REPORT *for* PROJECT PHASE - 3

NETFLIX MOVIES AND TV SHOWS RECOMMENDATION SYSTEM

CSE587C: Data Intensive Computing Spring 2024

Instructor: **Dr. Shamshad Parvin** *Deadline*: 03 May 2024

Team members:	*UBIT Name*	*Person Number*
Leela Satya Praneeth Posina	**leelasat**	**50540667**
Surya Suhas Reddy Sathi	**suryasuh**	**50538640**
Sai Venkat Reddy Sheri	**ssheri**	**50545752**

PROBLEM STATEMENT

As Netflix's content library continues to rapidly expand, subscribers face increasing difficulty in discovering new titles that align with their personal interests and tastes. The sheer volume of available movies and TV shows makes manual browsing an inefficient and frustrating process, as users must sift through vast catalogs to identify relevant options amidst nuanced preferences.

To address this challenge, there is a growing need for an accurate and personalized recommendation system that can effectively match users with enjoyment-optimized viewing options based on a specific movie or TV show they have already watched and enjoyed. By leveraging a user's positive experience with a particular title, the system can analyze its characteristics and identify other similar titles the user is likely to appreciate.

However, developing such a robust recommendation engine requires access to proprietary data on user viewing histories, content preferences, and behavioral patterns – information that streaming platforms like Netflix closely guard. This lack of transparency presents barriers to external entities aiming to innovate in this space, underscoring the value of a recommendation system tailored specifically to the Netflix platform and its unique dataset.

MODELS APPLIED AND RESULTS:

The following models are trained:

1. BIRCH
2. DBSCAN
3. K-Means Clustering
4. Mean Shift Clustering
5. Gaussian Mixture Model
6. Agglomerative Clustering

MODEL NAME	**Silhouette Score**	**Calinski-Harabasz Index**	**Davies-Bouldin Index**	**Homogeneity**	**Completeness**	**V-measure**
K-means Clustering	0.42	843.43	0.9534	0.346	1.00	0.514
BIRCH	0.856	8338.35	0.1402	0.6261	1.000	0.770
Mean – Shift Clustering	0.331	121.24	0.588	0.1108	1.0	0.199
Gaussian Mixture Model	0.171	383.23	2.292	0.232	1.000	0.377
Agglomerative Clustering	0.157	393.434	2.07	0.1967	1.0000	0.328
(DBSCAN).	0.700	46.22	1.483	0.5355	1.00	0.6975

Considered Model: Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

WEB UI

USED TECHNOLOGIES : HTML, CSS, JavaScript, Flask

First we have created a joblib file of the BIRCH Model and then used it in the back end code.

BACKEND DESCRIPTION:

1. Imports and Flask App Initialization:

- We import the necessary modules: Flask, render_template, send_from_directory, request, jsonify from Flask, load from joblib, numpy, pickle, StandardScaler from sklearn.preprocessing, cosine_similarity from sklearn.metrics.pairwise, pandas, and csv.

- It initializes a Flask application instance (app).

2. Loading Data and Model:

- The code loads data from a pickle file (data.pickle) and assigns it to the variable data.
- It loads a machine learning model from a joblib file (model.joblib) and assigns it to the variable model.
- It creates an instance of StandardScaler and assigns it to the variable scaler.
- It performs feature scaling on the movie features using scaler.fit_transform(movie_features) and assigns the scaled features to movie_features_scaled.
- It adds a new column 'cluster' to the data DataFrame, which contains the cluster labels from the loaded model.

3. Helper Function: birch_method_movies:

- This function takes a movie name, the data DataFrame, the loaded model, the scaler object, and an optional top_n parameter for the number of similar movies to return.
- It converts the movie name to lowercase.
- It finds the matching movie in the data DataFrame based on the title.
- If no matching movie is found, it returns "No such movie found".
- It extracts the movie features and scales them using the scaler object.
- It predicts the cluster for the given movie using the loaded model.
- It finds all movies in the same cluster as the given movie.
- It calculates the cosine similarity between the given movie's features and the features of other movies in the same cluster.
- It returns the titles of the top_n most similar movies.

4. Routes:

- The @app.route('/') route renders the index.html template.
- The @app.route('/eda') route renders the EDA.html template and passes a list of image paths (images) to be displayed.
- The @app.route('/movies.csv') route serves the movies.csv file from the server's root path.

- The @app.route('/results', methods=('POST', 'GET')) route handles the movie recommendation functionality:
- If the request is a GET request, it retrieves the movie name from the query string (request.args.get('movie')).
- It calls the birch_method_movies function with the provided movie name and other necessary parameters.
- If the function returns a string (indicating an error), it renders the result.html template with the error message.
- If the function returns a list of movie titles, it reads the movie details from the movies.csv file and renders the result.html template with the movie details.
- The @app.route('/image/<int:image_index>') route renders the full_screen.html template and passes an image URL based on the provided image_index.

5. Running the App:

- If the script is run directly (not imported as a module), the Flask app is run with debugging enabled (app.run(debug=True)).

This Flask application serves a web interface for exploring movie recommendations based on a machine learning model. Users can enter a movie name, and the application will recommend similar movies based on the provided model and data. Additionally, it provides routes for rendering static HTML templates and serving image assets.

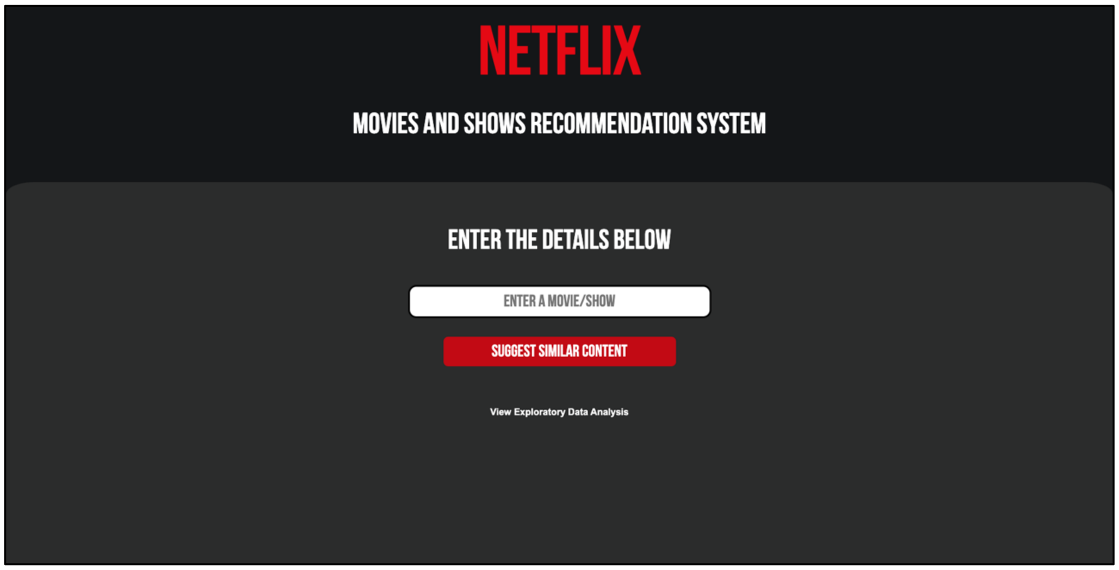
FRONT END DESCRIPTION:

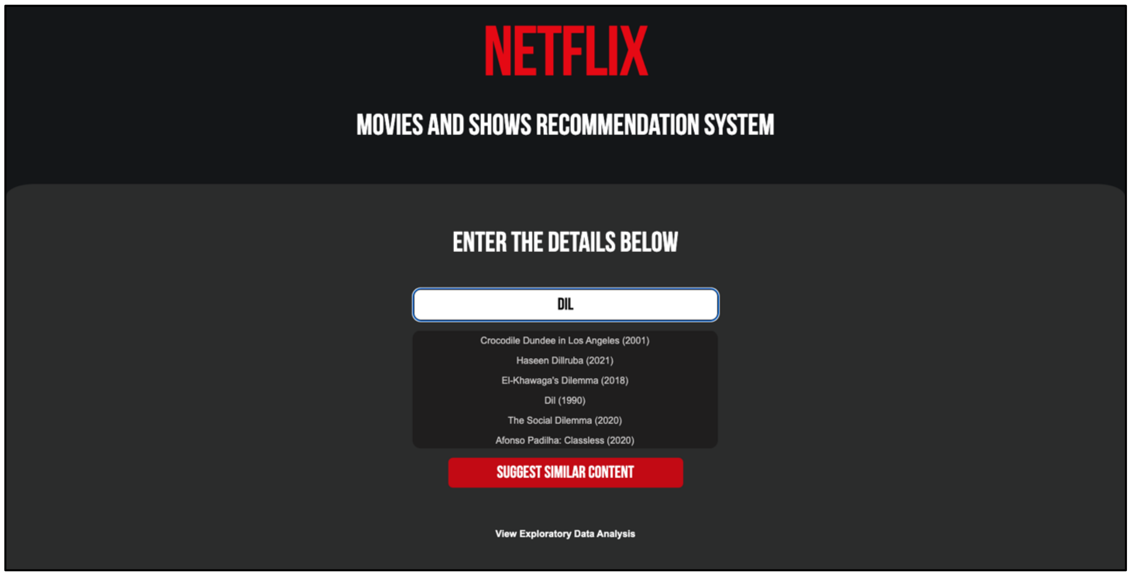
HTML:

1. Index.html

- This page shows the main page for the Netflix movie and show recommendation system.
- In the <body> section there is a container div (<div class="container-1">). Inside the container, there are two major sections:
- A base layer (<div class="base-layer">) with the title and subtitle.
- The second layer (<div class="second-layer">) contains a form (<form action="/results" method="get">) which sends data to the "/results" route using the GET method.

- User can enter a name of any movie or show and can get the related movies/shows.

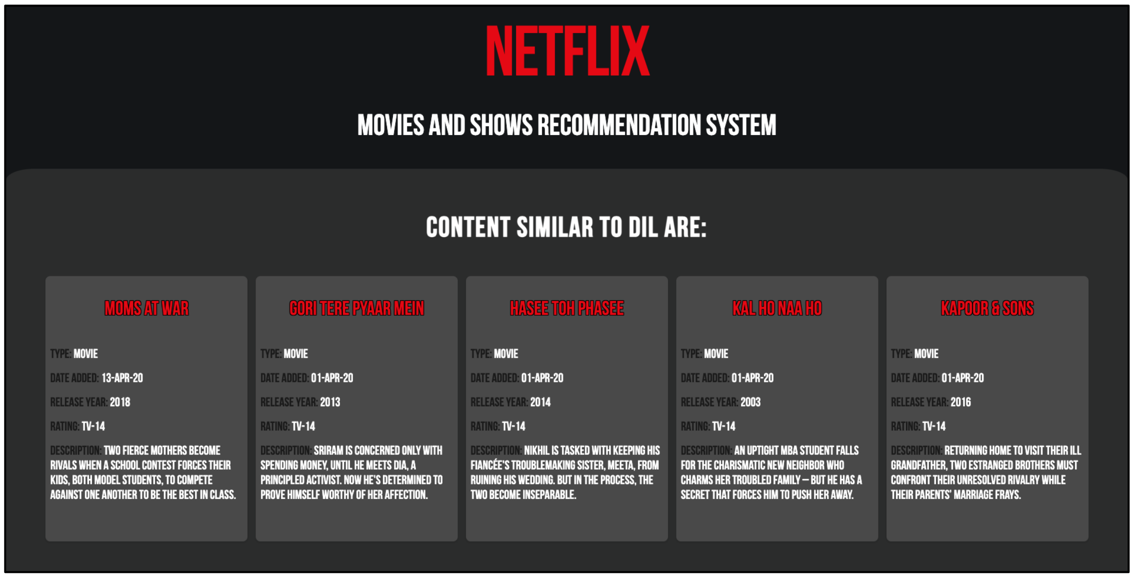



NETFLIX
MOVIES AND SHOWS RECOMMENDATION SYSTEM
ENTER THE DETAILS BELOW
DIL
SUGGEST SIMILAR CONTENT
View Exploratory Data Analysis

Crocodile Dundee in Los Angeles (2001)
Haseen Dillruba (2021)
El-Khawaga's Dilemma (2018)
Dil (1990)
The Social Dilemma (2020)
Afonso Padilha: Classless (2020)

2. Results.html

- This file renders the movie recommendation results page.
- If movie recommendations are available, it renders a grid of movie containers, each containing movie details like title, type, date added, release year, rating, and description.
- If no recommendations are found, it displays an error message.



NETFLIX

MOVIES AND SHOWS RECOMMENDATION SYSTEM

CONTENT SIMILAR TO DIL ARE:

MOMS AT WAR

TYPE: MOVIE

DATE ADDED: 13-APR-20

RELEASE YEAR: 2018

RATING: TV-14

DESCRIPTION: TWO FIERCE MOTHERS BECOME RIVALS WHEN A SCHOOL CONTEST FORCES THEIR KIDS, BOTH MODEL STUDENTS, TO COMPETE AGAINST ONE ANOTHER TO BE THE BEST IN CLASS.

GORI TERE PYAAR MEIN

TYPE: MOVIE

DATE ADDED: 01-APR-20

RELEASE YEAR: 2013

RATING: TV-14

DESCRIPTION: SRIRAM IS CONCERNED ONLY WITH SPENDING MONEY, UNTIL HE MEETS DIA, A PRINCIPLED ACTIVIST. NOW HE'S DETERMINED TO PROVE HIMSELF WORTHY OF HER AFFECTION.

HASEE TOH PHASEE

TYPE: MOVIE

DATE ADDED: 01-APR-20

RELEASE YEAR: 2014

RATING: TV-14

DESCRIPTION: NIKHIL IS TASKED WITH KEEPING HIS FIANCÉE'S TROUBLEMAKING SISTER, MEETA, FROM RUINING HIS WEDDING. BUT IN THE PROCESS, THE TWO BECOME INSEPARABLE.

KAL HO NAA HO

TYPE: MOVIE

DATE ADDED: 01-APR-20

RELEASE YEAR: 2003

RATING: TV-14

DESCRIPTION: AN UPTIGHT MBA STUDENT FALLS FOR THE CHARISMATIC NEW NEIGHBOR WHO CHARMS HER TROUBLED FAMILY – BUT HE HAS A SECRET THAT FORCES HIM TO PUSH HER AWAY.

KAPOOR & SONS

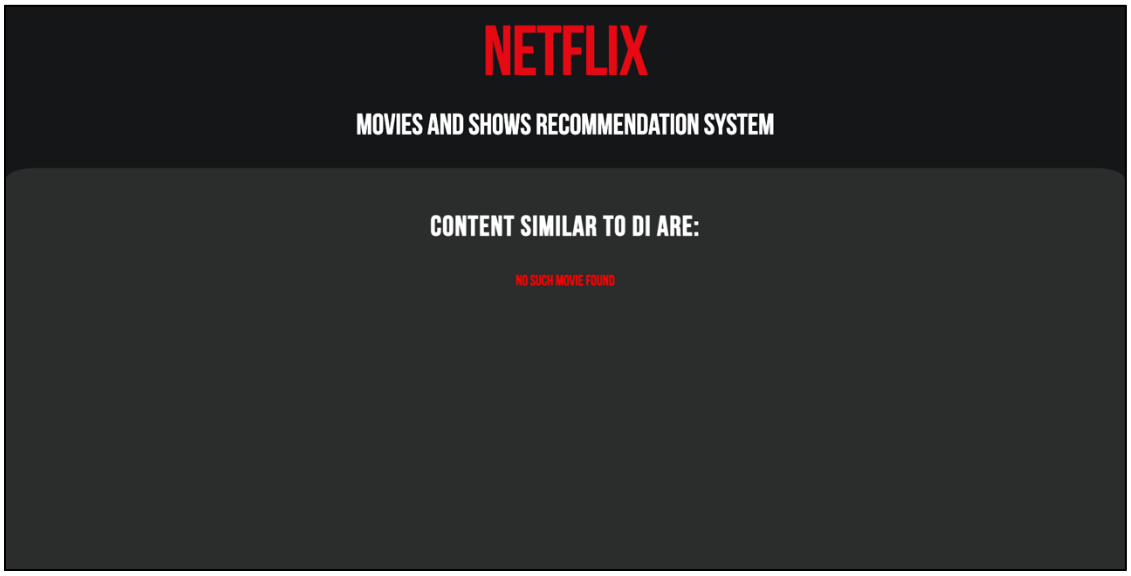
TYPE: MOVIE

DATE ADDED: 01-APR-20

RELEASE YEAR: 2016

RATING: TV-14

DESCRIPTION: RETURNING HOME TO VISIT THEIR ILL GRANDFATHER, TWO ESTRANGED BROTHERS MUST CONFRONT THEIR UNRESOLVED RIVALRY WHILE THEIR PARENTS' MARRIAGE FRAYS.



NETFLIX

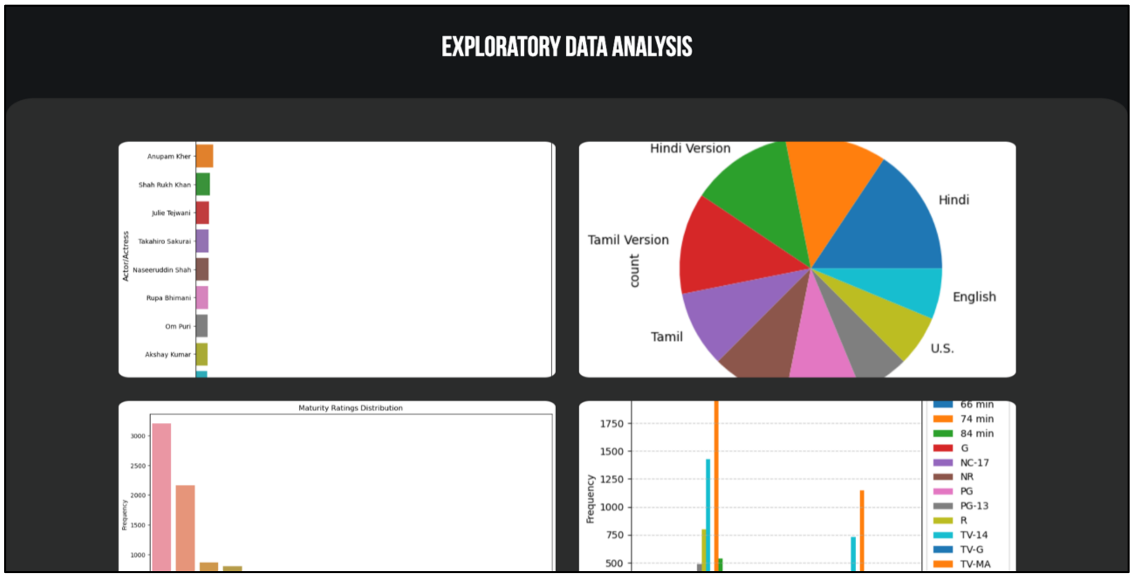
MOVIES AND SHOWS RECOMMENDATION SYSTEM

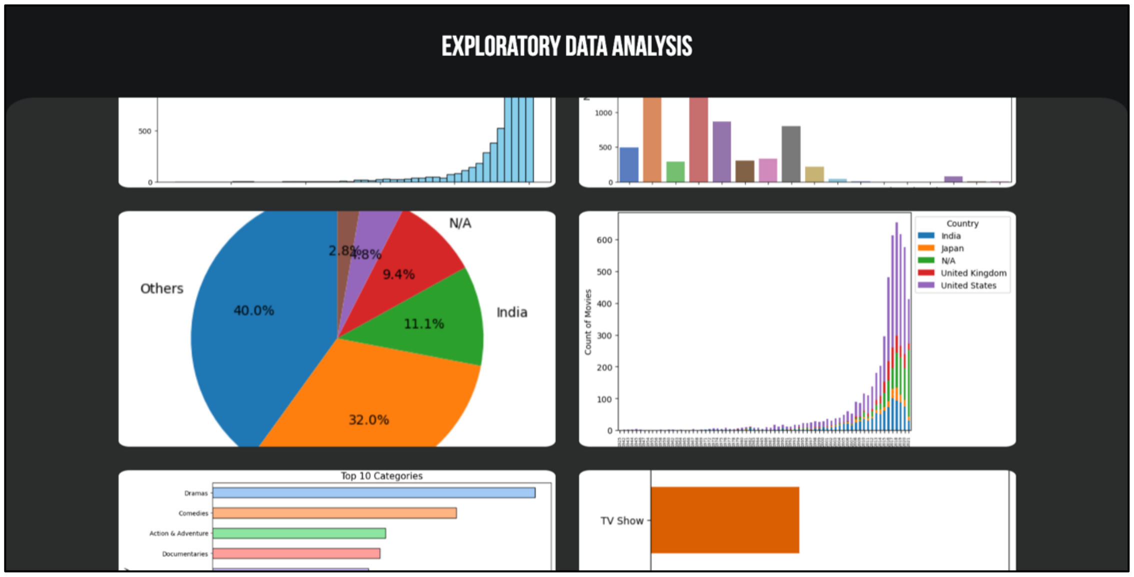
CONTENT SIMILAR TO DI ARE:

NO SUCH MOVIE FOUND

3. EDA.html

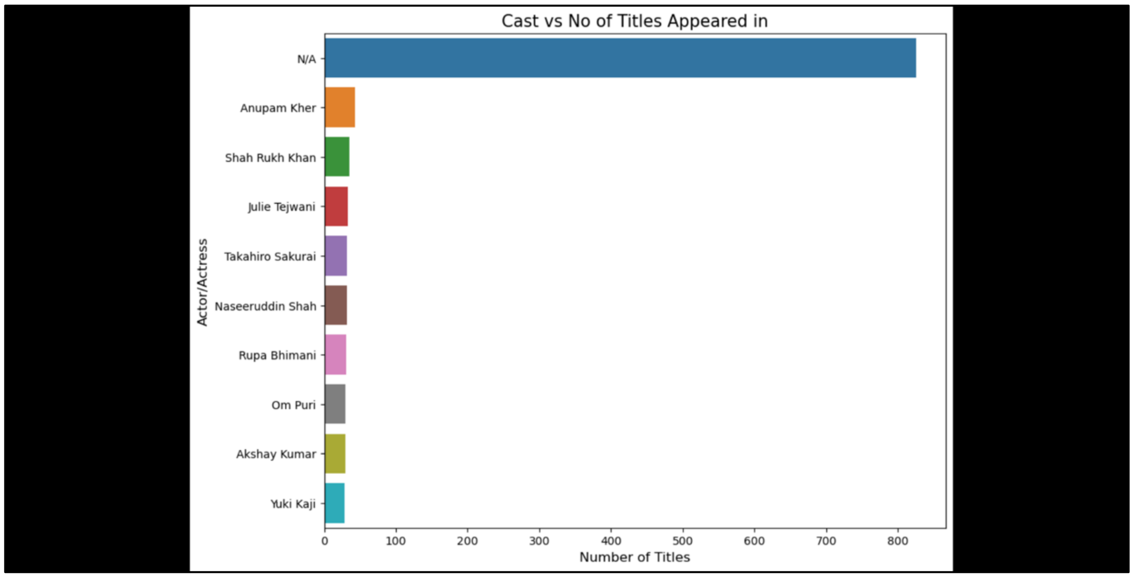
- This file renders the Exploratory Data Analysis (EDA) page.
- It contains gallery section for the Exploratory Data Analysis Images.
- The gallery section displays a grid of images, with each image being a link that opens the image in a full-screen view (/image/<loop.index>).
- The images are dynamically rendered using a Jinja2 template loop ({% for image in images %}).





4. Full_screen.html

- This file renders a full-screen view of an image.
- It displays the selected image in full-screen mode with a black background.
- The image source is dynamically set using a Jinja2 template variable ({{ url_for('static', filename=image) }}).



CSS:

Font import

- We imported a custom font named "CustomFont" from a local file (BebasNeue.ttf).

Global Styles:

- It adds 0 margin to the body element.

Container Styles:

- It sets up a relative positioning context using the class .container-1 for its child elements.
- It sets the background and padding for the layer using the class .base-layer.
- It sets the background, padding, and padding-bottom of another layer using the class. f-layer.
- It sets styles of a layer positioned absolutely, with a background, border-radius, positioning, dimensions, padding, and overflow using the class. s-layer.

Typography:

- It styles the main title with the color, font family, size, weight, alignment, letter-spacing, and margin using the class .title.
- It styles a subtitle using the class .sub-title with color, font family, size, weight, alignment, letter-spacing, padding, and vertical positioning.

- It styles another subtitle using the class. subtitle with color, font family, size, weight, and letter-spacing.

Responsive Styles:

- It includes media queries for different screen widths, adjusting the positioning and dimensions of certain elements to ensure responsive behavior.

Form Styles:

- It styles the input[type="text"] elements with padding, width, border, border-radius, font family, font size, and text alignment.
- It styles the next button with background color, color, border, padding, border-radius, font size, cursor, and font family using the class. next.
- It styles a form section using the class with settings to display, flex-direction, and alignment for form sections.

Suggestion List Styles:

- It styles the appearance of a suggestion list, including list-style-type, alignment, padding, maximum height, width, overflow behavior, background color, and border-radius for the ID:suggestions.
- It styles the appearance of the individual.

Styles:

- The styles for .gallery img define the style of images inside the gallery by setting the width, height, object-fit, and border-radius.
- The styles for .gallery img:hover define the hover effect for the gallery image.

JAVASCRIPT

The JavaScript code has only one functionality:

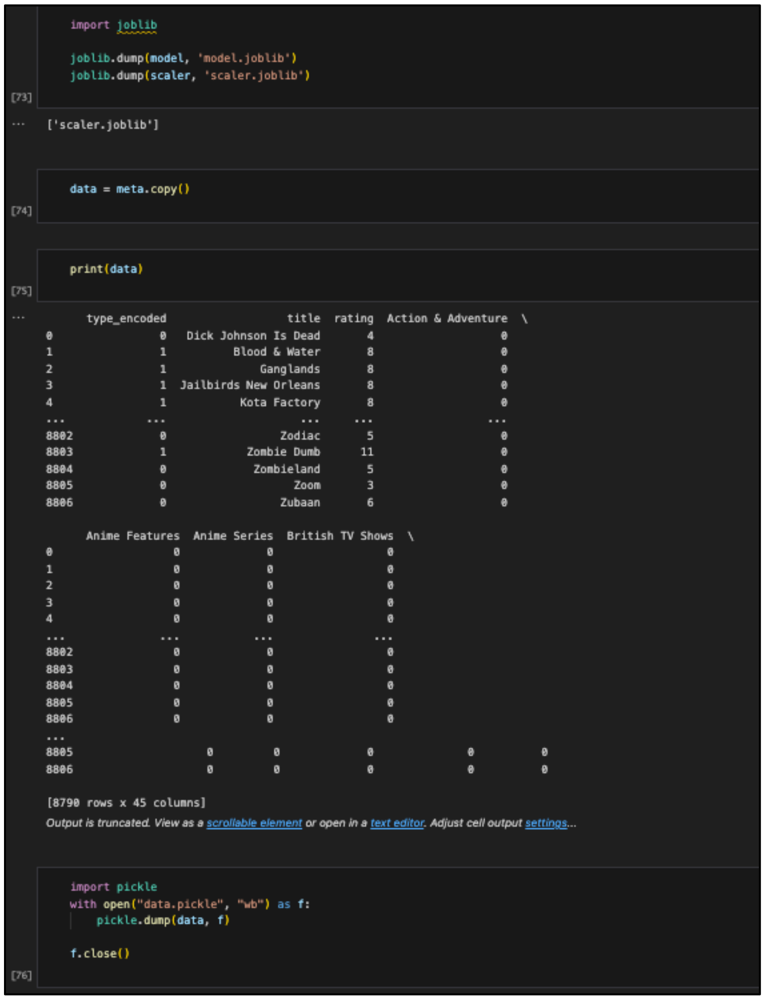
Movie Suggestion Functionality:

- This code listens for user input in the movie search input field.
- When the user types at least three characters, it fetches the movies.csv file from the server.
- It parses the CSV data to extract movie titles and years.
- It filters the movie titles based on the user's input and displays the matching titles as suggestions in a list.
- When the user clicks on a suggestion, the selected title is populated in the input field.
- This functionality helps users find and select movies more easily.

STEPS TO RUN THE WEB APP

Step 1:

First, we need to create the joblib files and the pickle files that are required for the running of the model in the flask application. These files include data.pickle and model.joblib files.



```
import joblib

joblib.dump(model, 'model.joblib')
joblib.dump(scaler, 'scaler.joblib')
```
[73]

```
['scaler.joblib']
```

```
data = meta.copy()
```
[74]

```
print(data)
```
[75]

```
      type_encoded                  title  rating  Action & Adventure  \
0                0   Dick Johnson Is Dead       4                   0
1                1          Blood & Water       8                   0
2                1              Ganglands       8                   0
3                1  Jailbirds New Orleans       8                   0
4                1            Kota Factory       8                   0
...            ...                    ...     ...                 ...
8802             0                 Zodiac       5                   0
8803             1            Zombie Dumb      11                   0
8804             0             Zombieland       5                   0
8805             0                   Zoom       3                   0
8806             0                 Zubaan       6                   0

      Anime Features  Anime Series  British TV Shows  \
0                  0             0                 0
1                  0             0                 0
2                  0             0                 0
3                  0             0                 0
4                  0             0                 0
...              ...           ...               ...
8802               0             0                 0
8803               0             0                 0
8804               0             0                 0
8805               0             0                 0
8806               0             0                 0
...
8805                    0        0           0             0         0
8806                    0        0           0             0         0

[8790 rows x 45 columns]
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

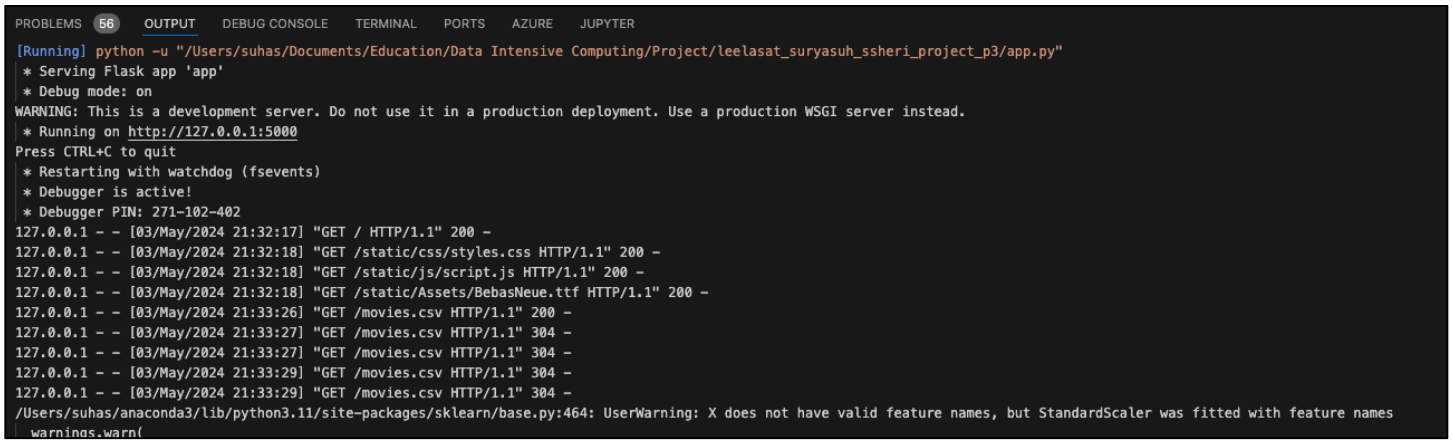
```
import pickle
with open("data.pickle", "wb") as f:
    pickle.dump(data, f)

f.close()
```
[76]

Joblib is a library in Python that provides efficient tools for saving and loading large data structures, such as NumPy arrays and machine learning models, to and from disk. It is optimized for scientific computing and is commonly used in machine learning tasks.

Pickle is a built-in Python module that allows serialization and deserialization of Python objects, enabling the saving and loading of almost any Python object to and from disk or over a network connection.

Step 2: Running the flask app using terminal.



```
[Running] python -u "/Users/suhas/Documents/Education/Data Intensive Computing/Project/leelasat_suryasuh_ssheri_project_p3/app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (fsevents)
 * Debugger is active!
 * Debugger PIN: 271-102-402
127.0.0.1 - - [03/May/2024 21:32:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2024 21:32:18] "GET /static/css/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2024 21:32:18] "GET /static/js/script.js HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2024 21:32:18] "GET /static/Assets/BebasNeue.ttf HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2024 21:33:26] "GET /movies.csv HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2024 21:33:27] "GET /movies.csv HTTP/1.1" 304 -
127.0.0.1 - - [03/May/2024 21:33:27] "GET /movies.csv HTTP/1.1" 304 -
127.0.0.1 - - [03/May/2024 21:33:29] "GET /movies.csv HTTP/1.1" 304 -
127.0.0.1 - - [03/May/2024 21:33:29] "GET /movies.csv HTTP/1.1" 304 -
/Users/suhas/anaconda3/lib/python3.11/site-packages/sklearn/base.py:464: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

A URL "http://127.0.0.1:5000" appears. This means, our program started giving input to the server through the POST method. And the website can be accessed through this link.

Step 3: When we open the above URL in any browser locally, we can access the webpage. Then the user can enter the movie/series name in the text box. There is also a auto fill kind of suggestion feature, that helps the user to see whether the title is available in the dataset.

Step 4: When clicked on the Suggest Similar Content Button, the flask app fetches the results from the model and then the results are displayed on the results.html page.

Step 5: If the user wants to see the insights of the data, he/she can click on the view EDA link, they can see the EDA in the gallery.

RECOMMENDATIONS AND INSIGHTS FOR USERS

RECOMMENDATIONS:

1. **Explore Similar Content**:

Encourage users to try out the recommended movies or TV shows that are similar to the one they entered. This can help them discover new content that aligns with their interests and preferences.

2. **Personalized Recommendations:**

Highlight that the recommendations are personalized based on the user's input and the underlying data analysis. This can build trust in the system and encourage users to rely on the recommendations.

3. **Expand Interests:**

Recommend that users occasionally try content from different genres or categories than their usual preferences. This can help them discover new interests and broaden their viewing horizons.

INSIGHTS:

1. **Popularity Trends:**

Provide insights into the most popular movies or TV shows based on user ratings, views, or other engagement metrics. This can help users stay updated on trending content and popular titles.

2. **Genre Analysis:**

Share insights into the most popular genres or categories among users. This can help users understand the general preferences of the user base and potentially discover new genres to explore.

3. **Demographic Preferences:**

If available, share insights into how different demographic groups (age, gender, location, etc.) tend to prefer certain types of content. This can help users understand the diversity of preferences and potentially discover content popular among specific groups.

4. **Release Year Patterns:**

Analyze and share insights into how user preferences vary based on the release year of the content. This can help users understand if they tend to prefer newer releases or if they gravitate towards classic or retro content.

5. **Data-driven Insights:**

Highlight that the recommendations and insights are driven by data analysis and machine learning techniques. This can build credibility in the system and demonstrate the power of data-driven decision-making.

PROJECT EXTENSIONS AND ENHANCEMENTS:

1. **Collaborative Filtering**:

   Implementing collaborative filtering techniques to incorporate user ratings and preferences into the recommendation algorithm. This would allow the system to provide recommendations based on similar users' preferences, in addition to content similarities.

2. **User Profiles**:

   Developing user profiles that capture individual viewing histories, preferences, and ratings. This would enable more personalized recommendations tailored to each user's unique interests and tastes.

3. **Content-based Filtering**:

   Enhance the current content-based filtering approach by incorporating additional content metadata, such as genre, director, cast, and plot summaries. This could improve the relevance and accuracy of recommendations.

4. **Hybrid Approach**:

   Combine collaborative filtering and content-based filtering techniques to create a hybrid recommendation system. This approach can leverage the strengths of both methods and potentially provide more robust and accurate recommendations.

5. **Social Integration**:

   Integrate social features that allow users to share recommendations, follow friends, and discover content based on their social connections' preferences.

6. **Interactive Interface**:

Enhance the user interface to provide more interactive and engaging experiences. This could include features like visual browsing, filtering options, and the ability to create personalized watchlists or queues.

CONCLUSION

In this project we developed a personalized recommendation system for Netflix movies and TV shows using unsupervised machine learning techniques. Various clustering algorithms like K-Means, BIRCH, Mean Shift, Gaussian Mixture Models, Agglomerative Clustering, and DBSCAN were explored and evaluated. BIRCH outperformed others, exhibiting excellent clustering quality with high Silhouette Score and well-separated, dense clusters.

A user-friendly web application built with Flask, HTML, CSS, and JavaScript enabled users to input preferred titles and receive recommendations from the trained BIRCH model. The Exploratory Data Analysis section provided valuable insights into content preferences and trends.

While promising results were achieved, potential enhancements include incorporating collaborative filtering, developing user profiles, implementing hybrid approaches, social integration, interactive interfaces, and leveraging additional content metadata for improved recommendation accuracy and personalization.

REFERENCES

1. Dataset Link - https://huggingface.co/datasets/hugginglearners/netflix-shows
2. Pandas Documentation – https://pandas.pydata.org/docs/
3. NumPy Documentation – https://numpy.org/doc/#
4. Matplotlib Documentation – https://matplotlib.org/stable/
5. Scikit Learn Documentation – https://scikit-learn.org/stable/
6. Seaborn Documentation - https://seaborn.pydata.org/
7. K-Means Clustering - https://stanford.edu/~cpiech/cs221/handouts/kmeans.html
8. BIRCH Explained - https://medium.com/birch-explained/
9. Mean-shift Clustering Tutorial - https://www.geeksforgeeks.org/ml-mean-shift-clustering/
10. Gaussian Mixture Model - https://medium.com/gmm/
11. Agglomerative Clustering - https://medium.com/dagglomerative-clustering/
12. DBSCAN Tutorial - https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/
13. Flask Documentation - https://flask.palletsprojects.com/

14. HTML Documentation - https://developer.mozilla.org/en-US/docs/Web/HTML

15. CSS Documentation - https://developer.mozilla.org/en-US/docs/Web/CSS

16. JavaScript Documentation - https://developer.mozilla.org/en-US/docs/Web/JavaScript

THE END