CS584 Machine Learning Report

Anwesha Nayak
(anayak8@hawk.iit.edu)
Illinois institute of technology
(A20512145)

Rishabh Dharmendra Darji
(rdarji@hawk.iit.edu)
Illinois institute of technology
(A20561536)

Taufeeq Ahmed Mohamaad
(tmohammed1@hawk.iit.edu)
Illinois institute of technology
(A20512082)

Abstract— In the present report, we conducted a thorough analysis of a Netflix data set with the aim of knowing the viewer's preferences and developing better recommendation algorithms. The initial step in our approach involved data cleaning and preprocessing procedures which then were followed by EDA in order to investigate and visualize data relations. We used machine learning algorithms like k-nearest neighbors, Gaussian naive Bayes, and XGB Classifier to sort the content as TV Shows or Movies, and the dataset is evaluated using measures such as precision, recall, F-score, and accuracy using the confusion matrix. Furthermore, we implemented a recommendation system with cosine similarity and TF-IDF vectorization that was compared with Sklearn's models to ensure quality. This extensive study not only delves into the user behavior of Netflix but also increases opportunities for users to have personalized experiences and provides stakeholders in content creation and operation of the platform with actionable intelligence.

Keywords— VGG-19, XGBoost, Image Analysis, Tabular Data, Machine Learning, Deep Learning, Netflix dataset analysis, content distribution patterns, TF-IDF vectorization, cosine similarity, geographical preferences, recommendation system, machine learning algorithms, strategic decisions, user engagement, personalized content suggestions, F-score, content creation, accuracy metrics

I. INTRODUCTION

Netflix goes to the same league as a primary actor in media circulation and video streaming This company possesses a high-tech ecosystem with more than 8000 films and TV shows available on their platform. This is reflected in their total subscribers of 200 million which reflects a global scale. A comprehensive listing with the data file presenting and giving vital details about all offerings on Netflix – cast, directors, ratings, release year, length and more.

Dataset Overview:

The Netflix Titles dataset, which is a complete collection of movies and series on Netflix and covers different properties, such as type of titles, directors, cast, country of origin, year of release, rating, size, and the genres and a short description. With the help of this database, we are able to perform all the necessary Netflix content coverage, genres popularity and distribution of shows within different time zones and the regions.

Key Details:

Total Entries: The information comprises 8809 entries, each containing an individual movie or a TV show.

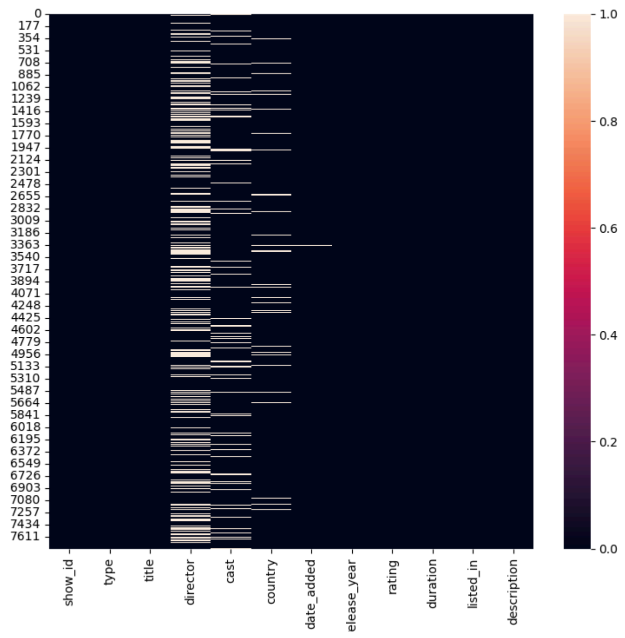
Columns: The observations of the dataset are 12 columns as presented below:

1. show_id: A unique identifier with a number for every title.

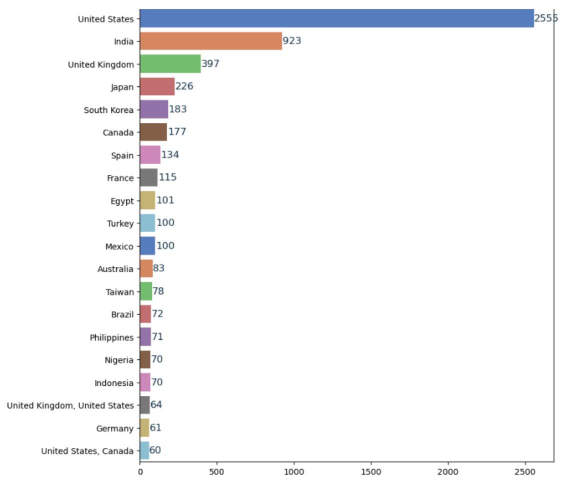
2. type: It is the genre of the title as it can be either 'movie' or 'TV show'.
3. title: The first thing is the name of the movie or TV series.
4. director: The director (or directors) of the movie (or TV shows). (Does not typify all types of entries; particularly those of TV shows where similar rating might not be appropriate.)
5. cast: The actors/actresses listed in the title underlines the main characters. (e.g. Some entries may miss out on this detail.)
6. country: The country or countries that the movie or show is produced in.
7. date_added: Release dates of the title by Netflix.
8. release_year: One of the most important facts about a motion picture is its release year.
9. rating: Classification into age groups of the title.
10. duration: The length of the title in length of minutes for movies and seasons of television shows.
11. listed_in: It is composed of the genres the title is a part of.
12. description: A brief summary of the title.

II. EXPLORATORY DATA ANALYSIS

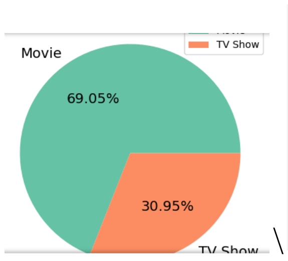
In this part, we discuss the results from the program, covering all aspects like our visualization, classification, recommendation systems, and other analyses.

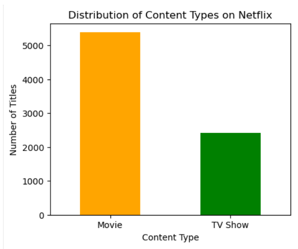


This code is for an algorithm that involves dual action for the handling of missing entries in a DataFrame. It begins by creating a heatmap with seaborn which determines the missing values and indicates areas of missing data by unique separate colors. This is oursided by a statistical analysis that determines the per column % of missing data, which is the exact measurment of the incompleteness of data. Together, these techniques provide an integrative approach to check the dataset integrity which provide necessary wisdoms in the outcomes of the data preprocessing phase, and the foundation of the analysis work is solid.

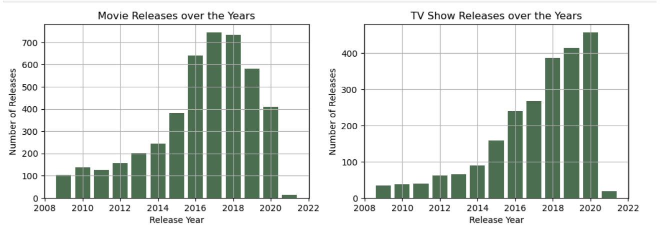


The bar chart is used to show a quantitative comparison among the top 20 entities which, likely, are either countries or combined-country groupings. This visual shows the clearly important role played by the US, the next most prominent country in terms of counts being India, followed by the overall stable decrease in counts among the remaining countries. The graph shows the country data points which have been merged. Together, these data points could accentuate shared metrics or common occurrences. The numerical annotations made directly on the bars of the graph strengthen the data readability and from visual counting it is also obvious which ones are in stark contrast and which ones are collaborative.

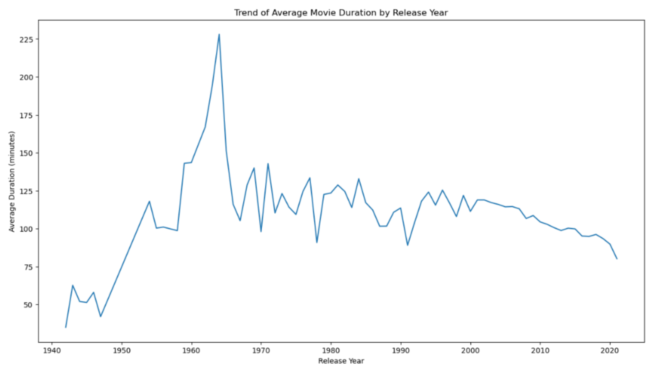




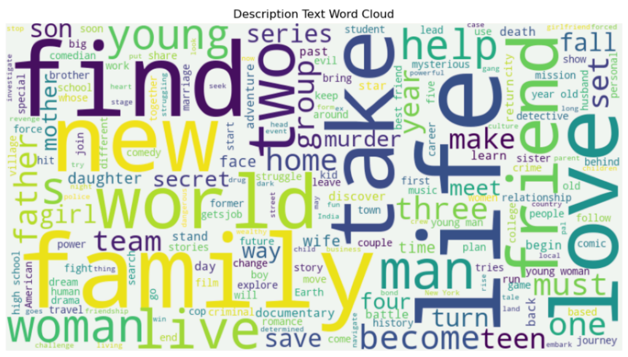
The two graphs reveal a visual calculation of a data table with content most likely representing Netflix titles marked as 'Movies' and 'TV Shows'. The first image is a pie chart plotted with the matplotlib library shows that the group of Movies (69.05%) has bigger data compared to TV Shows (30.95%). Inside the pie chart is the code that shows the use of `value_coulds()` to find out the distribution of content types and matplotlib settings that are responsible for configuring the appearance of the chart. The second picture has a bar chart with the same data set so that it is obvious that TV Shows (2410) have less titles than Movies (5377) which are presented side by side ratifying that there are more movies. Visual presentations, with and without the code, both demonstrate influence of movies over the TV shows in this specific set of Netflix material and also serve as the sample on data preparation and plotting methods to make those two plots.



The content release shown in the bar graphs over the years covers movies and TV shows from 2008 to 2022; with one looking at movies and the other at TV shows. Looking at the release of TV shows over time, a tendency of their numbers to go up is visible, so that it can be concluded that the attention is drawn to serials. Films appears later, on the contrary, their peak was in 2018, and there was a trend of descent after that. These numbers show that Netflix is moving towards TV series according to the rising trend of the consumerist market to be engaged with series. This customer interest for binge-watching and consuming narrative episodes accommodates Netflix well.



The decline trend of the 1940 to 2020 mean movie lengths analysis shows a remarkable slowdown down of about 20 minutes on average. Timewise, the format was quite short. The notable shift to a more lengthy film can be seen at the beginning of the 1960s, which is a possible indicator of longer or epic storylines being the cinematic trend of the era. Towards the end, the role of the genre has been shortened gradually, however, it's recent years' features at the top. Thus, it shows either a follow suit in the rapidly changing process of consumption for the audience or a reconsideration of economics of production for the films, that are inclined to be shorter nowadays.



The figure presents a word cloud of words taken from a text dataset that mainly include the words like "new," "family," "love," "help," "secret," and "woman." The font size of words correlates to their frequency, revealing that the aforementioned themes are frequently found. This visualization method is at the same time very quick and intuitive providing general information on the main subject of the narrative, focusing on personal and relational tales most likely within the genres of drama, mystery, and everyday life. The cloud of words is an analytic tool which will tell you which toppics or feelings are mentioned most often in the text.

III. DATA PREPROCESSING

The dataset consists of various attributes related to Netflix titles, including type, rating, country, release year, director, cast, and more. The dataset is loaded into a DataFrame from a CSV file named netflix_titles.csv.

1. Initially, df.info() and df.describe() functions are executed to check the DataFrame's structure to generate summary statistics for numerical columns.
2. To handle missing values, the director, country and cast columns are filled with 'Unknown' where data is missing.
3. The mode of the date_added column is used to fill its missing values.
4. The type column is converted into binary format, where 'TV Show' is mapped to 1 and otherwise to 0.
5. Ratings are mapped to integer codes to transform the rating column into numerical format.
6. The date_added column is converted into datetime format and a new column year_added is created to store only the year.
7. The duration column is split and converted to integers representing the duration in minutes or seasons.
8. Map_genre is defined to map genre of the movie to s integer codes.
9. A dictionary maps countries to continents, which is used to transform the country column into a new continent column indicating the continent code.
10. Columns such as title, date_added, country, cast, director, listed_in, and description are dropped from the dataset to focus on relevant features.

IV. MODELING TECHNIQUES

1. K Nearest Neighbors Algorithm: The objective of this implementation is to predict whether a Netflix title is a 'TV Show' or 'Movie'. The dataset features used for prediction include 'continent', 'year_added', 'release_year', 'genre', 'duration', and 'rating'.

1. New features such as num_cast (number of cast members) and num_genres (number of genres) are created by splitting the respective string entries and counting the elements.
2. The duration field is transformed into minutes using a custom function that handles different formats (e.g., minutes and seasons).
3. Missing values in numerical fields are filled with the mean of their respective columns.
4. The type column is converted into a binary format, where movies are represented as 1.
5. The dataset is split into training and testing sets with a ratio of 80:20.
6. The Euclidean distance between instances is calculated to measure the closeness between data points.
7. For each instance in the testing set, the k closest neighbors from the training set are identified based on the calculated Euclidean distances.
8. The most frequent type among the nearest neighbors is chosen as the prediction using a voting mechanism.
9. The number of neighbors (k) is set to 10, balancing between overfitting and underfitting.

2. Gaussian Naive Bayes classifier: The aim is to implement a Gaussian Naive Bayes (GNB) classifier to predict if Netflix titles are 'TV Shows' or 'Movies' based on selected features from the dataset.

1. Created new features like num_cast (the count of cast members), num_genres (the count of genres), and duration_min (conversion of duration into minutes).
2. Converted the type field to binary format where movies are represented as 1.
3. Removed columns that are less likely to be useful for the model.
4. For each class in the target variable, the prior probability, mean, and standard deviation are computed for each selected feature.
5. For each instance in the test set, the likelihood of each feature is computed.
6. These feature likelihoods are combined with the class prior probabilities to get the total likelihood for each class.
7. For each instance, the class with the highest total likelihood is predicted.

3. XgBoost Classifier: XGBoost classifier is a popular learning method based on boosting decision trees. The objective of this implementation is to predict whether a Netflix title is a 'TV Show' or 'Movie'.

1. Each node of the decision tree selects a feature and a threshold for splitting the data.
2. Fit method in the node class determines the best feature and threshold to split the data based on minimizing a loss function.
3. Predict method makes predictions for given input features based on the thresholds and features selected during the fitting process.
4. The XGBoost class has parameters for the number of estimators (n_estimators) and the learning rate.

5. Model Fitting: Fit the model using a loop over the number of estimators. In each iteration:
6. Compute residuals from previous predictions as targets for the next tree.
7. Calculate gradients and Hessians based on current predictions.
8. Train a new decision tree on these values to predict the residuals.
9. Update the model's predictions by adding the weighted predictions of the new tree.
10. Implement a method to predict the probabilities of the positive class for test data by combining the predictions from all trees and applying the sigmoid function.
11. Convert the probabilities into class labels using a threshold of 0.5.
12. Use the training data to fit the XGBoost model.
13. Apply the trained model to the test data to obtain class predictions.

V. RECOMMENDATION SYSTEM

The importance of a word in a document is measured statistically by TF-IDF (Term Frequency-Inverse Document Frequency). The movie recommendation system is implemented by loading movie data and computing a TF-IDF matrix that quantifies the importance of each word within the movie descriptions relative to their frequency across all movies. Then the cosine similarity between movies is calculated to estimate their textual similarity. Recommendations are generated by comparing the user-input movie to others in the dataset, identifying and listing the most similar movies based on these similarity scores.

VI. MODEL EVALUATION

For evaluating different models, it's crucial to select metrics that reflect the objectives and constraints of the problem . We have chosen the following metrics for evaluating the KNN, Gaussian Naive Bayes, and XGBoost models based on their performance outputs:
1.Accuracy: Accuracy measures the overall effectiveness of a model in classifying instances correctly. It is useful when classes are balanced, or when false positives and false negatives have similar costs.
2. Precision: It measures the accuracy of positive predictions. It is crucial when the cost of a false positive is high.
3. Recall: Measures the model’s ability to detect all actual positives. It is critical when the cost of missing a positive (false negative) is high.
4. F1-Score: The F1-Score is the harmonic mean of precision and recall. It's used when it's important to balance precision and recall, which might be the case if both false positives and false negatives are costly.

VII. TRADE OFFS

1. K-Nearest Neighbors (KNN): KNN is simple and effective in classification problems. From our observations, KNN has the highest accuracy and F1-score among the models. KNN can be computationally expensive and sensitive to the scale of the data and the choice of k. High accuracy suggests that the model was effective.

2. Gaussian Naive Bayes: Naive Bayes is fast and effective, especially for large datasets. It works well with categorical input features. We can see that it has lower accuracy compared to KNN but high recall. It assumes feature independence and normally distributed data, which might not always be true. Also, the high recall indicates it is good at identifying positive cases but at the cost of many false positives.

3. XGBoost: XGBoost is a powerful ensemble technique that uses gradient boosting to build predictive models. It is known for its performance and flexibility. It has a similar F1-score to Naive Bayes but slightly lower accuracy and precision. It is prone to overfitting if not properly tuned and can be more complex to implement and tune compared to simpler models.

VI. CONCLUSION

Each model has its strengths and trade-offs which makes them suitable for different scenarios. KNN has high precision and recall which makes it very reliable. Gaussian Naive Bayes is simple and fast but sacrifices precision for recall due to which it is suitable for scenarios where missing a positive is more problematic than handling extra false positives. XGBoost has the highest recall but lower precision which might be best when it is critical to identify all positives.