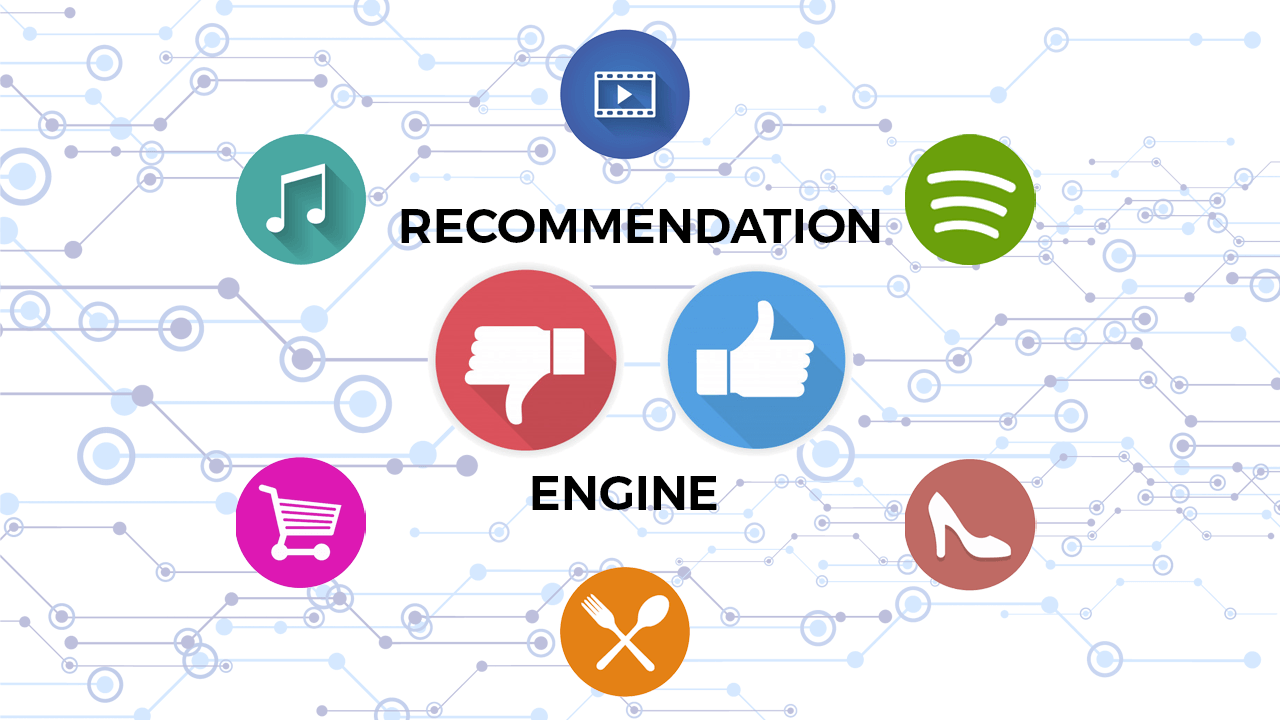
**Product Development using Different Recommendation Engine Algorithms**



-Akash Ramdev (AS21DXB005)

-Jinen Mirje(AS21DXB013)

-Manglam Jain(AS21DXB012)

-Vibhav Bagga(AS21DXB001)

**Problem statement**

We are provided with two different datasets.

1. IMDB Top 1000
2. Movie Lens

Develop an algorithm to give recommendations based on past data. Use collaborative, content-based filtering and Two Tower Model and develop an UI for the user to see the Top 10 recommendations.

**Objective**

* Understanding the 2 datasets
* Merge datasets together to create final data
* Formulation and decision of the algorithms to be used
* Make UI for recommendations

**About the Dataset**

**MovieLens:**

* MovieLens was founded in 1997 by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, to collect research data on personalised suggestions.
* The full data set contains 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. It also includes tag genome data with 12 million relevance scores across 1,100 tags (Last updated 8/2017).

**IMDB:**

* IMDb (also known as the Internet Movie Database) is an online database of information related to movies, it combines movie plot description, Metastore ratings, critic and user ratings and reviews, release dates, and many more aspects.
* This dataset includes IMDB top 1,000 movies of all time with attributes such as Title, Certificate, Duration, Genre, etc.

**Dataset Features:**

**Movie**

* Movie ID
* Title
* Genre
* Rows – 27,278

**Rating**

* User ID
* Movie ID
* Rating
* Time Stamp
* Rows – 20,000,263

**IMDB**

* Series Title
* Overview
* Rows - 1000

**Final Dataset**

A picture containing text, receipt, screenshot, algebra

Description automatically generated

**Collaborative Filtering**

Collaborative filtering is a widely used technique in recommender systems that leverages the preferences and behaviors of users to make personalized recommendations. It operates on the assumption that users who have similar preferences in the past will have similar preferences in the future. Collaborative filtering does not rely on explicit knowledge about items but rather on the collective wisdom of users to identify similarities and make recommendations.

**Types of Collaborative Filtering**

Collaborative filtering can be categorized into two main types: user-based collaborative filtering and item-based collaborative filtering.

**User-Based Collaborative Filtering:** User-based collaborative filtering focuses on finding similar users based on their past preferences and recommending items that those similar users have liked or rated highly. This approach assumes that users who have rated items similarly in the past will have similar tastes and preferences in the future.

**Item-Based Collaborative Filtering:** Item-based collaborative filtering, on the other hand, identifies similar items based on the ratings and preferences of users. It recommends items that are similar to the ones a user has liked or rated highly. This approach assumes that if a user has enjoyed or preferred a particular item, they are likely to enjoy other similar items as well.

Diagram

Description automatically generated

1. **Memory Based**

Memory-based techniques calculate the similarity between users or products using historical user ratings data. These techniques work by defining a similarity metric between users or items, then identifying the most comparable to suggest undiscovered stuff. Model-based

User-Based and Item-Based are the two primary categories of memory-based collaborative filtering algorithms. Even if the differences between them are slight, it can occasionally be difficult to determine which should be utilized when.

**User-Based Collaborative Filtering:**

Here, we look for users who have viewed or rated content similarly, and we leverage their preferences to suggest new stuff.

A disadvantage is that there are typically many more users than items, which results in much larger user similarity matrices (more on this in the section that follows), which causes performance and memory problems on larger datasets and forces the use of parallelization techniques or alternative methods entirely.

Another frequent issue is a cold-start: There may not be enough data on a new user's preferences, so there is nothing to compare against.

**Item-Based Collaborative Filtering:**

The concept is the same in item-based collaborative filtering, but instead of starting with a specific movie (or set of movies), we locate related movies based on user preferences.

Item similarity matrices are typically smaller than user-based techniques, which will lower the cost of locating neighbors in our similarity matrix. Additionally, the cold-start issue will not be an issue with this strategy because just one item is sufficient to suggest additional items that are comparable.

**User-item matrix:**

The user-item matrix needs to be created first. The dataframe is filled with the user's rating (if it exists, 0 otherwise), which is basically a pivoted table from the rating data where the rows are the users, the columns are the movies:

**Similarity Matrix:**

We'll create a similarity matrix next. We aim to identify a proximity metric among all users (or things) in the user-item matrix using the similarity between vectors. The cosine similarity is a widely used metric.

As is well known, the name of this similarity metric comes from the fact that it is equal to the cosine of the angle between the two vectors under comparison, in this case, the user (or item) similarity vectors of scores. The cosine will be higher and so produce a higher similarity factor the smaller the angle between two vectors. For further information on this, refer to the aforementioned section. Sklearn's metrics can be applied. In this instance, we'll be using the cosine\_similarity pairwise sub-module for pairwise distance or similarity metrics.

1. **KNN**

Users' actions are used by collaborative filtering-based systems to suggest other products. They can typically be either user-based or item-based. User-based approaches are typically not preferred over item-based approaches. Due to users' tendency to change, user-based approaches are frequently more difficult to scale than item-based approaches. Because items typically don't change much, item-based approaches can frequently be computed offline.

For this use scenario, KNN is the ideal go-to model, and KNN is a great starting point for developing recommender systems. In item-based collaborative filtering, KNN will employ a predefined distance metric to identify groups of related items based on user ratings, and will then make recommendations using the distance metric in item ratings of the top-k nearest neighbours.

1. **SVD**

Collaborative filtering is a widely used technique in recommender systems to make personalized recommendations by leveraging the behaviour and preferences of similar users. One popular approach in collaborative filtering is Singular Value Decomposition (SVD), which has proven to be effective in capturing latent factors and reducing the dimensionality of the user-item interaction matrix.

The SVD-based collaborative filtering algorithm consists of the following steps:

**Data Preparation:** The user-item interaction data is typically represented in a sparse matrix format, where rows correspond to users, columns correspond to items, and the cells contain ratings or indicators of interactions. This matrix is then preprocessed to handle missing values, normalize the ratings, and account for any biases.

**SVD Decomposition:** The preprocessed user-item matrix is decomposed using SVD to obtain the three component matrices: U, Σ, and V^T. The rank, or the number of singular values considered, determines the dimensionality of the approximation and affects the trade-off between accuracy and computational complexity.

**Dimensionality Reduction:** To reduce the dimensionality of the data, a lower-rank approximation of the original user-item matrix is constructed using a subset of the singular values and corresponding singular vectors. By truncating the singular values and the associated vectors, we retain the most important latent factors that explain the variations in the data.

**Generating Recommendations:** Once the dimensionality reduction is performed, the lower-rank approximation of the user-item matrix can be utilized to generate recommendations. This is achieved by employing similarity measures (e.g., cosine similarity) to identify similar users or items. Recommendations are made based on the predicted ratings derived from the reconstructed matrix.

**Content Based**

Content-based filtering is a recommendation technique that focuses on the characteristics and attributes of items to make personalized recommendations. It leverages the features and properties of items, such as textual descriptions, metadata, or item attributes, to understand user preferences and match them with relevant items. Content-based filtering is particularly useful when there is limited or no historical user-item interaction data available.

Content-based filtering involves the following key steps:

**Item Representation:** The first step in content-based filtering is to represent items in a suitable format for analysis. This can involve extracting relevant features from item descriptions, metadata, or any available textual information. These features can include keywords, genre, actors, directors, or any other relevant attributes that describe the items.

**Profile Creation:** Each user is assigned a profile that represents their preferences based on their previous interactions or explicit feedback. This profile is created by analyzing the features and attributes of the items they have rated, liked, or interacted with in the past.

**Item Similarity:** Content-based filtering calculates the similarity between items based on their feature vectors. Various similarity measures, such as cosine similarity or Jaccard similarity, can be used to quantify the similarity between item vectors. The more similar two items are, the higher their similarity score.

**Recommendation Generation:** To generate recommendations, content-based filtering identifies items that are similar to the ones a user has shown interest in. This is done by comparing the user's profile with the features of other items and selecting the most similar items. The top-N similar items or items with the highest similarity scores are recommended to the user.

Diagram

Description automatically generated

1. **TF-IDF**

Content-based filtering is a recommendation technique that utilizes the characteristics and attributes of items to make personalized recommendations. One popular method within content-based filtering is TF-IDF (Term Frequency-Inverse Document Frequency), which quantifies the importance of terms in a document based on their frequency in the document and their rarity across the entire corpus. TF-IDF is commonly used to represent and compare the textual content of items in order to make recommendations.

TF-IDF is a numerical representation that reflects the significance of terms in a document or item description. It consists of two components:

**Term Frequency (TF):** Term Frequency measures how often a term occurs in a document. It assigns higher weights to terms that appear more frequently, as they are considered more important in describing the document.

**Inverse Document Frequency (IDF):** Inverse Document Frequency calculates the rarity of a term across the entire corpus of documents. It assigns higher weights to terms that are less common across documents, as they are considered more informative and distinctive.

The combination of TF and IDF results in a TF-IDF score for each term in a document, representing its importance in describing the document relative to the entire corpus.

1. **Transformer (BERT)**

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art language model that has shown remarkable performance in natural language processing tasks, including understanding the context and meaning of textual content. By leveraging BERT, content-based filtering can capture the semantic representation of item descriptions and provide more accurate recommendations.

BERT is a transformer-based model that uses a deep bidirectional architecture to capture the context and meaning of text. It is pre-trained on a large corpus of text data and can then be fine-tuned on specific downstream tasks, such as content-based recommendation. BERT's ability to learn contextual relationships in language makes it suitable for capturing the nuanced meanings and relationships within item descriptions.

**Two Tower**

The Two-Tower Model is a powerful recommendation approach that leverages deep neural networks to learn user and item representations in a shared space. This model has gained popularity in recent years due to its ability to capture complex user-item interactions and make accurate recommendations. The Two-Tower Model comprises two separate neural networks, the user tower and the item tower, which jointly learn to map users and items into a shared latent space.

**Architecture** of the Two-Tower Model

The Two-Tower Model consists of the following components:

**User Tower:** The user tower takes user-related input, such as user demographics, historical interactions, or user profiles, and processes it through a series of layers. These layers typically include embedding layers, fully connected layers, and non-linear activation functions. The output of the user tower is a user representation vector in the latent space.

**Item Tower:** The item tower takes item-related input, such as item descriptions, metadata, or item features, and passes it through a similar set of layers as the user tower. The output of the item tower is an item representation vector in the same latent space as the user tower.

**Similarity Calculation:** The user and item representations from the user tower and item tower, respectively, are used to calculate the similarity between users and items. This can be done using various similarity metrics, such as cosine similarity or Euclidean distance. The higher the similarity score between a user and an item, the more likely the item is to be recommended to the user.

**Recommendation Generation:** Based on the calculated similarity scores, the Two-Tower Model generates a list of top-N recommendations for each user. These recommendations are typically the items with the highest similarity scores to the user representation.

Diagram

Description automatically generated

**Performance Metrics in Recommendation Engines**

Evaluating the performance of a recommendation engine is crucial to assess its effectiveness and make informed decisions about system improvements. Various performance metrics are used to measure the quality and accuracy of recommendations. These metrics help understand how well the recommendation engine is performing and enable comparisons between different approaches. In this section, we discuss some commonly used performance metrics in recommendation engines.

**1. Accuracy Metrics**

**a) Precision:** Precision measures the proportion of relevant recommendations among all the recommendations made. It focuses on the quality of recommendations by calculating the ratio of true positives (relevant recommendations) to the total recommendations made. Precision helps evaluate how precise and targeted the recommendations are.

**b) Recall:** Recall measures the proportion of relevant recommendations that are successfully identified. It focuses on the coverage of recommendations by calculating the ratio of true positives to the total relevant items. Recall helps evaluate the ability of the recommendation engine to capture all relevant items.

**c) F1 Score:** The F1 score combines precision and recall into a single metric. It is the harmonic mean of precision and recall, providing a balanced measure of both precision and coverage. The F1 score is useful when there is an imbalance between precision and recall, and a single metric is desired to evaluate the overall performance.

**2. Diversity Metrics**

**a) Catalog Coverage:** Catalog coverage measures the proportion of unique items in the recommendation catalog that are actually recommended to users. It indicates how well the recommendation engine explores the entire catalog and avoids over-representing a few popular items. Higher catalog coverage indicates a broader range of recommended items.

**b) Novelty:** Novelty measures the degree to which recommended items are different or novel to users. It assesses the diversity of recommendations by considering how distinct the recommended items are from the items that the user has interacted with in the past. Higher novelty suggests the recommendation engine is introducing users to new and unfamiliar items.

**3. Ranking Metrics**

**a) Mean Average Precision (MAP):** MAP measures the average precision of the recommended items at different positions in the recommendation list. It considers both the relevance of recommended items and their ranking position. MAP is particularly useful when the order of recommendations is important, such as in scenarios where only a few top recommendations are shown.

**b) Normalized Discounted Cumulative Gain (NDCG):** NDCG is a ranking metric that assesses the quality of the recommendation list based on the relevance of items and their position. It calculates the gain of each recommended item, discounting it based on its position in the list. NDCG provides a measure of how well the recommendations are ranked.

**4. User Engagement Metrics**

**a) Click-Through Rate (CTR):** CTR measures the percentage of users who click on the recommended items out of the total impressions or recommendations shown. It indicates the effectiveness of the recommendations in capturing user interest and driving user engagement. Higher CTR suggests more relevant and engaging recommendations.

**b) Conversion Rate:** Conversion rate measures the percentage of users who take a desired action, such as making a purchase or completing a specific task, after receiving the recommendations. It helps evaluate the impact of recommendations on user behaviour and business goals.

**UI Development**

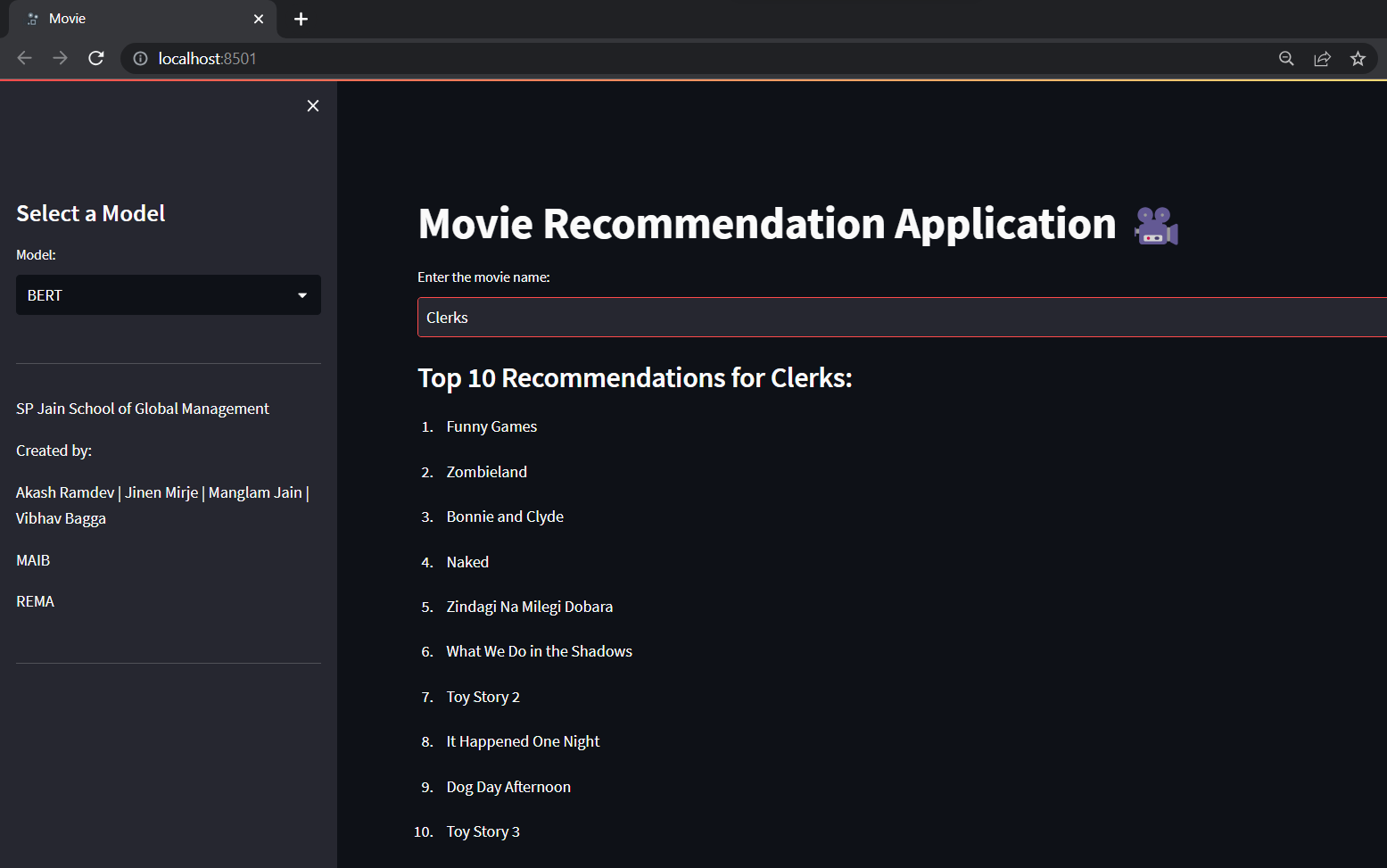
The user interface (UI) development phase involved designing and implementing an interactive web application using the Streamlit framework. The goal was to create a user-friendly and visually appealing interface for the Movie Recommendation App.

In the initial stages of development, we created a basic UI layout using Streamlit components such as titles, text inputs, and dropdown menus. The UI allowed users to select a recommendation model and provide the necessary input, either a movie name or a user ID, based on the selected model.

To enhance the visual appeal and user experience of the application, several improvements were made to the UI design. These changes included:

* Adding appropriate titles and headings to provide clear instructions and guidance to the users.
* Beautifying the UI by applying styles, such as custom colors and fonts, to create a cohesive and visually pleasing design.
* Organizing the UI elements in a logical and intuitive manner, ensuring easy navigation and understanding.
* Displaying recommendations in a more readable format, such as a numbered list, to improve clarity.

After multiple iterations and refinements, we arrived at a final UI design that met the project requirements and provided an engaging user experience. The final UI features a clean and intuitive layout, clear instructions, and a visually appealing presentation of movie recommendations.

****

**Conclusion**

In this report, we explored different approaches in recommendation systems, including collaborative filtering, content-based filtering, and the Two-Tower Model. Each approach offers unique advantages and considerations for building effective recommendation engines. Let's summarize the key points discussed:

**Collaborative Filtering:** Collaborative filtering methods, such as Singular Value Decomposition (SVD) and K-Nearest Neighbors (KNN), leverage user-item interactions to make recommendations. SVD captures latent factors to identify similarities between users and items, while KNN calculates item similarities based on user behavior. These methods are effective in capturing user preferences and generating personalized recommendations.

**Content-Based Filtering:** Content-based filtering utilizes item characteristics and user profiles to make recommendations. TF-IDF and BERT are popular techniques in content-based filtering. TF-IDF quantifies the importance of terms in item descriptions, while BERT captures semantic understanding of textual content. Content-based filtering is valuable for recommending items similar to those a user has already shown interest in.

**Two-Tower Model:** The Two-Tower Model leverages deep neural networks to learn user and item representations in a shared latent space. By capturing complex user-item interactions, it offers accurate and personalized recommendations. This model allows flexibility in incorporating different types of user and item data, making it suitable for various recommendation scenarios.

In addition to discussing different recommendation approaches, we also explored performance metrics for evaluating recommendation engines. Accuracy metrics such as precision, recall, and F1 score assess the quality of recommendations. Diversity metrics such as catalog coverage and novelty measure the range and uniqueness of recommendations. Ranking metrics like MAP and NDCG evaluate the effectiveness of recommendation ordering.

**References**

1. Dey, V. (2022b). Collaborative Filtering Vs Content-Based Filtering for Recommender Systems. *Analytics India Magazine*. <https://analyticsindiamag.com/collaborative-filtering-vs-content-based-filtering-for-recommender-systems/>
2. Muvi. (2022, January 6). The Difference Between Collaborative and Content Based Recommendation Engines. *Medium*. https://muvi-com.medium.com/the-difference-between-collaborative-and-content-based-recommendation-engines-ade24b5147f2
3. Roy, A. (2021, December 15). Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering. *Medium*. https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421
4. Nixon, A. E. (2022, October 15). Building a movie content based recommender using tf-idf. *Medium*. https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5
5. Michaelgruberx. (2022, April 25). Build a Memory-Based Collaborative Filter with Python | Python in Plain English. *Medium*. https://python.plainenglish.io/build-a-memory-based-collaborative-filter-with-python-a3d366a0ad1e
6. Nixon, A. E. (2022a, October 15). Building a memory based collaborative filtering recommender. Medium. <https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331>
7. Longo, C. (2023, April 20). Evaluation Metrics for Recommender Systems - Towards Data Science. Medium. <https://towardsdatascience.com/evaluation-metrics-for-recommender-systems-df56c6611093>
8. Deutschman, Z. (2023). Recommender Systems: Machine Learning Metrics and Business Metrics. *neptune.ai*. https://neptune.ai/blog/recommender-systems-metrics
9. Jones, A. (2023, May 20). Streamlit from Scratch: Build a Data Dashboard with Streamlit’s Layout and UI features. *Medium*. https://towardsdatascience.com/streamlit-from-scratch-build-a-data-dashboard-with-streamlits-layout-and-ui-features-a2fc2f0a6a59
10. *How to Build a UI for your Model using Streamlit*. (n.d.). Engineering Education (EngEd) Program | Section. https://www.section.io/engineering-education/streamlit-ui-tutorial/