Pre-release Movie Ratings Predictor

Austin Min, Rushil Patel, Riley Wong, Krish Ananth

CSCI 183: Data Science

Department of Computer Science, Santa Clara University

Dr. Smita Ghosh

December 9, 2024

Introduction

This project aims to predict rating metrics for movies before they are even released. By analyzing a large dataset of movies and their most influential features, we train and test models that will provide the best predictions on the success of a movie from the standpoint of ratings. The goal is to create a tool that lets fans, movie-makers, and distributors alike understand what makes a movie successful. Unlike the plethora of other research that focuses on predicting gross income, we propose that ratings can serve as a more nuanced and qualitative measure of a movie's success, reflecting audience perceptions and movie popularity more broadly.

As we continue through the project, we approach aspects of data collection, evaluation, and preprocessing as well as model training and analysis. We then discuss the limitations of our models, perform an informal experiment on upcoming movies, and conclude with future research that could be done to enhance this project.

Data Collection

We originally planned to use IMDb because they provide a non-commercial dataset on movies here: https://developer.imdb.com/non-commercial-datasets/. However, the data here is split into relational tables, which takes extra effort to process, and also lacks some data we particularly want, for example, budget and distributor. While the data that we see this dataset lacks is available publicly on their website, to obtain the data files with this additional data we need to coordinate a subscription to their service.

We shifted our focus to Rotten Tomatoes and found two datasets on Kaggle relating to such data:

1. https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset?select=rotten_tomatoes_movies.csv
2. https://www.kaggle.com/datasets/andrezaza/clapper-massive-rotten-tomatoes-movies-and-reviews?select=rotten_tomatoes_movies.csv

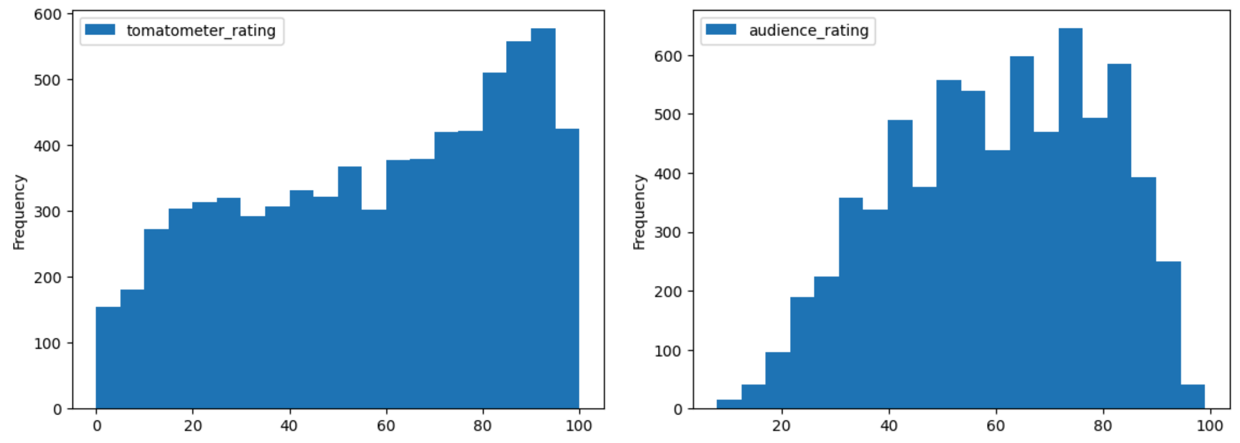
Of the two, the first dataset looked the most promising because it did not have many missing values unlike the second. For the rest of this project, we continued with the first dataset.

There are also public API's that would let us gather additional data about movies. One example is the Open Movie Database API, https://www.omdbapi.com/. However, collecting this data through the API would require a lot of manual effort. Furthermore, we are limited to 1,000 API calls a day on the free version. For the scope of this project, we deemed this potential resource unnecessary to pursue. However, this API would have given us a lot more data on ratings such as from Metacritic, IMDb, and Rotten Tomatoes ratings all in one.

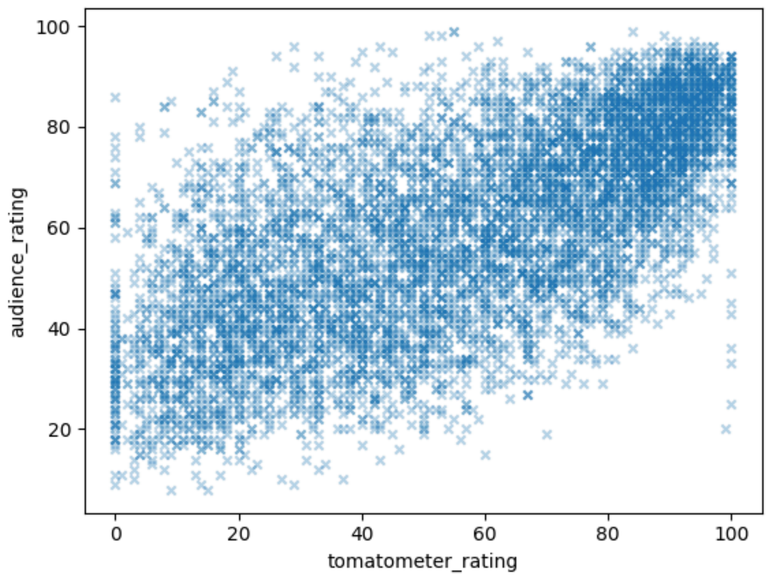
Data Evaluation and Preprocessing

1. Feature Selection

To refine the dataset for predicting audience ratings, we began by analyzing which columns had predictive potential and discarded irrelevant features. For instance, we determined that summarizing fields like "rotten_tomatoes_link" and "movie_info" provided no numerical significance that could impact our model in any sort of way, so they were removed. We focused on maintaining features with intuitive relevance including "runtime," "content_rating," "genre," "directors," "actors," and "production_company" which we later evaluated. Meanwhile, we retained "audience_rating" as the target variable over the "tomatometer_rating" because we believed the abundance of audience ratings would be more reliable than the ratings of a few critics per movie. To back this assumption, we see below that audience ratings generally are more normally distributed than critic ratings, which is likely due to their large numbers.



However, the two generally correlate as they increase together as shown in the plot below:



2. Instance Selection

To ensure the dataset is relevant to modern trends, we excluded movies released before the year 2000, since their audience preferences might differ significantly from contemporary films. Additionally, movies with fewer than 1,000 audience ratings were removed to improve data reliability since this is generally a very small audience size for the scale and reach of Rotten Tomatoes. These filters refined the dataset and focused on recent, widely viewed movies to enhance the accuracy of audience rating predictions.

3. Feature Evaluation

Content ratings include the ratings “R,” “PG,” “PG-13,” etc. The average audience rating for each content rating assigned is listed in the table below. We see that most categories perform generally the same overall, with an average rating localized around 60.00 with similar deviations. This suggests that content rating doesn’t influence the performance of movies much.

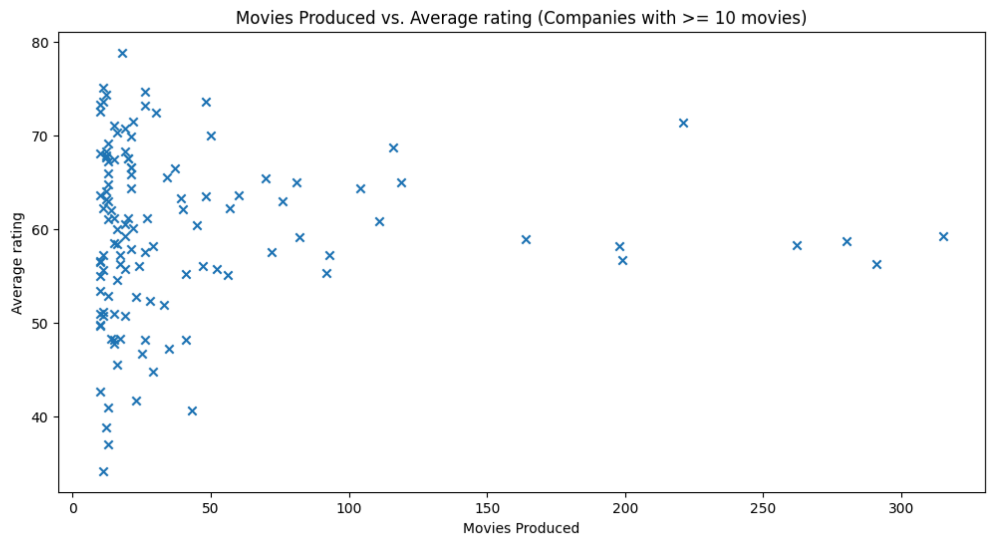
Content Rating	**Average**	**Std. Dev.**	**Mean Abs. Dev.**	**Instances**
R	57.19	19.92	16.98	3151
PG-13	59.68	19.01	15.97	1946
NR	66.43	18.17	14.84	1025
PG	63.67	18.67	15.78	763
G	64.89	18.17	15.11	190
NC17	65.92	20.13	17.10	12

There are 21 unique genres in total. The 5 highest average-rated and 5 lowest average-rated movie genres are listed below. We see a large disparity in average audience ratings between these genres. This suggests that genre can have a significant impact on movie ratings.

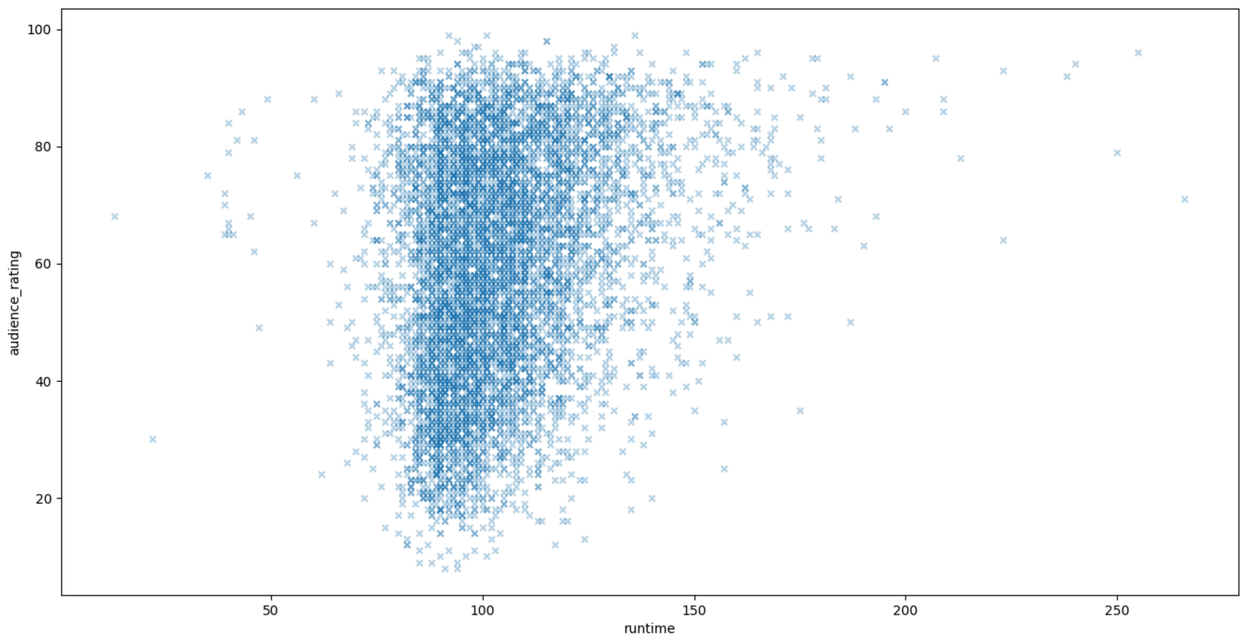
Genre	**Average**	**Std. Dev.**	**Mean Abs. Dev.**	**Instances**
Documentary	76.39	13.07	9.95	665
Sports & Fitness	75.9	12.57	9.72	73

Special Interest	73.84	14.82	11.44	460
Anime & Manga	72.5	13.59	11.00	6
Faith & Spirituality	71.53	13.83	10.22	30
…	…	…		…
Comedy	56.73	18.21	15.35	2461
Science Fiction & Fantasy	56.64	20.52	17.22	786
Mystery & Suspense	53.66	19.43	16.45	1478
Cult Movies	52.42	17.01	14.42	12
Horror	45.44	17.92	14.87	767

Production companies include studios and distributors like “Disney,” “Warner Bros.,” and “Paramount.” Overall, we notice that as production companies increase in total movies they’ve produced, the average ratings for their movies tend to normalize around 55-60%. This may be due to the “law of large numbers” since the ratings of 55-60% are the ratings we see most common in the dataset. Meanwhile, production companies with a generally low number of movies have distributed average ratings.



Run time appears distributed in a “filled-in” v-shape. In other words, short movies tend to perform well, average-length movies spread the entire spectrum, and long movies perform well again. This could be because short and long movies alike are generally special features and may tend to be more highly celebrated.



4. Feature Encoding

Preprocessing steps were applied to standardize and encode categorical features. Categorical fields like "genre" and "content_rating" were one-hot encoded to facilitate their incorporation into machine learning models. This turned our 2 columns into 27 columns. We did not do the same for “directors,” “actors,” and “production company” since they had a larger pool of classes which, if one-hot encoded, would create over 20,000 unique columns. This dimensionality would make our models very computationally expensive as well as throw off our models due to the inherent difficulties of handling high-dimensionalities. Therefore, we aimed to try alternative approaches. Ultimately, we decided on a frequency-based encoding where the most common actors were given the lowest encodings. Both the “actors” and “directors” features were integer encoded to integrate this data into our models. We set a 10% threshold on actors and encoded the discarded actors simply as the largest encoding plus one. We did the same for directors but, instead, our threshold was that they needed to have directed at least 2 movies—this alone removed over half of the pool of directors. This strategy helped remove most of the actors and directors that were lesser known. We deemed the lesser-known actors as the ones that appear at a lower frequency throughout the dataset. Setting a 10% threshold allowed the more popular actors and directors to be featured more prominently within our model.

The concept of our model revolves around the idea that famous actors and directors tend to produce movies with more box office success. As a result, the lower integer encoded actors and directors would tend to be more favorable in our model when it comes to determining box office success for a particular movie. We decided to keep all production companies for encoding since there were about 1,500 in total which was manageable. We later tried changing these thresholds and retraining our models on the altered datasets, but overall we found that these values worked the best error-wise and computationally.

One additional method we used was to only take the top 3 actors listed per movie via the lowest encodings, and create distinct columns “actors1,” “actors2,” and “actors3.” We did this while keeping the threshold percentage method, so, if there were no significant 2nd or 3rd supporting actors, then their encoding would still be the highest encoding plus one. However, this may not be the most accurate method, as larger movies would have a bigger cast with much more popular actors than smaller movies. Nevertheless, it improved the model marginally so it was kept.

5. Normalization

To ensure that the various machine learning algorithms that we implemented are not greatly influenced by the relative size of the features, we decided to normalize the data. This was done using Sklearn’s MinMaxScaler which divided the actual value by the max value of the column it belonged to. The features that we scaled were “runtime,” “directors,” “actors1,” “actors2,” “actors3,” and “production company.” Since most of our algorithms were dependent on a distance function, this would improve the accuracy of our models. Without this scaler, the distance function would amplify the features with relatively larger values.

Model Training and Analysis

To train our models, we split our data into train and test with 80% going to the training dataset and 20% going to the testing dataset. To accurately compare our models against each other, we used mean absolute error. Additionally, we can intuitively use mean absolute error to evaluate the quality of an individual model without needing comparisons, as the true values of audience ratings are bounded between 0 and 100. For instance, a model's mean absolute error of 10 indicates that, on average, the model's predictions deviate by 10 rating points, making it easy to judge whether the error is acceptable in general.

1. Linear Regression

The first model that we implemented was Linear Regression. Initially, we had a feeling that this model would not be the best representation as our data could not have a linear trend. However, when implemented, we got a mean absolute error of 13.00 to 14.00 which depended on the random sampling of the train-test split. Similarly, when we tested on the same training dataset it got very similar results, validating the lack of overfitting.

2. Logistic Regression

The second model that we implemented was the Logistic Regression model. Using mean absolute error as our error function, this model outputted a training error usually around 14.75 while the test error was usually slightly higher around 15.25.

3. Linear SVC

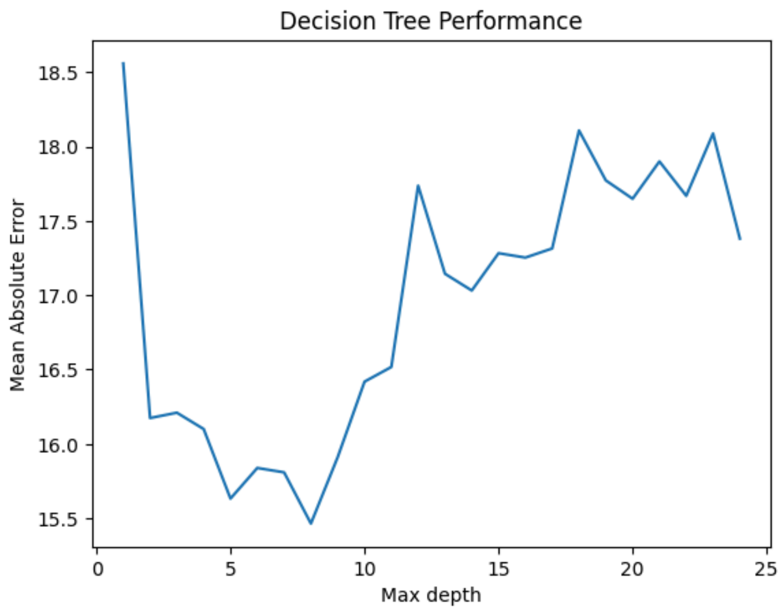
We were curious as to whether using an SVC model would lead to better results than the regression models, so we decided to test it out on our dataset. The test error came to about 15.18 on average, while the training error for the model came out to approximately 15.46, also depending on the sampling of the train-test split.

4. Kernel SVM

We were also curious about the effects of kernelization, and if our models could be improved upon using Kernel SVM. Our Kernel SVM model performed with a test error of about 14.70 and the training error came to be approximately 14.54 depending on the sampling, slightly better than using Linear SVC.

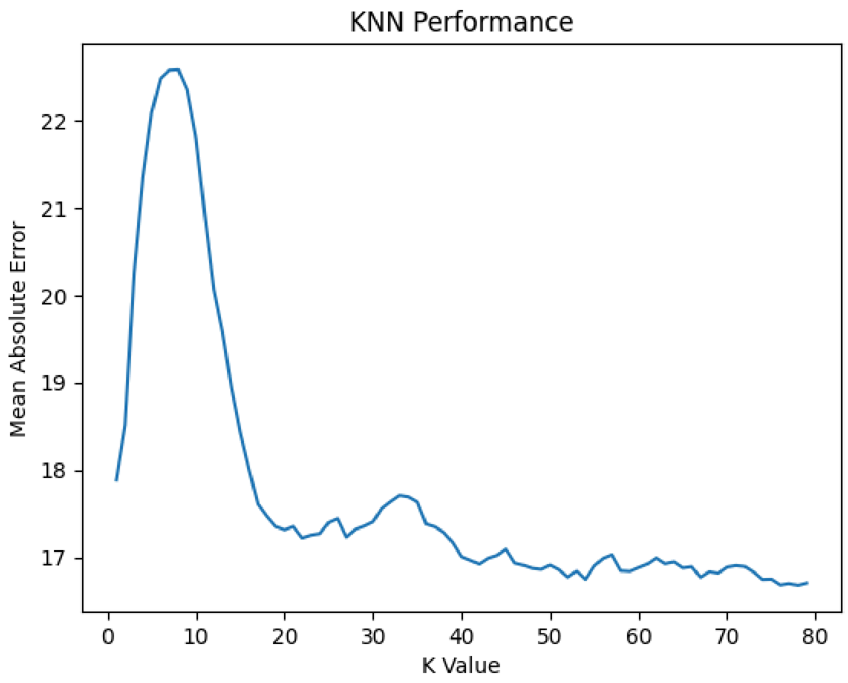
5. Decision Tree Classification

The Decision Tree Classifier can handle non-linear data well while handling both numerical and categorical data properly. We implemented this model and usually got a mean absolute error of around 15.00 to 16.00 depending on the train-test sampling. We noticed that the best depth of the decision tree was usually around 6 to 8.



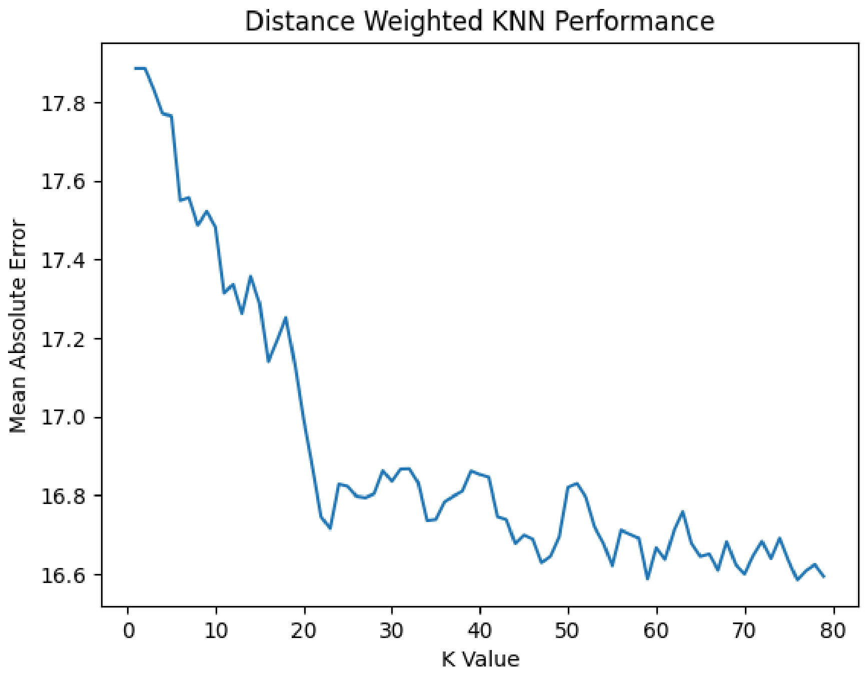
6. *k*-Nearest Neighbor

The next approach that was implemented was the *k*-Nearest Neighbor model. In our code, we implemented this model by trying all *k* values in the range 0 to 80. The best minimum absolute error that we got from all values of *k* was approximately 16.00 to 17.00. We generated a graph to see the association between the *k* values and the error and saw that the error value increased for values between 1 to 10, meaning it could be overfitting using those *k*'s or being subject to outliers. However, the general trend after *k* equaled 10 was that the error began to generally decrease with some minor deviations along the way. Using the elbow method, the ideal value of *k* would be around 20 to 30.



7. Distance Weighted *k*-NN

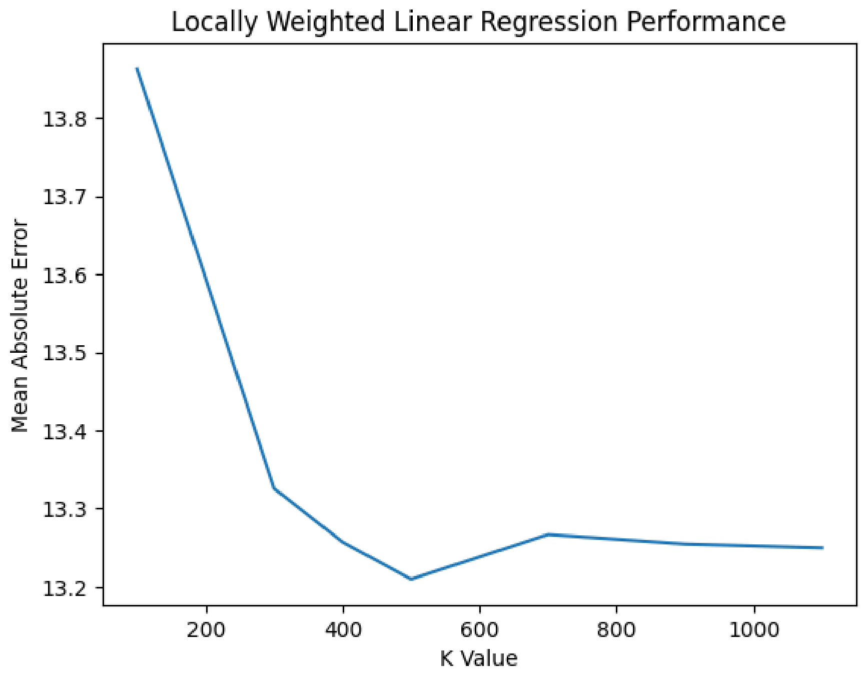
The implementation of distance-weighted *k*-NN was quite similar to the regular *k*-NN approach defined above. However, the plot of distance-weighted *k*-NN was different, especially for small values of *k*. In regular *k*-NN we saw the error increase in the range *k* equaling 1 to 10. However, for distance-weighted *k*-NN, the error generally decreased throughout all *k* values. This showcases that since we are using weights, this model can handle outliers better while not overfitting the data. Using the elbow method to determine the ideal value of *k* would help us determine the value of *k* to be approximately 20 to 30 again, just like the previous *k*-NN model. The best minimum absolute error was similar to the regular *k*-NN model, with this model giving a minimum error of 16.00 to 17.00 as well.



8. Locally Weighted Linear Regression

So far, Linear Regression seems to be the model that performs the best. Another modified version of *k*-NN that we had implemented was Locally Weighted Linear Regression to try to stick to the success of this model. This model would take the *k* nearest neighbors of the new data point and then fit a linear regression line only through those points. Therefore, this model does not focus on generalization from the entire data but instead emphasizes making proper predictions based on local data. We generally saw that a large *k* value was needed for this model: depending on the train-test split, the optimal *k* was usually between 400 to 600. In general, the minimum absolute error occurred around *k* as 500 which tended to range from 12.80 to 14.00

depending on the sample. This was a generally solid model in comparison to the previously implemented models. However, this model was very computationally expensive and took over 10 times longer to run than other models. Furthermore, there existed no module in our Python libraries that provided this, so we had to implement it ourselves and it may not be as robust.



9. Conclusions

The best and most reliable model appeared to be linear regression with a mean absolute error of around 13.00 to 14.00. It is important to contextualize this performance. Given that the true values of ratings range only from 0 to 100, a mean absolute error of even 13.00 represents a notable percentage of the total range, which suggests there is a limit to the model's practical utility. However, considering that most individual features we evaluated in the dataset originally had mean absolute deviations ranging from about 9.00 to 17.00 when calculating their respective averages, most of our models' performances align with the inherent variability in the data. That said, we had hoped that aggregating the data altogether would help our models identify stronger patterns and would lead to significantly improved predictive accuracy, but the results suggest that the underlying complexity or noise in the data may still be a limiting factor.

Applying The Model

For the sake of interest, we wanted to apply our model to upcoming movies to predict how they will do before their release. We retrained the most promising models on all the data available (keeping the best *k*-values we found where necessary) and got these predictions for these four upcoming movies:

Movie Name	**Predicted Audience Rating**		
	Lin. Reg.	**Decision Tree**	***k*-NN**
A Minecraft Movie	58.17	49.00	51.00
Mufasa: The Lion King	78.17	85.00	72.00
Captain America: Brave New World	51.91	49.00	49.00
Flight Risk	54.70	59.0	38.0

Conclusion

There are many challenges associated with movie predictions. The feature we wanted the most but could not obtain reasonably was budget. We hypothesize that budget is a large predictor because it can serve as a metric that correlates to cast quality, production quality, and marketing quality. However, given the scope of this project, there was not enough time to acquire this data.

We should also consider the variability of ratings due to factors such as competitive releases, social media buzz, and revivals through streaming services. With more time, we would like to add more features to the data set. Along with the aforementioned features, it would likely prove useful to have data such as if a movie is a sequel, if it is based on an existing franchise (book, game, TV show), or what type of role significant actors play (protagonist, villain, etc).

Another important consideration is how our preprocessing techniques themselves influence prediction accuracy. While we aimed to uncover patterns through feature engineering and data aggregation, our results suggest that the inherent variability in audience ratings might limit the achievable precision without additional data. Exploring advanced ways of encoding data and other learning techniques could potentially capture relationships between features and ratings more effectively.

Additional Information

The data parsing, evaluation, and encoding processes, as well as the model training and analysis, can be found below:

https://github.com/au-s-ti-n/movie_predictor

The relevant directions for installing necessary packages and running the code repository can be found in the README.md file.

References

Khalid Ibnal Asad, T. Ahmed and M. Saiedur Rahman, "Movie popularity classification based on inherent movie attributes using C4.5, PART and correlation coefficient," *2012 International Conference on Informatics, Electronics & Vision (ICIEV),* 2012. https://ieeexplore.ieee.org/document/6317401

Vikranth Udandarao, Pratyush Gupta. “Movie Revenue Prediction Using Machine Learning Models,” 2024. https://arxiv.org/pdf/2405.11651v1