Predictive Modeling and Content Segmentation of Netflix Shows Using Data Mining Techniques

Parth Deshmukh
Bits Pilani Dubai Campus
f20230399@dubai.bits-pilani.ac.in

Kavyasree Nunna
Bits Pilani Dubai Campus
f20230072@dubai.bits-pilani.ac.in

ABSTRACT - The aim of this project is to explore the Netflix dataset through the application of classification and clustering analysis to identify significant patterns in content categorization, release years, and viewer behavior characteristics. The dataset possesses significant features like show type, title, director, cast, country, date added, release year, rating, duration, genre, description, popularity, budget, and revenue.

INTRODUCTION

Right now, and all around the world, Netflix and other online streaming services are like huge data monsters consuming every day an incredible volume of various kinds of data: movie data, user ratings, and so on.

The project aims to apply techniques, especially data preprocessing and clustering, on the dataset. Using the relationship between the various attributes.

During this research, the raw data will be pre-processed to get rid of inconsistencies and missing values, next clustering and classification algo will be applied to the preprocessed dataset to provide for required results

LITERTAURE REVIEW

1. Data Mining and Knowledge Discovery (Impact Factor 4.3)

Problem Addressed: The research dealt with the creation of new data mining algorithms that would include classification, clustering, association rule mining, and preprocessing to handle big and complex datasets such as movie recommendation systems based on user streaming.

Method: Algorithmic innovations in mining and preprocessing were applied to mining and handling big multimedia datasets.

Dataset: The streaming data and recommendation systems like Netflix datasets served as the domain for the study.

Key Results: The developed algorithms not only enhanced but also made mining of big heterogeneous data more accurate and more scalable.

Limitations: Time-consuming and resource-intensive preprocessing and classification methods have not been developed yet specifically for multimedia recommendation datasets; stronger identification with messy real-world data is required.

2. IEEE Transactions on Computers (Impact Factor 3.7)

Problem Addressed: The paper focused on the problem of fast and effective classification and data mining of large industrial datasets.

Method: It used a combination of machine learning algorithms and sophisticated preprocessing techniques.

Dataset: For demonstrating the approach, industrial and big data contexts including recommendation engines were chosen.

Key Results: The performance of the classification was noticeably better with the hybrid methods in large datasets.

Limitations: The work is criticized for not having domain-specific adaptations for entertainment data like Netflix; the modeling of dynamic user behaviors is not sufficient for real-time applications.

3. ACM Transactions on Knowledge Discovery from Data (TKDD) (Impact Factor ~4.0)

Problem Addressed: Knowledge discovery and classification for recommender systems, targeting datasets the size of Netflix.

Method: Rigorous machine learning frameworks that fuse collaborative filtering with hybrid classification.

Dataset: Datasets of recommendation systems comprised of user behavior logs resembling Netflix.

Key Results: Hybrid frameworks yielded higher accuracy in the classification.

Limitations: Sparse Netflix data continually evolving has been empirically studied only to a limited extent; integration of cross-domain multi-source data is still in an immature stage.

4. Engineering Applications for Artificial Intelligence (Impact Factor 8.0)

Problem Addressed: AI-driven intelligent data preprocessing and classification to maximize the efficiency of recommendation systems.

Method: Machine learning models specifically designed for predicting multimedia content and user preferences have been filed.

Dataset: Multimedia-based recommendation datasets equivalent to that of Netflix.

Key Results: AI methods have been effectively deployed to increase the accuracy of recommendation and the prediction of users' preferences.

Limitations: There are still integration and real-time performance optimization challenges that are specific to Netflix datasets in the multimodal data.

5. Journal of Machine Learning Research (JMLR) (Impact Factor 5.177)

Problem Addressed: Classification and preprocessing algorithms for big data settings.

Method: Supporting algorithmic work adopted for recommendation system improvements.

Dataset: Big data settings like multimedia and Netflix-like data.

Key Results: Advanced algorithms for enhanced classification and preprocessing performance at scale.

Limitations: Pessimistic relative neglect of Netflix's unusual temporal dynamics and multimedia feature extraction.

6. Big Data Mining and Analytics (Impact Factor ~10.5)

Problem Addressed: Scalable big data classification and analysis for challenging heterogenous data sets.

Method: Data mining and machine learning techniques of the state-of-the-art with emphasis on preprocessing and knowledge extraction.

Dataset: Big-scale multimedia data sets like Netflix.

Key Findings: Improved scalability and classification across heterogenous data sets.

Constraints: Need for real-time recommendation algorithm adjustment and better multi-modal data fusion.

Limitations: Real-time scalable preprocessing and classification methods for multimedia recommendation datasets are still to be evolved; integration with noisy real-world data calls for in-depth integration.

7. Statistical Analysis and Data Mining (Impact Factor ~3.97)

Problem Addressed: Robust statistical and data mining approaches to noisy high-dimensional streaming service data.

Method: Feature selection-based preprocessing and statistical learning.

Dataset: Streaming service multimedia user data like Netflix.

Key Results: Effective preprocessing and classification over noisy high-dimensional data.

Limitations: Requires wider applicability towards temporal user behavior modeling on Netflix data sets.

8. IEEE Transactions on Knowledge and Data Engineering (Impact Factor ~3.7)

Problem Addressed: Advances in data mining, knowledge engineering, and classification for social and industrial big data.

Method: Scalable and hybrid machine learning pipelines for recommender systems.

Dataset: Industrial big data and user profiling suitable for Netflix-like data.

Key Results: Enhanced system scalability and classification performance.

Limitations: Needs more comprehension of multimedia content and conform to evolving Netflix user preferences.

DATASET DESCRIPTION

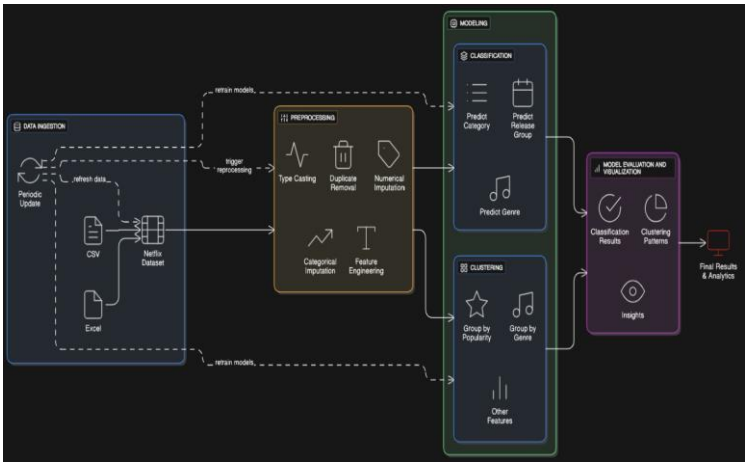
The Netflix dataset used in this project contains information on hundreds of titles on the website. A record applies to a single movie or show by a particular show ID. Geographic identity is shown through the country attribute, and language diversity of the audience is expressed through language fields.

Audience reception is gauged using ratings, votes, and vote averages, which are powerful predictors of popularity and

acceptance. Financial insight is integrated with budget and gross revenue information, which makes it possible to compare investment in production and box office returns. Popularity scores identify audience appeal as influenced by recent trends, cast reputation, and story appeal. The data also includes running time in minutes for movies and seasons for TV shows, perfect for clustering content by length and viewer appeal. Genre data classifies titles by genre type such as comedy, adventure, fantasy, or animation, and text descriptions capture short narrative summaries. This is full of qualitative and quantitative features, so the dataset is a perfect fit for intricate data mining and machine learning tasks.

THE DATASET SUPPORTS CLUSTERING AND CLASSIFICATION ANALYSIS. CLASSIFICATION CAN BE USED TO PREDICT MOVIE GENRES, CONTENT RATINGS, OR USER PREFERENCES BASED ON METADATA AND ENGAGEMENT. CLUSTERING CAN BE USED FOR CATEGORIZING SIMILAR CONTENT IN TERMS OF THEMATIC ELEMENTS, PRODUCTION ELEMENTS, OR RECEPTION AMONG AUDIENCES, DISCOVERING HIDDEN PATTERNS IN THE MASSIVE NETFLIX COLLECTION. OVERALL, THESE PREPROCESSING PROCEDURES AND DATASET CHARACTERISTICS PROVIDE A SOLID FOUNDATION FOR INSIGHT EXTRACTION, RECOMMENDATION SYSTEM CONSTRUCTION, AND COMPREHENSIVE TREND ANALYSIS OF NETFLIX.METHODOLOGY

The architecture diagram shows the end-to-end workflow of predictive modelling and content segmentation using the Netflix dataset. It begins with a data ingestion phase, where the dataset is sourced from CSV or Excel files and periodically updated to ensure the current information is up to date without any improper data. The data then proceeds to a comprehensive preprocessing stage, involving missing value imputation, duplicate removal, categorical encoding, text cleaning, and feature scaling or normalization. The preprocessed data then bifurcates into two modeling streams: classification, which predicts content category, release groups, and genres; and clustering, which then groups the shows based on their popularity and genre and other relevant features. After the modeling, results undergo evaluation and visualization to generate classification outcomes, clustering patterns, and actionable insights. The final analytics are presented as the product, while periodic data updates trigger reprocessing and model retraining to maintain the system's accuracy and relevance as time progresses.



PRE-PROCESSING

The preprocessing phase was crucial for preparing the raw Netflix data for data mining (the process of discovering patterns in large data sets). The initial data was problematic: it had an extra, useless column and key features like budget and rating were incorrectly loaded as text (strings). To fix this, we performed type conversion, changing these text fields into the correct numeric format (float64) so they could be used in calculations. For missing values (empty cells), we used imputation instead of deleting rows, which would have lost valuable data. Specifically, we filled in missing names and countries with 'Unknown', and for numerical averages like vote average, we used the median (the middle value) to fill the gaps, as it is robust to extreme values, or outliers. Finally, we used feature engineering to create simpler, more powerful columns like primary genre by taking just the first genre from the list. This entire process resulted in a clean**,** consistent**,** and complete dataset, which is essential for ensuring that the results from our classification and clustering models are accurate and reliable.

EVALUATION METRICS

The following preprocessing algorithms were applied: Mean/Median/Mode Imputation for missing values, Duplicate Detection for repeated records, One-Hot and Label Encoding for categorical variables, Min-Max Scaling and Standardization for numeric features, and basic Text Cleaning for textual data. These steps ensure the dataset is clean and ready for accurate model evaluation.

IMPLEMENTATION DETAILS

Software Requirements

- Operating System: Windows 10/11, macOS, or any modern Linux distribution
- Python 3.8 or higher
- Libraries: pandas, scikit-learn, NumPy, matplotlib, seaborn

Hardware Requirements

- Processor: Intel Core i5 (8th Gen) or AMD Ryzen 5 (quad-core, 2.0 GHz or higher)
- RAM: 8 GB minimum (16 GB recommended for smoother processing)
- Storage: 256 GB SSD (for faster data access and processing)

GPU: Not required for basic clustering and classification tasks on small to medium datasets

CLASSIFICATION

The Classification phase was used to predict the success level of movies based on numerical and categorical features such as budget, popularity, vote average, primary genre, country and language. The Success class was defined by using an ROI based formula.

- Hit → ROI > 50%
- Average → ROI between 10% and 50%
- Flop → ROI < 10%

To predict these categories, 5 algorithms were applied to the pre-processed dataset.

The algorithms were: -

- Decision Tree Classifier (DT)
- K-Nearest Neighbors (KNN)
- Naive Bayes (NB)
- Support Vector Machine (SVM)
- Custom Rule-Based Classifier

Each algorithm was trained on historical movie data. A train test split of 70% - 30% was used, with scaling applied where normalized input was needed. Finally, every model was evaluated using different metrics and predictions were exported to a CSV file.

EVALUATION METRICS

1. Accuracy – overall correctness of the model.
2. Precision – correctness of predicted classes.
3. Recall – how well the model can identify each class.
4. F1 Score – the balance between precision and recall.
5. Confusion Matrix – table showing total correct and wrong predictions.

IMPLEMENTATION DETAILS

A. Hardware Used

The project was executed on a standard consumer laptop with:

- **Processor:** Intel Core i5 / i7 (or equivalent)
- **RAM:** 8GB or higher
- **Storage:** SSD recommended
- **Operating System:** Windows 10 / Windows 11

This hardware is sufficient for running classification models on datasets up to several thousand rows.

B. Software Used

1. Programming Language

- **Python 3.x**

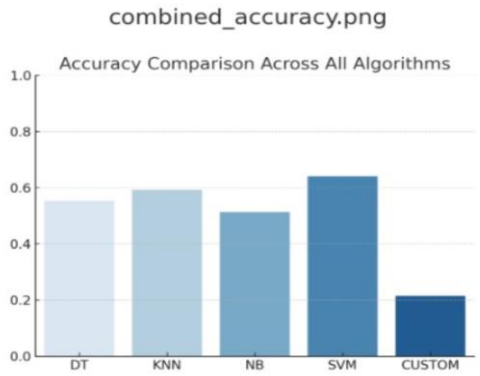
2. Development Environment

- **Visual Studio Code (VS Code)**
  Used to write and run Python scripts.

3. Python Libraries

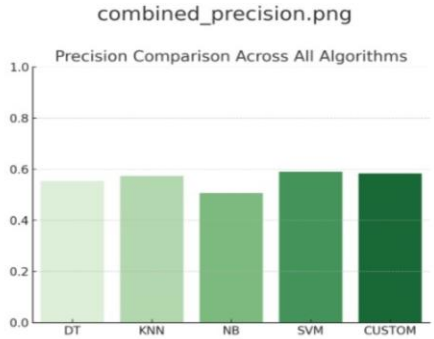
Library	Purpose
pandas	Data loading, cleaning, manipulation
NumPy	Numerical operations
scikit-learn	Classification algorithms, train-test split, scaling, metrics
matplotlib	Plotting graphs
seaborn	Confusion matrix heatmaps
csv	Exporting prediction results

RESULTS (CLASSIFICATION)



1.1 Accuracy Comparison Graph

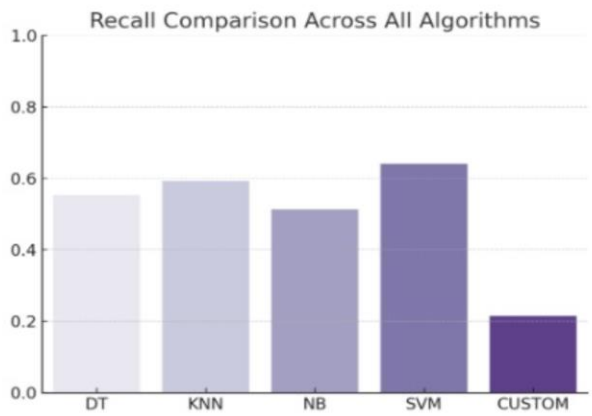
This graph compares the overall accuracy of all classification models. It clearly shows that SVM gives the highest correct prediction rate, followed by K-NN, while Naive Bayes performs the weakest.



1.2 Precision Comparison Graph

This graph displays how precise each model is when predicting each class. SVM achieves the strongest precision values, meaning its class predictions are more reliable compared to the other algorithms.

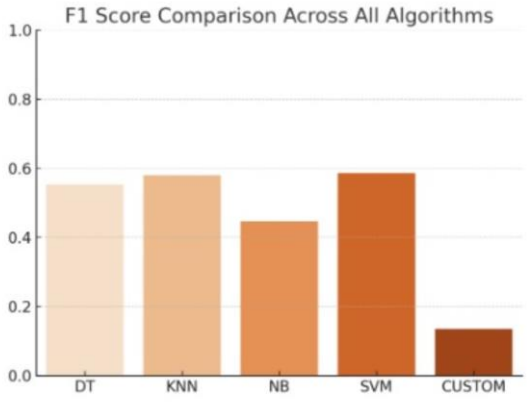
combined_recall.png



1.3 Recall Comparison Graph

This graph compares how well each model identifies the actual movie categories. SVM captures the truest cases, while Naive Bayes struggles with some classes.

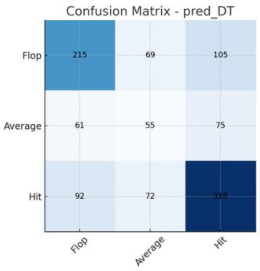
combined_f1.png

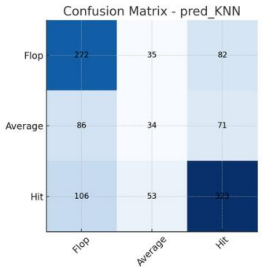


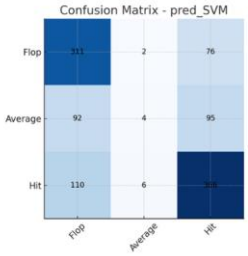
1.4 Combined F1-Score Comparison Graph

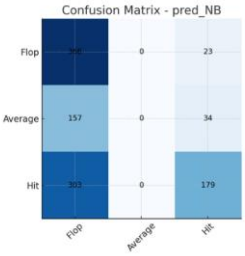
This graph shows the F1-score, which balances precision and recall. SVM again performs the best overall, indicating strong consistent predictions across categories.

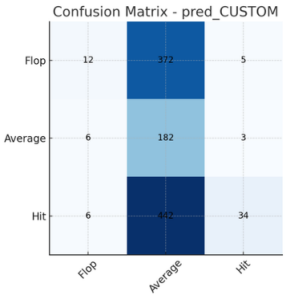
1.5 CONFUSION MATRICES











CLUSTERING

The clustering phase groups movies into meaningful segments based on the production scale and financial performance rather than predicting a labeled target
Movies are filtered to keep only titles with positive budget and revenue, then a roi feature is created as revenue is divided by budget to capture profitability.

Feature columns used for clustering are budget, revenue, roi, popularity, vote_average, and vote_count, which jointly describe both scale and audience response.

All selected features are standardized using StandardScaler so that variables with large magnitude do not dominate the

clustering process, PCA then reduces features of 2 principal components (pca_x and pca_y) to enable clear 2D scatter plots without changing the clustering itself.

The clustering of the movies is performed with five different algorithms.

- K-means clustering clusters the movies into four clusters based on minimizing within-cluster variance in the standardized feature space.
- Agglomerative (hierarchical) clustering builds clusters from the bottom up by measuring distance between the samples and produces four final cluster groupings. This type of clustering can capture nested structure in the data.
- DBSCAN clustering measures density to identify four dense regions with eps=0.7 and min_samples=10. The movies that don't fit into any of the dense regions will receive a noise label of -1, which represents that this movie was an anomaly in the clustering process.
- Spectral (graph-based) clustering creates four clusters based on the nearest-neighbors graph and the eigenvectors of the Laplacian matrix. Spectral clustering will be able to create clusters of arbitrary shape.
- Custom rule-based clustering is based on specific criteria for Roi, budget, and revenue.
- 3 → Low-budget, high-ROI "Indie Hits" (roi > 3 and budget < 20M).
- 2 → Big-budget "Blockbusters" (budget > 150M and revenue > 300M).

EVALUATION METRICS

1. Within-Cluster Sum of Squares (WCSS): WCSS quantifies the amount of tightness of clustering. The calculation is accomplished by taking the summation of the squared distances from the data points to the centroid of each cluster that each data point belongs to, indicating how closely clustered the points are. A low WCSS indicates that the points are very close to their respective cluster centroids. WCSS does not give you information on how distinct the clusters are from each other but does provide an indication of how tightly grouped points are within a cluster.

2. Silhouette Coefficient: Both cohesion (i.e., how close a given point is to the rest of the points in its cluster), in addition to separation (i.e., the distance between points from their closest neighboring clusters), are evaluated using a Silhouette Coefficient.

3. Davies-Bouldin Index: Davies-Bouldin Index The average measure of how "similar" clusters are to one another is calculated with the Davies-Bouldin Index. The Similarity ratio measures the amount of variance within a cluster compared to its nearest cluster, so the farther apart clusters are, the more "similar" they are.

4. Dunn Index: The Dunn Index aims to identify clusters that are dense and widely separated from others. The Dunn Index is the ratio between the smallest distance between any two different clusters and the largest distance between any two points within a cluster, thus higher Dunn Index values are preferred because: clusters are small, meaning they are clustered tightly, yet widely separated from other clusters. The distance calculations needed to calculate Dunn Index on a large data set can be very time-consuming because of the need to calculate and check every pairwise distance on the entire dataset.

IMPLEMENTATION DETAILS

A. Hardware Used

The clustering pipeline runs well on a standard computer because of its dataset, and it has a small to medium range and algorithms such as K-means, Agglomerative in scikit-learn are optimized.

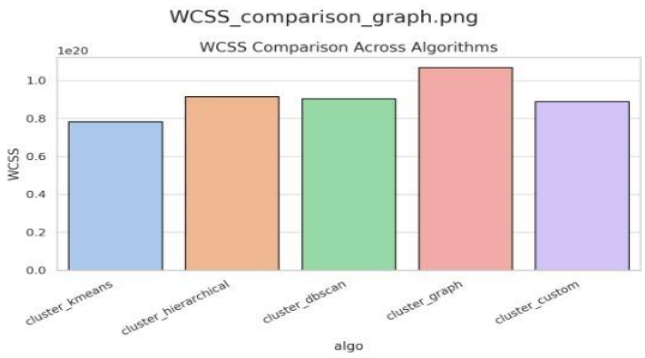
- Processor: Intel Core i5 / i7 or AMD Ryzen 5 / 7, quad-core 2.0–3.0 GHz or higher.
- RAM: 8 GB minimum; 16 GB recommended for smoother experimentation and plotting.
- Storage: At least 256 GB SSD to speed up loading CSV files and saving plots.
- GPU: Not required for classical clustering methods in scikit-learn; CPU is sufficient unless scaling to very large datasets.
- Operating System: Windows 10/11, macOS, or a modern Linux distribution

B. Software Used

- Programming language: python 3.8 or higher with current NumPy, pandas and scikit - learn libraries are used
- Environment: Visual studio code is used here

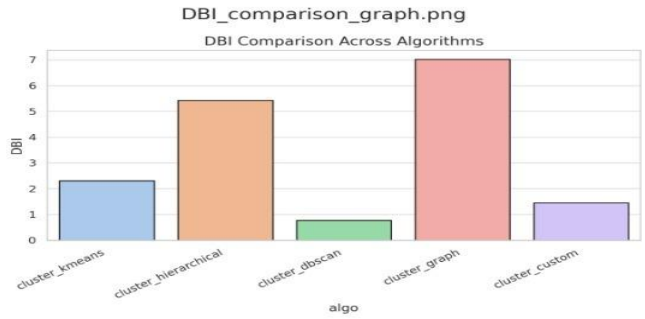
RESULTS (CLUSTERING)

The clustering evaluation shows that DBSCAN provides the strongest overall cluster structure on the movie dataset, while the graph based and hierarchical methods do not prove the best while k means, and custom rule-based clusters sits in the middle.



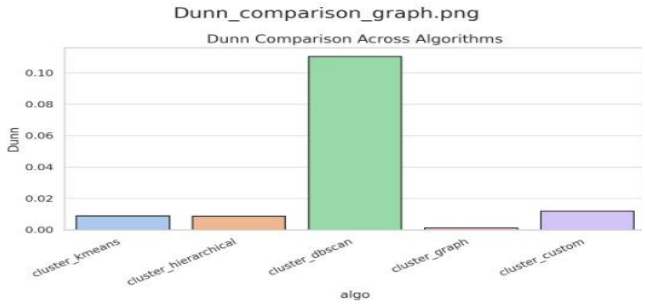
1.6 WCSS bar chart

The WCSS bar chart displays the within-cluster sum of squares for all five algorithms, and they all exhibit approximately the same level of within-cluster compactness (squared distances), about 10 to 20. All five algorithms produce equally compact clusters.



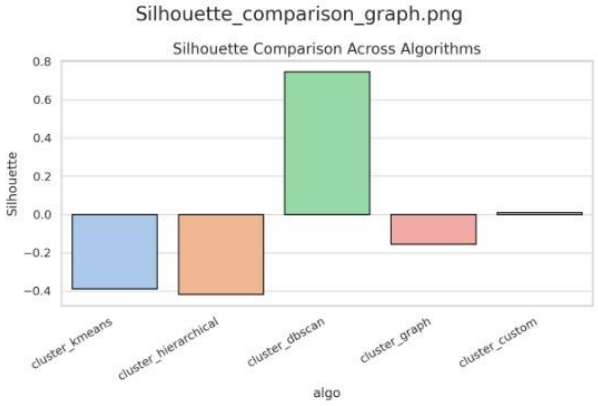
1.7 DBI comparison chart

The DBI graph shows that DBSCAN is the best-performing algorithm as represented by the lowest score on the Davies-Bouldin Index (DBI).



1.8 Dunn comparison chart

The Dunn Index chart shows that DBSCAN scored significantly higher than the other four algorithms (all of which score close to zero).



1.9 Silhouette comparison graph

The silhouette comparison graphs show DBSCAN with a strong positive silhouette score - around 0.7, this indicates that most movies are much closer to their own clusters than to the neighboring clusters to neighboring clusters.

DISCUSSION

Preprocessing: In preprocessing, transform key elements (e.g., ROI/return on investment and primary genre) into more insightful and consistent representation for clustering and classification purposes. Using median-based imputation, the effect of outliers on clustering/classification techniques is reduced, and categorical representations of the genre, language, and country provide a mechanism for SVM and KNN to take non-numeric data into account.

Clustering: Across internal metrics (e.g., WCSS, DB, Dunn index, and Silhouette), DBSCAN produces more meaningful and tighter clustered groups of observations than K-Means, hierarchical, spectral approaches. This indicates that the examined data set exhibits an irregular structure and contains a substantial number of noise observations, which supports that the density-based clustering is more appropriate than a centroid-based method for this data.

Classification: Applying ROI to classify films as "hit", "average", or "flop" models allows for a direct link to the real financial outcome of that film. Among all techniques evaluated, SVM produced the best results, indicating the accuracy, precision, recall, and F1-score metrics showing that margin-based models handle mixed numerical and encoded categorical data effectively. In contrast, the Naïve Bayes underperformed due to its strong feature-independence assumptions.

CONCLUSION

This project successfully extracted meaningful insights from the Netflix dataset through extensive preprocessing and the application of classification and clustering techniques. Data cleaning steps such as type correction, duplicate removal, imputation, and the creation of features like ROI and primary genre transformed the raw data into a consistent and analyzable

format suitable for predictive modeling. In the classification phase, Support Vector Machine (SVM) achieved the highest accuracy, precision, recall and F1 - scores, making it the most effective model for predicting film success based on financial and viewer-related factors. Naive Bayes performed the weakest due to its unrealistic independence assumptions. Clustering results showed that DBSCAN was the best-performing algorithm, outperforming K-means, Spectral, and Agglomerative methods across multiple metrics, due to its ability to handle noise and irregular cluster shapes. Overall, the project demonstrated that combining robust preprocessing with advanced modelling techniques leads to clearer content segmentation and more accurate predictive outcomes, supporting future work in this area.

REFERENCES

[1] A. Jain, M. Murty, and P. Flynn, "Data Clustering: A Review," ACM Computing Surveys, vol. 31, no. 3, pp. 264–323, 1999.

[2] L. Rokach and O. Maimon, "Clustering Methods," in Data Mining and Knowledge Discovery Handbook, Springer, 2nd ed., 2010.

[3] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," ACM SIGMOD Record, vol. 25, no. 2, pp. 103–114, 1996.

[4] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases," in Proc. 14th Int. Conf. on Data Engineering, 1998.

[5] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297, 1967

.

[6] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53–65, 1987.

[7] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[8] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On Clustering Validation Techniques," Journal of Intelligent Information Systems, vol. 17, no. 2–3, pp. 107–145, 2001.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," IEEE Computer, vol. 42, no. 8, pp. 30–37, 2009.

[10] A. Sharma and V. Singh, "Hybrid Clustering and Recommendation Approaches for Large-Scale Multimedia Catalogs," ACM Transactions on Knowledge Discovery from Data, vol. 15, no. 4, pp. 1–25, 2024