deployment: Implementing real-time recommendation generation based on trained embeddings and deploying the system to provide users with personalized movie suggestions.

3. Data Description

3.1. Source

The dataset used for this project is the MovieLens 100K dataset, which is a widely recognized dataset for evaluating recommendation systems. The dataset is provided by the GroupLens Research lab at the University of Minnesota and is publicly accessible. The MovieLens 100K dataset comprises 100,000 ratings from 943 users on 1,682 movies. Each user has rated at least 20 movies, ensuring a baseline level of engagement. This dataset is particularly suitable for collaborative filtering tasks and has been extensively used in the research community for bench-marking recommendation algorithms.

3.2. Size and Format

Data Overview

The MovieLens 100K dataset is compact yet rich in information, making it ideal for educational purposes and preliminary research. It consists of several files in plain text format, each serving a specific purpose in the recommendation process:

- u.data: Contains 100,000 ratings by users for movies, with each entry representing a user-movie-rating-timestamp tuple.
- u.item: Lists 1,682 movies along with their metadata, including movie ID, title, release date, video release date, IMDb URL, and genre information.
- u.user: Includes demographic information for 943 users, such as user ID, age, gender, occupation, and zip code.
- u.genre: Enumerates the movie genres, linking genre IDs to genre names
- u.occupation: Provides a list of possible user occupations.
- u.base and u.test: These files are split versions of the u.data file, used for training and testing purposes respectively.

These files are organized in a tab-separated format, ensuring easy parsing and manipulation.

3.3. Features

The MovieLens 100K dataset encompasses a diverse set of features that are crucial for building robust recommendation models: user features (userid, age, gender, occupation, zip code), movie features (movieid, title, release date, video release date, imdburl, genres) and rating features (userid, movieid, rating, timestamp). This comprehensive set of features allows for a rich exploration of user preferences and movie characteristics, enabling the development of sophisticated recommendation algorithms. By leveraging these features, we can build models that not only predict user ratings but also uncover underlying patterns and trends in the data.

4. Data Cleaning and Preprocessing

4.1. Cleaning Steps

Data cleaning and preprocessing are critical steps in preparing the MovieLens 100K dataset for use in our recommendation system. The initial stage involves parsing the dataset files to load the data into a suitable format, typically a Pandas DataFrame for ease of manipulation and analysis. The u.data file, which contains user ratings, is read and columns are labeled appropriately as user ID, movie ID, rating, and timestamp. Similarly, the u.item file is processed to extract movie metadata, including movie titles and genres. Once the data is loaded, the next step is handling missing values and inconsistencies. Although the MovieLens 100K dataset is well-curated, it is essential to verify that there are no missing values or erroneous entries. Any missing values are imputed or removed based on their significance and impact on the analysis.

4.2. Challenges and solutions

Challenge 1: Handling Missing Data

Problem: Missing data can skew analysis results, lead to biased conclusions, and degrade model performance.

Solution:

- Use mean, median, or max values based on the distribution of the data.
- Use the mode (most frequent value) for imputation.
- Remove columns with more than 30% missing values to preserve data integrity.

Challenge 2: Data Integration

Problem: Integrating data from multiple sources or tables can be complex, especially if there are inconsistencies or discrepancies.

Solution:

- Ensure that data from different sources have a consistent schema and format.
- Use unique identifiers and robust matching algorithms to merge data accurately.
- Develop rules to resolve conflicts or discrepancies in the integrated data, ensuring data consistency and reliability.

Challenge 3: Ensuring Data Quality

Problem: Poor data quality can result in inaccurate analysis and unreliable models.

Solution:

- Implement validation checks during data entry and preprocessing to catch errors early.
- Conduct regular data quality audits to identify and rectify issues.
- Establish a feedback loop with data sources to continuously improve data quality over time.

By proactively addressing these challenges with targeted solutions, the data preprocessing and cleaning process can be more efficient, leading to higher quality data and better analytical outcomes.

5. Exploratory Data Analysis

5.1. Key Findings

Objective:

Exploratory Data Analysis (EDA) is a crucial phase in any data-driven project, particularly for recommendation systems. It involves a thorough examination of the MovieLens 100K dataset to uncover initial insights, detect patterns, and identify anomalies. The primary objective is to understand the structure and relationships within the data, whichguides the subsequent modeling efforts. The first step in EDA is

to analyze the distribution of ratings. The MovieLens 100K dataset contains ratings on a scale from 1 to 5. By visualizing the distribution of these ratings, we observe that the ratings are not uniformly distributed. Typically, users tend to give higher ratings more frequently, with ratings of 4 and 5 being the most common. This skewed distribution can introduce bias in the recommendation system, favoring items that already have high ratings. Understanding this distribution helps in deciding whether to apply any normalization techniques or consider alternative loss functions during model training. Next, we analyze user behavior by examining the number of ratings per user. This analysis reveals that while some users are highly active, rating hundreds of movies, others have rated only the minimum required (20 movies). This variation in user activity can impact collaborative filtering algorithms, as highly active users provide more data points for learning user preferences. To visualize this, we can create a histogram of the number of ratings per user, which typically shows a right-skewed distribution with a long tail of highly active users.

5.2. EDA and Visualization

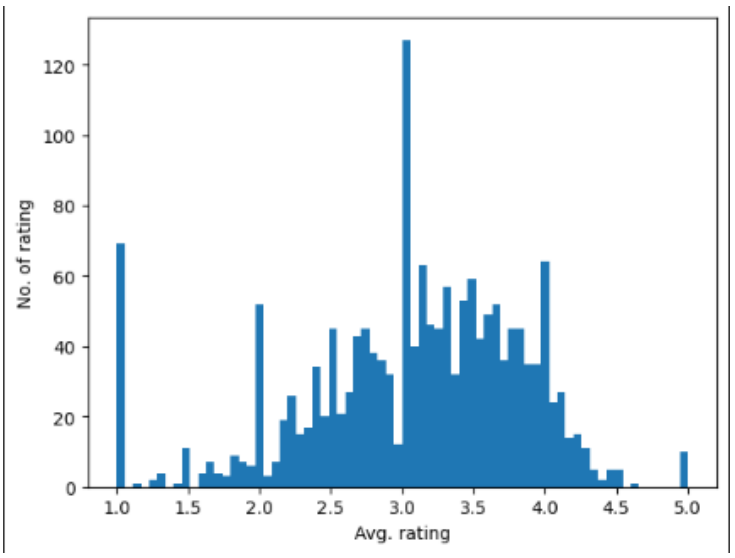
Objectives

Visualizations play a pivotal role in EDA by providing intuitive and interpretable representations of the data. Various visualization techniques are employed to illustrate the key findings discussed above.

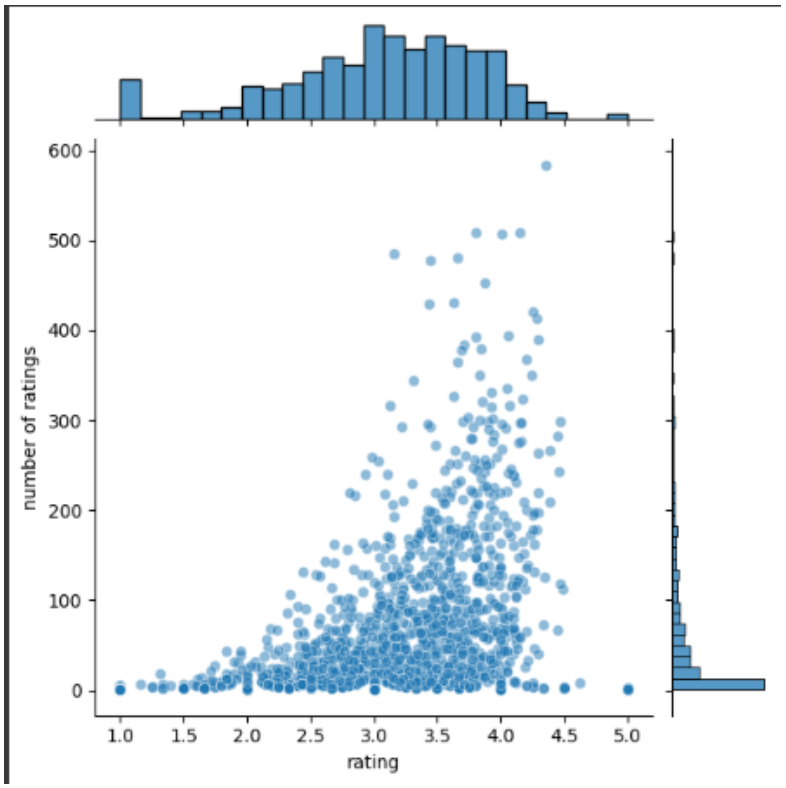
.

Visualization

- A bar plot or histogram of the rating distribution shows the frequency of each rating value. This visualization highlights the skewness towards higher ratings and helps in understanding the overall rating behavior of users.



- The plot that visualizes the relationship between two variables: the movie rating (on the x-axis) and the number of ratings each movie received (on the y-axis). The plots combine scatter plots and histograms (or density plots) to provide a detailed view of how two variables are distributed and related.



6. Methodology

6.1. Model Selection

In our project, we have chosen to utilize Graph Convolutional Networks (GCNs)

due to their superior ability to capture relationships within graph-structured data. Traditional recommendation methods like Collaborative Filtering (CF) and Content-Based Filtering (CBF) offer baseline solutions for recommending items based on user interactions and content similarities. However, these approaches often fail to capture the nuanced and complex relationships that exist within user-item interaction networks. GCNs extend the concept of Convolutional Neural Networks (CNNs) to graphs, which allows them to effectively aggregate and propagate information across nodes. In a GCN, nodes represent users and items, while edges represent interactions such as ratings. The core operation in a GCN layer can be expressed as:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \quad (6.1.1)$$

In this equation, H(l) represents the node feature matrix at layer l, with H(0) being the initial feature matrix (e.g., one-hot encoded features or pre-trained embeddings). ˜ A=A+I is the adjacency matrix with added self-loops, where A is the original adjacency matrix and I is the identity matrix. ˜ D is the degree matrix of ˜ A, W(l) is the learnable weight matrix for layer l, and sigma is a non-linear activation function such as ReLU. This layer-wise propagation rule allows the GCN to aggregate information from a node's immediate neighbors, and stacking multiple layers enables information propagation across the graph, capturing higher-order neighborhood information. This makes GCNs particularly effective for recommendation systems as they can naturally incorporate both user-user and item-item similarities into the learning process.

6.2. Data Splitting Strategy

To evaluate the performance of our GCN-based recommendation system, we implement a temporal train-test split to simulate real-world conditions where recommendations are based on past interactions. The MovieLens 100K dataset includes timestamps for each rating, allowing us to split the data chronologically. The temporal split involves using interactions up to a certain point in time for training and those occurring after that point for testing. This method ensures that the model is trained on historical data and evaluated on future interactions, providing a realistic measure of its predictive capability. Additionally, we use k-fold cross-validation on the training set to fine-tune our model. In k-fold cross-validation, the training data is divided into k subsets (folds). The model is trained on k − 1 folds and validated on the remaining fold, rotating through all

folds to ensure comprehensive evaluation. This technique helps prevent overfitting and provides a robust estimate of the model's performance. We also consider stratified sampling to maintain the distribution of ratings and demographic information in both the training and test sets. This approach ensures that the model is exposed to a representative sample of the dataset during training and is evaluated on a similarly distributed test set. 23 Chapter 6: Methodology 6.3. Feature Engineering and Selection Feature engineering is a critical step in enhancing the performance of our GCN-based recommendation system. For the MovieLens 100K dataset, we utilize a variety of features derived from user demographics, movie metadata, and user-item interactions. User features include age, gender, occupation, and zip code. These categorical variables are encoded using techniques such as one-hot encoding or embedded in continuous vectors through embedding layers. Movie features encompass genres, which are one-hot encoded, and other metadata like release year and title. Interaction features are extracted from the user-movie ratings matrix. User profiles are constructed by aggregating the ratings a user has given to various movies, capturing their preferences. Similarly, movie profiles aggregate the ratings received from different users, reflecting the movie's popularity and appeal. For our GCN model, the adjacency matrix A represents the user-movie interaction graph. Nodes in this graph correspond to users and movies, while edges indicate interactions, weighted by the rating values. We also consider constructing similarity graphs, where edges between users represent similarity in rating patterns, and edges between movies represent similarity in genres or other attributes. The feature selection process involves identifying the most relevant features that contribute to the model's performance. Techniques such as mutual information, correlation analysis, and Principal Component Analysis (PCA) are used to evaluate feature importance. Redundant or irrelevant features are eliminated to reduce model complexity and improve generalization. In summary, our methodology focuses on leveraging GCNs to model the intricate relationships in the MovieLens 100K dataset. Through careful model selection, strategic data splitting, and meticulous feature engineering, we aim to build a robust and accu-rate recommendation system. This approach ensures that our model not only captures user preferences effectively but also adapts to the dynamic nature of user behavior in a recommendation system.

7. Model Development

7.1. Model Architecture

The architecture of our Graph Convolutional Network (GCN) for movie recommendation is designed to effectively capture and propagate information through the user-item interaction graph. The model comprises multiple layers, each responsible for aggregating features from neighboring nodes and transforming them through learned weight matrices. Our model's architecture typically starts with an input layer where the feature vectors for users and items are initialized. This is followed by multiple GCN layers, each performing the aforementioned graph convolution operation. The depth of the network, or the number of GCN layers, is chosen based on the dataset's complexity and the need to capture higher-order relationships. For example, in our movie recommendation system, the first GCN layer might aggregate immediate neighbors' features, providing a basic level of interaction. The second layer would then aggregate information from two-hop neighbors, capturing more complex interactions and similarities. This hierarchical aggregation helps in building a comprehensive understanding of the graph structure. Additionally, dropout layers are incorporated between GCN layers to prevent overfitting by randomly deactivating a subset of neurons during training. Batch normalization layers are also used to stabilize and accelerate training by normalizing the inputs of each layer. The final layer of the GCN outputs node embeddings for both users and items. These embeddings are used to predict user preferences for items. Specifically, the predicted rating for a user u and item i can be computed as the dot product of their respective embeddings:

$$\hat{r}_{ui} = z_u^T z_i$$

where z_u and z_i are the embeddings of user u and item i, respectively. This prediction can be further refined using additional neural network layers or combining it with other collaborative filtering techniques.

7.2. Training Procedure

The training procedure for our GCN-based recommendation system involves several steps to ensure the model learns effectively from the data and generalizes well to unseen interactions. We employ a supervised learning approach, where the model is trained to predict user ratings for movies based on historical interactions. The loss function used during training is the Mean Squared Error (MSE) between the predicted ratings $\hat{r}_{ui}$ and the actual ratings r_{ui}:

$$L = \frac{1}{|R|} \sum_{u,i \in R} (r_{ui} - \hat{r}_{ui})^2$$

Here, R represents the set of user-item interactions in the training set. The objective is to minimize this loss function, thereby reducing the prediction error. To optimize the loss function, we use the Adam optimizer, which is well-suited for handling sparse data and adaptive learning rates. The optimizer updates the weight matrices W^l of each GCN layer based on the gradients computed during backpropagation. Training the model involves the following steps:

1. Initialization: Initialize the model parameters, including the weight matrices W^l , using a suitable initialization method such as Xavier initialization. This ensures that the weights start from a reasonable range, preventing issues with gradients during the initial stages of training.
2. Forward Pass: For each training batch, perform a forward pass through the GCN layers. Compute the node embeddings and predict the ratings $\hat{r}_{ui}$ for the user-item pairs in the batch.
3. Loss Computation: Calculate the loss using the MSE loss function. This involves computing the difference between the predicted ratings and the actual ratings.
4. Backpropagation: Compute the gradients of the loss concerning the model parameters using backpropagation. This step involves the chain rule to propagate the error gradients through the GCN layers.

5. Parameter Update: Update the model parameters using the Adam optimizer. The learning rate and other hyperparameters of the optimizer are tuned to ensure efficient convergence.
6. Regularization: Apply regularization techniques such as L2 regularization (weight decay) to prevent overfitting. Dropout layers, as mentioned earlier, are also used to deactivate neurons randomly during training, adding another layer of regularization.
7. Evaluation: Periodically evaluate the model's performance on a validation set. This helps in monitoring the training process and preventing overfitting. Evaluation metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used to measure performance.
8. Hyperparameter Tuning: Perform hyperparameter tuning using techniques such as grid search or random search to find the optimal values for the learning rate, number of GCN layers, dropout rate, and other hyperparameters. Cross-validation is used to ensure that the selected hyperparameters generalize well to unseen data.
9. Early Stopping: Implement early stopping to halt training if the model's performance on the validation set does not improve for a certain number of epochs. This prevents overfitting and reduces training time.

By following these steps, we ensure that our GCN-based recommendation system is trained effectively, capturing the complex relationships in the user-item interaction graph. The final trained model can then be used to generate personalized movie recommendations for users, leveraging the rich information encoded in the graph structure.

8. Model Evaluation and Fine-Tuning

8.1. Evaluation Metrics

Evaluating the performance of our Graph Convolutional Network (GCN)-based recommendation system is crucial to ensure it effectively predicts

user preferences and provides accurate recommendations. We employ several evaluation metrics that are standard in the field of recommendation systems. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are the primary metrics used to measure the prediction accuracy of our model. Mean Absolute Error (MAE) is another metric used to evaluate the model. It measures the average magnitude of errors in predictions, without considering their direction. Precision, Recall, and F1 Score are also employed to evaluate the recommendation system, especially in the context of binary relevance (e.g., whether a user liked or disliked an item). These metrics are defined as follows:

• Precision is the fraction of relevant items among the recommended items:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

• Recall is the fraction of relevant items that have been recommended out of all relevant items:

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

• F1 Score is the harmonic mean of Precision and Recall:

$$F1_{score} = \mathbf{2} * \frac{Precision * Recall}{Precision + Recall}$$

These metrics help us understand the trade-offs between recommending more items (Recall) and recommending relevant items (Precision).

8.2. Hyperparameter Tuning

Hyperparameter tuning is an essential step in optimizing the performance of our GCN model. Hyperparameters such as the learning rate, the number of GCN layers, the hidden layer size, dropout rates, and the regularization parameters significantly influence the model's ability to generalize. The process of hyperparameter tuning involves searching through a predefined space of hyperparameters and evaluating the model's performance on a validation set. Techniques such as Grid Search and Random Search are commonly used. In Grid Search, we define a grid of hyperparameter values and exhaustively evaluate the model for each combination. For example, if we are tuning the learning rate and the number of layers, we might have a grid like this: Each combination is evaluated using

Learning Rate	Number of Layers
0.001	2
0.001	3
0.005	2
0.005	3

cross-validation, and the combination that yields the best performance on the validation set is selected. Random Search improves efficiency by randomly sampling a subset of the hyperparameter space. This method can find good hyperparameter values faster than Grid Search, especially when the hyperparameter space is large. We also use Bayesian Optimization for hyperparameter tuning. Bayesian Optimization models the function that maps hyperparameters to the objective (e.g., validation loss) and uses this model to choose hyperparameters likely to improve performance. This method is particularly effective for expensive evaluations. Regularization techniques, such as L2 regularization (weight decay), prevent overfitting. The L2 regularization term is added to the loss function:

$$L_{reg} = L + \lambda \sum_{1} ||W^{(1)}||_2^2$$

where λ is the regularization coefficient and $W^{(1)}$ are the weights l-th layer.

8.3. Cross-Validation Techniques

Cross-validation is a robust technique for assessing the generalizability of our GCN model. K-Fold Cross-Validation is the most commonly used method, where the dataset is divided into K equally sized folds. The model is trained K times, each time using a different fold as the validation set and the remaining K - 1 folds as the training set. The performance metric is averaged over the K runs:

$$CV_{score} = \frac{1}{K} \sum_{i=1}^{K} Metric_i$$

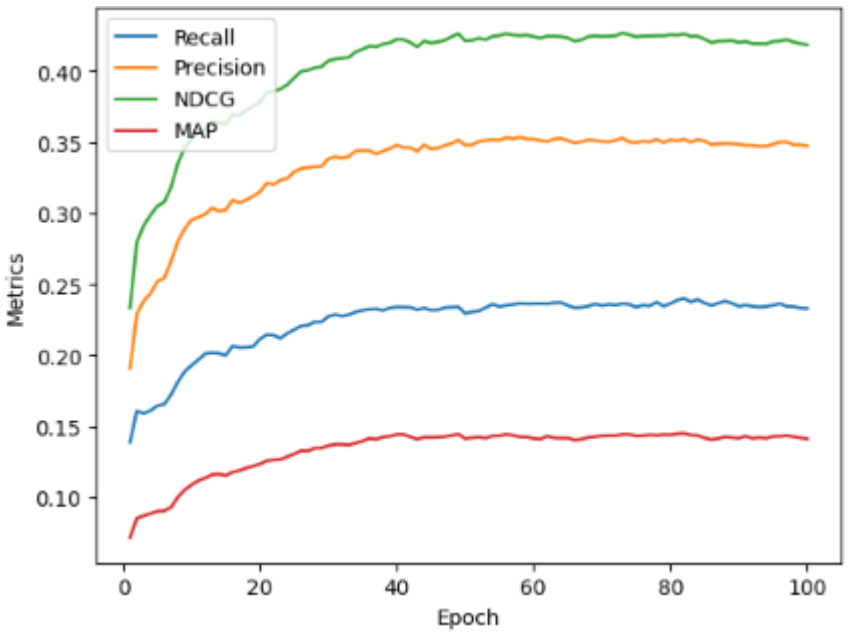
where Metrici is the performance metric for the i-th fold. This method helps in

reducing variance and bias in the model evaluation. In the context of recommendation systems, Leave-One-Out Cross-Validation (LOOCV) is often used. Here, each user-item interaction is used once as the validation set while the rest are used for training. This method provides a thorough evaluation but can be computationally expensive. Timebased Cross-Validation is another technique, particularly relevant for dynamic recommendation systems. The dataset is split based on time, ensuring that the validation set contains interactions that occurred after the training set. This method simulates real-world scenarios where the model needs to predict future interactions based on past data. By employing these rigorous evaluation and fine-tuning techniques, we ensure that our GCN-based recommendation system achieves high accuracy and generalizes well to unseen data, providing users with relevant and personalized movie recommendations.

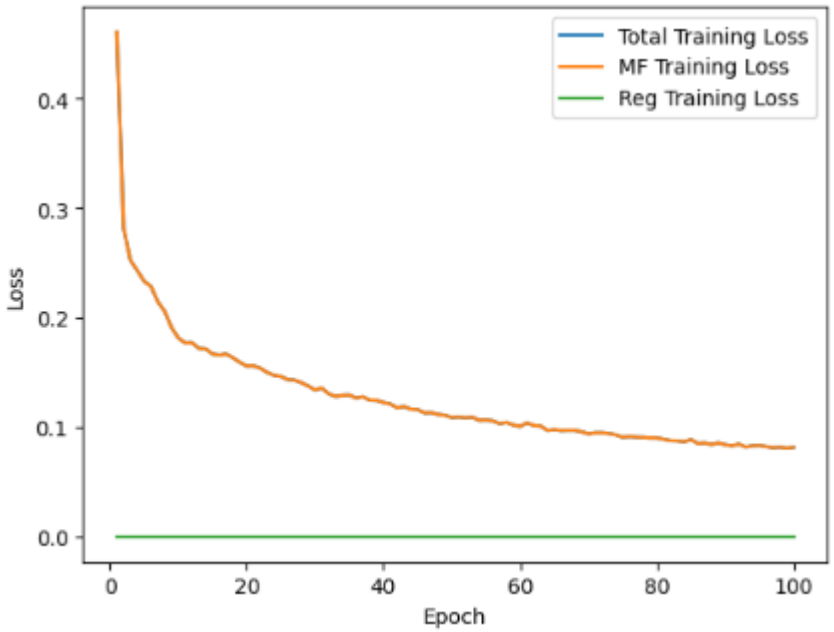
9. Results Interpretation and Visualization

Visualizations play a crucial role in understanding and interpreting the results of our GCN-based recommendation system. We employed several types of visualizations to illustrate the performance and insights derived from our model. Rating Distribution: A histogram of the predicted versus actual ratings provides a visual comparison of how well the model's predictions align with the actual ratings. This visualization highlights the concentration of ratings around certain values, allowing us to see if the model tends to overestimate or underestimate ratings. Precision-Recall Curve: The Precision-Recal curve is a graphical representation of the trade-off between precision and recall for different threshold settings. It helps us understand the performance of the recommendation system across various thresholds, which is crucial for fine-tuning the model. A high area under the Precision-Recall curve indicates a model that maintains high precision and recall across different thresholds. Confusion Matrix: A confusion matrix for binary classification (like a like/dislike scenario) helps visualize the number of true positives, false positives, true negatives, and false negatives. This matrix provides a detailed breakdown of how well the model distinguishes between relevant and non-relevant items, which is particularly useful for understanding the model's strengths and weaknesses. User and Item Embeddings: Visualizing the embeddings of users and items learned by the GCN can provide insights into how

the model perceives user preferences and item characteristics. Techniques such as t-SNE or PCA can reduce the dimensionality of these embeddings to 2D or 3D, allowing us to plot them and observe clusters of similar users or items. This can reveal interesting patterns, such as groups of users with similar preferences or clusters of movies with similar genres.



Performance numbers comparison chart over time



Loss function graph during training

10. Conclusion and Recommendation

10.1. Evaluation Metrics

Based on the comprehensive segmentation criteria provided, customers are categorized into four distinct groups to better understand their behaviors and potential relationship with the bank.

Firstly, the Main Group encompasses customers actively involved in CASA transactions and displaying high scores, indicating a strong affinity with the bank. These individuals are likely to engage with multiple core products or rely on a single core product with high scores, highlighting their significant contribution to the bank's business. By nurturing relationships with these customers through targeted marketing efforts and personalized services, the bank can enhance loyalty and foster long-term engagement.

In contrast, the Likely Main Group comprises customers who currently engage in CASA transactions and are forecasted to transition into main customers. Although their usage patterns may vary, ranging from interaction with one core product with average scores to engagement across multiple products, they represent a promising segment for future growth. By identifying and catering to the evolving needs of these customers, the bank can capitalize on opportunities to deepen relationships and increase their lifetime value.

Conversely, the Less Likely Main Group includes customers with CASA transactions but are deemed less likely to become main customers based on predictive modeling. These individuals may exhibit behaviors indicative of a weaker relationship with the bank, such as utilizing one core product with low scores or engaging with multiple products, including a core product with low scores. While their current engagement levels may be suboptimal, targeted strategies aimed at enhancing satisfaction and addressing pain points could potentially convert them into more valuable customers over time.

Lastly, the Non-Main Group encompasses customers with CASA transactions but behaviors suggesting a lower propensity to become main customers. Some members of this group may display disloyalty by directing a higher percentage of their funds to other banks compared to what is retained

within the bank. Others may exhibit minimal engagement by abstaining from the use of any core or ancillary bank products. To mitigate attrition and maximize retention, the bank could implement strategies to re-engage these customers, such as tailored incentives or improved service offerings.

In conclusion, the segmentation of customers into these distinct groups provides invaluable insights for the bank to tailor its marketing strategies, product offerings, and customer engagement initiatives. By prioritizing efforts to cultivate relationships with existing main customers, capitalize on the potential of likely main customers, and strategically address the needs of less likely main and non-main customers, the bank can optimize its customer retention and growth strategies for sustainable long-term success.

10.2. Recommendation

In our endeavor to optimize customer engagement and product offerings, we aim to conduct A/B testing across different customer segments, each with specific characteristics and preferences. The primary objective of this A/B testing is to identify the most suitable customer value propositions (CVP) for each group, determine the effectiveness of increased interest rates for FD products, evaluate cashback campaigns for new and existing card products, and explore the combined impact of these strategies.

For the groups characterized by high FD, low card amount, low FD, high card amount, and similar FD and card amounts, we will use the notification IBMB channel. The goal here is to find the suitable CVP for each group by testing various offers and messaging strategies. Additionally, we aim to measure the impact of increased interest rates for FD products and cashback campaigns on customer engagement and satisfaction. For instance, we can test different value propositions such as premium features, personalized financial advice, and exclusive rewards, and then measure engagement rates, response rates, and customer satisfaction to identify the most appealing CVP.

In parallel, for groups with average FD, average loan, and average MCC, we will utilize multiple channels including email, SMS, calls, and in-person interactions at transaction offices. This segment of testing aims to determine the most effective communication channels and CVPs for these groups. We will

evaluate the impact of increased interest rates for FD products, decreased interest rates for loan products, and cashback campaigns for card products. By testing tailored financial solutions, loyalty programs, and targeted offers, we can track engagement metrics and customer feedback to determine the optimal value proposition.

To specifically address the goal of increasing interest rates for FD products, we will conduct A/B tests across all groups by comparing FD products with increased interest rates against control groups with current rates. Monitoring uptake rates, deposit amounts, and overall customer interest will help us analyze the impact on both engagement and profitability. Similarly, to evaluate the effectiveness of cashback campaigns for new and existing card products, we will implement these campaigns across all groups, testing different cashback amounts and conditions. By measuring card application rates, transaction volumes, and customer satisfaction, we can assess the effectiveness of these campaigns.

Furthermore, we plan to explore the combined strategies of increasing FD interest rates and implementing cashback campaigns. For the Average Loan group, we will also examine the effect of decreased interest rates for loan products. Conducting A/B tests that combine these strategies will allow us to assess their combined impact on customer engagement, satisfaction, and overall financial performance.

For each A/B test, we will define clear hypotheses, establish specific and measurable goals, and select appropriate metrics such as engagement rates, conversion rates, and customer satisfaction scores. By setting up control and treatment groups randomly, we ensure unbiased results. Rigorous analysis using statistical methods will help us evaluate the effectiveness of each strategy, looking for significant differences in performance metrics to determine the best approaches.

By implementing targeted A/B testing across these customer segments and communication channels, we aim to uncover the most effective strategies for enhancing customer engagement and optimizing product offerings. This comprehensive approach will enable us to tailor our value propositions, financial product terms, and marketing campaigns to better meet the needs and preferences of our diverse customer base, ultimately driving increased satisfaction and loyalty.

10.3 Future Works

Looking ahead, several avenues for future work can build upon the findings and recommendations of this project. One promising direction is the integration of temporal dynamics into the GCN model. User preferences often change over time, and capturing these dynamics can lead to more accurate and timely recommendations. Temporal Graph Convolutional Networks (T-GCNs) or Recurrent Graph Neural Networks (RGNNs) can be explored to model these temporal aspects. Another area for future work is the incorporation of explainability into the recommendation system. Understanding why a particular recommendation was made is crucial for user trust and satisfaction. Techniques such as attention mechanisms or interpretable embeddings can be employed to provide explanations for the recommendations, enhancing the transparency of the model. Additionally, investigating the scalability of the GCN model is important for practical applications. As the size of the user-item graph grows, efficient training and inference become critical. Exploring distributed training techniques or graph sampling methods can help address scalability issues, making the model suitable for large-scale datasets. Collaborative research and development of hybrid models that combine the strengths of GCNs with other recommendation techniques, such as collaborative filtering or content-based filtering, can also be pursued. Hybrid models can leverage the complementary advantages of different approaches, potentially leading to more robust and versatile recommendation systems. In summary, the conclusion of this project highlights the effectiveness of GCNs in recommendation systems, while the recommendations and future work outline a clear path for further enhancements and broader applications. By continuing to explore and refine GCN-based models, we can develop more accurate, scalable, and interpretable recommendation systems that cater to diverse user needs and preferences.

REFERENCES

References

[1] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: IEEE Transactions on Neural Networks and Learning Systems 32.1 (2021), pp. 4–24. doi: 10.1109/TNNLS.2020.2978386.

[2] Rex Ying et al. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems”. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD ’18. London, United Kingdom:

Association for Computing Machinery, 2018, pp. 974–983. isbn: 9781450355520. doi: 10.1145/3219819.3219890. url: https://doi.org/10.1145/3219819.

3219890.

[3] Jianxun Lian et al. “xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems”. In: Proceedings of the 24th ACM SIGKDD

International Conference on Knowledge Discovery & Data Mining. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 1754–1763.

isbn: 9781450355520. doi: 10.1145/3219819.3220023. url: https://doi.org/

10.1145/3219819.3220023.

[4] Fan-Yun Sun et al. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. 2020. arXiv: 1908 .01000 [id=’cs.LG’ fullname =′ M achineLearning′isactive = T ruealtname = Noneinarchive =′cs′isgeneral = F alsedescription =′ P apersonallaspectsofmachinelearningre

[5] Rianne van den Berg et al. Graph Convolutional Matrix Completion. 2017. arXiv:1706.02263[id=’stat.ML’fullname=′MachineLearning′isactive=Truealtname =Noneinarchive=′stat′isgeneral=Falsedescription=′Coversmachinelearningpapers(s upervisupervisedlearning,graphicalmodels,reinforcementlearning,bandits,highdim ensionalinferen

[6] Xiangnan He et al. “Neural Collaborative Filtering”. In: Proceedings of the 26th International Conference on the World Wide Web. WWW ’17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.isbn:9781450349130.doi: 10.1145/3038912.3052569. url: https://doi.org/10.1145/3038912.3052569.

[7] Tianyi Lin et al. “Sparsemax and Relaxed Wasserstein for Topic Sparsity”. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. WSDM ’19. Melbourne VIC, Australia: Association for Computing Machinery,2019,pp.141–149.isbn:9781450359405.doi:10.1145/3289600.3290957. url: https://doi.org/10.1145/3289600.3290957.

[8] Yehuda Koren et al. “Matrix Factorization Techniques for Recommender Systems”.In: Computer 42.8 (2009), pp. 30–37. doi: 10.1109/MC.2009.263.

[9] Meng Qu et al. GMNN: Graph Markov Neural Networks. 2020. arXiv: 1905 . 06214[id=’cs.LG’fullname=′MachineLearning′isactive=Truealtname=Noneinarchive=′cs′isgeneral= F alsedescription =′ P apersonallaspectsofmachinelearningre

[10] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. 2017. arXiv: 1609.02907 [id=’cs.LG’ fullname =′ MachineLearning′isacT ruealtname = Noneinarchive =′cs′isgeneral = F alsedescription =′ P apersonallaspectsofma

[11] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. 2016. arXiv:1611.07308 [id=’stat.ML’ fullname=′MachineLearning′isactive = T ruealtnam=Noneinarchive=′stat′isgeneral=Falsedescription=′Coversmachinelearningpapers(supervisupervisedlearning,graphicalmodels,reinforcementlearning, bandits, highdimensionalinferen