Netflix Movie Recommendation System

Mikhail Sinitcyn, Nicholas Zhang
CMPT459

November 27, 2024

Data Preprocessing

In this project, we combined Netflix ratings data from the Netflix Prize dataset with IMDb data to map each movie to its features, including cast, composers, directors, producers, genre and release year. This took a lot of preprocessing.

Firstly, we removed films past 2005 due to the scope of the Netflix dataset. We only kept movies, as individual TV show episodes have unique id's that are difficult to map to the originating series, greatly complicating preprocessing and downstream tasks.

We then mapped the resulting 10,000 movies to their IMDb id's. We used this mapping to extract features for each movie from the IMDb dataset. Computing a count of each feature, we only kept those with at least 20 entries, resulting in 450 features.

Furthermore, the Country feature does not indicate country of origin, rather every country of major release. Thus, this feature was explicitly removed as noise. We also didn't use isAdult since it was a particularly rare feature from the early days of Netflix, with only 20 movies out of the selected 9,000 that were rated adult. The numerical values (release year and duration) are np.float64, we treat them as categorical features (decades and duration grouped, respectively).

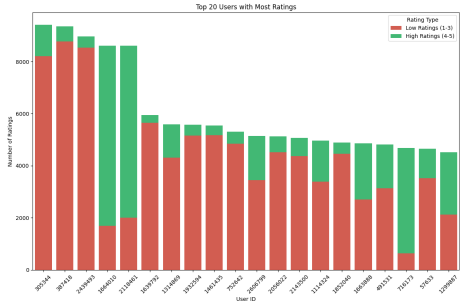
After extracting movie features, we created user preference features to represent each Netflix user. We compiled the top 10,000 users with the most ratings across the selected 9,000 movies, and computed preference distributions as high(4-5) and low ratings(1-3) ratios for each feature. The resulting feature vectors values ranged from 0.0 to 1.0, indicating lowest and highest preference, respectively. Missing preference values where imputed as -1 for downstream tasks.

Exploratory Data Analysis

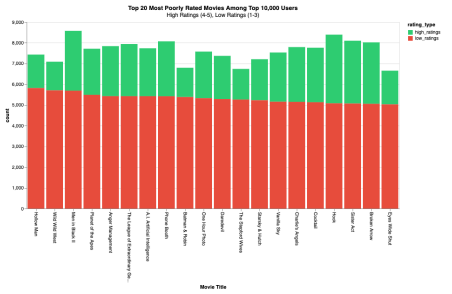
Netflix Analysis

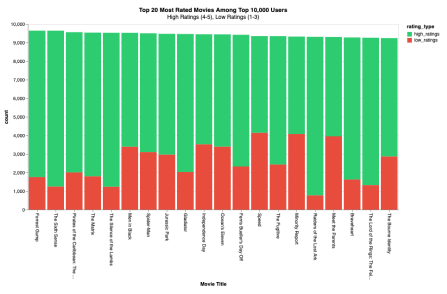
Among the top 10,000 users, we find that the most reviewed movies have more high ratings than low. Conversely, we find that the users with the most ratings

have more low ratings than high.

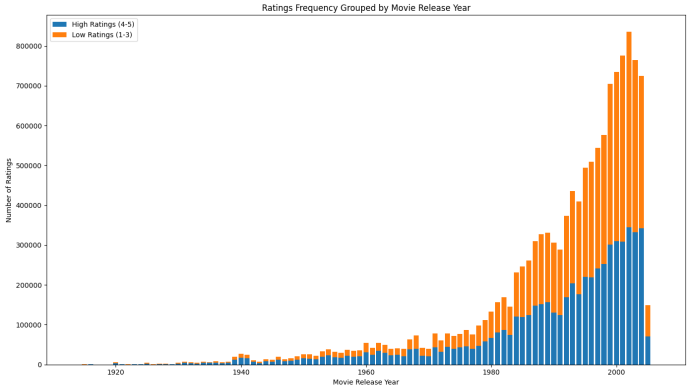


Unsurprisingly, we find that the list of top 20 most rated movies consists of box office hits like *Forrest Gump, The Sixth Sense, Pirates of the Caribbean,* and *The Matrix*; among which, *Raiders of the Lost Ark* is rated the most highly. The top 20 most poorly rated movies include famously panned films such as *Wild Wild West, Men In Black II, Planet of the Apes,* and *The League of Extraordinary Gentlemen.* Interestingly, two of these movies star *Will Smith*, and many of them were box office successes despite the low ratings.



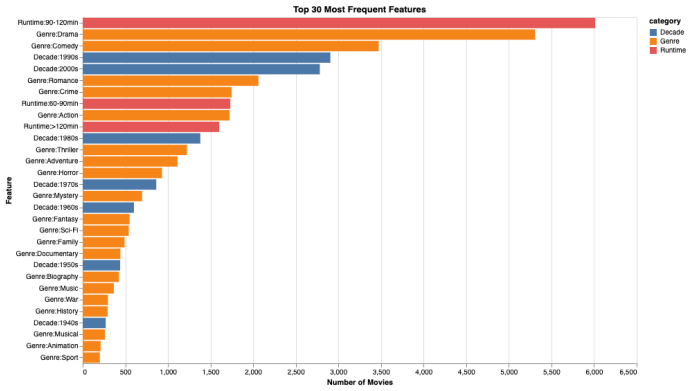


Plotting the distribution of movie ratings by release year, we find that users leave more reviews for most recent movies. There is an apparent trend that the ratio of low of ratings seems to increase each year, with 2002-2005 films having a majority negative ratings.

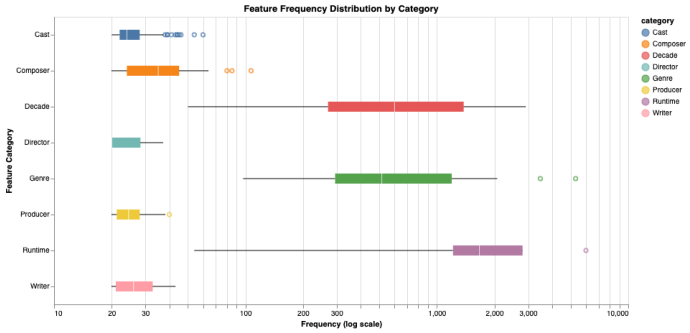


IMDb Analysis

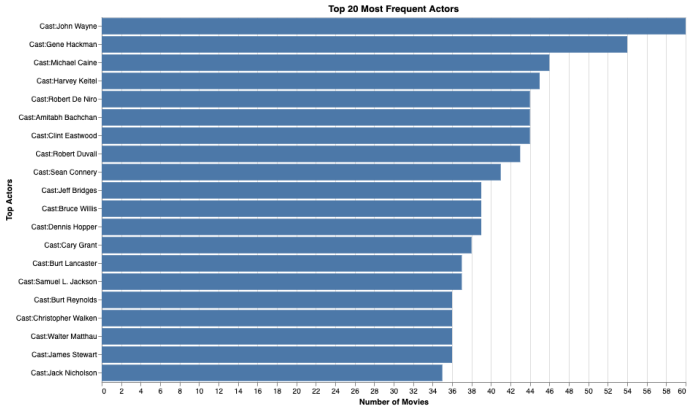
We analyze the distribution of our engineered popular features across the 9,000 movies in our dataset. Naturally, we find runtime, release decade, and genre to be the most popular features as they are necessary for every movie. 90-120 minute is the most popular feature, followed by Drama, Comedy, and 1990's, 2000's films.



We then plot the distribution of features by category. This plot confirms the overwhelming presence of release decade, genre, and runtime features.

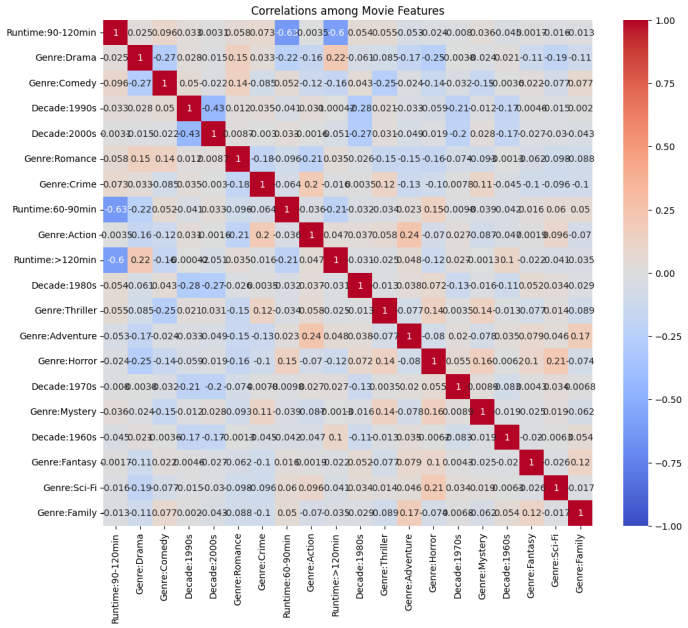


Notably, we observe an uneven frequency distribution of cast features, indicating that some actors have as many as 30-60 credits. Ranking the top 20 most frequent actors, we find legends such as *John Waynes, Gene Hackman, Michael Caine, Robert De Niro, Clint Eastwood*, and *Sean Connery*.



Similarly with directors, *Alfred Hitchcock, Woody Allen*, and *Clint Eastwood* have credits in over 30 movies.

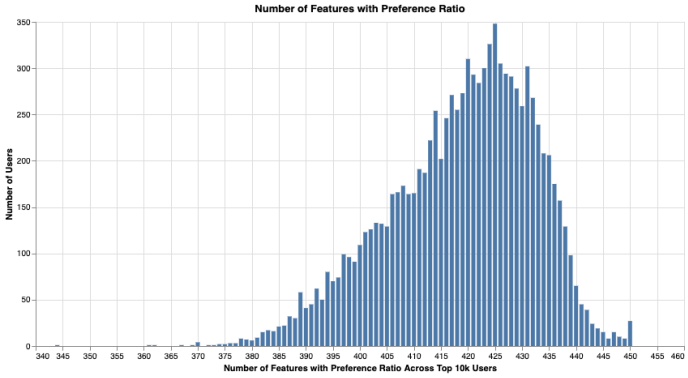
We then analyze correlations between user preference features, again finding the strongest correlations between genres like *Action-Adventure, Horror-Sci-Fi*, and *Crime-Action*. Strongest negative correlations include *Comedy-Runtime over 120min*, and *Romance-Horror*.



User Analysis

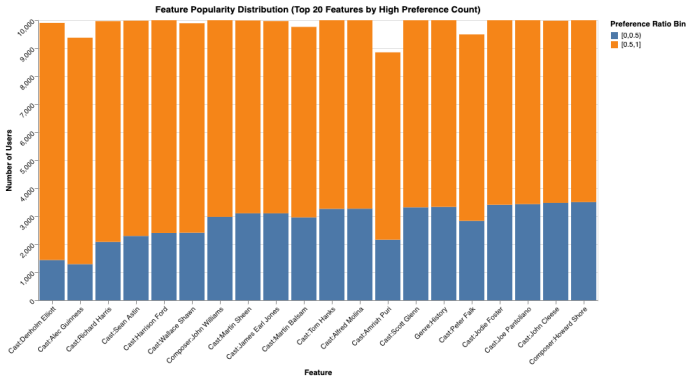
We analyze the engineered user preference features, specifically the distribution of high and low ratings amongst features and the sparsity of the user preference feature vectors.

We find that the user feature vectors are sufficiently dense, with a median of 420 out of 450 features filled in. Many of the engineered features have been rated by all top 10,000 users.



We rank features by highest preference count. We find that the most popular features are predominantly actors, most notably *Denholm Elliott, Alec Guiness, Harrison Ford*, and *James Earl Jones*. Among the top 20 features, two com-

posers stand out: *John Williams (Jaws, Star Wars, Indiana Jones)* and *Howard Shore (Lord of the Rings, The Hobbit)*. Naturally, these actors and composers earn such high preference scores due to their work in beloved movies.

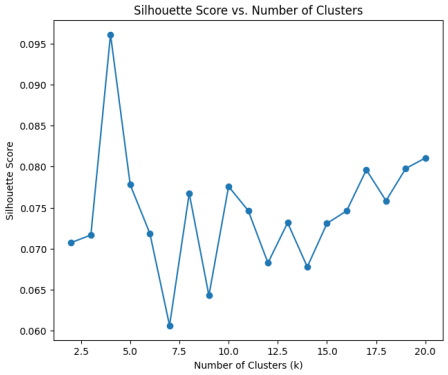


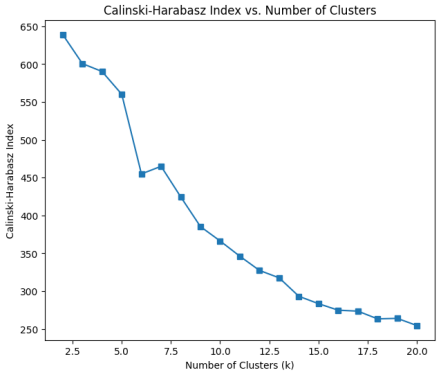
Clustering (Movies)

We cluster movies using K-Means and DBSCAN algorithms. Each movie is represented as a one-hot encoded vector indicating the presence or absence of the engineered 450 most popular movie features.

K-Means

Starting with k-means clustering, we evaluate the Silhouette Score and Calinksi-Harabasz Index over a range of values for hyperparameter k, the number of clusters. Using the former metric plot, we perform k-means clustering with 4 cluster groups, plotting the resulting cluster assignments in 2 dimensions via PCA.





Inspecting the cluster entries using an interactive tooltip legend, we find that the top-left group 0 cluster consists of War and Drama movies over 2 hours in runtime. This includes *Troy, The Punisher*, and *The Lord of The Rings*.

The orange group 1 cluster below it consists of Romance and Drama movies between 60 and 90 minutes in runtime, notably including *Gaslight*.

Immediately below it, the group 0 cluster continues with Action-Adventure movies over 120 minutes in runtime. It notably includes *Star Wars, The Mummy*, and *The Matrix*.

The remaining group 0 and group 1 clusters (blue and orange) include more *Comedy* and *Family* movies, again separated by runtime in the 60-90 and over 120 minute categories.

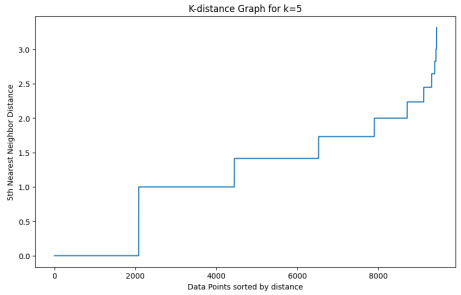
The group 2 and group 3 clusters on the right closely follow the feature combinations of the clusters on the left, separated by the runtime category 90-120 minutes. The nested clusters differ by genre, *Drama, Crime, Thriller*, then *Action* and *Adventure*, and finally *Family* and *Comedy*.

Thus, we find that k-means clustering creates top-level clusters based on *Runtime* features, then creates nested clusters based on *Genre*.

Applying PCA to reduce the dimensionality from 450 features to 50 principal components, we find 6 cluster groups to be the optimal value per the Silhouette Score. Again plotting the cluster assignments in 2 dimensions, we find similar cluster assignments based on *Runtime* and *Genre*, now with finer *Genre* granularity within nested clusters.

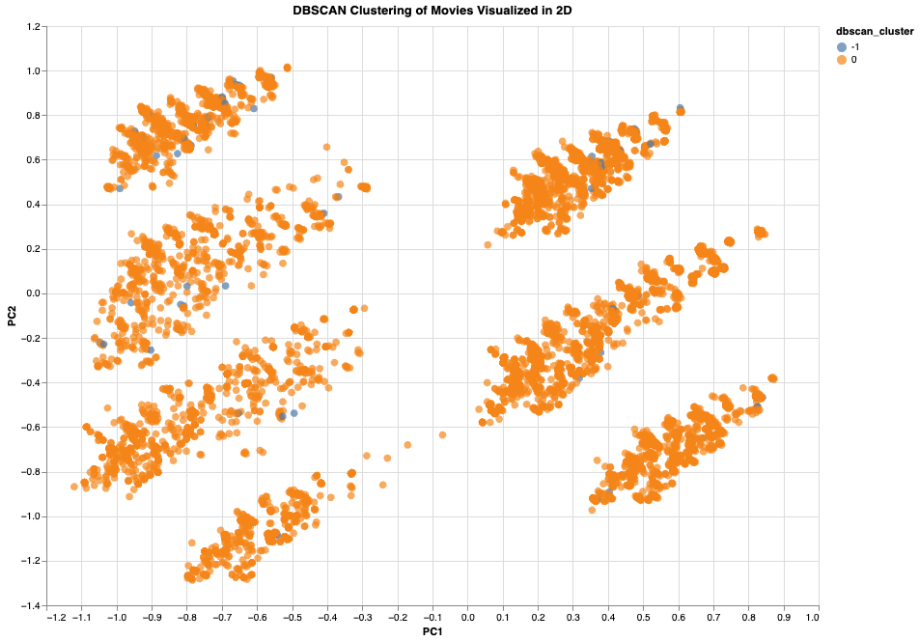
DBSCAN

We select appropriate values for hyperparameters *epsilon* and *min_distance* via the elbow method using a k-distance graph, with a default value of k=5 by convention. Note that this plot is discrete due to the one-hot encoded feature vectors.



Interestingly, we find entirely different results using DBSCAN clustering. Due to the sparse one-hot encoded movie representations, DBSCAN functions as an outlier detection model. The overwhelming majority of movies are grouped into one cluster, as feature-rich movies with exceptionally popular "superstar" cast are detected as outliers. This includes *Batman, Batman Returns, Antz,*

Hook, The Royal Tenenbaums, and *Con Air*.



Intuitively, this makes sense as the average movie has 3-5 popular features. Thus, a movie with 10-15 popular features will simply not have enough neighbors within a distance epsilon to be assigned to a cluster. We find that this result is reduced when performing DBSCAN on PCA-reduced features as pure counts are replaced with principal components, resulting in less interpretable outlier assignments without movie domain knowledge.

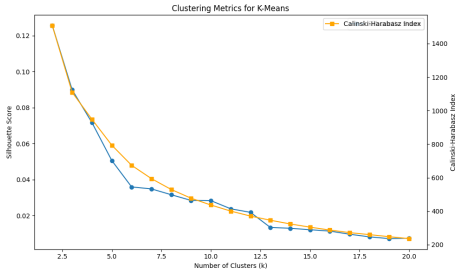
Clustering (Users)

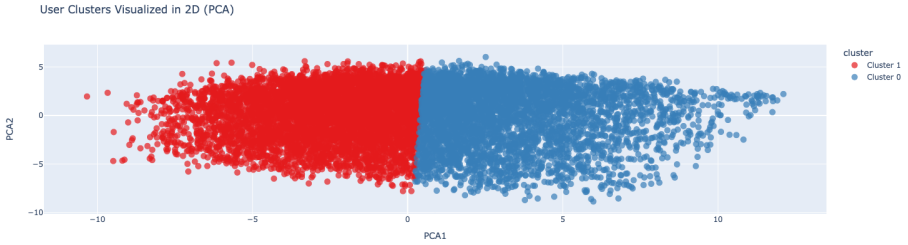
We cluster the top 10,000 Netflix users, represented by their preference ratios across the 450 top engineered features. We find that the movie feature preferences of the top 10,000 users follow an elliptical Gaussian distribution for the first 2 principle components (explaining 30% of the variance), resulting in no valid cluster separation. These results are in stark contrast with the movie feature clusters which have surprisingly well-separated hierarchical cluster groups. Inferring from this scatterplot, k-means clustering seems appropriate.

K-Means

Again using Silhouette Score and Calinski-Harabasz Index to select the hyperparameter value k=2, we find that the cluster assignments across the elliptical

Gaussian distribution of user preferences is effectively trivial.

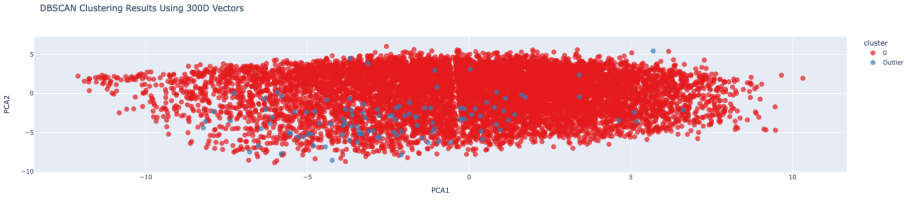




Applying k=2 clustering to the 300 principal components (explaining 67% of the variance), we see an apparent linear separating line when visualizing the first 2 principal components (explaining 30% of the variance). Again, the plot follows an elliptical Gaussian distribution.

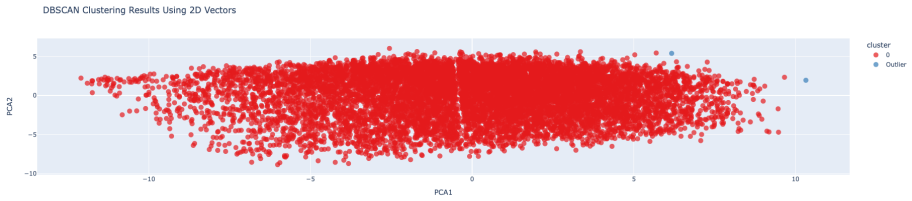
DBSCAN

Given the distribution of the user feature vectors, DBSCAN clustering essentially functions as an outlier detection model, resulting in difficult to interpret outliers. Intuitively, outliers are users with rating distributions across the popular features that deviate significantly from the norm among the top 10,000 users.



As an extreme experiment, we perform a very conservative DBSCAN clustering on a 2D PCA projection (explaining only 30% of the variance) of the dataset and find only two outliers:

- A user with Anthony Quinn and Donald Pleasence as their two most highly rated features: 60's action fan who loves the Halloween movie series
- A user who, despite rating over 800 movies, has not rated a single movie with our engineered features of interest (the right-most datapoint)



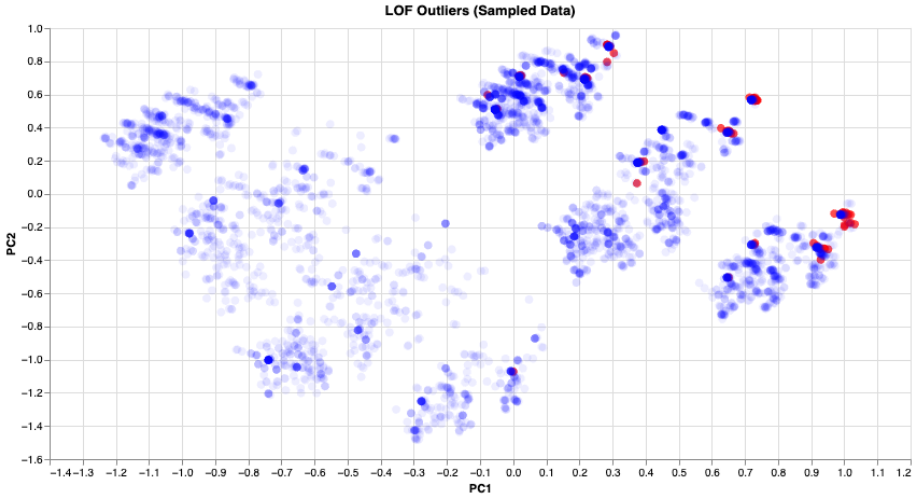
Outlier Detection

We compute Local Outlier Factor and Isolation Forest scores for each movie represented by its popular features to detect outliers. Given the integrity guarantees of the dataset, we interpret no outlier entry as noise, rather an expression of interesting patterns.

Local Outlier Factor

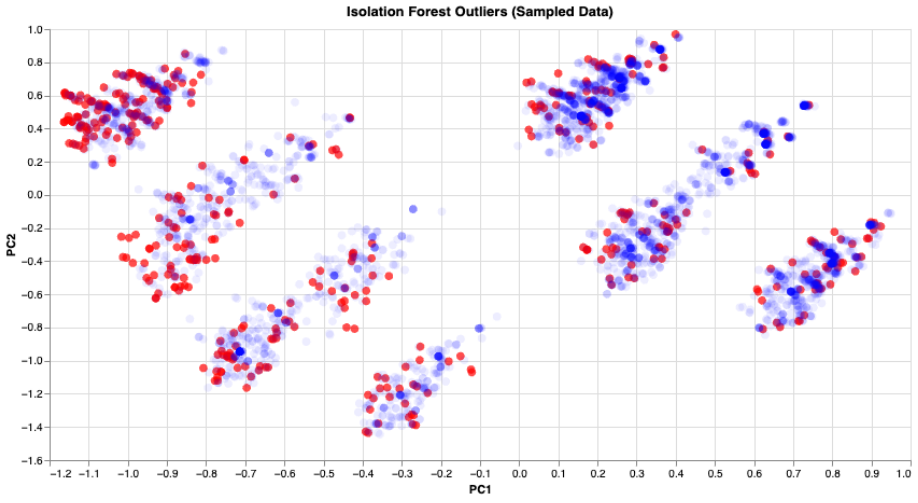
Per the Data Mining Textbook, LOF is a "normalized distance-based approach where the normalization factor corresponds to the average local data density."

Intuitively, we find that movies with unusual feature combinations, namely unusual cast ensembles or surprising individual casting choices for the genre, are given the highest absolute LOF score, thus indicating the highest "outlier" rating. This includes the star-studded *The Royal Tenenbaums, Indecent Proposal*, and *Mars Attacks*. Interestingly, many of the outlier entries form outlier clusters when visualized in 2 dimensions.



Isolation Forest

Similar to DBSCAN, we find that feature-rich movies with exceptionally popular "superstar" cast are detected as outliers by Isolation Forest. In particular, among the 462 detected outliers, many have A-list actors from a wide variety of genres, with some surprising casting choices given the movie genres. Notable outliers include *Lawrence of Arabia, Predator 2, Antz, Sleepy Hollow*, and *Batman Returns*.



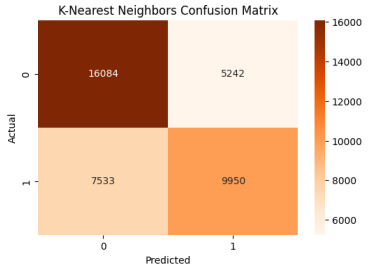
Classification

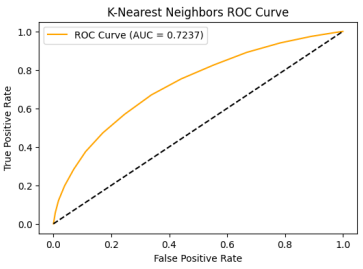
We evaluate classification models in predicting user ratings for movies given movie features and user preference features. Given that our dataset contains millions of ratings from the top 10,000 users across 9,000 movies, we create our training dataset of 39,000 movie ratings using a subset of ratings across 30 users sampled at random.

Model Evaluations

K Nearest Neighbor

We first evaluate the performance of K Nearest Neighbors classifier with default hyperparameter value of k=10. Per the evaluations, this model correctly predicts high user ratings 65% percent of the time, and low ratings 57% of the time, resulting in a combined 61% F1 score and 72% ROC AUC Score. We find that this model's performance does not significantly lag behind that of Logistic Regression and XGBoost, suggesting that the assumption of similarity between nearby data points holds true for this dataset.

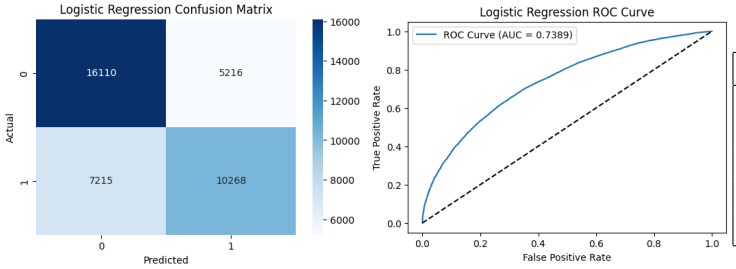




Metric	Score
Accuracy	0.6708
Precision	0.6549
Recall	0.5691
F1 Score	0.6090
ROC AUC Score	0.7237

Logistic Regression

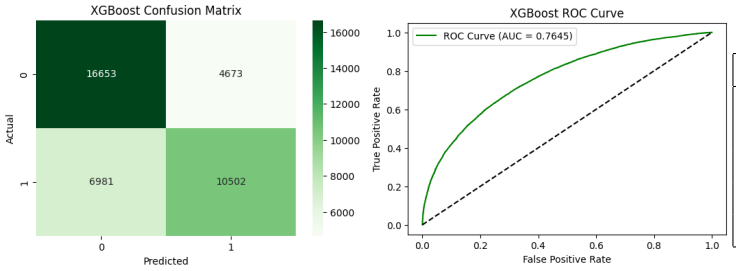
We evaluate the Logistic Regression classifier with default configurations. Per the evaluations, this model correctly predicts high user ratings 66% percent of the time, and low ratings 59% of the time, resulting in a combined 62% F1 score and 73% ROC AUC Score. Given the nature of the movie and user features, we note that including feature interactions in a future experiment may lead to better model performance.



Metric	Score
Accuracy	0.6797
Precision	0.6631
Recall	0.5873
F1 Score	0.6229
ROC AUC Score	0.7389

XGBoost

Per the evaluations, this model correctly predicts high user ratings 69% percent of the time, and low ratings 60% of the time, resulting in a combined 64% F1 score and 76% ROC AUC Score.

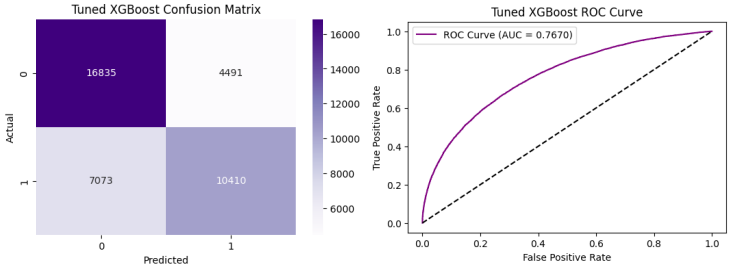


Metric	Score
Accuracy	0.6997
Precision	0.6921
Recall	0.6007
F1 Score	0.6432
ROC AUC Score	0.7645

Hyperparameter Tuning

We also perform hyperparameter tuning on the XGBoost model using Randomized Search over 250 hyperparameter configuration, and evaluate the resulting model performance as well as the learned weights assigned to each feature in the Feature Selection step.

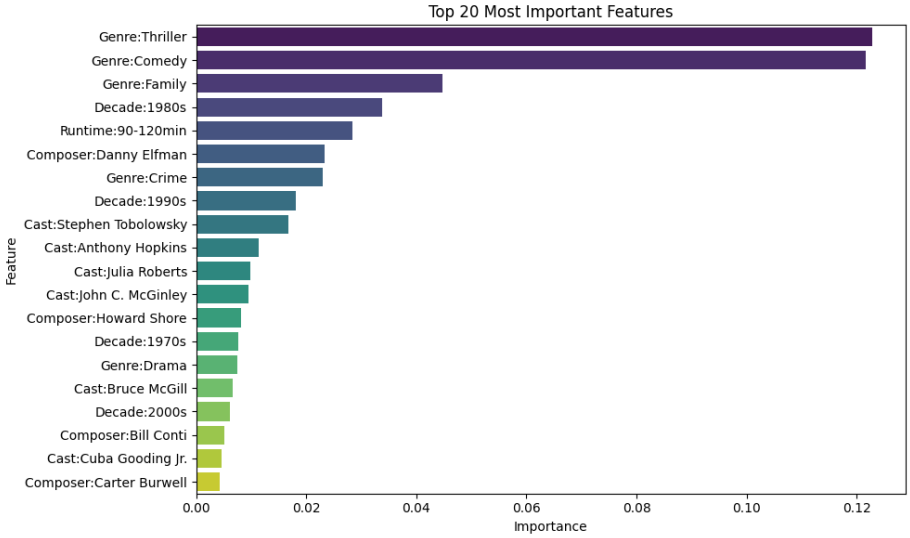
Per the evaluations, this model correctly predicts high user ratings 70% percent of the time, and low ratings 59% of the time, again resulting in a combined 64% F1 score and 77% ROC AUC Score. We find that the model performance is not improved from the baseline settings over this range of hyperparameter values.



Metric	Score
Accuracy	0.7020
Precision	0.6986
Recall	0.5954
F1 Score	0.6429
ROC AUC Score	0.7670

Feature Selection

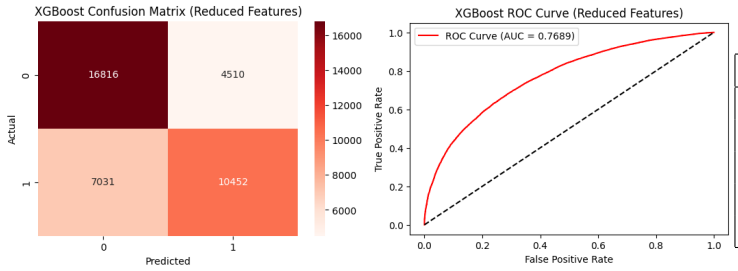
Per our Data Preprocessing step, we performed explicit feature selection in selecting only movies, analyzing only the top 10,000 movies, and using features with over 20 occurrences. Now, as part of our implicit feature selection, we analyze the learned weights of our Logistic Regression classifier as direct "importance" scores across all features.



We find that *Genre* features like *Thriller, Comedy*, and *Family* are overwhelmingly the highest predictors of a high rating. We then find that the *1980's* are the most highly rated decade, followed by the *1990's* and *2000's*, and *90-120* minutes is the most preferred runtime, neither of which come as a surprise. Interestingly, Danny Elfman is the highest individual predictor of high ratings, confirming his status as a beloved composer for his work on *Batman, Spiderman, Beetlejuice, Good Will Hunting, Nightmare Before Christmas*, and *Men In Black*.

We also find that a high number of features are assigned negligible weights, indicating that many actors are insignificant predictors of high movie ratings. This especially includes cast members with neutral user preference ratios.

Re-evaluating the XGBoost model with the bottom 386 features dropped, we find that the performance is not impacted. Thus confirming that the features have no effect on the predictive abilities of the model.



Metric	Score
Accuracy	0.7026
Precision	0.6986
Recall	0.5978
F1 Score	0.6443
ROC AUC Score	0.7689

Model Interpretation

Given a user's movie feature preferences and an individual movie's features, the model predicts whether the user would give the movie a low rating (1-3 stars) or high rating (4-5 stars). Per the evaluations, this model correctly predicts high user ratings 69% percent of the time, and low ratings 60% of the time, resulting in a combined 64% F1 score and 76% ROC AUC Score.

In practice, such a model would compute batch predictions for all users periodically via a cronjob (per Designing Machine Learning Systems) and improve over time as users left more reviews.

Challenges, Limitations, and Potential Future Work

The sheer volume of data presented the biggest challenge. As a result, data preprocessing (ETL) was much harder than expected, taking almost a week of work. Downstream in the classification task, there were too many ratings to train a classifier on, so aggressive sampling methods had to be used to limit the training dataset to 39,000 entries. In the future, this work could benefit greatly from increased compute.

Conclusion

Through our analysis, we gained insights into movie preferences from the Netflix-IMDb dataset. Box office hits consistently receive the most ratings, though commercial success doesn't guarantee positive reception (e.g., Wild Wild West, Men in Black II). The most recent movies (2002-2005) in the dataset tend to receive more negative ratings despite higher review volumes, which speaks to

declining movie quality or changing user preferences. Genre preferences show strong correlations, particularly Action-Adventure and Horror-Sci-Fi pairings, while Romance-Horror shows expected negative correlation.

We find that different algorithms excel at different tasks. K-means effectively clustered movies hierarchically by runtime then genre, but failed to meaningfully separate users, suggesting viewer preferences don't follow clear groupings. DBSCAN and Local Outlier Factor proved most valuable at finding movies with unusual feature combinations, particularly unexpected casting choices outside typical genres.

For predicting user ratings, we achieved 76% ROC AUC with XGBoost, with genre being the strongest predictor followed by decade and runtime. Surprisingly, most individual actors had negligible impact on ratings. The explicit feature selection of keeping only features with 20+ occurrences proved effective, as further dropping the bottom 386 features did not impact model performance.