Milestone 3 Report

Project Title: Movie Recommendation Engine
Course: Introduction to Data Science
Student: Ashfaq Ahmed Mohammed (UFID: 86835927)
GitHub: **GitHub Repository**

Section 1: Objective, Tools, and Datasets Used

1.1 Project Objective

This project aims to develop a **Movie Recommendation Engine** that suggests movies to users based on their preferences. The system leverages user ratings, movie metadata, and streaming availability attributes to generate personalized movie suggestions.

- **Milestone 1** focused on **Exploratory Data Analysis (EDA)**, **handling missing values**, **outlier treatment**, and **preprocessing** the datasets to understand trends, biases, and structure the data for modelling.
- **Milestone 2** concentrated on **Feature Engineering**, **encoding** of categorical and text data (via TF-IDF and FastText), **Feature Selection**, and Unsupervised Model Training using clustering algorithms **(KMeans, DBSCAN, GMM) and semantic embedding methods.**
- **Milestone 3** focuses on:
    - Evaluation and Interpretation of model performance.
    - Identification of model biases and limitations.
    - Develop an interactive dashboard using **Streamlit** to present recommendations.

1.2 Tools and Libraries Used

Data Handling and Processing

- **pandas** – For loading, cleaning, merging, and manipulating structured datasets.
- **numpy** – For numerical operations, matrix operations, and array processing.

Data Visualization

- **matplotlib** – For creating visualizations and graphs.
- **seaborn** – For statistical and distribution-based data visualization.

Machine Learning, Modeling, and Evaluation

- **scikit-learn** – Comprehensive ML framework used for:
    - **Clustering Algorithms**:
        - K-Means
        - DBSCAN
        - Gaussian Mixture Models (GMM)
    - **Feature Engineering**:
        - **TfidfVectorizer** – For text vectorization.
        - **TruncatedSVD** – Dimensionality reduction for sparse TF-IDF matrices.
        - **PCA** – For reducing dimensions and visualization.
    - **Evaluation Metrics**:
        - **Silhouette Score** – To assess cluster cohesion and separation.
        - **Davies-Bouldin Index** – To measure cluster quality.
        - **Calinski-Harabasz Index** – To assess inter-cluster variance and compactness.
    - **Cosine Similarity**:
        - Used to measure the similarity between movies' feature vectors.

Text Embedding

- **FastText** – For generating dense, semantically rich embeddings of text-based movie attributes.

Interactive Application and Dashboard Development

- **Streamlit** – For building a dynamic, user-interactive movie recommendation web application.
- **pickle** – For loading pre-saved models, data frames, and vectorizers efficiently inside the Streamlit app.

File Handling and Utilities

- **os** – For system operations like path navigation and file management.

1.3 Project Timeline

S.No	Task	Deadline
1	Data Collection and Preprocessing	23rd February 2025
2	EDA	23rd February 2025
3	Feature Engineering and Feature Selection	4th April 2025
4	Data Modelling	4th April 2025
5	Evaluation and Testing	15th April 2025
6	User Interface design and Implementation	23th April 2025

1.4 Cleaned and Merged Dataset

The Google Drive link to the final merged dataset that has the newly engineered features and the cluster labels can be found from this link: Milestone 3 Dataset

1.5 Recap of work done in Milestone 2

As part of Milestone 2, the cleaned movie dataset was transformed into a feature-rich space and produced the first working recommendation models. Three domain-specific numeric features; **Budget-Popularity, Movie-Age, and OTT-Score** were engineered, while text attributes (director, lead actor, genres, production houses, plot overview) were encoded with **TF-IDF** and **FastText**. The resulting vectors were combined with scaled numerical features and fed into four unsupervised approaches: **K-Means, DBSCAN, Gaussian Mixture Model (GMM), and a FastText cosine-similarity method**. The trained models, vector matrices, and lookup dictionaries were exported to disk so they can be reloaded in Milestone 3 for comprehensive evaluation, bias investigation, and integration into the Streamlit dashboard.

Section 2: Evaluation and Interpretation

2.1 Model Performance Evaluation

Because the movie recommendation engine is an unsupervised learning problem, there are no predefined labels or ground truth classes available for evaluation. In supervised learning, evaluation typically involves comparing model predictions directly against known true values using metrics such as accuracy, precision, recall, or F1-score. However, in the absence of such labelled ground truth in unsupervised settings, concrete metrics of correctness do not exist.

Instead, the evaluation must rely on internal validation metrics that assess clustering quality based on intrinsic properties of the data distribution, I used the following intrinsic parameters to evaluate my model's performance:

- **Silhouette Score** measures how well each movie fits within its assigned cluster compared to other clusters, serving as a proxy for cohesion and separation.
- **Davies-Bouldin Index** evaluates the average similarity between each cluster and its most similar one, offering a measure of separation compactness.
- **The Calinski-Harabasz Index** calculates the ratio of between-cluster variance to within-cluster variance, indicating how well the clusters are formed.

Model	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz index
K-Mean	0.041	1.88	12472.554
DBSCAN	0.672	1.638	858.332
GMM	-0.066	7.078	4751.553

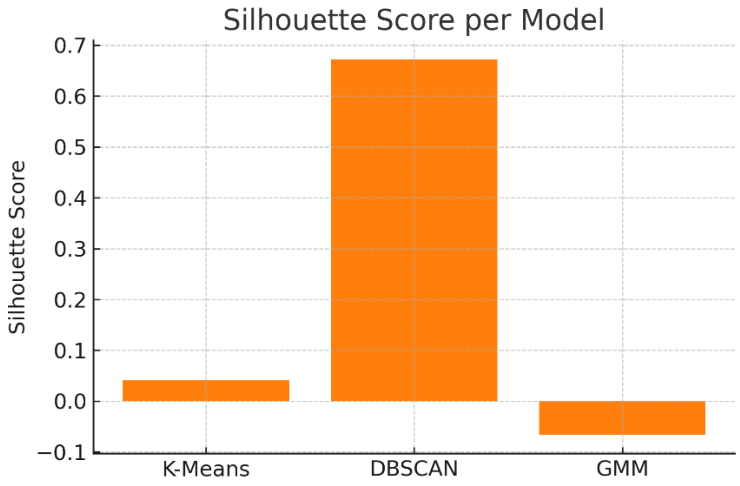

Figure 1 Sihouette Score

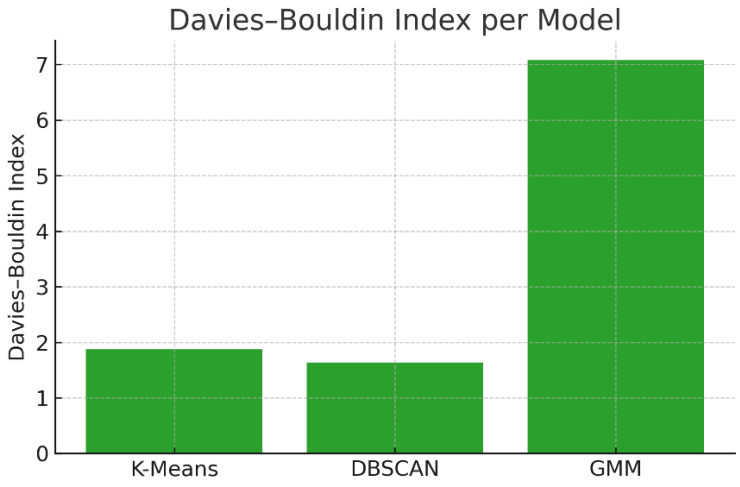

Figure 2 Davies-Bouldin Index

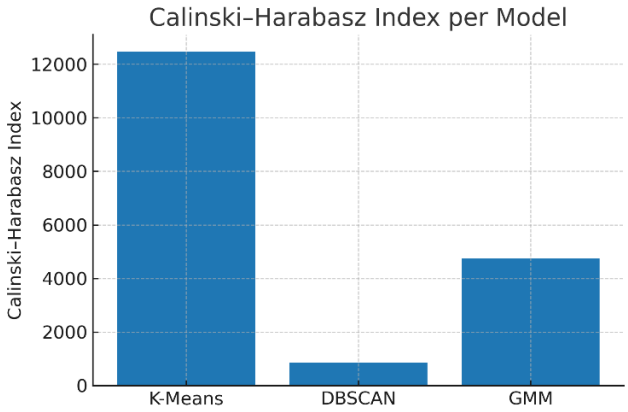

Figure 3 Calinski - Harabasz Index

Interpretation of Scores:

- **K-Means Clustering**:
  Achieved a **low Silhouette Score** (0.041), indicating that the cluster boundaries were not very well-defined, but a **very high Calinski-Harabasz Index** (12472.554) suggested that when clusters were formed, they exhibited strong inter-cluster variance. This implies that although K-Means formed tight groups, the decision boundaries between some clusters may have been ambiguous.
- **DBSCAN Clustering**:
  Produced the **best Silhouette Score** (0.672), demonstrating strong intra-cluster cohesion and inter-cluster separation. The Davies-Bouldin Index (1.638) also remained relatively low, reinforcing the quality of clustering. However, DBSCAN's Calinski-Harabasz Index (858.332) was lower compared to K-Means, reflecting the fact that DBSCAN's clusters might be smaller and irregularly shaped due to its density-based nature.
- **Gaussian Mixture Model (GMM)**:
  GMM had a **negative Silhouette Score** (-0.066) and a **high Davies-Bouldin Index** (7.078), suggesting that clusters overlapped significantly. This is expected behaviour because GMM assigns soft probabilities rather than hard labels, meaning a single movie can partially belong to multiple clusters. Despite this, the Calinski-Harabasz Index (4751.553) indicates that there was still meaningful separation between cluster centers.

The **fastText-based** model was not evaluated using clustering metrics, since it relies on cosine similarity for nearest neighbour retrieval rather than hard clustering.

2.2 Evaluation Beyond Metrics (Ground Truth)

It is important to note that these internal validation metrics cannot determine whether the cluster groupings are meaningful from a user perspective. For example, two films may belong to different clusters based on these metrics but be very similar from a user's perspective (for example, genre, storyline, director style).

To supplement the internal metrics, a qualitative ground-truth evaluation was carried out. This entailed selecting specific input movies (such as Skyfall, Interstellar, and Toy Story) and manually inspecting the top recommendations generated by each model. The tables below show recommendations by each model for a specific input movie.

Input Movie: Skyfall

K-Means Clustering	Casino Royale, Mission: Impossible - Ghost Protocol, Django Unchained, Captain America: The Winter Soldier, Eagle Eye, Total Recall, The Avengers, Quantum of Solace, Star Wars: The Force Awakens, Avengers: Age of Ultron.

DBScan	Casino Royale, Mission: Impossible - Ghost Protocol, Django Unchained, Captain America: The Winter Soldier, Eagle Eye, Total Recall, The Avengers, Quantum of Solace, Star Wars: The Force Awakens, Avengers: Age of Ultron.
GMM	Captain America: The Winter Soldier, Thor: The Dark World, The Dark Knight, The Chronicles of Narnia: The Lion, The Witch and the Wardrobe, Hugo, Captain America: Civil War, Batman Begins, Guardians of the Galaxy, G.I. Joe: The Rise of Cobra, Eight Below.
FastText Embeddings	Captain America: The Winter Soldier, Iron Man 2, Pirates of the Caribbean: Dead Men Tell No Tales, The Dark Knight, Total Recall, Mission: Impossible - Ghost Protocol, The Expendables 3, Star Trek Into Darkness, Avengers: Age of Ultron, Casino Royale.

Input Movie: Interstellar

K-Means Clustering	Inception, Watchmen, Pacific Rim, The Dark Knight Rises, The Matrix, The Matrix Reloaded, Valerian and the City of a Thousand Planets, The Revenant, Troy, The Hunger Games: Catching Fire.
DBScan	Inception, Watchmen, Pacific Rim, The Dark Knight Rises, The Matrix, The Matrix Reloaded, Valerian and the City of a Thousand Planets, The Revenant, Troy, The Hunger Games: Catching Fire.
GMM	Inception, The Dark Knight Rises, The Prestige, Into the Wild, Superman Returns, Mission: Impossible - Rogue Nation, The Pursuit of Happyness, Iron Man, Maze Runner: The Scorch Trials, Saving Private Ryan.
FastText Embeddings	Fantastic Beasts and Where to Find Them, The Hobbit: The Battle of the Five Armies, Valerian and the City of a Thousand Planets, Tomorrowland, Transformers: Age of Extinction, Aliens, Pacific Rim, Prometheus, Battleship, Transformers: The Last Knight.

Input Movie: Toy Story

K-Means Clustering	Toy Story 2, Chicken Run, Shrek, Toy Story 3, The Princess Bride, Ratatouille, Kung Fu Panda, Sister Act, Wreck-It Ralph, Finding Nemo.
DBScan	Toy Story 2, Chicken Run, Shrek, Toy Story 3, The Princess Bride, Ratatouille, Kung Fu Panda, Sister Act, Wreck-It Ralph, Finding Nemo.
GMM	Toy Story 2, Ratatouille, Aladdin, The Polar Express, Penguins of Madagascar, The Addams Family, Philadelphia, Austin Powers: The Spy Who Shagged Me, The Italian Job, Quantum of Solace.
FastText Embeddings	Toy Story 2, Toy Story 3, Burn After Reading, Finding Nemo, Bolt, Cars 3, The Emperor's New Groove, Cloudy with a Chance of Meatballs 2, The Incredibles, The Twilight Saga: Breaking Dawn - Part 1.

2.3 Interpretations of Ground Truth

2.3.1 K-Means Ground Truth

In the ground truth evaluation, K-Means clustering generated genre-consistent recommendations.

When **Skyfall** was provided as input, K-Means primarily recommended spy-action thrillers like *Casino Royale, Mission: Impossible - Ghost Protocol, and Quantum of Solace* while remaining thematically consistent.

Similarly, for **Interstellar**, the model returned sci-fi titles such as *Inception, The Matrix, and The Dark Knight Rises,* indicating a strong correlation between futuristic and science fiction themes.

In the case of **Toy Story**, K-Means suggested family-friendly animations like *Toy Story 2, Shrek, and Kung Fu Panda*, which remained true to the animated family genre.

2.3.2 DBSCAN Ground Truth

DBSCAN behaved quite similarly to K-Means in ground truth testing.

Using **Skyfall** as input, it retrieved other action-thrillers such as *Casino Royale and Mission: Impossible* while maintaining genre alignment. For **Interstellar**, DBSCAN again returned well-known sci-fi and dystopian films such as *Inception, The Matrix, and Pacific Rim*. After choosing **Toy Story**, DBSCAN suggested cartoons such as *Toy Story 2, Ratatouille, and Finding Nemo* to ensure a tight genre focus and little drift.

2.3.3 Gaussian Mixture Model (GMM) Ground Truth

GMM's ground truth results reflected the soft clustering behaviour.

GMM recommended action films like The *Dark Knight* for **Skyfall**, as well as less tightly related fantasy films like *The Chronicles of Narnia and Hugo*.

GMM mixed genres for **Interstellar** by combining sci-fi films like *The Dark Knight Rises, The Prestige with emotional dramas like Saving Private Ryan*.

While relevant family animations such as *Toy Story 2 and Ratatouille* were recommended for **Toy Story**, some unexpected titles such as *Philadelphia and Quantum of Solace* appeared.

2.3.4 FastText Embeddings Ground Truth

The FastText-based semantic model generated flexible and broad suggestions based on deeper textual similarities.

FastText recommended Skyfall-style action thrillers like Mission: Impossible - Ghost Protocol and Avengers: Age of Ultron.

When Interstellar was entered, the model returned futuristic, effects-heavy sci-fi films like Valerian and the City of a Thousand Planets, Prometheus, and Transformers.

In the case of Toy Story, FastText primarily recommended animated films such as Finding Nemo and The Incredibles, but it also occasionally introduced outliers such as Burn After Reading.

2.4 Deriving Actionable Insights from Model Predictions

Based on the evaluation and ground truth results, actionable insights about model usability were derived.

K-Means and DBSCAN models are ideal for users who prefer genre-specific recommendations. These models consistently recommend films from the same thematic cluster as the input, making them ideal for users looking for more of the "same type" of film.

The Gaussian Mixture Model (GMM) is appropriate for users looking to discover films from different genres. Its probabilistic clustering produces a broader, more exploratory set of recommendations, albeit at the expense of occasional relevance loss.

FastText embeddings allow for the discovery of semantically related films even when the genre tags differ. This model helps users find novel or stylistically similar content and can recommend hidden gems that would not otherwise be clustered by metadata alone.

Section 3: Biases and Limitations

3.1 Model-Specific Biases and Limitations

Now that the models' recommendation behaviours have been thoroughly evaluated and interpreted through both internal metrics and ground truth analysis, we proceed to discuss the biases and limitations inherent to each model.

3.1.1 K-Means

Genre Rigidity:

K-Means provided consistent but narrow genre recommendations. For example, with Skyfall as input, it suggested only spy thrillers with no thematic variety. This behaviour can limit user discovery outside of dominant genres.

Cluster Count Sensitivity:

The number of clusters used (Using the Elbow Method) significantly affected the performance and diversity of recommendations. Too few clusters resulted in the grouping of disparate films, whereas too many fragmented closely related films.

3.1.2 DBSCAN

Clustering Bias Against Sparse Movies:

DBSCAN uses dense neighbourhoods to form clusters. Using an epsilon value of 0.2 and 13 minimum samples, the model concentrated on forming densely packed groups. As a result, niche, less popular, or independent films frequently failed to meet the density threshold and were labelled as noise, thus being excluded from recommendations.

Parameter Sensitivity:

The hyperparameters (epsilon=0.2, min_samples=13) had a significant impact on the model's performance. A minor change could either divide relevant films into separate clusters or combine unrelated films into large, less meaningful clusters.

3.1.3 GMM

Cluster Boundaries:

Since clusters are modelled as overlapping distributions, semantic clarity between genres became weaker, confusing the genre-based recommendation logic; for instance, it mixed unrelated movie types like recommending fantasy movies alongside spy thrillers.

Poor Separation Evidence:

GMM's low Silhouette Score (-0.066) and high Davies-Bouldin Index indicated that its clusters were not cleanly separated, making recommendations less thematically focused compared to K-Means or DBSCAN.

3.1.4 FastText

Keyword Overlap Bias:

When two movie descriptions share common words, “mission,” “adventure,” “destiny,” etc. FastText may link them even if their genres differ sharply. For example, it can recommend the romantic drama The Notebook for the sci-fi epic Interstellar simply because both synopses emphasise “love” and “fate.”

No Genre Boundary:

Recommendations are driven purely by cosine similarity, without hard cluster boundaries or validation scores. This lack of guardrails means loosely related titles can slip through, so thematic coherence must be verified qualitatively rather than by quantitative metrics.

3.2 Dataset-specific biases and Limitations

3.2.1 Genre Imbalance

The dataset is heavily skewed towards genres such as drama, comedy, and thriller, which results in over-representation of these categories in clustering and recommendations.

Less common genres, such as Documentary and Musical, are under-represented, making it difficult for the system to accurately recommend niche or unconventional movies.

3.2.2 Temporal Bias

More recent movies (1990s onward) dominate the dataset due to better metadata availability and user engagement.

This introduces a temporal bias, favouring modern movies over classic or older films, and reduces exposure to historically significant cinema in recommendations.

3.2.3 Sparse Metadata for Niche Movies

Many lesser-known or independent films have incomplete metadata.

This sparsity has an impact on vectorisation quality and clustering effectiveness, ultimately reducing the likelihood of such films appearing in recommendations, resulting in a bias favouring well-documented, mainstream titles.

Section 4: Tool Development

4.1 Overview

To improve the accessibility and usability of the movie recommendation engine, an interactive dashboard was created with Streamlit. The goal was to create a user-friendly tool that allowed users to explore movie recommendations based on various models without requiring any technical knowledge.

4.2 Exporting and Importing the Models

[147]:
```
import pickle
```

[148]:
```
with open("kmeans_model.pkl", "wb") as f:
    pickle.dump(kmeans, f)

with open("dbscan_model.pkl", "wb") as f:
    pickle.dump(dbscan_clusterer, f)


with open("gmm_model.pkl", "wb") as f:
    pickle.dump(gmm, f)
```

[149]:
```
with open("fasttext_vectors.pkl", "wb") as f:
    pickle.dump(combined_matrix, f)
```

[150]:
```
df.to_csv('Data_clustered', index=False)
```

[152]:
```
with open("title_to_index.pkl", "wb") as f:
    pickle.dump(title_to_index, f)

np.save("combined_features.npy", combined_features)
```

Figure 4 Code Snippet - Exporting Models

The snippet above serialises every object the Streamlit dashboard needs at runtime.

First, each trained clustering model, K-Means, DBSCAN, and GMM, is written to disk as a .pkl file with *pickle.dump()*. Pickle preserves the full Python object (parameters, centroids, etc.), so the app can reload the models instantly rather than re-train them on start-up. Next, the dense FastText feature matrix is stored the same way, and the title-to-index dictionary is pickled so the app can map a user's movie title directly to its vector row. Finally, a CSV snapshot of the clustered data is exported for quick offline inspection. Together, these files let the dashboard launch in seconds, guarantee consistent recommendations across machines, and keep the deployment package small and self-contained.

```
88  def load_resources():
89      df = pd.read_csv("models/Data_clustered.csv")
90      df['title'] = df['title'].astype(str).str.strip()
91      df['popularity'] = pd.to_numeric(df['popularity'], errors='coerce')
92
93      with open("models/kmeans_model.pkl", "rb") as f:
94          kmeans = pickle.load(f)
95      with open("models/dbscan_model.pkl", "rb") as f:
96          dbscan = pickle.load(f)
97      with open("models/gmm_model.pkl", "rb") as f:
98          gmm = pickle.load(f)
99      with open("models/fasttext_vectors.pkl", "rb") as f:
100         fasttext_vectors = pickle.load(f)
101     with open("models/title_to_index.pkl", "rb") as f:
102         title_to_index = pickle.load(f)
103
104     combined_features = np.load("models/combined_features.npy")
105
106     return df, kmeans, dbscan, gmm, fasttext_vectors, title_to_index, combined_features
107
108 df_full, kmeans, dbscan, gmm, fasttext_vectors, title_to_index, combined_features = load_resources()
```

Figure 5 Code Snippet - Importing Models

The above code snippet shows how the exported models and arrays are being imported into the streamlit application, the load_resources() function runs as soon as the UI is displayed on the user's screen.

4.2 Key Functionalities

- **Random Movie Selection Grid**:
  Users are initially presented with a grid of **9 randomly selected movies**. They can choose one to receive recommendations based on that choice.

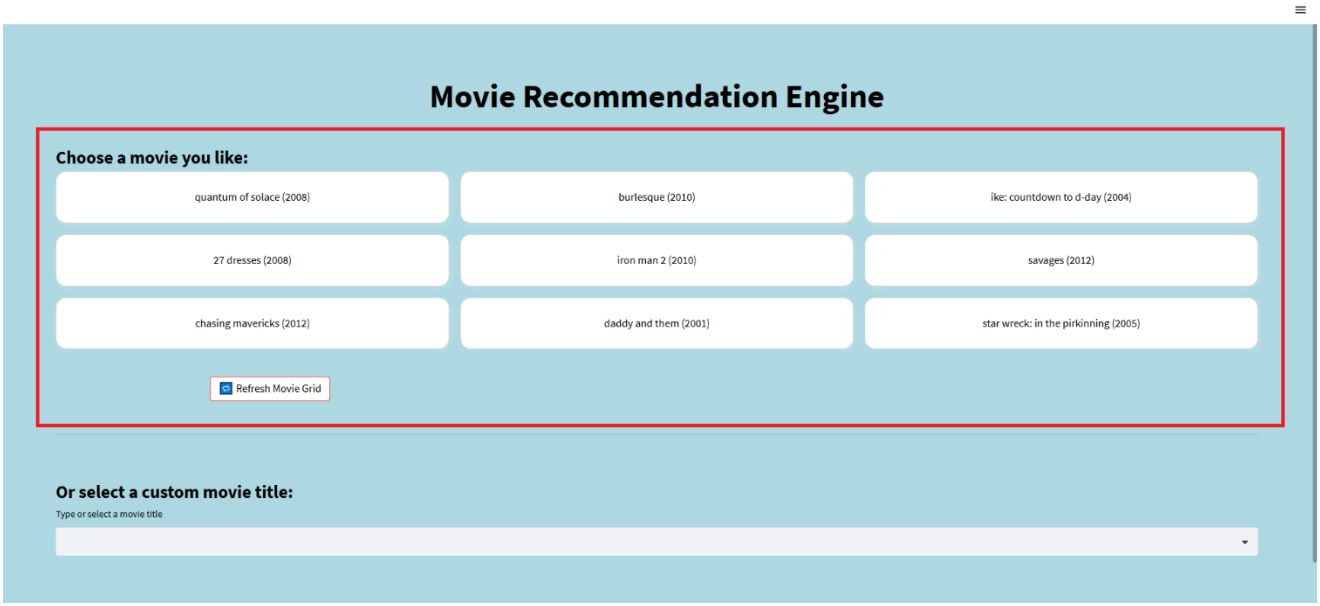

Figure 6 3x3 Movie Selection grid

- **Manual Movie Search**:
  Alternatively, users can manually search and select any movie from the database using a custom input text box.

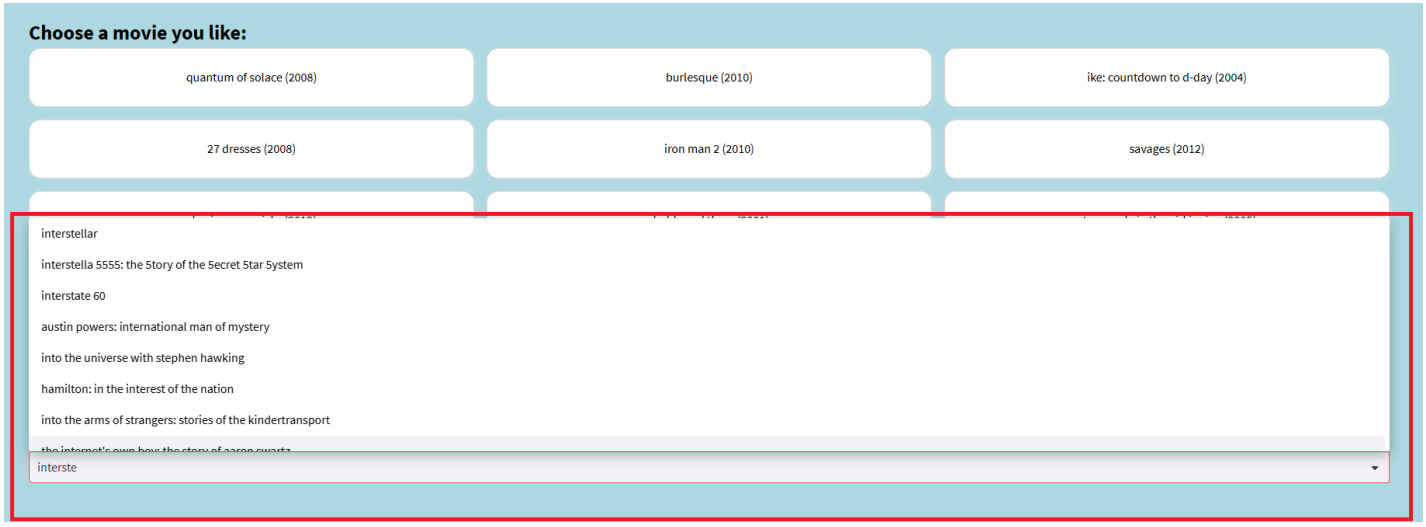

Figure 7 Search Box (With autofill feature)

- **Model Selection**:
  Users can choose the recommendation strategy by selecting among:
  - **K-Means Clustering**
  - **DBSCAN**
  - **Gaussian Mixture Model (GMM)**
  - **FastText Embeddings**

- **Number of Recommendations Slider**:
  A slider allows users to select how many movies they want to receive, offering flexibility based on user preference (minimum 5, maximum 15).

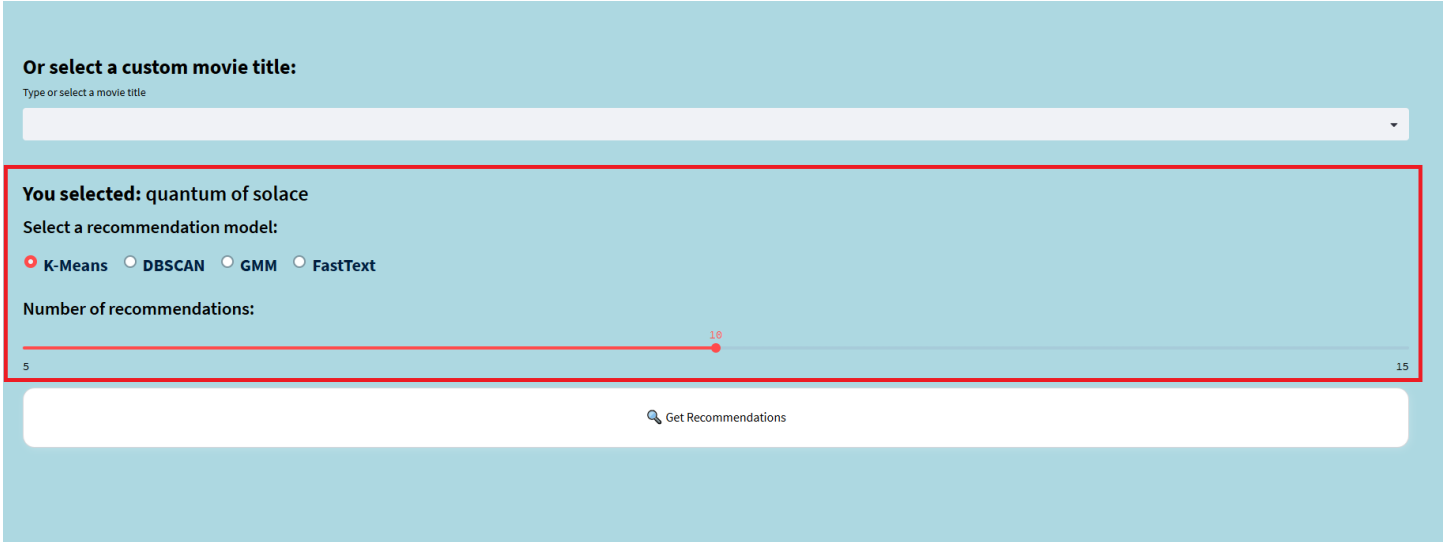

Figure 8 Model Selection

- **Dynamic Recommendations Display**:
  Upon selection, the tool displays recommended movies, including:
  - **Movie Title**
  - **Director**
  - **Genres**

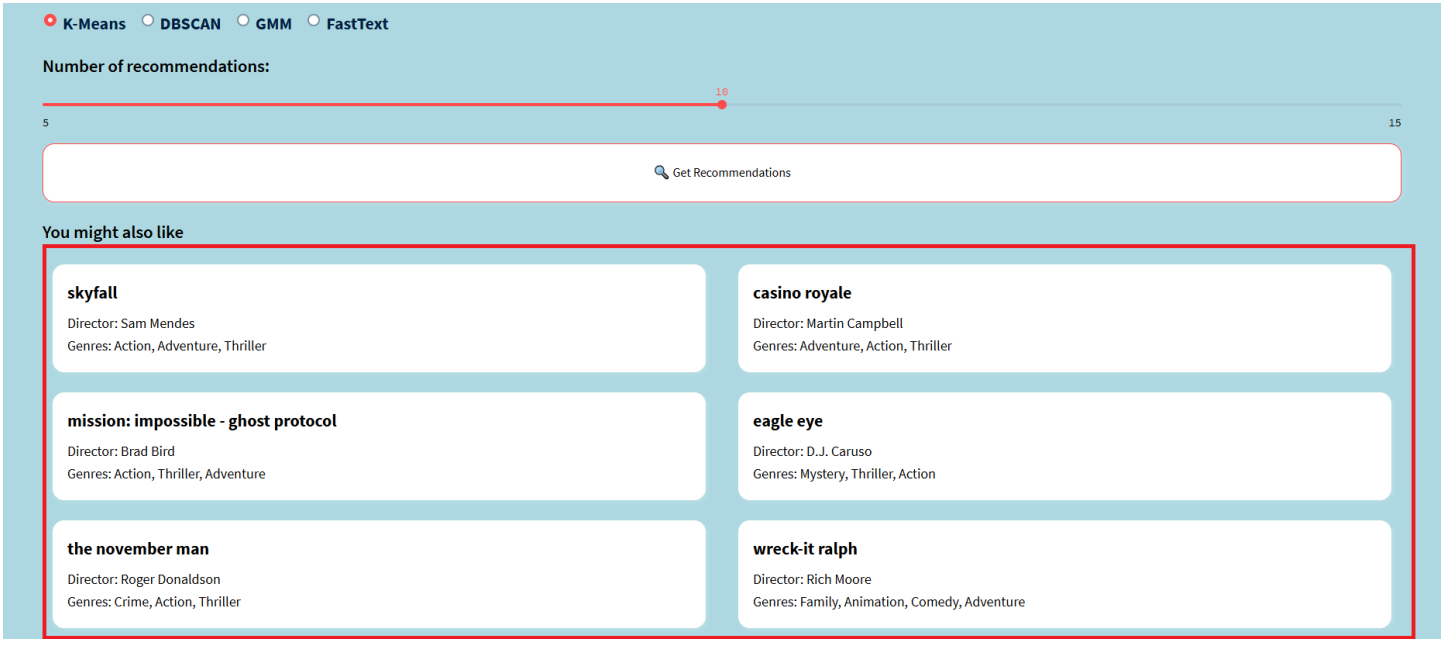

Figure 9 Movie Recommendations

- **Refresh Movie Grid Button**:
  Users can refresh the random grid to explore different starting options by clicking a dedicated **"Refresh Movie Grid"** button.
- **Cluster-Visualization Feature:**

To better understand the recommendations, the dashboard shows a “Cluster-Insight” scatter plot beside every recommendation list. The recommendations themselves are computed with cosine similarity in the full feature space; afterward, those same high-dimensional vectors are dropped into two PCA axes just for a quick visual. Because PCA flattens hundreds of dimensions into two, exact distances get warped—dots that look far apart on the plot may be close in reality, and vice versa. Think of the chart as a rough sketch of the cluster layout, not a ruler for measuring similarity.

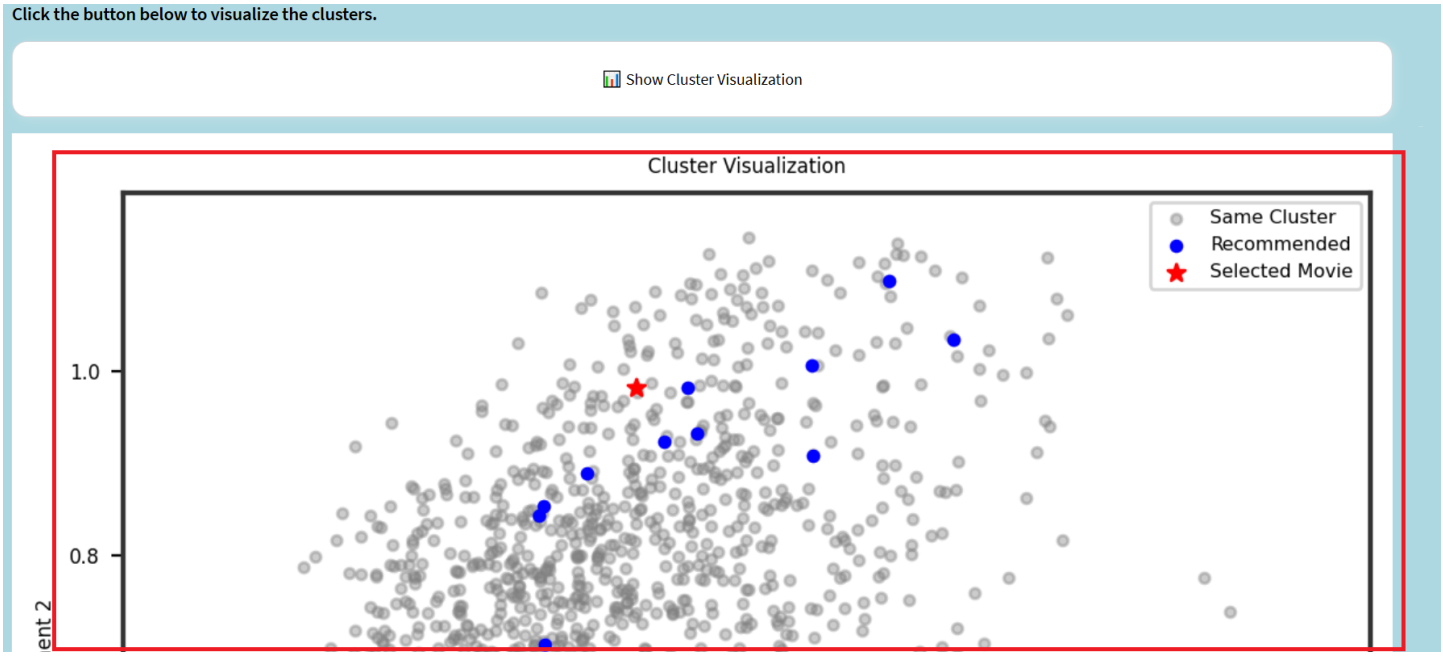

Figure 10 Cluster Visualization

```python
def get_recommendations_kmeans(title, df, title_to_index, combined_features, top_n=10):
    title = title.strip().lower()
    if title not in title_to_index:
        return f"Movie titled '{title}' not found."
    idx = title_to_index[title]
    cluster_label = df.loc[idx, 'cluster']
    cluster_movies = df[df['cluster'] == cluster_label].copy()
    cluster_movies = cluster_movies[cluster_movies.index != idx]
    cluster_indices = cluster_movies.index
    if cluster_movies.empty:
        return f"No other movies found in the same cluster as '{title}'."
    cos_sim = cosine_similarity(combined_features[idx].reshape(1, -1), combined_features[cluster_indices]).flatten()
    cluster_movies['similarity'] = cos_sim
    return cluster_movies.sort_values(by='similarity', ascending=False)[['title', 'director', 'genre_list', 'similarity']].head(top_n)

def get_recommendations_dbscan(title, df, title_to_index, combined_features, top_n=10):
    title = title.strip().lower()
    if title not in title_to_index:
        return f"Movie '{title}' not found."
    idx = title_to_index[title]
    target_cluster = df.loc[idx, 'dbscan_cluster']
    if target_cluster == -1:
        return f"Movie '{title}' is labeled as noise by DBSCAN and has no cluster."
    cluster_movies = df[df['dbscan_cluster'] == target_cluster].copy()
    cluster_movies = cluster_movies[cluster_movies.index != idx]
    cluster_indices = cluster_movies.index
    target_vec = combined_features[idx].reshape(1, -1)
    cos_sim = cosine_similarity(target_vec, combined_features[cluster_indices]).flatten()
    cluster_movies['similarity'] = cos_sim
    return cluster_movies.sort_values(by='similarity', ascending=False)[['title', 'director', 'genre_list', 'similarity']].head(top_n)
```

Figure 11 Code Snippet - Recommendations Function

4.3 User Flow

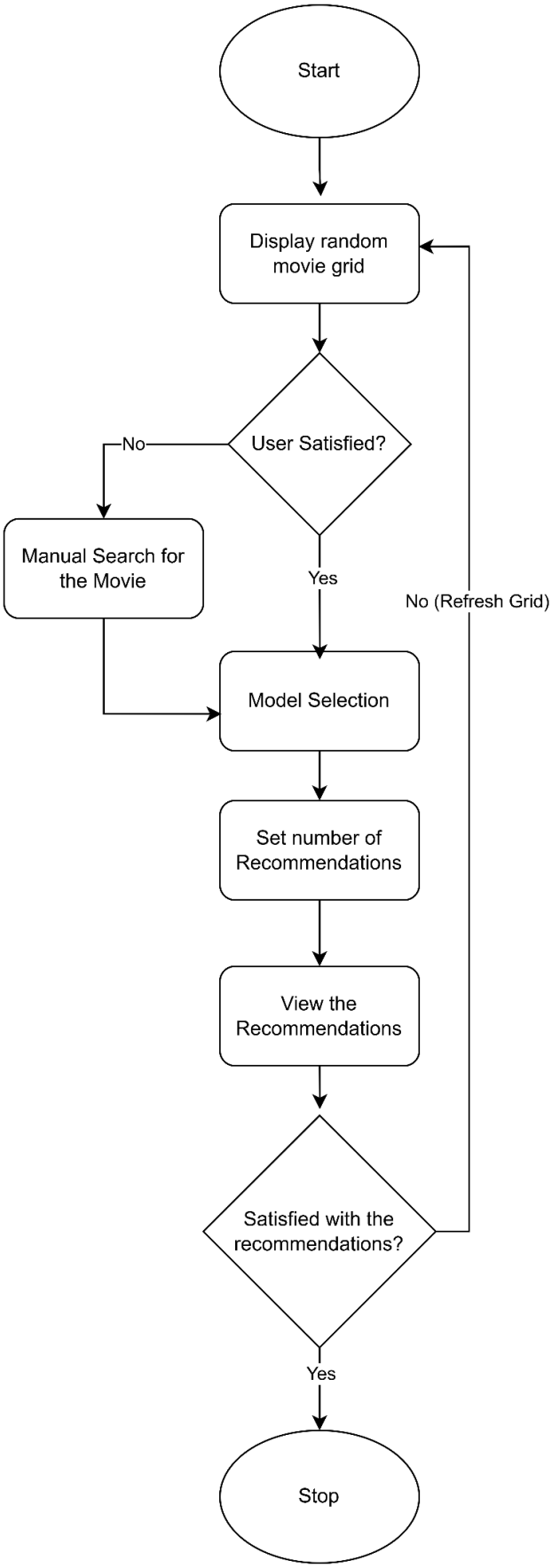

Figure 12 User Flow Diagram

The user journey through the Streamlit dashboard is designed to be intuitive and smooth:

1. **Landing and Selection**:
   Upon opening the application, the user is presented with a grid of **9 randomly selected movies**.
2. **Choosing Input**:
   The user can either:
   - **Select a movie** directly from the grid, or
   - **Search for a custom movie** using the text box provided.
3. **Model Selection**:
   After selecting a movie, the user chooses the desired recommendation model (**K-Means, DBSCAN, GMM, or FastText**) using radio buttons.
4. **Adjusting Output**:
   The user sets the **number of recommendations** using a slider control (between 5 to 15).
5. **Generating Recommendations**:
   Upon clicking **"Get Recommendations"**, the system displays a curated list of similar movies, including their **title**, **director**, and **genres**.
6. **Visualizing Cluster**:
   To better understand how the movies were recommended, the user can use a button to visualize the movie vectors and clusters through a scatter-plot.
7. **Exploring Further**:
   Users can **refresh the movie grid** at any time to discover new starting points or restart the recommendation process.

Section 5: Conclusion

5.1 Milestone 3 Conclusion

As part of the third milestone, I have confirmed the Movie recommendation engine's validity on three fronts. First, quantitative tests Silhouette, Davies–Bouldin, and Calinski–Harabasz made it clear where each model excels or falters. Second, ground-truth examinations translated those numbers into practical insight, showing DBSCAN's genre cohesion, K-Means' rigidity, GMM's exploratory breadth, and FastText's semantic reach based on these intrinsic and extrinsic metrics. I was able to identify the biases and limitations of each approach. Third, a Streamlit dashboard packaged the work for real users: pre-serialised models load in milliseconds, and an on-screen Cluster-Insight plot gives an immediate, qualitative explanation for every recommendation.

5.2 Overall Project Conclusion

The project progressed through three well-defined stages:

- **Milestone 1 – Data Preparation and Exploration**
  Comprehensive preprocessing unified six movie datasets, removed anomalies, normalised numerical attributes, and handled outliers. Exploratory analysis revealed core patterns from the data.
- **Milestone 2 – Feature Design and Model Selection**
  Domain-specific numeric features (Budget-Popularity, Movie-Age, OTT-Score) and rich text embeddings (TF-IDF, FastText) were engineered. Four unsupervised learning models, K-Means, DBSCAN, GMM, and a FastText cosine-similarity approach, were trained and chosen for further evaluation.
- **Milestone 3 – Evaluation, Bias Analysis, and Tool Development**
  Each model was rigorously benchmarked, its biases documented, and the entire pipeline was wrapped in an interactive Streamlit application that requires no re-training at run time.

The finished engine now combines flexible algorithm choices, transparent evaluation evidence, and an intuitive web interface. Future enhancements like metadata enrichment, automated hyperparameter tuning, and hybrid ranking can further improve both accuracy and diversity.