Predicting Film Audience Scores Using a Graphical Neural Network

Team Number: 15
Submission Date: 12/2/25

Abstract

This project builds a graphical model predicting the audience score of upcoming films and visualizes relationships among previously released films. Using data from TMDB, Rotten Tomatoes, and YouTube API, we constructed a dataset of 66,000+ films spanning 2010-2025, containing key film attributes such as cast, genres, release information, viewer engagement metrics, and trailer sentiment. Additionally, we incorporated diversity indicators like female cast percentages to uncover patterns between gender representation and film success. After cleaning and merging these data sources, we applied graphical modeling techniques (Graph Neural Networks and XGBoost baselines) to capture dependencies among film features and trained a prediction model to estimate audience scores. Our results show meaningful relationships between attributes such as genre clusters and the influence of cast popularity, and demonstrate moderate predictive accuracy for new film scores. Our Streamlit interface enables users to interactively explore these relationships while recommending films based on likes and dislikes.

1. Introduction & Motivation

The film industry faces mounting pressure to predict audience reception while also addressing debates about representation and diversity. Audience scores directly influence marketing strategies, streaming decisions, and long-term commercial success. Being able to predict these scores before release could offer studios and creators valuable insights into audience expectations and potential film performance. Systematic analysis of films beyond the basics of budget, runtime, etc. has been limited by data fragmentation as film metadata, engagement metrics, and diversity information exist across disparate platforms. Additionally, gender and demographic information are not systematically tracked in standard film databases, and simple regression and black-box models don't offer in-depth predictions for analyzing complex relationships. By integrating multiple data sources, enriching them with gender representation features, and building a structured graphical model, we can explore how different film attributes interact and contribute to audience responses.

1.3 Objectives

- Construct a comprehensive dataset integrating multiple APIs for films containing features relevant to audience score prediction and webscraping critic and audience scores.
- Build a complex knowledge-based graphical model showing dependencies among film attributes and audience scores.
- Use the graphical model to predict scores for upcoming films, based on gathered metrics of previously scored films.
- Interpret the structure of the learned model to understand relationships among released films and provide actionable insights on patterns across these factors
- Create an interactive dashboard that allows for users to both see the prediction of upcoming films, and as well see the relationship and similarities that the films have with one another.

2. Data and Methods

2.1 Data Sources

1. **TMDB (Movie Database)**: 66,233 films
    a. Features collected: genres, runtime, release dates, popularity, and vote counts, budget, vote average, overview, production companies, production countries, cast, directors, is_successful (vote_average > 6), description sentiment score, urls for posters and trailers

    b. Collected using TMDB Bearer Token authentication processing 2010-2017 and 2018-2025 separately due to API limits
    c. Excluded films with less than five votes and with a vote average of 0.
2. **Rotten Tomatoes (RT)**: 6,800 films
    a. Features collected: has_rt_url (binary indicator of url present), rt_url (full RT page url), audience scores and critic information, critic_score (Tomatometer percentage 0-100), audience_score (0-100), review_sentiment (from critic reviews)
    b. Webscraped using BeautifulSoup with URL mapping from TMDB titles and implemented retry logic and rate limiting to abide by Rotten Tomatoes' terms of service
3. **YouTube API**: 42,156 trailers
    a. Features collected: Trailer metadata (video_name, published_at, official, tags, category_id), viewer engagement (view_count, like_count, comment_count, favorite_count), video descriptions, and timing information (release_date, days_until_release, before_release)
    b. Used the YouTube Data API and filtered for films that appeared on the collected TMDB dataset, and then scraped the above metrics from all of the trailers that each movie had within the TMDB dataset.
    c. If there were multiple trailers per film, we first filtered for if the trailer was either stated to be “official” or the “main” trailer. Then we filtered based on the date of release for the trailer, and we decided to use the first trailer that was released for the film as it would not have any biases towards the metrics.

2.2 Cleaning

- Merging datasets on film IDs or matched titles and release years.
- Removing duplicate trailers using heuristics such as filtering by “official trailer” and selecting the earliest valid upload when multiple versions exist.
- Handling missing values for audience scores, popularity metrics, or cast data.
- Standardizing numeric features (e.g., log-transforming skewed trailer view counts).
- Extracting sentiment or keywords from trailer descriptions and comments.

2.3 Data Integration and Storage

We created a database for storage using MongoDB and designed a document-oriented schema optimized for storage efficiency and query performance. The complete organization of our MongoDB setup can be seen in the Appendix. The structure of this schema was beneficial as it accommodated varying data completeness, as not all films have Rotten Tomatoes scores or trailer data. The nested format of the documents naturally represents hierarchical relationships (production metadata, people, metrics, etc.), and it has easy integration with Python and our GNN preprocessing pipeline.

2.4 Modeling / Data Techniques

Sentiment Analysis:

- In order to create the quantitative description and critic sentiment values, we utilized the DistilBERT-base-uncased-finetuned-sst-2-english model, accessed via the Hugging Face transformers library. It takes a text input, such as a movie description or an aggregated critic review, and predicts whether the dominant tone is Positive or Negative with a corresponding confidence score. This confidence score is then normalized to yield the Sentiment Score, a continuous metric ranging from -1 (maximum negative sentiment) to +1 (maximum positive sentiment). This value is a representation of the overall emotional polarity and strength of the text, enabling direct, scalable comparison of marketing material and critical reception across the entire movie dataset.

Modeling:

- Our GNN models audience scores by treating the film dataset as a graph of 66k movies interconnected through shared genres, directors, production countries, and diversity based similarity via K-nearest neighbors in a ten-dimensional gender-representation space with Gaussian weighted edges. The layers of our GNN attempt to capture different features of the graph by building on information from the neighbors surrounding each node. Each film is encoded as a 16-dimensional feature vector capturing production attributes, sentiment, and diversity metrics. Only ~5k films have audience scores, so we train the GNN in a semi-supervised setting where unlabeled nodes still contribute to message passing as the vectors are passed through each layer. We achieved this using a 70/10/20 training, validation and test split. We use a two-layer GraphSAGE model with batch normalization, dropout, and a residual connection for stability. Since SAGEConv does not support weighted edges, we implemented a custom weighted aggregation step using scatter operations. Edges are normalized to prevent dense groups from dominating. Training used Adam optimization with early stopping based on validation RMSE. This pipeline enables the model to learn both feature-level patterns and relational structure across the film graph.
- The XGBoost model predicts Rotten Tomatoes audience scores using 150+ engineered features including production metadata (budget, runtime, genres), cast/director features (performance and top actor/director indicators), diversity metrics (gender balance, female representation) and trailer engagement with time decay. Time decay features apply exponential decay (30-day half-life), Gaussian recency weighting (peak at 14 days), and engagement rates (views/likes per day) to capture trailer buzz. Hyperparameters were tuned using RandomizedSearchCV (50 iterations, 3-fold cross-validation) across learning rate, tree depth, and regularization parameters.
- The KCGN model predicts audience scores using a knowledge graph with semantic relationships between films and certain metadata features. The graph has over 300,000 nodes consisting of movies, genres, directors, and more. Each node is connected by a relationship that two nodes share, allowing the model to learn how specific relationships influence each other. Because only about 5,000 movies have known audience scores, we use a supervised learning strategy in which labeled movies supervise the learning of entity embeddings, refining neighboring movies through their relations. Since KCGN models are relation aware, they are able to capture more detailed patterns, such as the predictive value of genre similarity against the predictive value of a shared director or cast member. Normalization and weighting had to be used to prevent overly common relations from dominating.
- The final ensemble model of these three models is currently being worked on for implementation on the website. Currently, it combines the other three by making a weighted average of their predictions.

3. Results and Interpretation

3.1 Results

After training with early stopping, the GNN achieved a validation RMSE of 0.238, Test RMSE of 0.197, test MAE of 0.161, test R^2 of 0.221, RMSE of 0.188, and a Pearson correlation of 0.627. These results indicate that while the model does not perfectly recover audience scores, it captures meaningful structure in film attributes and their graph neighborhoods. The improvement from validation to test to full-inference suggests that the model generalizes well and benefits from graph propagation across many unlabeled nodes. The KCGN achieved an RMSE of 0.1775, MAE of 0.1418, R^2 of 0.3672, and Pearson correlation of 0.6114. Similarly to the GNN, this indicates a clear capturing of patterns by the model, though does not perfectly predict audience scores. The XGBoost model had an RMSE of 0.1773, MAE of

0.1355, R^2 of 0.3429, and MSE of 0.0314. The final ensemble does not quite yet have results, as we are still implementing it for the website.

3.2 Interpretations

- Although the GNN does not outperform the XGBoost model, it captures relational signals that feature-only models cannot. The relatively strong Pearson correlation (0.63) indicates that the learned node embeddings encode meaningful semantic information about films. These embeddings will be especially valuable in the ensemble phase.
- Figure 6.3, the density of predictions plot made for the GNN model, reflects that the audience rating predictions are skewed towards the higher side, with a majority of predictions being above 50%. This could be because, in general, audiences tend to rate movies positively, a pattern that the model seems to have detected.
- XGBoost feature importance shows that critic-score presence, runtime, US production, director quality, and engagement metrics are the strongest predictors of audience ratings.
- Across the three models, we can see individual strengths that together reveal how both feature-level and relational information can shape audience perception. The GNN captures meaningful film-to-film similarity, reflected in its Pearson correlation of 0.63, indicating that the graph structure successfully encodes similarity in genre, director, and diversity characteristics. Its full-inference performance (RMSE 0.188) improves over the test split, showing that message passing across all 66k films helps the model generalize, though its predictions tend to regress toward the mean due to heavy-tailed degree distributions and relational smoothing. In contrast, XGBoost achieves the strongest point-prediction accuracy (RMSE 0.177, MAE 0.136) by leveraging nonlinear interactions among 150+ curated features; its feature importances highlight that critic information, runtime, production attributes, and trailer engagement are the dominant predictors of audience sentiment. However, since it operates purely on tabular features, it cannot capture structural similarities between films. The KCGN model then is able to fall between the two models. While slightly less accurate than XGBoost, it effectively models semantic relationships between films, directors, genres, and production entities by distinguishing which relational types carry predictive weight. Altogether, the models show that no single approach is sufficient, XGBoost captures detailed feature-level patterns, the GNN captures global similarity across the film graph, and KCGN captures meaningful metadata relationships, providing a foundation for an ensemble that combines their strengths.

4. Discussion and Reflection

4.1 Challenges

1. One of the first challenges we faced was creating a consistent and meaningful scoring metric across platforms. Audience scores differ in scale and format between TMDB and Rotten Tomatoes, requiring normalization and careful handling during data merging. To solve this, we standardized the scoring metric across data sources to a common 0-100 scale. Along with this problem of creating the dataset prediction metric, we initially limited the cohort of films to those that had a high audience score. This both limited the number of films within the created dataset and created a lack of diversity among the films within the different scoring ranges. We then decided to include all of the films within a given time frame rather than filtering using audience score, which then allowed for more accurate predictions across the entire range of scores.
2. One of the biggest challenges we faced was the diversity component of our project. Originally, we wanted to analyze whether diversity in film (eg. cast, race, gender) and audience sentiment can help predict a movie's success pre-release and uncover systematic biases in audience perception. However, we reassessed this portion of our project because there was no standardized

race/ethnicity data in TMDB, and using name-based ethnicity prediction is problematic and unlikely to be accurate. We decided to prioritize data quality over quantity and narrow our focus to gender only as TMDB provides the gender field and it's not inferred by us. In doing so, we explored how gender representation in film is important in its own right. In the event that this didn't work out, our next strategy would've been to analyze structural diversity (production countries, languages, etc.) or focus entirely on sentiment analysis instead.

3. Major challenges when building the GNN included building a multi-million-edge graph without overwhelming memory, preventing oversmoothing, and balancing extremely uneven relation sizes. We solved these by limiting edge group sizes, normalizing weights, and reducing the feature space. A further challenge involved training on only 5,091 labeled films while still extracting value from the remaining 60k. Graph-based semi-supervised learning allowed us to overcome this limitation, but only after implementing careful masking logic to prevent mislabeled or missing targets from contaminating training. If this approach had failed, our fallback plan was to reduce the graph to a sparser network or shift to node2vec-style embeddings for downstream regression. The final model demonstrates that thoughtful graph construction and careful architectural choices are essential when deploying GNNs at this scale. In addition, training a GNN requires alignment of data into a single numeric node-feature matrix, so missing or inconsistent fields cause repeated errors and misaligned rows. We addressed this by creating filled missing fields with zeros, normalized all numeric inputs, and encoded diversity information consistently across all movies.
4. One of the biggest challenges in implementing the XGBoost model was hyperparameter tuning. Unlike linear models, which use 1-2 parameters, XGBoost uses dozens that interact in complex ways. We had to balance tree depth, regularization, and sample rates to avoid overfitting. Through grid search and validation curve analysis, we found the best hyperparameters to ensure the model generalizes well to new movies. Another challenge was figuring out how to incorporate time decay features for trailer data into our model. Raw trailer metrics like view and like counts treat all engagement equally, regardless of when it occurred. A trailer with 1 million views accumulated over 6 months tells a different story than one with 1 million views accumulated over one week. To capture this, we implemented exponential decay with a half-life of 30 days to down-weight older engagements, Gaussian recency weighting to give more weight to engagement up to 14 days before release, and velocity metrics which calculate engagement per day.
5. When it came to the KCGN model, the biggest challenges were to do with the hyperparameter tuning and particularly with ensuring that it was not too exponentially large. Because of there being over 60,000 movies in the dataset, creating edges for each feature would blow up the amount of memory that the model needed to run, quickly crashing the runtime. This problem required restrictions to be added to the amount of edges per feature, as well as an intense amount of optimization. Google colab, with its access to GPUs, was particularly helpful in running it. With such a complex model, the hyperparameter tuning was also complex and took time to get right.
6. Due to working on the different models separately, the final ensemble of them was challenging, particularly when it came to aligning, configuring and producing predictions and parameters for our final website. Artifacts of each model needed to be saved independently so that each model was able to be referenced and stacked, multiplying any previous compile times that were long but necessary.

4.2 Lessons Learned

- Readjusting the scope of our project isn't a bad thing, as it's better to do one dimension well rather than multiple dimensions poorly.

- Filtering down the dataset based on the prediction metric leads to a bias and an inability to predict the entire spectrum of films. Instead, we should include all of the films as it represents the diversity among the different films and the broad range of audience scores.
- API datasets often require extensive cleaning before they can be used meaningfully.
- Graphical models offer interpretability but require careful feature selection to avoid dense or unreadable graphs.
- Balancing prediction accuracy with model interpretability is a key part of designing useful data science solutions.

5. Acknowledgment and References

- TMDB API - create list of films within our given time period
- Rotten Tomatoes Website - find audience score for films within the TMDB csv
- YouTube Data API - find all possible trailers for films within TMDB csv
- Tools: Python, R, statistical modeling libraries
- AI assistance: ChatGPT, Google Gemini, GitHub Copilot
    - We used AI tools in order to help with guiding the process of creating our dashboard, along with writing code scripts in order to create datasets and manipulate them in ways that would best suit the project.

6. Appendix

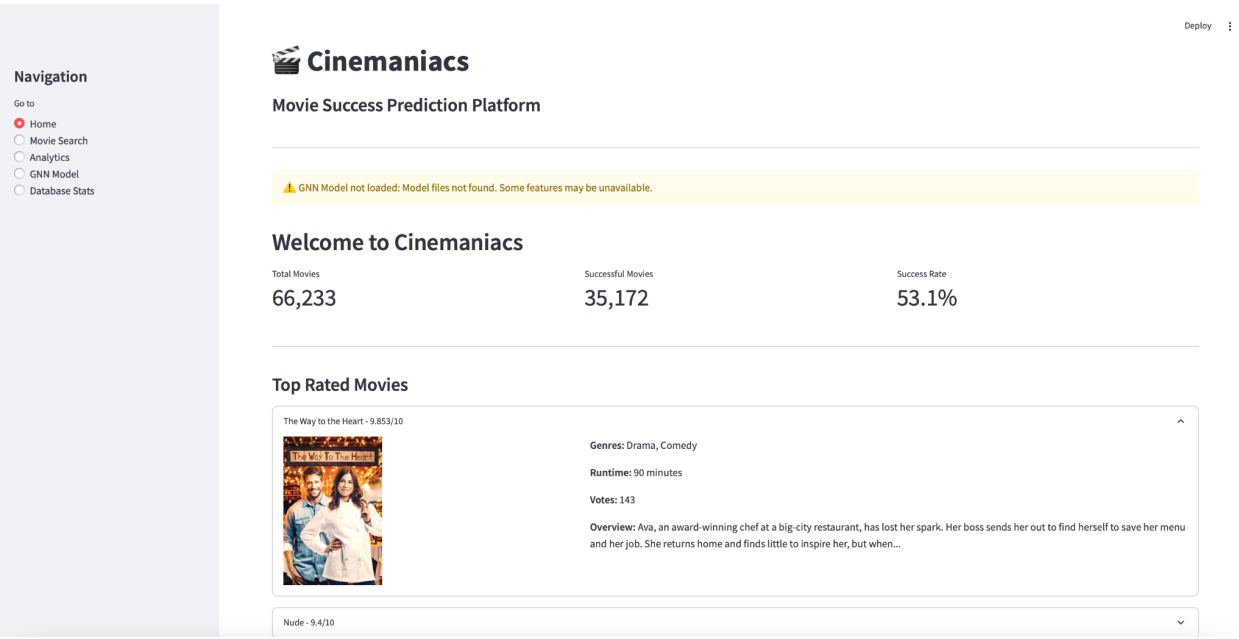

Figure 1.1: Website Home Page

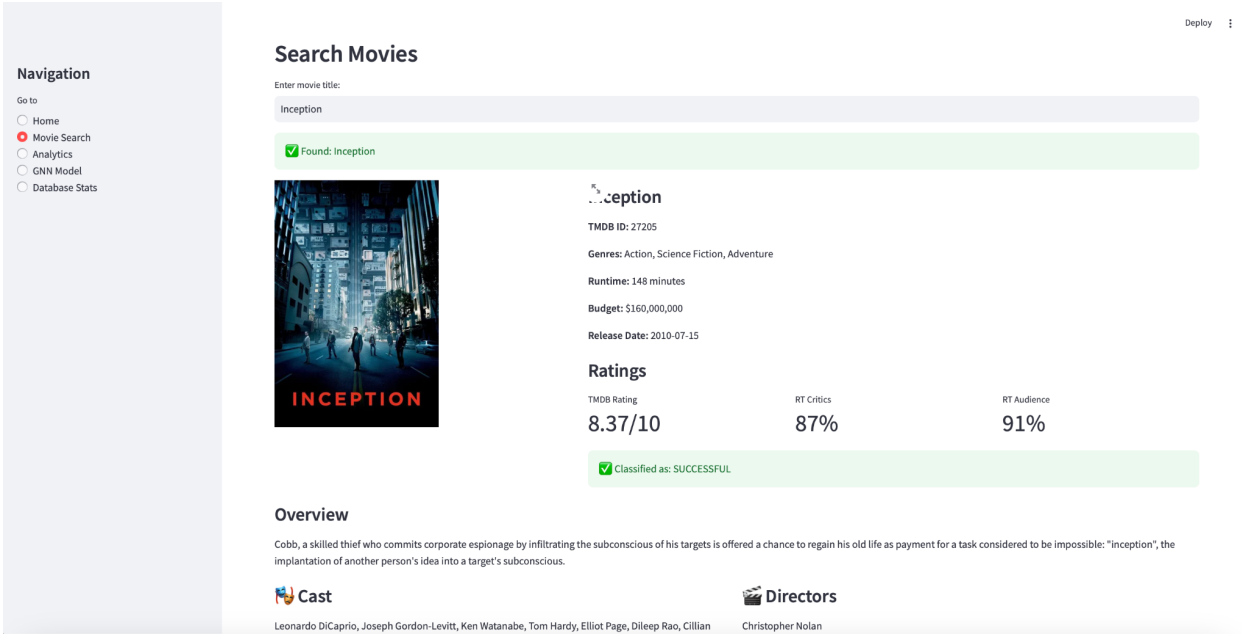

Figure 1.2: Website Movie Search

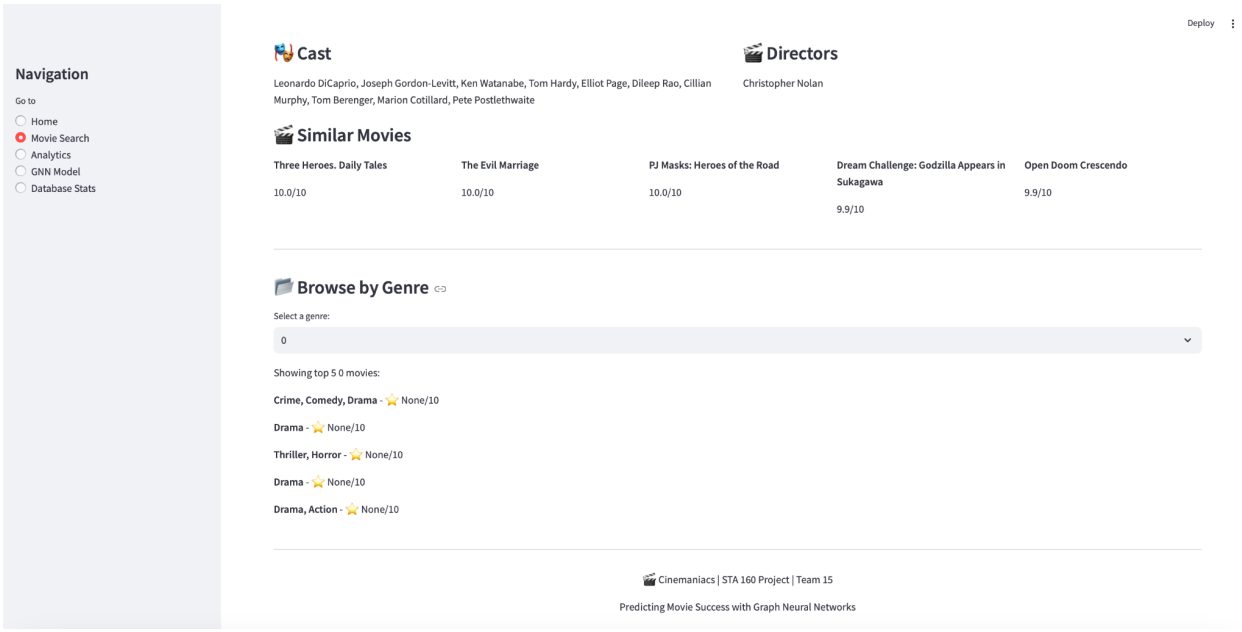

Figure 1.3: Website Browsing

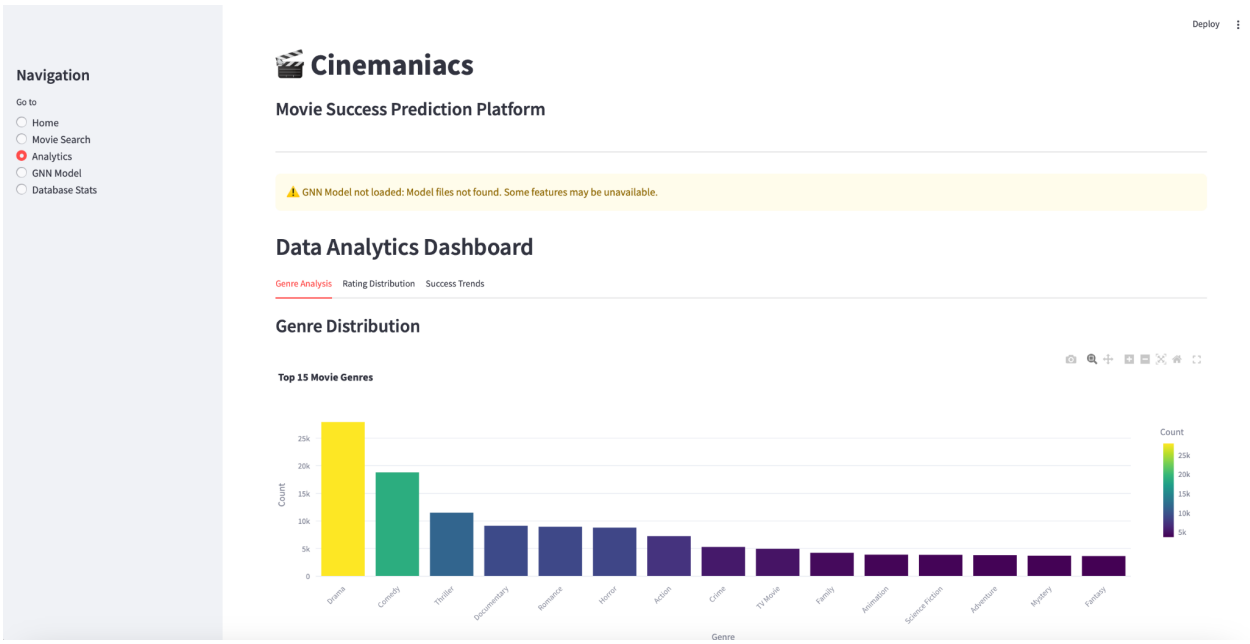

Figure 1.4: Website Analytics

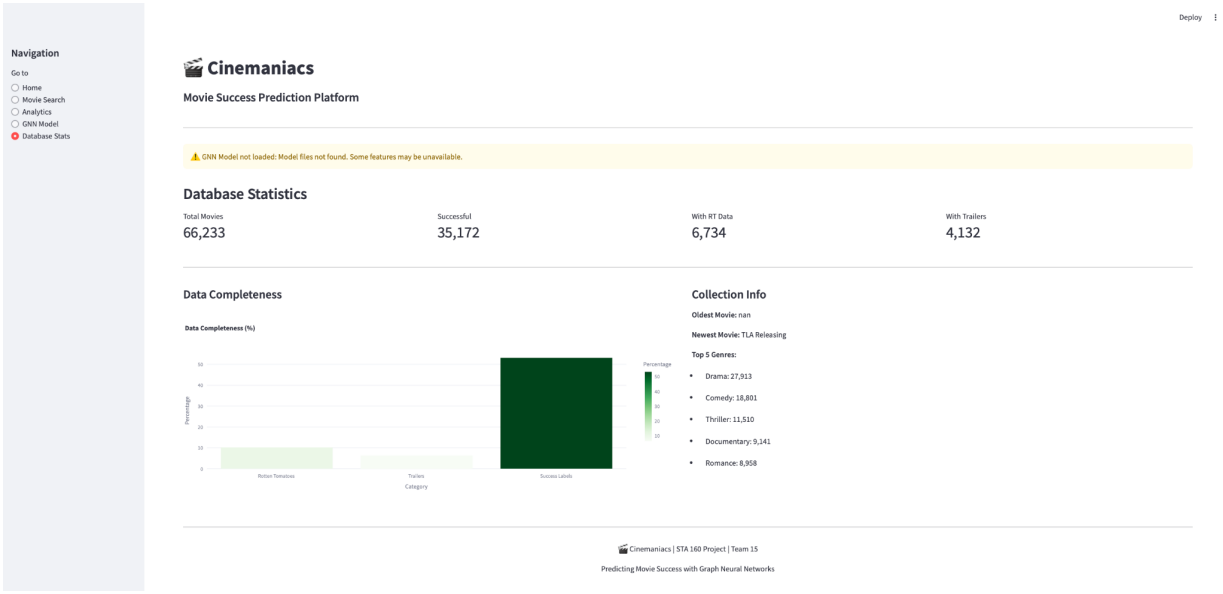

Figure 1.5: Website Database Stats

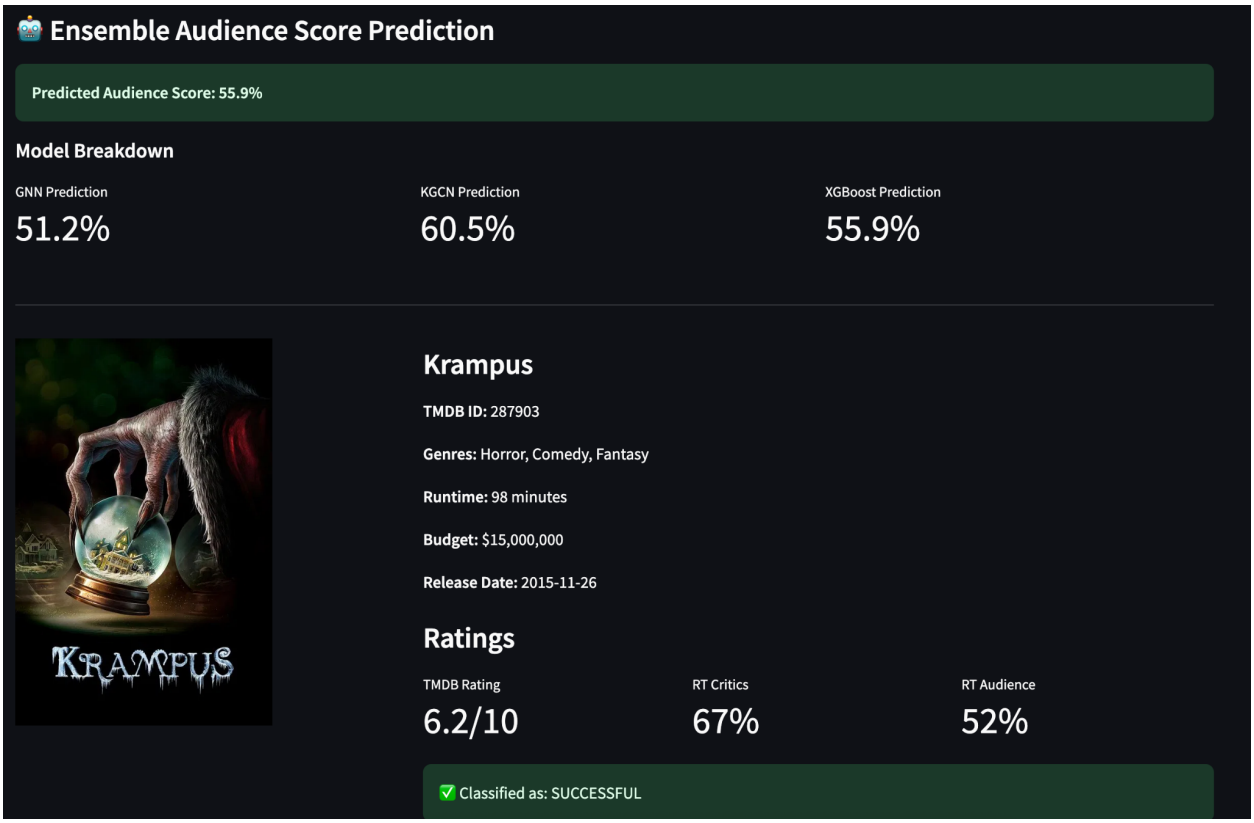

Figure 1.6: Website Ensemble Model Implementation

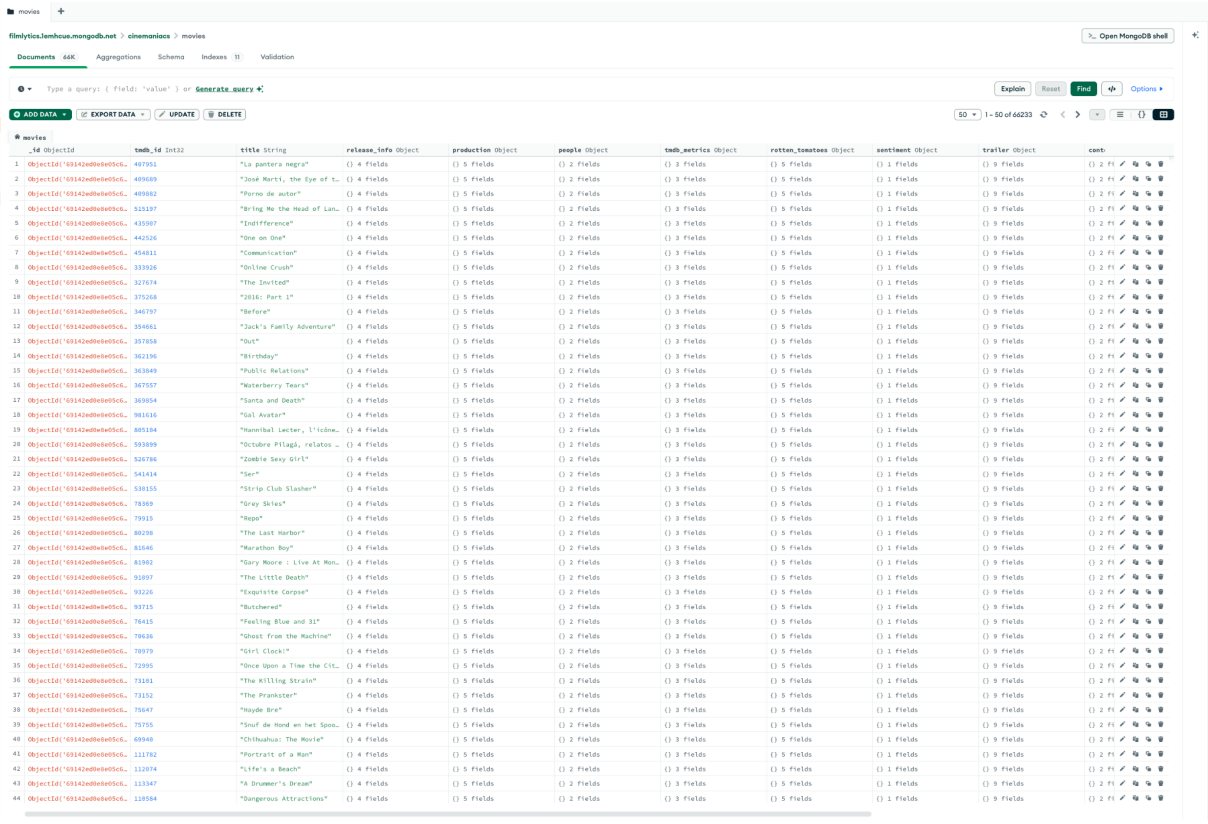

Figure 2.1: MongoDB Database

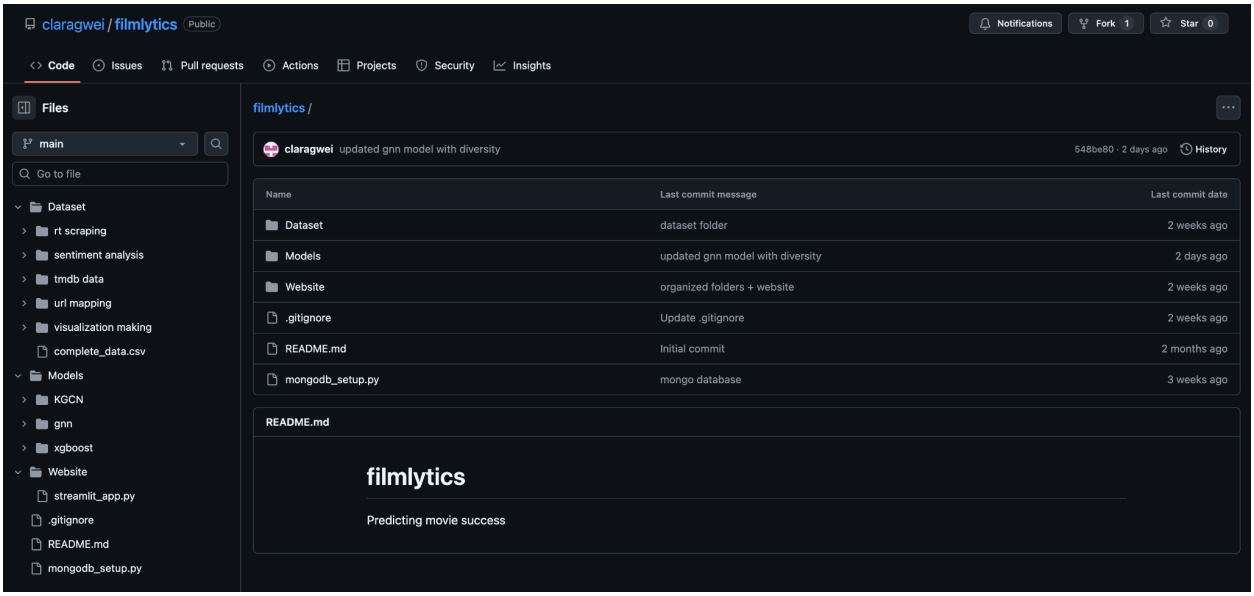

Figure 3.1: Github Page

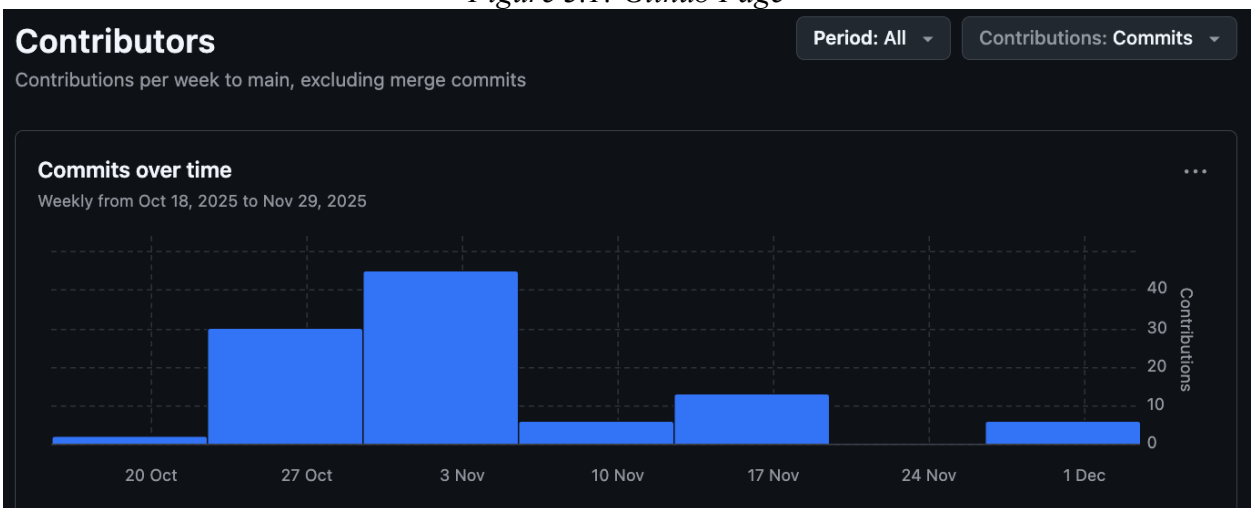

Figure 3.2: Github Commit History

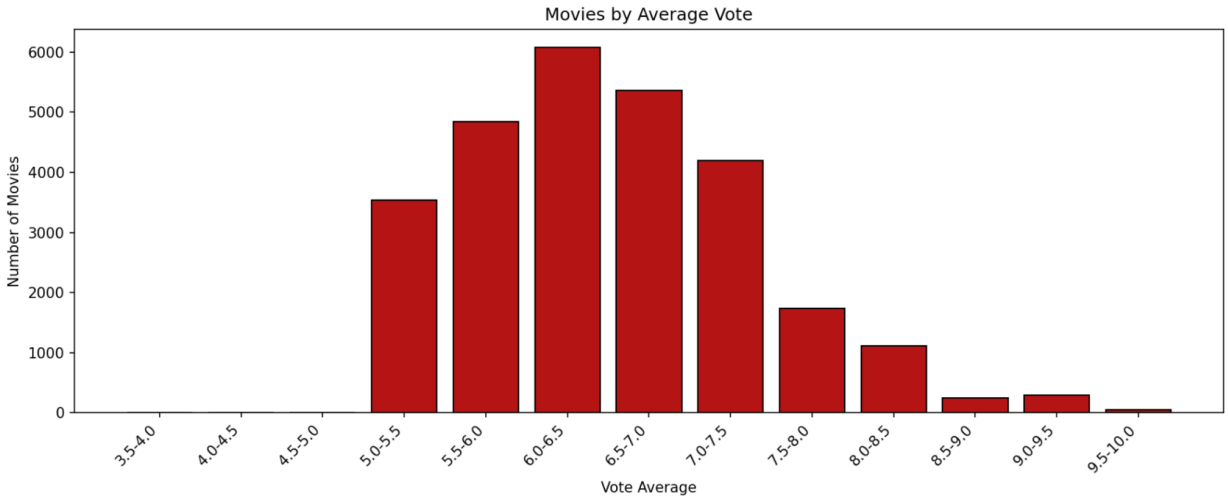

Figure 4.1: Dataset - Movies by Vote Average

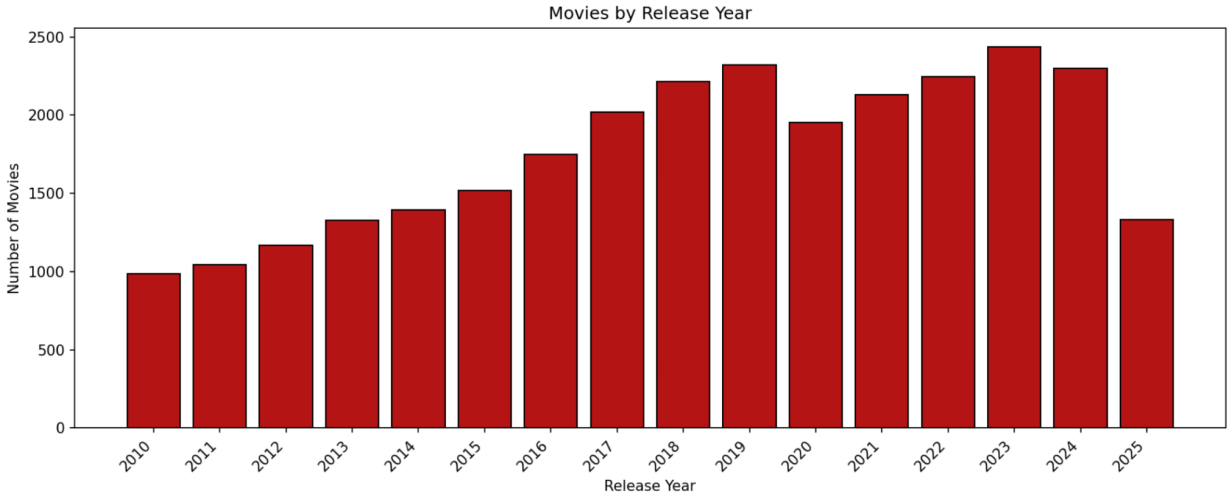

Figure 4.2: Dataset - Movies by Release Year

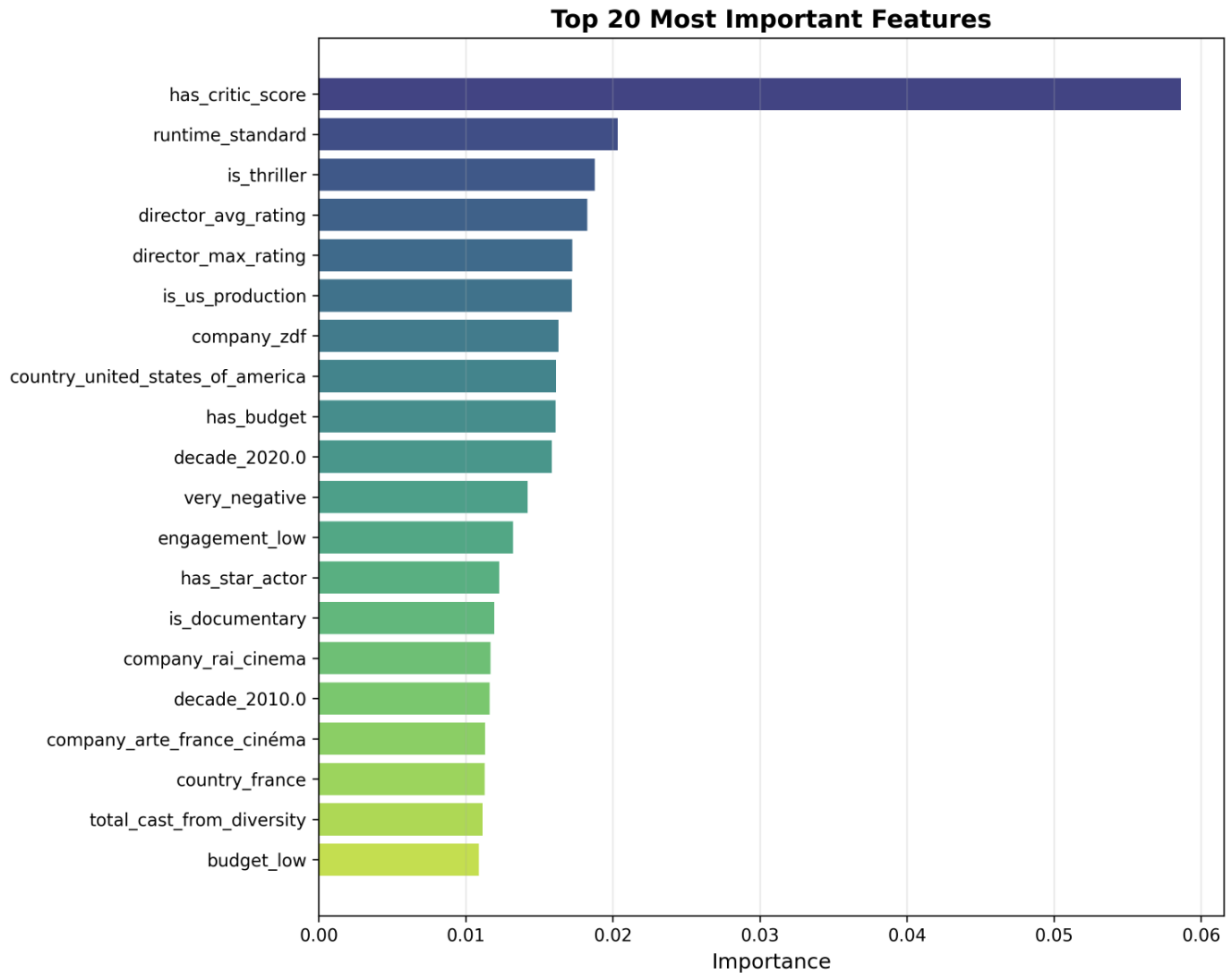

Figure 4.3: XGBoost Feature Importance

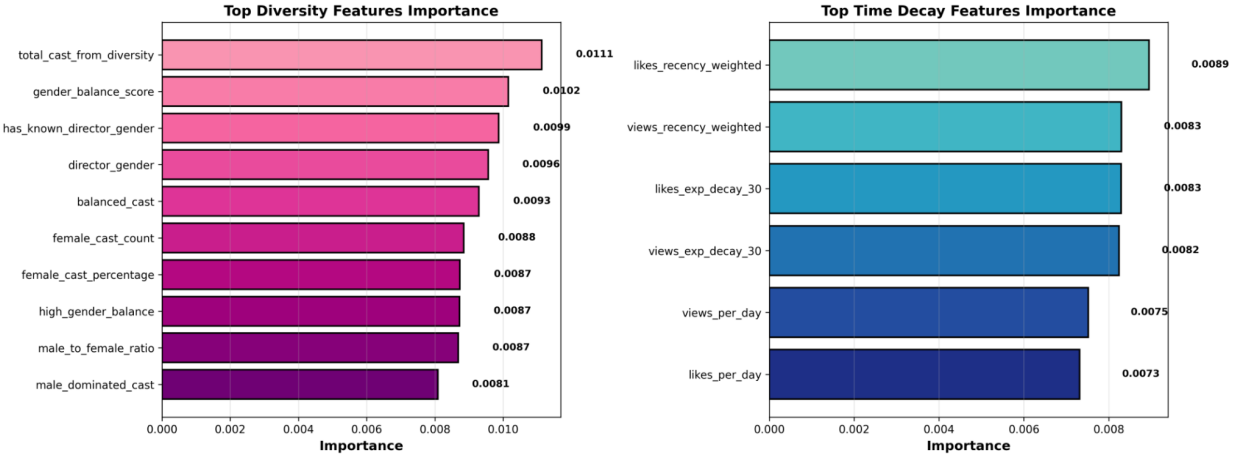

Figure 4.4: XGBoost Diversity and Time Decay Feature Importance

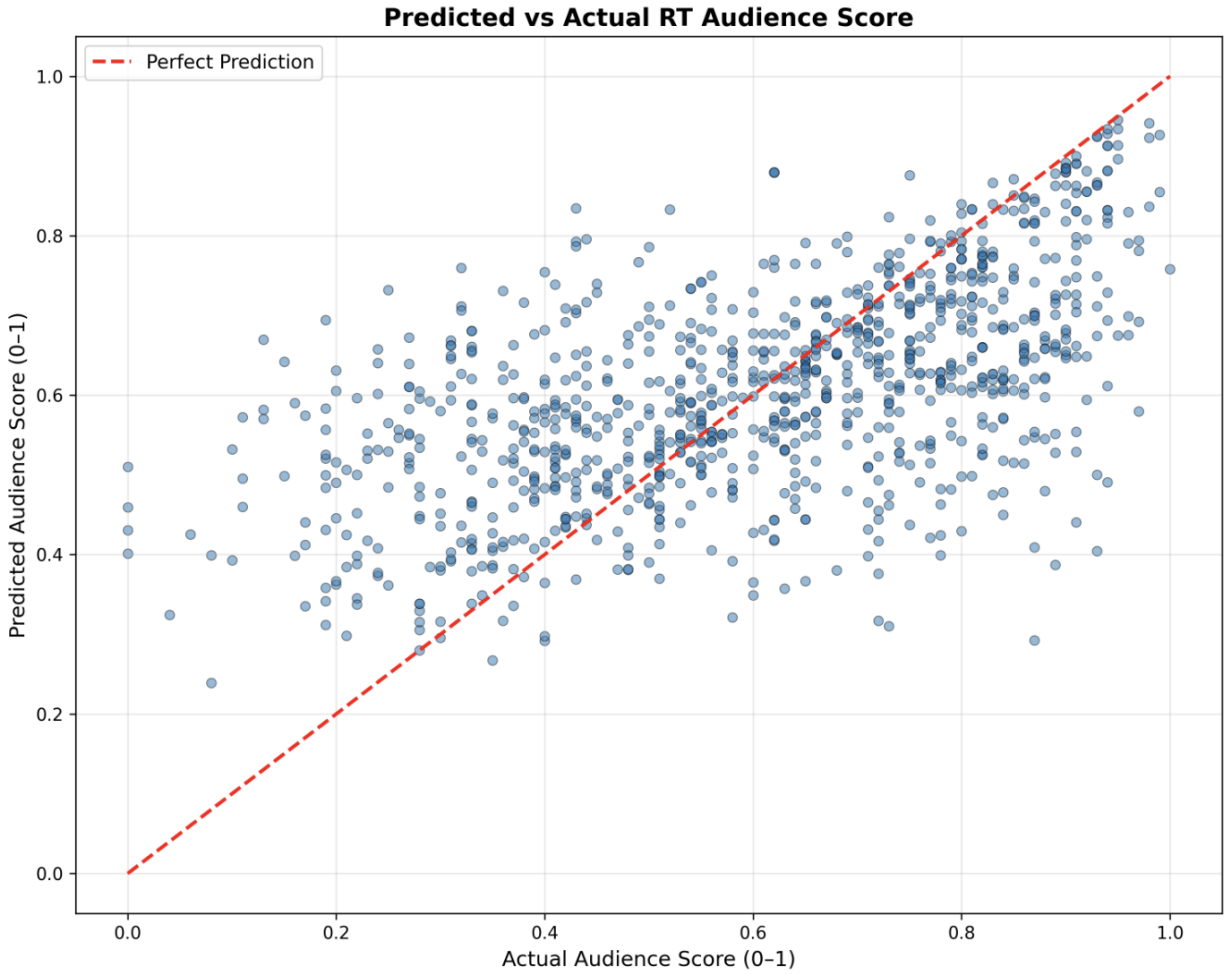

Figure 4.5: XGBoost Predicted vs Actual Plot

Figure 5: Complete Dataset - CSV

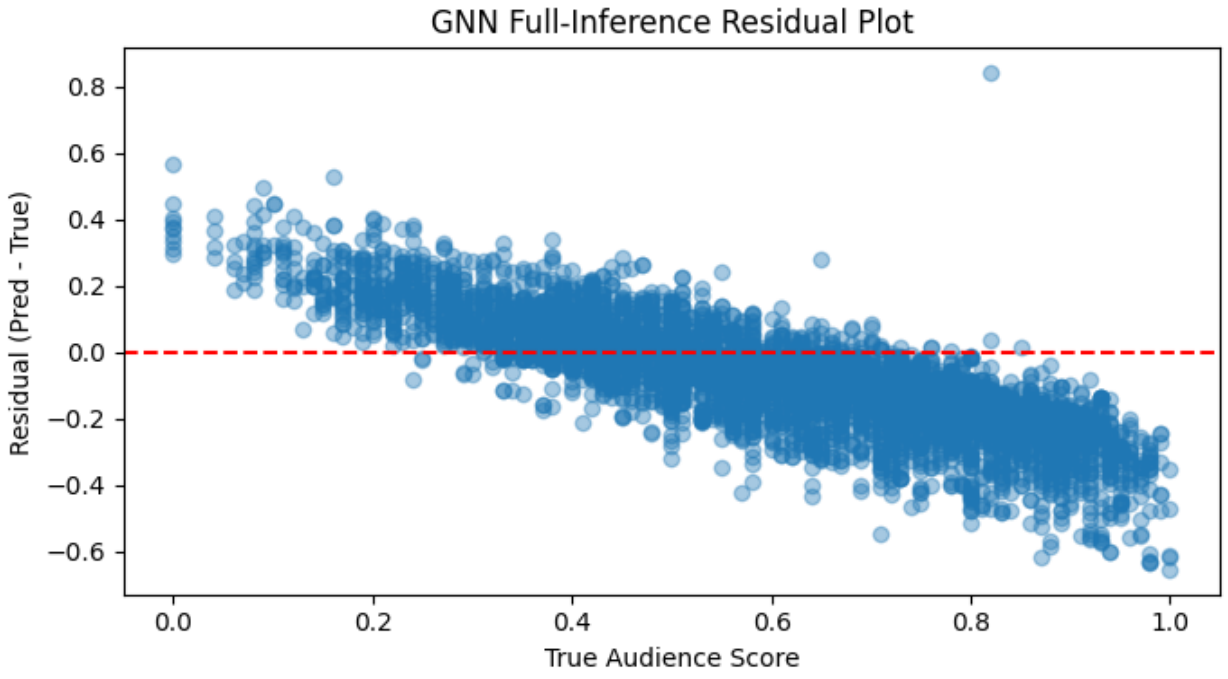

Figure 6.1: GNN Residual Plot

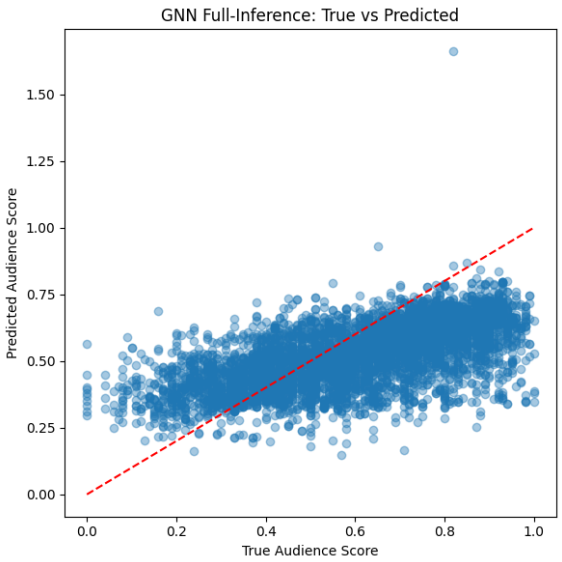

Figure 6.2: GNN True vs Predicted Plot

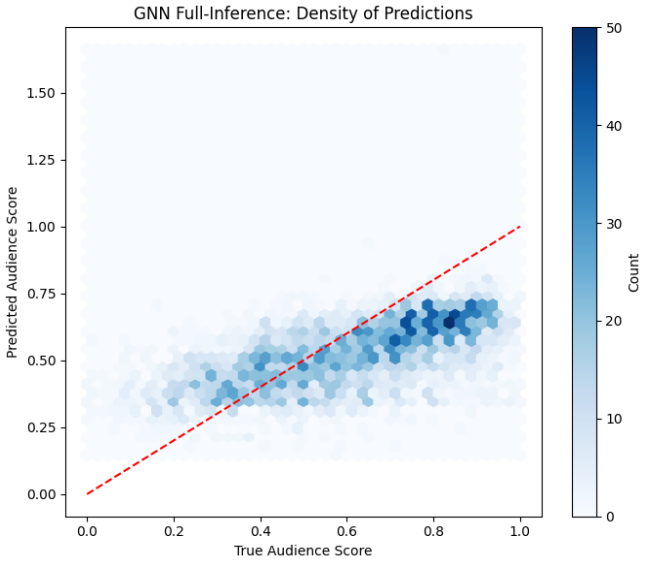

Figure 6.3: GNN Density of Predictions Plot

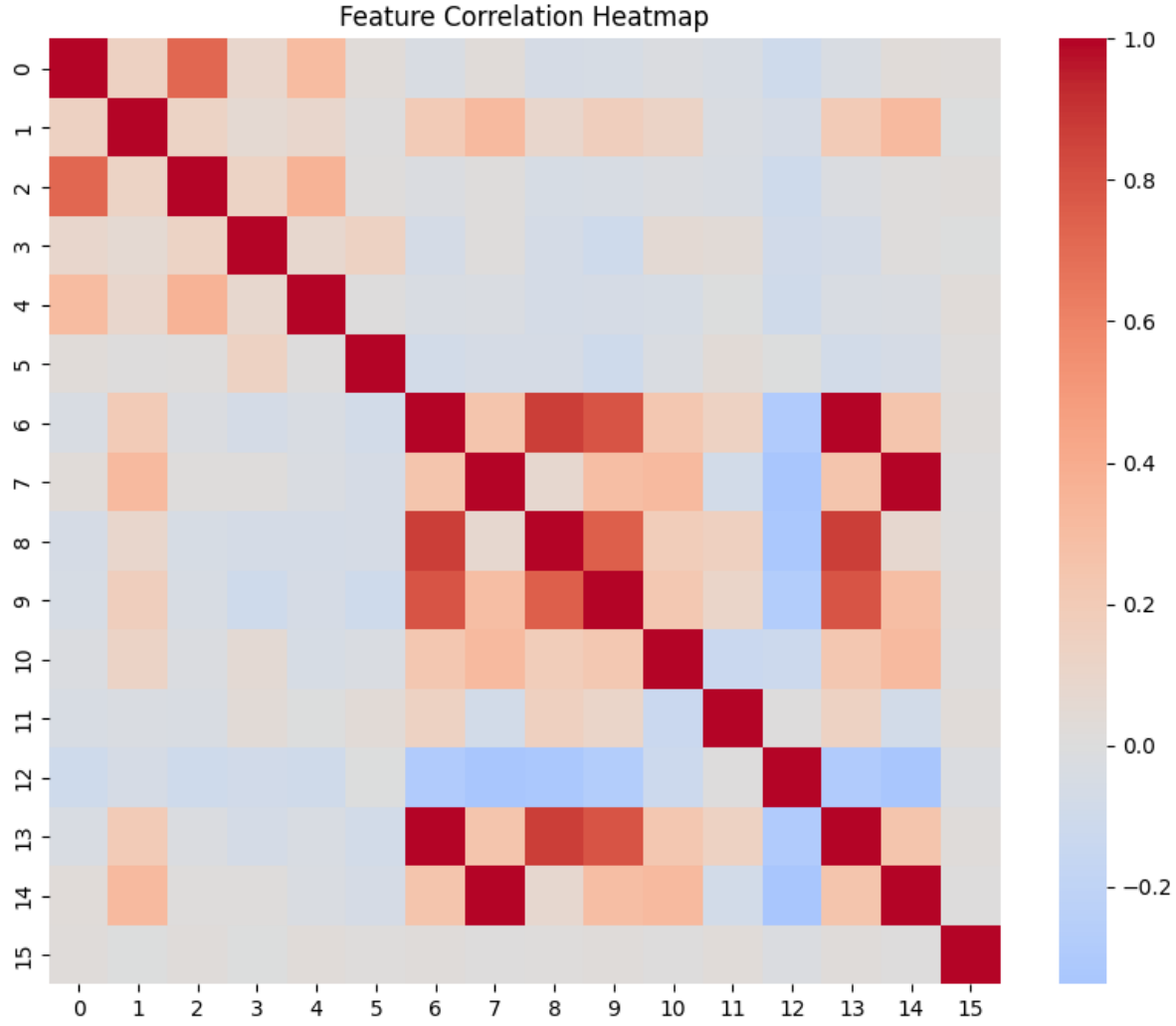

Figure 6.4: GNN Feature Correlation Heatmap

Figure 7.1: KCGN Truth vs. Predicted Scatterplot

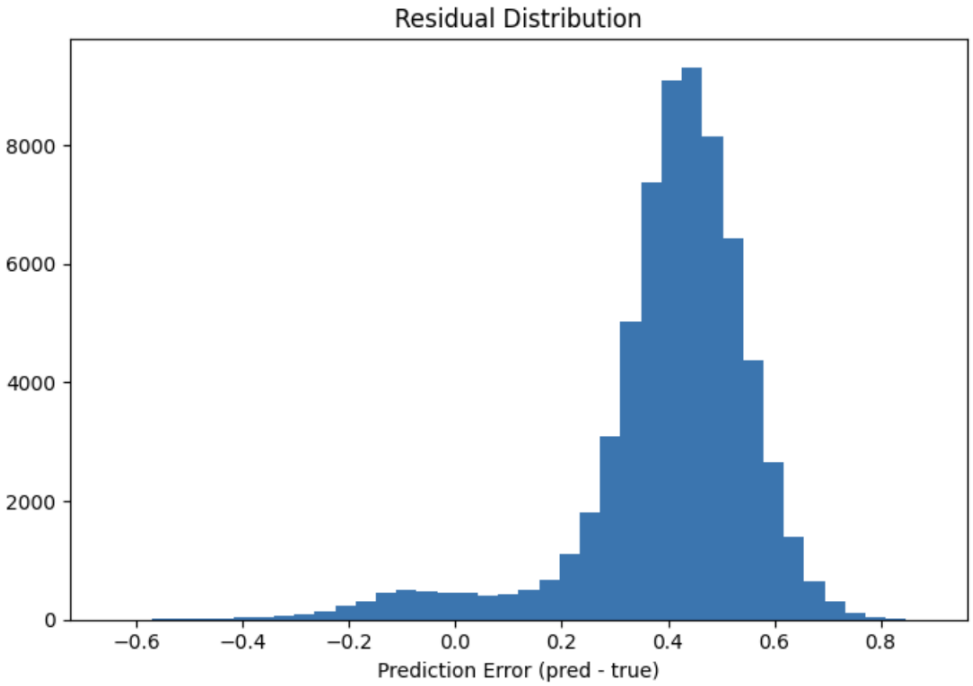

Figure 7.2: KCGN Residual Distribution

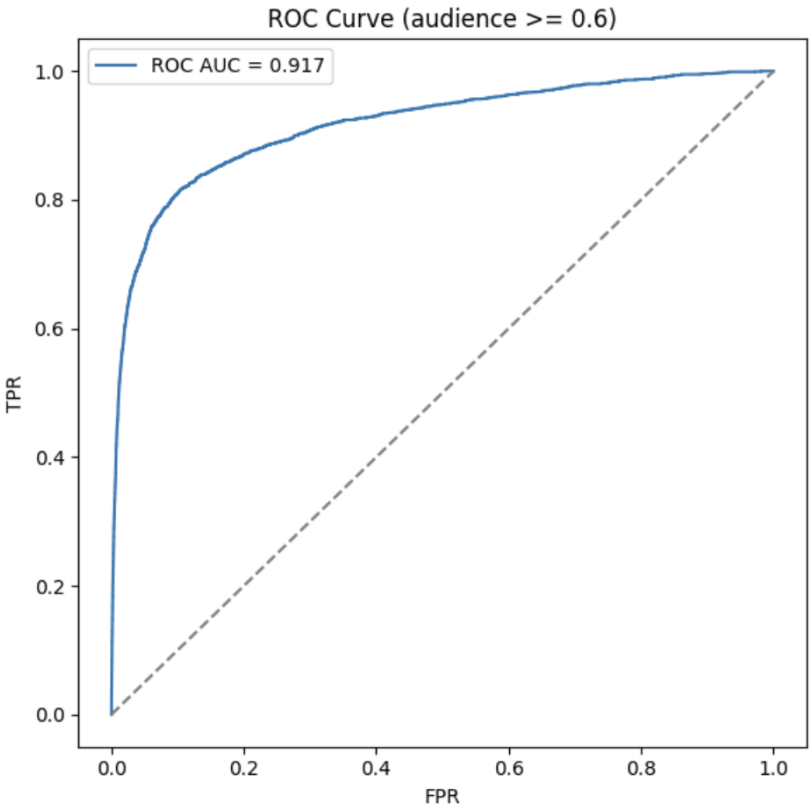

Figure 7.3: KCGN ROC Curve

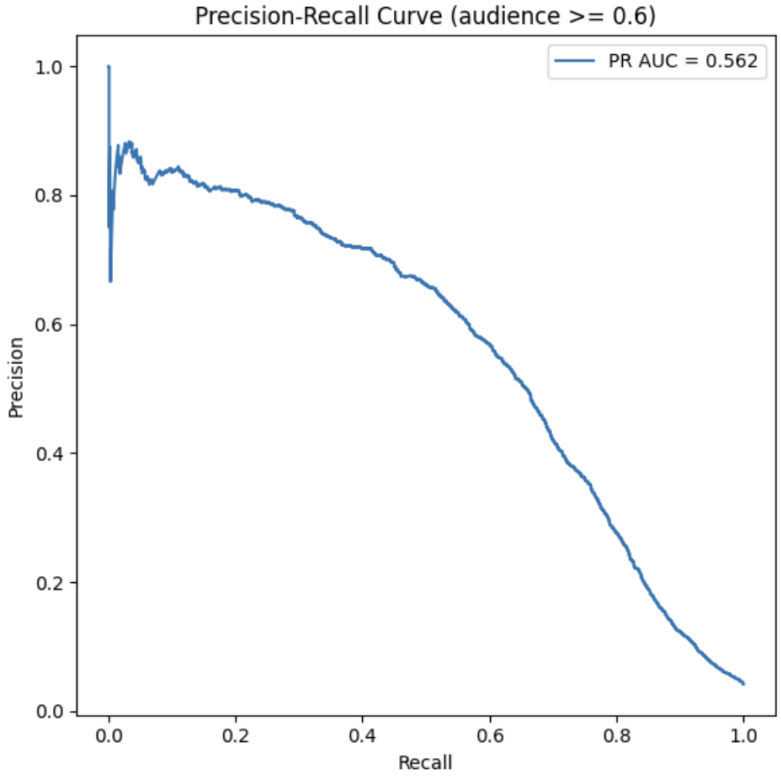

Figure 7.4: KCGN Precision-Recall Curve