Netflix Movies and TV Show

1. Data Exploration

Data Exploration

1. Are the variables in the dataset related? If yes what does it mean?

The (co)relation can be checked within dependent variables and between dependent and independent variables.

within independent variables: This relation helps in understanding the distribution of the data.

- From 3rd and 4th chart we can observe similar pattern of duration distribution of Movies and Genre movies where newly releases movies takes less years to get added on platform as this is dependent on the release and popularity of netflix.

- From chi square test we can observe that there is a significant relation between director and country, director and genre, cast and director.

- There is a relation between type and the duration range. The range of Movies duration is high indicating the units ‘Mins’ and the range of TV Show is lower indication the units season.

between dependent and independent variables: This relation helps in predicting the dependent variable.

- The 1st and second chart helps in clearly dividing the classes based on the type and then further by duration. This suggests that we can create two different models for each of the type.

- From chi square test we can observe that there is a significant relation between (director and genre)and (cast and genre).

- For example, the actor Anupam Kher works in these genres: Dramas, Comedies, Action, Children, Crime TV, International Movies

1. Do the features in this dataset contain outliers?

- The movies that are represented by only one country can be considered as outlier but cannot be removed, as test data can contain this scenario.
- The derived feature, release_diff (lag between movies release and add) contains some values outside IQR which are considered to be outlier.
- Based on movie duration, a lot of movies can be removed, however this is only due to the difference of units for movies and TV shows and hence won't be treated as outliers.
- Movies with uncategorised genre can be considered outliers, as they can clearly be classified with more description. Therefore, they are considered outliers and will either be removed or considered after adding more information.

2. Who is the actor who has worked in the cast with the most actors?

Answer: Anupam Kher with 33 movies.

3. Which actor has worked in more films?

Answer: Anupam Kher

4. How many films has each director made, and what is the genre of each film he has made?

Answer: Group by 'director' and aggregate to count films and list genres.

5. In which year have the most films been made in the U.S.? And in which year in Brazil?

Answer:

Year with most films in the U.S.: 2017

Year with most films in Brazil: 2020

6. Taking into consideration the duration of each film, which director shot the most minutes?

Answer: The director who shot the most minutes is Martin Scorsese with 1579 minutes

8. Which actor shot the most minutes in total?

Answer: Anupam Kher

2. Unsupervised Learning

We will performed this task in 3 steps.

1. Embedding Generation:

- Embeddings for the description can be created with Text vectorization, word or sentence embedding methods. We selected a pretrained lightweight transformer based sentence embedding model for its ability to create context-aware sentence embeddings, which are crucial for distinguishing genres with overlapping vocabulary.
- The Title and description of the movies are combined into ‘title_desc’ to enhance the accuracy of the clustering. The desc are then passed to a tokenizer and then model to generate embeddings.
- Embeddings are evaluated manually by checking the description of a few show_ids.

more model details: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

2. Clustering:

- Clustering of embeddings can be done with clustering and topic modelling algorithms, but since our scope is to cluster only descriptions based on genres we will focus on these clustering algorithms. {kmeans, DBScan}
- We will used the embeddings as features for these algorithms and silhouette score for an evaluation of the models performance. After clustering, we see the purity score for each cluster to evaluate the clustering based on genres.
- The score (0-1) corresponds to the homogeneity of genres within the cluster.

3. Visualisation:

- Since cluster features are in high dimensions, we will compress them to 3 dimensions for visualisation.
- The dimensionality reduction is performed using PCA.

Recommended Improvements:

- Plot Elbow plot containing WCSS, and Silhouette plot containing Silhouette scores to find the optimal number of clusters.
- Compare the results of clusters with full embeddings and reduced embeddings.
- Use a scree plot to determine the ideal number of components wrt to variance explanation

OBSERVATIONS:

- Sampling movies with similar characteristics based on their embeddings produced promising results.
- We opted for the K-Means algorithm over DBSCAN because DBSCAN identified more than 15% of the records as outliers while K-Means, being a centroid-based method, worked better for our data, producing more meaningful clusters with fewer outliers.
- K-Means yielded relatively sub optimal purity scores (0.3-0.4) for the clusters, indicating that the clustering was reasonably accurate in grouping movies by genre. However, there is room for improvement to lower this score with further tuning of the algorithm and preprocessing of the data (such as cleaning text or adding more description).

3. Supervised Learning

Data preparation

- The length of the title_desc is observed to have a near identical normally distributed length. Using this distribution we decide the max number of tokens to be 50.
- Uncategorized movies (~45) are considered as outlier and removed.
- The training data is divided into 2 sets. movies_df (Movie Type) and tvshows_df (TV Show) as each of them has unique classes.
- The durations are normalised for each of the df.
- Reduced sentence embedding (with 20 components) is used as a for each of the set.
- The training data for each of set in divided into train and validation set using stratified sampling

Recommended Improvements:

- Top Actors: Since there is a significant relation between actors, we can add the top actors as feature is that also creates a significant proportion of the test set.
- Top Directors: Similarly, the top directors can be added.
- Top Countries : All 75 countries in the test set matches with the training data.

Model Building:

- We used spatial models like Logistic Regression, KNN, and SVC, as well as tree-based models like XGBoost, due to the nature of our dataset, which contains text descriptions that are vectorized into spatial representations. For example, terms like "comedy" and "comedies" or "anime" and "manga" are semantically similar and close to each other in the vector space. Spatial models like KNN and SVC can capture these relationships by measuring the distances between words and classifying based on their proximity. On the other hand, tree-based models like XGBoost are better at handling complex, non-linear relationships between features, such as interactions between multiple attributes like "director," "cast," and "genre."

- Logistic Regression

TRAINING:

LogisticRegression(max_iter=500, random_state=42,
class_weight='balanced',penalty='l1',solver='liblinear',)

- The use of class_weight='balanced' helps handle class imbalances by adjusting weights.
- Using an L1 penalty helps in feature selection, as it tends to drive less significant features to zero.
- The solver determines the type of algorithm to be used for fitting.

RESULT:

- The prediction accuracy of the model is 48% and the f1 score is 51%. The training f1 score is 51%. This indicates the model has low bias and is underfitting the data. This suggests that the model is not effectively capturing the complex relationships

Recommended Improvements:

- Asses the metric against different classification threshold.
- Use the OLS model for understanding model.

- KNN

TRAINING:

KNeighborsClassifier(n_neighbors=12,weights='distance',p=1, metric='cosine'),

- The use of metric='cosine' helps in selecting neighbors with similar direction.
- The weights='distance' giving more importance to closer neighbors.

RESULT:

- The prediction accuracy of the model is 48% and the f1 score is 51%. The training f1 score is 100%.
- These results suggest that the model is overfitting the training data and struggles to generalize to unseen data.

Recommended Improvements:

- Regularise KNN with hyper parameter tuning.
- Update the embedding models as the performance of KNN heavily depends on the quality of the features.
- Asses the data points of the classes that had lowest f1 score.

- XGBOOST

TRAINING:

XGBClassifier(objective='multi:softmax',eval_metric='merror',class_weght='balanced',scale_pos_weight=10, random_state=42, max_depth=5, gamma=1e-3, min_child_weight=2)

- max_depth=5, gamma=1e-3, min_child_weight=2: These parameters control the complexity of the trees. The model is set with relatively shallow trees, which prevents overfitting.

RESULT:

- The prediction accuracy of the model is 54% and the f1 score is 52%. The training f1 score is 100%.
- These results again suggest that the model is overfitting the training data and struggles to generalize to unseen data.

Recommended Improvements:

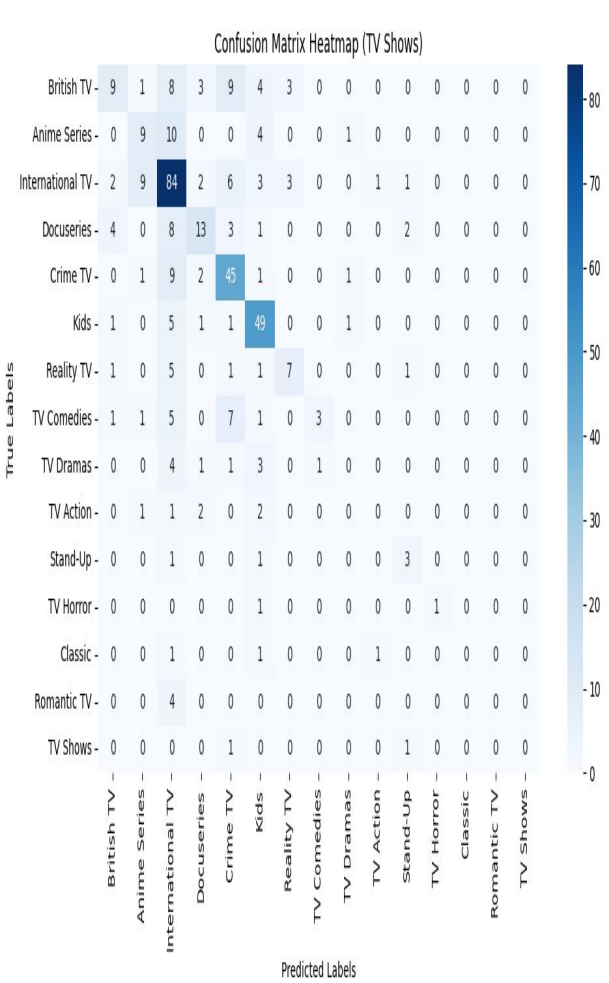
- Regularising the model.
- Asses the data points of the classes that had lowest f1 score.
- Add more features that explains the pattern of these low performing models.

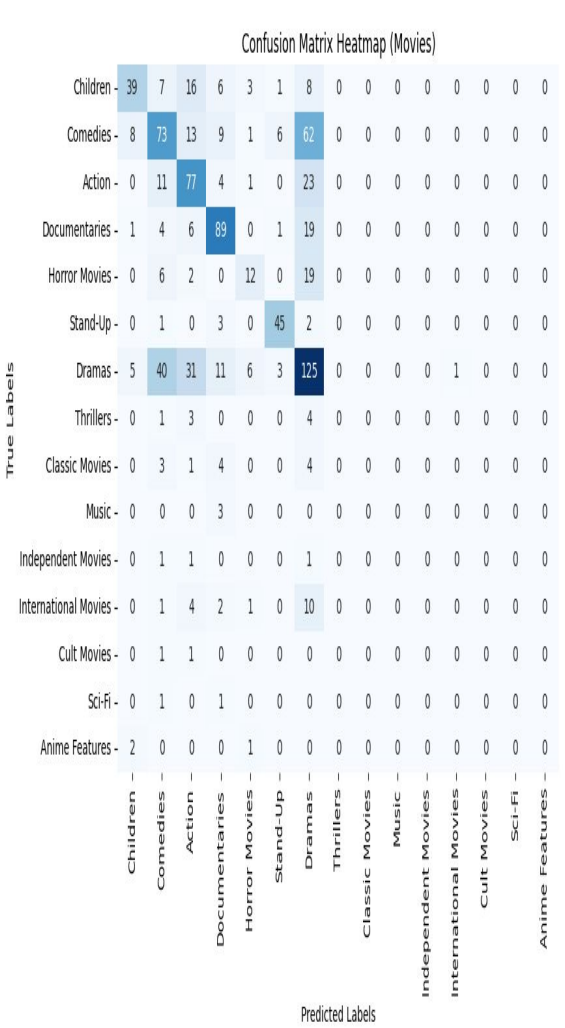
Model Selection

* Xgboost model was selected based on its prediction accuracy on the test

* However, the confusion matrix explains that features with low frequency such as Sci-Fi and Anime Features could not be learned. Suggesting that we should check the data within train and test set of such features since the training accuracy was 100%.

* Model Selection should be done with cross validation.





Data Preparation

- The length of the title_desc is observed to have a near identical normally distributed length. Using this distribution we decide the max number of tokens to be 50.
- Uncategorized movies (~45) are considered as outlier and removed.
- The training data is divided into 2 sets. movies_df (Movie Type) and tvshows_df (TV Show) as each of them has unique classes.
- The durations are normalised for each of the df.
- Reduced sentence embedding (with 20 components) is used as a for each of the set.
- The training data for each of set in divided into train and validation set using stratified sampling

Recommended Improvements:

- Top Actors: Since there is a significant relation between actors, we can add the top actors as feature is that also creates a significant proportion of the test set.
- Top Directors: Similarly, the top directors can be added.
- Top Countries : All 75 countries in the test set matches with the training data.

Custom Sequential DL Model

TRAINING:

- The transformer-based model utilizes a Token and Position Embedding layer followed by a Transformer Encoder block to process sequential text data. The Token and Position Embedding layer combines token embeddings (to represent word meanings) with positional embeddings (to account for word order in a sequence). The Transformer Encoder block uses Multi-Head Attention to capture dependencies between words across the sequence and a Feed-Forward Neural Network (FFN) to model complex relationships. Outputs from the encoder are globally aggregated using Global Average Pooling, followed by a Dropout layer to prevent overfitting. Finally, a Dense softmax layer predicts class probabilities for multi-class classification tasks.

RESULT:

- The model has fairly poor performance with 30% accuracy. It could only classify the classes with higher frequency.

Recommended Improvements:

- Use Simple layers or Sequence based layers like LSTM or RNN.
- Train for more epochs.
- Visualise the weight update in tensorboard.
- Balance the dataset.
- Add more layers before the classification layer.

Transformer Based architecture

TRAINING:

- The transformer-based model utilizes a Token and Position Embedding layer followed by a Transformer Encoder block to process sequential text data. The Token and Position Embedding layer combines token embeddings (to represent word meanings) with positional embeddings (to account for word order in a sequence). The Transformer Encoder block uses Multi-Head Attention to capture dependencies between words across the sequence and a Feed-Forward Neural Network (FFN) to model complex relationships. Outputs from the encoder are globally aggregated using Global Average Pooling, followed by a Dropout layer to prevent overfitting. Finally, a Dense softmax layer predicts class probabilities for multi-class classification tasks.
- The model is trained for 50 epochs with categorical_focal_crossentropy loss, Adam optimiser and validation split to monitor the loss and accuracy on un trained dataset.
- The model is weights till the last softmax layer is then frozen, and trained on tv_shows dataset.

RESULT:

- The training and validation loss can be seen decreasing significantly at each epoch which is good.
- The encoder based model performed better than the custom DL model on movies dataset with 48% accuracy, but was able to predict only the classes with more than 40 samples.
- Transfer learning for TV shows did not yield better results than the base model, suggesting that training the model from scratch may be more effective in this scenario
- Fairly poor performance with 30% accuracy. It could only classify the classes with higher frequency.

Recommended Improvements:

- Train for more epochs.
- Visualise the weight update in tensorboard.
- Balance the dataset.
- Utilise pretrained weights.
- Use BertForClassification based architecture.

OBSERVATION

- During the ML model classification we observed the models like XGBoost and k-NN demonstrated a tendency to overfit the training data, effectively identifying patterns but struggling with generalization on the test set, particularly for underrepresented classes with fewer than 20 records.
- The following strategies can employed to improve the model performances.
- Generating synthetic data for under represented class.
- Combining the classes with less frequency into one class and then perform further classification with another model.

4. Statistics for model selection

Model Selection: MCNemar Test

In the previous section, the best-performing models identified were XGBoost for machine learning (ML) and the Transformer-based Encoder model for deep learning (DL). To validate their comparative performance, we conducted the McNemar test, which evaluates paired nominal data to test the hypothesis of equal error rates.

Hypotheses:

- Null Hypothesis (H_0): The error rates of the ML and DL models are equal.
- Alternate Hypothesis (H_1): The error rates of the ML and DL models are different.
- Test Statistics:
  - P-value = 9.37×10^{-26}
  - Significance Level (α): 0.05
  - Decision: Reject H_0

Conclusion: The ML and DL models have statistically different error rates. Based on the classification report, the ML model outperforms the DL model, and the McNemar test confirms that this performance difference is significant.