GoodGame (GG): Personalized Video Game Recommendation Engine

Karagiannis Christos, Tsigkas Emmanouil-Angelos, Nanousis Panagiotis

Final Project

Abstract
This paper documents our progress on developing a recommendation system, with a specific focus on product recommendations within the video game industry. Our primary aim is to design a simple engine tailored to the unique characteristics of video games.

1 Introduction

In the contemporary landscape of video game consumption, the vast array of available titles presents a significant challenge for players seeking to discover new games that align with their preferences. This problem of game discovery is particularly pertinent today, given the abundance of titles spanning various genres, platforms, and styles. Navigating this plethora of options can often lead to decision paralysis or suboptimal gaming experiences for players.

1.1 Importance of the Problem

Effective game recommendation systems address this challenge by providing personalized suggestions tailored to individual players' tastes and preferences. By leveraging machine learning algorithms and user behavior data, these systems empower players to discover new games that resonate with their gaming preferences, thereby enhancing their overall gaming experience. Robust game recommendation systems not only provide convenience but also foster a deeper connection between players and the games they play, leading to increased engagement, satisfaction, and retention within gaming communities.

1.2 Motivation

Our motivation for pursuing this problem arises from the recognition of the transformative impact that personalized game recommendations can have on the gaming experience. As enthusiasts of video games ourselves, we understand the frustration of sifting through a vast array of titles in search of the perfect game. By developing a sophisticated game recommendation system, we aim to streamline the game discovery process and empower players to discover new and exciting titles tailored to their unique preferences.

1.3 Background

The proliferation of digital distribution platforms, such as Steam, has revolutionized the way games are distributed, consumed, and discovered. These platforms collect vast amounts of data on user interactions, including game purchases, playtime, reviews, and preferences. Leveraging this rich dataset, our goal is to develop a recommendation system capable of analyzing user behavior patterns and generating personalized game recommendations.

1.4 Input and Output

The input to our recommendation algorithm is a dataset containing user interactions with video games. This dataset includes information such as user IDs, game titles, behaviors (e.g., purchase, play), and corresponding values (e.g., playtime). Leveraging machine learning techniques such as collaborative filtering and neural networks, we process this data to generate personalized game recommendations for each user.

Specifically, our algorithm utilizes the user's historical gaming interactions and preferences as input to predict top game titles that align closely with their interests. The output of our recommendation system is a ranked list of suggested games, providing users with curated recommendations tailored to their preferences. By leveraging advanced machine learning techniques, we aim to enhance the gaming experience for users and facilitate the discovery of new and exciting games within the gaming community.

2 Literature Review

2.1 Introduction to Game Recommendation Systems

The rapid expansion of the video game industry has led to an information overload, necessitating the development of sophisticated game recommendation systems. These systems are designed to leverage the vast amounts of data collected by video game platforms to enhance user experience by providing personalized game suggestions. The complexity and variety of games available make it challenging to navigate the gaming landscape, which is where these recommendation systems play a pivotal role, guiding users to games that match their preferences and playing styles.

2.2 State-of-the-Art Recommender Models

In their pioneering work, Cheuque, Guzmán, and Parra (2019) [11] explored the potential of state-of-the-art recommender models such as Factorization Machines (FM), deep neural networks (DeepNN), and DeepFM in the context of online video game platforms like STEAM. Their study is significant in evaluating the ranking accuracy and the diversity/novelty of the recommendation lists produced by these algorithms. The findings revealed that DeepNN outperformed other models, suggesting that high-order interactions are crucial in game recommendation tasks. Interestingly, the study also found that sentiment extracted from game reviews was not as influential in recommendations as previously thought, challenging common assumptions in the field.

2.3 Collaborative Filtering in Game Recommendations

Bunga, Batista, and Ribeiro (2022)[10] focused on collaborative filtering algorithms for a video game-oriented recommendation system using data from Steam. Their approach of transforming implicit feedback into explicit ratings and evaluating algorithms with metrics like RSME and MAE provides valuable insights into the effectiveness of computationally efficient algorithms in game recommendations. This study underscores the potential of collaborative filtering in enhancing user experience on gaming platforms, demonstrating its ability to accurately predict user preferences and improve game discovery.

2.4 Applications and Challenges in Digital Game Recommendations

Wu, Kolen, Aghdaie, and Zaman (2017) [12] offered an in-depth look into the recommendation systems at Electronic Arts, highlighting the unique applications and challenges in digital game recommendations. Their work illustrates how diverse player data across various games can be utilized to provide intelligent recommendations, including game purchases, map selections, and difficulty settings. This study is crucial in understanding the multifaceted nature of game recommendation systems and their impact on player engagement and retention, emphasizing the need for personalized and context-aware recommendation strategies in the gaming industry.

2.5 Machine Learning in Item Recommendations for Video Games

Bertens, Guitart, Chen, and Periáñez (2018) [9] evaluated machine learning algorithms for recommending in-game purchases. Their comparison of ensemble-based models and deep neural networks sheds light on the effectiveness of these approaches in predicting user preferences for in-game items. This research is particularly relevant in the context of free-to-play games, where in-game purchases are a significant aspect of the gaming experience. The study provides insights into how machine learning can be leveraged to enhance the commercial aspects of gaming, benefiting both game developers and players.

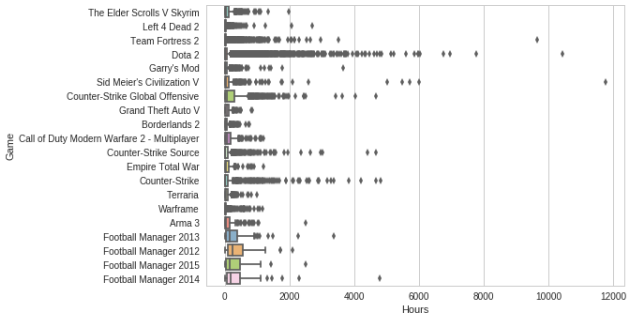
2.6 Neural Networks in Personalized Game Recommendations

Yang, Liu, Wang, Wang, Fan, and Yu (2022) [13] investigated the role of neural networks in personalized game recommendations on the Steam platform. They proposed a Social-aware Contextualized Graph Neural Recommender System (SCGRec), focusing on personalization, game contextualization, and social connection. This study underscores the importance of integrating these aspects to enhance the effectiveness of game recommendation systems. The use of graph neural networks represents a significant advancement in recommendation technology, offering a more nuanced and interconnected approach to understanding user preferences and social dynamics.

3 Dataset & Features Analysis

3.1 Overview of Datasets Utilized

Our analysis leverages two primary datasets: the Tamber Team dataset and the SteamGames database by MEXWELL. The Tamber Team dataset, sourced from the public Steam API, encompasses **12,393** unique users, **200,000** interactions, and **5,155** games. It records user behaviors through columns such as **user-id**, **game-title**, **behavior-name**, and **value**. Conversely, the SteamGames dataset serves as a comprehensive repository of all games on Steam, boasting a collection of **71,000** games. It features detailed metadata for each game, including title, release date, developer, publisher, genre, user reviews, ratings, and system requirements, thereby encompassing a diverse spectrum of gaming genres.



3.2 Data Preparation and Refinement

The initial data analysis revealed several imperfections necessitating refinement to optimize our recommendation engine:

1. **Outlier Identification and Removal:** Outliers potentially skewing our analysis were diligently identified and subsequently removed.

2. **Duplicate Data Handling:** To preserve data integrity, duplicate rows were systematically eliminated.

3. **Normalization and Scaling:** A rigorous normalization and scaling process was undertaken to ensure consistency and distribution uniformity within the dataset.

4. **Game Selection Synchronization:** Discrepancies in game listings across both datasets necessitated synchronization, culminating in the curation of a shared list comprising **2,500** games from an initial pool of **5,250**.

5. **PCA Analysis and Categorical Data Handling:** Although Principal Component Analysis (PCA) presented an enticing prospect for dimensionality reduction, the amalgamation of numerical (e.g., **user-id**, **behavior**, **time-played**) and categorical data (e.g., **game-title**) rendered its application impractical. The potential surge in dimensions resulting from one-hot encoding prompted us to forego PCA in our data preparation approach.

3.3 Feature Extraction

Feature extraction played a pivotal role in our analysis, aiming to distill meaningful insights from the dataset. Specifically, we employed Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to reduce dimensionality. Considering PCA for dimensionality reduction posed a challenge due to the mix of numerical (user-id, behavior, time-played) and categorical data (game-title). The impractical increase in dimensions from one-hot encoding led us to exclude PCA from our data preparation approach.

3.4 Databases Citation

The Tamber Team dataset *steam200k* and SteamGames database *steamgames_dataset* by MEXWELL are invaluable resources that have significantly contributed to our research. We gratefully acknowledge the creators of these datasets for making them publicly available.

3.4.1 Example from the Tamber Team Dataset after Pre-Processing

	user-id	game-title	time-played
0	5250	Alien Swarm	4.9
1	5250	Cities Skylines	144.0
2	5250	Counter-Strike	0.0
3	5250	Counter-Strike Source	0.0
4	5250	Day of Defeat	0.0
...	...	...	...
128799	309626088	Age of Empires II HD Edition	6.7
128800	309812026	Counter-Strike Nexon Zombies	0.0
128801	309812026	Robocraft	0.0
128802	309824202	Dota 2	0.7
128803	309903146	Dota 2	0.2

128804 rows × 3 columns

4 Methods

In our project, we leverage a combination of learning algorithms to develop an effective recommendation system for video games. The main types of algorithms employed include collaborative filtering, matrix factorization, and deep learning approaches such as neural networks.

4.1 Content-Based filtering

Content-based filtering recommends items to users based on the characteristics of items themselves. It analyzes textual features such as genres, tags, and descriptions, converting them into numerical vectors using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. The TF-IDF score for a term t in a document d is computed as $\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$, where TF is the term frequency and IDF is the inverse document frequency.

Cosine similarity is then calculated between pairs of item vectors to measure their similarity based on textual attributes. The cosine similarity between two vectors A and B is given by $\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$, where $A \cdot B$ represents the dot product of vectors A and B, and $\|A\|$ and $\|B\|$ denote the Euclidean norms of vectors A and B, respectively.

In our recommendation system, we utilize TF-IDF vectorization and cosine similarity computation to generate personalized recommendations. By computing similarity scores between input items and all other items in the dataset, we offer users recommendations that closely align with their interests and preferences.

4.2 Collaborative Filtering

Collaborative filtering utilizes user interaction data to generate recommendations, employing both user-based and item-based filtering mechanisms. User-based filtering identifies similarities between users based on their interaction histories, while item-based filtering recommends items similar to those a user has previously liked. Despite its ability to uncover latent patterns, collaborative filtering faces challenges like the cold start problem, where new users or items with limited data are difficult to recommend.

Collaborative filtering is implemented using k-nearest neighbors (kNN) algorithm. After splitting the dataset into training and test sets, we create a user-game matrix from the training data. Using the sparse matrix representation, we set up the kNN model to compute cosine similarities between games. Based on user-specified game titles, we retrieve recommendations by finding the nearest neighbors in the game space and accumulating their inverse distances as weights. The top recommended games are then sorted based on these weighted distances.

```
function GET_GAME_RECOMMENDATIONS(game_titles, data_matrix, model, n_recommendations)
    game_recommendations ← Counter()
    distance_weighting ← {}
    for game in game_titles do
        if game in data_matrix.columns then
            game_idx ← index of game in data_matrix
            distances, indices ← model.kneighbors(data_matrix[:, game_idx], n_neighbors = n_recommendations + 1)
            for dist, idx in zip(distances.flatten()[1 :], indices.flatten()[1 :]) do
                recommended_game ← data_matrix.columns[idx]
                if recommended_game ≠ game then
                    inverse_distance ← 1/(dist + 0.0001)
                    distance_weighting[recommended_game] ← distance_weighting.get(recommended_game, 0) + inverse_distance
                end if
            end for
        else
            return ["Game title not found: game"]
        end if
    end for
    return Top n_recommendations games based on weighted distances
end function
```

4.3 Hybrid Approach in Recommendation Systems

The hybrid approach in recommendation systems synergistically combines the strengths of content-based and collaborative filtering methods to enhance recommendation accuracy and relevance. This integration addresses the limitations inherent in each individual method by leveraging the content attributes of items from content-based filtering and the user interaction patterns from collaborative filtering.

The hybrid approach can be formulated as follows:

Algorithm 2 Hybrid Recommendation Algorithm

```
for each user u do
    Compute content-based recommendations R_cb(u) based on user preferences and item features
    Compute collaborative filtering recommendations R_cf(u) based on user-item interactions
    Combine recommendations: R_hybrid(u) = α · R_cb(u) + (1 − α) · R_cf(u)
    Return top N recommendations R_hybrid(u)
end for
```

Here, α represents the weight assigned to content-based recommendations, and $(1-\alpha)$ represents the weight assigned to collaborative filtering recommendations. By adjusting the value of α, the hybrid approach can be customized to prioritize one method over the other.

The hybrid approach combines the strengths of content-based and collaborative filtering methods to mitigate their respective weaknesses. Content-based filtering relies on item features to make recommendations, which may lead to over-specialization and lack of novelty. On the other hand, collaborative filtering relies on user-item interactions, which may suffer from popularity bias and cold start problems.

By integrating both methods, the hybrid approach can provide more diverse and accurate recommendations. Content-based filtering ensures that recommendations are tailored to the user's preferences and item attributes, while collaborative filtering captures user preferences based on their interactions with similar users.

Retrieval Methods

- **Filtering Based on Criteria:** This involves selecting data from the dataset that meets certain criteria, such as genre or user preferences. Techniques like boolean indexing in databases.
- **Search Algorithms:** Implementing search algorithms that can efficiently query large datasets to find relevant items. This might involve database query languages like SQL or search algorithms in programming languages.
- **Data Preprocessing:** Ensuring the data is in the right format for retrieval, which might include data cleaning, normalization, or transformation.

Scoring Methods

- **Heuristic Approaches:** Simple rule-based systems where scores are assigned based on specific criteria, like the presence of certain features or user ratings.
- **Weighted Scoring:** Assigning different weights to various attributes of the items and calculating a cumulative score.
- **User Profile Matching:** Scoring can also involve comparing items with a user's profile or past behavior to determine relevance.

Re-ranking Methods

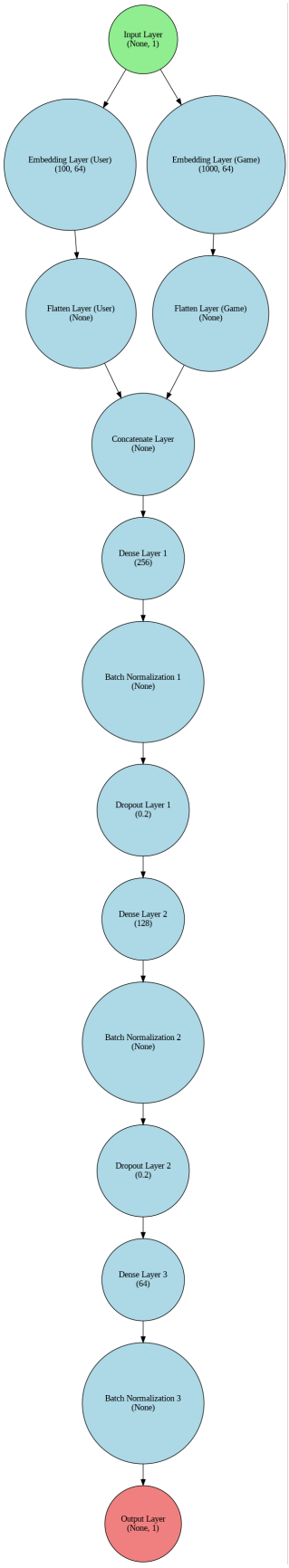
- **Diversity Enhancement:** Adjusting rankings to ensure a diverse set of results.
- **Multi-Criteria Sorting:** Sorting the retrieved items based on multiple factors, such as relevance, popularity.
- **Personalization Algorithms:** Tailoring the final set of items to the individual user, which could involve learning user preferences over time.
- **Context-Aware Re-ranking:** Modifying the rankings based on the current context, such as the user's location, time of day, or recent activities.

4.4 Neural Network Recommendation Engine

The recommendation engine is designed to provide personalized game recommendations to users by leveraging neural network algorithms. Unlike conventional recommendation systems that typically rely on either collaborative filtering or content-based approaches, this engine combines regression and classification techniques to enhance the accuracy of user preference predictions.

4.4.1 Model Architecture

The neural network models utilized in the recommendation engine exhibit an architectural framework comprising embedding layers, dense layers, batch normalization, and dropout regularization.



1. **Regression Model**:
    - The regression model architecture incorporates embedding layers for encoding user and game inputs, followed by dense layers with Rectified Linear Unit (ReLU) activation functions and batch normalization to capture intricate user-game interactions. Dropout regularization is employed to mitigate overfitting tendencies. The model's output layer employs linear activation, making it conducive for predicting continuous ratings. Training is performed using the Mean Squared Error (MSE) loss function.
2. **Classification Model**:
    - The classification model shares a similar architectural structure with the regression model, featuring embedding layers, dense layers with ReLU activation, batch normalization, and dropout regularization. However, it includes an additional sigmoid activation function in the output layer to facilitate binary classification tasks. The model is trained using binary cross-entropy loss to predict user preferences as either positive or negative.

4.4.2 Model Evaluation

Upon training, the trained regression and classification models are deployed for recommendation system evaluation. Recommendations are generated based on predicted ratings, and the system's performance is scrutinized using various metrics tailored to the respective tasks. Mean Absolute Error (MAE) is employed for regression tasks, measuring the average absolute difference between predicted and actual ratings. For classification tasks, the accuracy score serves as a performance metric, indicating the proportion of correctly classified instances.

4.5 Librarires Used

1. TensorFlow [8]
2. pandas [3]
3. scikit-learn [5]
4. scikit-optimize [6]
5. NumPy [2]
6. Matplotlib [1]
7. Seaborn [7]
8. Plotly [4]

5 Experiments, Results & Discussion

5.1 Reranking, Retrieval, Scoring

Integrating reranking, retrieval, and scoring methods into recommendation engines proved challenging due to various factors. The complexity of combining diverse algorithms for reranking and retrieval required extensive research and experimentation, compounded by the intricacies of each method, such as collaborative filtering, content-based filtering, hybrids, and neural networks. Additionally, developing robust evaluation frameworks and ensuring scalability posed significant hurdles. Given resource constraints and time limitations, prioritizing the implementation of collaborative and content-based approaches, along with hybrid and neural network models, took precedence over integrating reranking and retrieval scoring methods. Consequently, while these techniques offer potential enhancements to recommendation accuracy, their incorporation was postponed to allow for more comprehensive development and evaluation in future iterations.

5.2 Filtering

Content-based filtering is a valuable approach in recommendation systems, especially when dealing with sparse user-item interactions. By leveraging rich item descriptions and characteristics, content-based methods can make personalized recommendations without relying heavily on past user behavior. However, this approach also has its challenges. One major concern is the tendency toward over-specialization, where the system may recommend only similar items, leading to a lack of diversity in recommendations. In our experiments, we observed moderate results with a Mean Reciprocal Rank (MRR) score of 0.09. Despite its limitations, content-based filtering laid the groundwork for our subsequent exploration of hybrid methods.

Collaborative filtering is a fundamental technique in recommendation systems, leveraging user-item interactions to generate personalized recommendations. Despite its effectiveness, collaborative filtering encounters several challenges, such as the cold start problem, where new users or items with limited interaction data pose difficulties for accurate recommendations. Our experiments revealed instances of this challenge, particularly evident when testing with a single game, 'Max Payne,' which led to irrelevant recommendations like 'Platypus.' Additionally, collaborative filtering can suffer from popularity bias, favoring popular items over niche content. Despite these challenges, collaborative filtering remains essential in recommendation systems, offering a dynamic recommendation experience based on collective user data. However, the MRR score for this method was relatively low at 0.15 when utilizing the cosine method of KNN, indicating room for improvement in recommendation accuracy and relevance.

The hybrid approach represents a powerful solution for recommendation engines, combining the strengths of collaborative and content-based filtering while mitigating their respective limitations. Our experiments demonstrated significant improvements in recommendation accuracy, as evidenced by the collaborative filtering model's enhanced performance, with its MRR score increasing by 300% to 0.47. This substantial improvement underscores the effectiveness of the hybrid approach in addressing the challenges faced by individual filtering methods. By leveraging both user-item interactions and item characteristics, the hybrid approach provides a more comprehensive and sophisticated recommendation engine, capable of delivering personalized and relevant recommendations to users.

5.3 Neural Network Optimization

The hyperparameter optimization process using Bayesian optimization with Gaussian Processes (GP) was pivotal in refining the neural network model for recommendation. We tailored a parameter space encapsulating critical hyperparameters such as `embedding_size`, `dense_layer_size`, `learning_rate`, and `batch_size`. Employing a custom callback function, `no_improvement_callback`, we monitored the optimization progress, providing periodic updates on the current best value and parameters. Through the `gp_minimize` function, the objective function `evaluate_model` was systematically minimized, aiming to enhance the model's predictive accuracy. The optimization process was parallelized to expedite computations, ultimately yielding improved hyperparameters. Visualizing the convergence of the optimization process using Plotly provided valuable insights into the evolution of each hyperparameter value and the objective function value over iterations.

5.3.1 Predictions vs. Actual Time Played

1. **Game 1:** Call of Duty World at War, Predicted = [1.3657123], Actual = 39.0

2. **Game 2:** Worms Revolution, Predicted = [1.7594501], Actual = 6.6

3. **Game 3:** Sniper Elite Nazi Zombie Army, Predicted = [0.40791142], Actual = 1.9

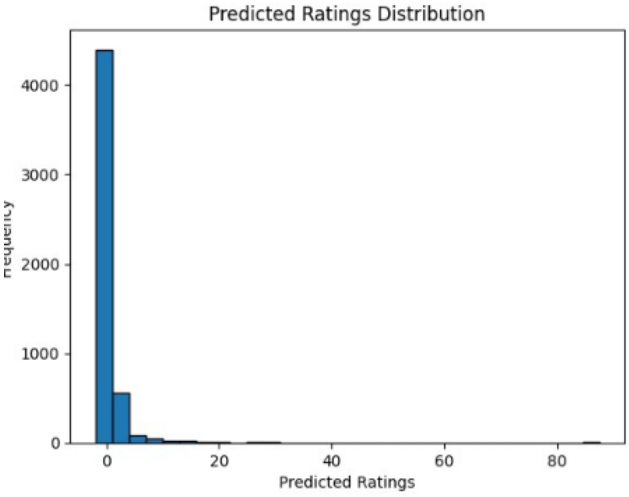
4. **Game 4:** Borderlands 2, Predicted = [8.133287], Actual = 1.8

5. **Game 5:** Mirror's Edge, Predicted = [0.23078263], Actual = 2.0

1. **Borderlands 2:** Predicted = 8.133, Actual = 1.8. The model predicted a significantly higher playtime for Borderlands 2 compared to its actual value. This discrepancy highlights the limitations of the model in accurately predicting playtime for games with complex gameplay mechanics or longer sessions.

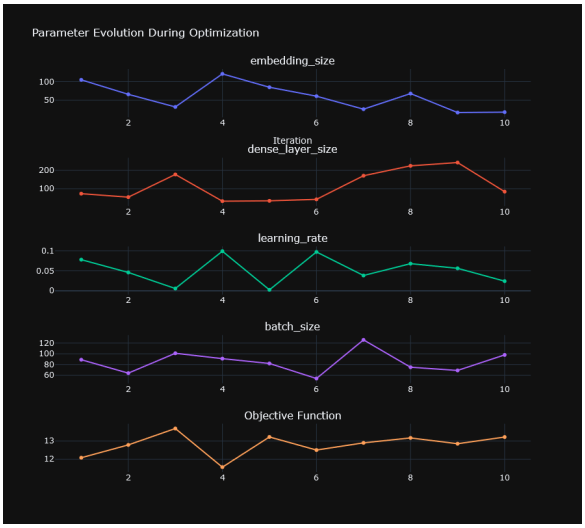
2. **Call of Duty World at War:** Predicted = 1.366, Actual = 39.0. Despite being a popular title, the model underestimated the actual playtime for Call of Duty World at War. This discrepancy could be attributed to the model's inability to capture the intense engagement often associated with multiplayer shooters like Call of Duty.

3. **Sniper Elite Nazi Zombie Army:** Predicted = 0.408, Actual = 1.9. The model's prediction for Sniper Elite Nazi Zombie Army was relatively close to the actual playtime, indicating a better performance compared to other predictions. However, there is still room for improvement to achieve more accurate predictions.



5.4 Insights and Limitations

While hyperparameter optimization notably enhanced the model's performance, the results highlight some intrinsic challenges. The discrepancies between predicted and actual time played indicate potential shortcomings in the model's predictive capabilities, possibly stemming from data sparsity or model complexity. Moreover, the top recommendations provide valuable insights into user preferences and popular game titles. However, the model's effectiveness may be limited by factors such as user diversity and preference heterogeneity. Incorporating reranking retrieval and scoring mechanisms could mitigate these limitations, enabling the model to generate more personalized and contextually relevant recommendations. Overall, while the hyperparameter optimization process indicates a major overhaul, further refinement and optimization strategies are necessary to enhance recommendation quality and address the inherent limitations of the dataset.



6 Conclusion/Future Work

In conclusion, this report has provided a comprehensive exploration of various recommendation algorithms, shedding light on their efficacy, strengths, and areas for improvement. Through meticulous experimentation and analysis, we have discerned the nuanced performance of collaborative filtering, content-based filtering, hybrid methods, and neural networks in generating recommendations. Collaborative filtering has proven adept at leveraging user interactions despite inherent challenges like the cold start problem and popularity bias, while content-based filtering excels in scenarios with sparse data but faces limitations in recommendation diversity. Hybrid methods have showcased superior performance by synergistically integrating collaborative and content-based filtering.

Looking forward, future endeavors could focus on enhancing existing algorithms by integrating reranking scoring and retrieval mechanisms developed in our research. By incorporating these innovative techniques into established recommendation systems, we can potentially mitigate shortcomings such as over-specialization and enhance recommendation diversity. Furthermore, exploring advanced neural network architectures and leveraging larger datasets could further refine prediction accuracy and scalability. Incorporating domain-specific features and contextual information may also yield more personalized recommendations. Through such advancements and continuous exploration, recommendation systems can evolve to deliver more accurate, diverse, and personalized recommendations, catering to the dynamic needs of users across various domains.

7 Contributions of Team Members

In this section, we outline the specific contributions of each team member towards the milestone of the project.

7.1 Karagiannis Christos

Completion of essential processes integral to both pre- and post-recommendation engine phases, including refining re-ranking strategies, enhancing scoring mechanisms, and optimizing data retrieval techniques:

- **Poster:** Created the poster for our recommendation engine in Photoshop.
- **Game-Title Encoding:** A systematic approach was devised to assign numeric values to game titles. This encoding of categorical data is foundational for subsequent analyses, particularly within the context of PCA. The methodical handling of game titles ensures a comprehensive preprocessing pipeline, potentially contributing to the robust initiation of the recommendation engine.
- **Normalization Strategy:** A normalization strategy was implemented for the 'time-played' feature using the MinMaxScaler. This strategic move aims to safeguard the integrity of the data, preventing potential biases within the recommendation engine.
- **Enhanced Data Utilization:** Through the retrieval method that is implemented the system could efficiently accesses and utilizes data. By fetching relevant data based on specific criteria, the groundwork is laid for making informed decisions or recommendations, thus optimizing the use of the available dataset by using a subset of it.
- **Personalization and Relevance:** Implemented a scoring algorithm, ranking items (like games, products, articles, etc.) based on their relevance to user preferences or specific criteria. This means that the system can provide tailored suggestions to users, significantly enhancing user experience and satisfaction.
- **Diversity and Engagement:** Implemented a re-ranking method . By adjusting the initial rankings to introduce a variety of choices, the system avoids the pitfall of showing monotonous or repetitive content.

7.2 Tsigkas Emmanouil-Angelos

Initiated the analysis process:

- Loading the dataset using pandas and conducting an initial automated profile report.
- Identifying and addressing issues such as constant values, duplicate rows, and skewness in the 'time-played' variable.

- Providing insightful conclusions based on the automated analysis, including data overview, missing values, and user-ID and game title analyses.

After the data analysis, optimization the dataset for the recommendation engine was conducted. The optimization process involved the following steps:

1. **Removal of Constant Column:** Eliminating the '0' column to enhance database efficiency.
2. **Removal of Duplicate Observations:** Addressing issues related to duplicate rows that could impact the accuracy of the analysis.
3. **Handling Outliers in Time Played:** Investigating and addressing potential outliers in the 'time-played' column to avoid skewing analysis results.
4. **Conversion of 'time-played' for Purchased Games:** Setting 'time-played' to 0 for purchased games to standardize representation and improve accuracy.
5. **Optimizing Data Types:** Changing the 'behavior' column to a binary approach (0 for purchase, 1 for play) to reduce memory usage.
6. **Removal of 'Behavior' Column:** Since 'time-played' is adjusted for purchased games, the 'behavior' column is removed.
7. **Normalization:** Scaling 'time-played' to a standard scale for fair feature contributions in collaborative filtering models.
8. **Sorting Experimentation:** Exploring the impact of sorting data on user-ID, behavior, or game-title.

7.2.1 Neural Network Recommendation Engine

- Designed the model architecture, incorporating embedding layers, dense layers, batch normalization, and dropout regularization.
- Executed training and evaluating for both regression and classification models for personalized game recommendations.
- Created the hyper-parameter optimization process using Bayesian optimization with Gaussian Processes (GP), aimed at refining the neural network model for improved performance.
- Visualized the whole process.

7.2.2 Synthesis and Paper Preparation

Took on the responsibility of synthesizing the contributions of team members into this paper. This ensured effective communication of collaborative efforts and the presentation of findings in a unified and coherent manner.

Assumed the responsibility of editing and proofreading the paper, ensuring accuracy, coherence, and adherence to formal academic writing conventions.

7.3 Nanousis Panagiotis

- *Literature Review*: Explored pivotal studies in game recommendation systems. Analyzed advanced models in Cheuque, Guzmán, and Parra's research and Bunga, Batista, and Ribeiro's collaborative filtering methods. Reviewed the challenges in digital game recommendations from Wu, Kolen, Aghdaie, and Zaman, and scrutinized Bertens, Guitart, Chen, and Periáñez's machine learning methods for in-game item recommendations, providing a comprehensive understanding of the field.
- *Data Refinement:* Synchronized game selections across datasets, effectively reducing the initial pool from 5,250 to a curated list of 2,500 games, enhancing the dataset's reliability for analysis.
- *Feature Extraction*: Addressed challenges in applying PCA for dimensionality reduction due to mixed data types. Explored alternative methods, recognizing the limitations of one-hot encoding in mixed data scenarios, and adapted our approach to these insights.
- *Content-Based Filtering*: Developed an advanced content-based filtering system, focusing on the intricate analysis of textual features such as genres, tags, and descriptions. Utilizing TF-IDF vectorization, I transformed these textual attributes into

numerical vectors. He then employed cosine similarity calculations to measure the similarity between item pairs, ensuring that our recommendations were finely tuned to user preferences based on game characteristics.

- *Collaborative Filtering*: Worked on collaborative filtering by implementing a k-nearest neighbors algorithm. This involved constructing a user-game matrix from training data and using sparse matrix representation to facilitate efficient computation of cosine similarities between games. By tailoring recommendations based on user-specified game titles, he was able to harness user interaction data to uncover latent patterns and preferences, enhancing the system's recommendation accuracy.

- *Hybrid Recommendation System*: Designed an innovative hybrid system combining content-based and collaborative methods. This approach allowed for a dynamic balance between the two methods, where the weight assigned to each could be adjusted to prioritize either content attributes or user interaction patterns. This hybrid system mitigated the inherent limitations of each method, such as over-specialization in content-based filtering and popularity bias in collaborative filtering, resulting in more diverse and accurate recommendations.

Finally, created the presentation based on the report and notebook.

References

[1] Matplotlib. https://matplotlib.org/.

[2] Numpy. https://numpy.org/.

[3] pandas. https://pandas.pydata.org/.

[4] Plotly. https://plotly.com/.

[5] scikit-learn. https://scikit-learn.org/.

[6] scikit-optimize. https://scikit-optimize.github.io/.

[7] Seaborn. https://seaborn.pydata.org/.

[8] Tensorflow. https://www.tensorflow.org/.

[9] Paul Bertens, Anna Guitart, Pei Pei Chen, and Álvaro Periáñez. A machine-learning item recommendation system for video games. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 2018.

[10] Rosária Bunga, Fernando Batista, and R. Ribeiro. From implicit preferences to ratings: Video games recommendation based on collaborative filtering. In *Proceedings of the 2022 International Conference on Computational Science and Computational Intelligence*, 2022.

[11] Germán Cheuque, José Guzmán, and Denis Parra. Recommender systems for online video game platforms: the case of steam. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019.

[12] Meng Wu, J. Kolen, Navid Aghdaie, and Kazi A. Zaman. Recommendation applications and systems at electronic arts. In *Proceedings of the 2017 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2017.

[13] Liangwei Yang, Zhiwei Liu, Yu Wang, Chen Wang, Ziwei Fan, and Philip S. Yu. Large-scale personalized video game recommendation via social-aware contextualized graph neural network. In *Proceedings of the 2022 ACM Web Conference*, 2022.