**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | Machine Learning for business & Data Visualisation Techniques |
| **Assessment Title:** | Integrated CA2 |
| **Lecturer Name:** | Sam Weiss & David McQuaid |
| **Student Full Name:** | Jason Liu Doyle |
| **Student Number:** | 2024200 |
| **Assessment Due Date:** | 4th of December 2024 |
| **Date of Submission:** | 4th of December 2024 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Table of Contents

[TABLE OF TABLES 1](#_Toc184229634)

[TABLE OF FIGURES 1](#_Toc184229635)

[MLB 2](#_Toc184229636)

[LOADING AND PREPERATION 2](#_Toc184229637)

[CONTENT-BASED FILTERING 3](#_Toc184229638)

[COLLABORATIVE BASED FILTERING 4](#_Toc184229639)

[CLUSTERING OUTLINE 4](#_Toc184229640)

[KNN 5](#_Toc184229641)

[HDBSCAN 9](#_Toc184229642)

[FUZZY-C MEANS 15](#_Toc184229643)

[RESULTS 19](#_Toc184229644)

[DASHBOARD 19](#_Toc184229645)

[REFERENCES 23](#_Toc184229646)

[Word Count – 2747 23](#_Toc184229647)

# TABLE OF TABLES

[Table 1: Train, Test Shape 2](#_Toc184153123)

[Table 2: Individual user score 2](#_Toc184153124)

[Table 3: Recommendation Generation Function 2](#_Toc184153125)

[Table 4: DBSCAN results 9](#_Toc184153126)

[Table 5: Fuzzy C Means results 15](#_Toc184153127)

# TABLE OF FIGURES

[Figure 1: Elbow method for PCA 5](#_Toc184153168)

[Figure 2: Elbow method for KNN 5](#_Toc184153169)

[Figure 3: Distribution of Clusters 6](#_Toc184153170)

[Figure 4: Genre Distribution of Clusters 6](#_Toc184153171)

[Figure 5: Top 3 Genres Proportion per Cluster 7](#_Toc184153172)

[Figure 6: Average rating by cluster 7](#_Toc184153173)

[Figure 7: Ratings by Genre 8](#_Toc184153174)

[Figure 8: Top movies for cluster 0 8](#_Toc184153175)

[Figure 9: Top movies for cluster 1 8](#_Toc184153176)

[Figure 10: Top movies for cluster 2 8](#_Toc184153177)

[Figure 11: Top movies for cluster 3 8](#_Toc184153178)

[Figure 12: Top movies for cluster 4 9](#_Toc184153179)

[Figure 13: HDBSCAN experiment results 10](#_Toc184153180)

[Figure 14: Cluster distribution 10](#_Toc184153181)

[Figure 15: Top Ten Cluster distribution Zoomed In 11](#_Toc184153182)

[Figure 16: Genre Distribution 11](#_Toc184153183)

[Figure 17: Genre Proportion 12](#_Toc184153184)

[Figure 18: Average Rating per cluster 12](#_Toc184153185)

[Figure 19: Ratings by genre 13](#_Toc184153186)

[Figure 20: Top movie titles for cluster 0 13](#_Toc184153187)

[Figure 21:Top movie titles for cluster 1 14](#_Toc184153188)

[Figure 22Top movie titles for cluster 2 14](#_Toc184153189)

[Figure 23: Top movie titles for cluster 3 14](#_Toc184153190)

[Figure 24: Top movie titles for cluster 4 14](#_Toc184153191)

[Figure 25: Top movie titles for cluster 5 15](#_Toc184153192)

[Figure 26: Distribution of Cluster 16](#_Toc184153193)

[Figure 27: Genre Distribution of Clusters 16](#_Toc184153194)

[Figure 28: Top 3 Genres per cluster 17](#_Toc184153195)

[Figure 29: Average Ratings per Cluster 17](#_Toc184153196)

[Figure 30: Ratings per cluster (1st half) 18](#_Toc184153197)

[Figure 31: Ratings per cluster (2nd half) 18](#_Toc184153198)

[Figure 32: Dashboard zoomed in 20](#_Toc184153199)

[Figure 33: Brainstorming layout 20](#_Toc184153200)

[Figure 34: Tab 1 Graph 1 21](#_Toc184153201)

[Figure 35: Tab 1 Graph 2 21](#_Toc184153202)

[Figure 36: Tab 1 Graph 3 21](#_Toc184153203)

[Figure 37: Tab 1 Graph 4 21](#_Toc184153204)

[Figure 38: Tab 1 cards 22](#_Toc184153205)

[Figure 39: Tab 2 22](#_Toc184153206)

# MLB

## LOADING AND PREPERATION

I started with detecting file encodings for CSV files using the chardet library for proper loading into pandas df’s (Python Software Foundation, 2024), the library had moderate confidence levels for encoding detection. For movies and tags files I applied detected encodings but there were loading errors. I fixed this by skipping problematic lines or just ignoring errors. The ratings file loaded no problem so no need for encoding detection. I then checked for successful loading by printing the first rows of each df.

I merged the three datasets (movies, ratings, tags) into a single combined df. Then checked for nulls and duplicates but found none. The final dataset was large with over 1.2 GB of memory, 17 million rows and nine features of different data types. This is the dataset used for the rest of the project.

## CONTENT-BASED FILTERING

For the content-based filtering part of the project I started by splitting the dataset into training and testing sets, with the training set having 13 million rows and the test set 1.3 million rows (Table 1).

|  |  |
| --- | --- |
| Train set shape: | (143679129) |
| Test set shape: | (35919799) |

Table 1: Train, Test Shape

I transformed the ratings data by centring the ratings around the dataset’s average to adjust for bias, this was for consistency and fairness in similarity calculations. The main features I picked were user IDs, movie IDs and ratings, because these numerical values are needed for calculating out the user-to-user similarity and predicting preferences.

Then I created a user-item matrix where rows were users, columns were movies and values were the adjusted ratings. I filled missing value with user-average ratings to make the matrix full for similarity scoring. I used this matrix to compute cosine similarity between users. To keep accuracy, the diagonal of the similarity matrix was set to zero to keep out self-similarity. I kept the similarity values in a separate df for easy reference. From the similarity matrix itself, I was able to identify the top 30 most similar users for each active user, making neighbourhoods of similar users for generating recommendations (Koren, Bell & Volinsky, 2009).

I then looked at shared movies between users and calculated out predicted ratings for test users based on their similarity to others. I needed to combine user ratings and generate predictions for comparison. The model’s performance was tested with Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), the results were a MAE of 0.7912 and RMSE of 1.8806, which is reasonable accuracy in predicting ratings. The MAE shows that predictions deviate by around 0.79 points on a 1-5 scale and the RMSE reflects slightly larger deviations because of the squared error term.

|  |  |
| --- | --- |
| MAE | 0.7911 |
| RMSE | 1.0806 |

Table 2: Individual user score

To get the recommendations I wrote a function to recommend the top-K items for test users. Recommendations were then ranked by predicted ratings and scores using precision and recall. But as you can see in Table 3, the results were poor with a precision of 0.145 and recall of 0.0006 (Resnick et al., 1994). The results show a strong limitation in identifying and ranking the most relevant items for users.

|  |  |
| --- | --- |
| Precision | 0.145 |
| Recall | 1.0006 |

Table 3: Recommendation Generation Function

The collaborative filtering model performed well in predicting ratings, it’s recommendation quality was weak showing a need for further improvement. In future iterations I would like to include content-based filtering features like genres or timestamps or release date to improve recommendation relevance. A hybrid approach mixing collaborative and content filtering could address data sparsity issues and improve the top-K recommendations. Also, including contextual data could improve the model’s effectiveness even more. Improving the feature engineering and using more advanced algorithms, the system’s recommendation quality and user satisfaction could go up by a lot.

## COLLABORATIVE BASED FILTERING

Although I have created a collaborative based filtering system in my notebook, the brief was later changed to include a discussion instead, so that is what I’ll include here.

Incorporating elements of content filtering into the recommender system could point out to a few limitations of collaborative filtering and improve it’s overall performance. Content filtering uses item attributes like genres, release dates or actors to recommend items like those the user has already liked. But a difference with collaborative filtering is it doesn’t rely on user interaction data which makes it less prone to issues like data sparsity or some ‘cold start’ problems for new users or items (Lops, de Gemmis & Semeraro, 2011).

Using features like movie genres or release years could help generate recommendations focused on user preferences for specific categories. This could balance out the poor results in my recommendation relevance seen in my collaborative filtering, especially since user similarity scores are weak. But content filtering has limitations as well like it’s inability to suggest diverse items or consider collaborative patterns like trends between similar users (Adomavicius & Tuzhilin, 2005).

A hybrid approach as I mentioned before could combine the strengths of the two, using collaborative filtering for more social based recommendations and content filtering then for more personalized suggestions. Having content based features would improve top-K recommendation relevance as well, address sparse data issues and also improve the system’s overall robustness.

## CLUSTERING OUTLINE

The objective of this clustering part of my project was to evaluate three clustering models, identify the best model using silhouette scores and gain insights into user behaviour. To manage computational efficiency I used a subsample of 1 million rows from the original 17 million on all the models, ones like DBSCAN and HDBSCAN scale poorly with large datasets which made subsampling essential. I did get KNN to handle the full dataset but subsampling was still applied uniformly across models to keep fairness (Rocke and Dai, 2003). A smaller subset of 100,000 rows was further used to work out silhouette scores for each model.

To improve the clustering performance, I used Principal Component Analysis (PCA) to reduce dimensionality as high dimensionality can negatively affect clustering due to the curse of dimensionality where distance metrics lose effectiveness. PCA transformed the data into 9 principal components, which I found with the elbow method, to retain 95% of the variance while filtering out noise (Figure 1). A graph of a graph showing the number of components

Description automatically generated

Figure 1: Elbow method for PCA

This improved computing efficiency and helped models like DBSCAN and HDBSCAN find more meaningful structures (Jolliffe and Cadima, 2016). By optimizing the dataset through PCA and subsampling, I made sure to get robust clustering outcomes with clear patterns and actionable insights.

### KNN

*Elbow Method Silhouette score + Cluster analysis*

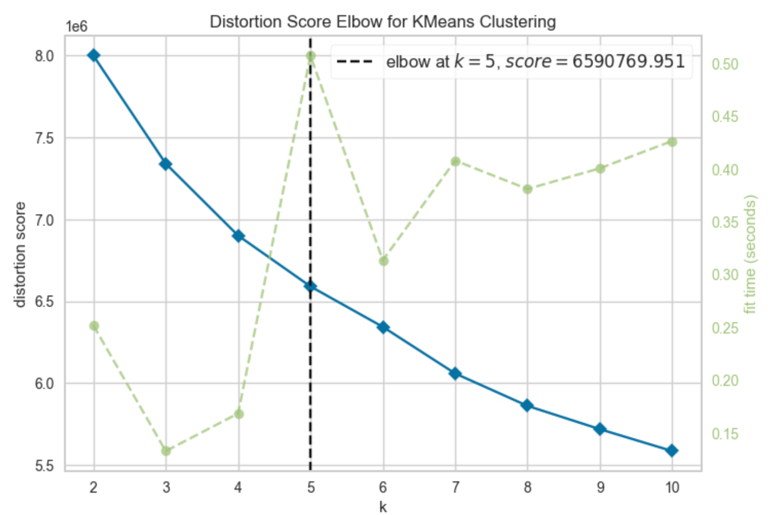
I started with K-Nearest Neighbors (KNN) clustering because of it’s simplicity and efficiency in handling larger datasets, particularly for finding patterns based on proximity in feature space (Zhang et al., 2017). KNN groups data points into clusters by giving each point to the cluster of it’s nearest neighbours, usually done by using Euclidean distance and then iteratively minimizes within cluster variance. To find the optimal number of clusters, I used the elbow method again (Fig. 2), which shows the point where additional clusters give diminishing returns (Thorndike, 1953).

Figure 2: Elbow method for KNN

The silhouette score I got was 0.096 which shows poorly separated or overlapping clusters, but the five clusters were relatively balanced, each having about 150,000–200,000 points (Fig 3). The clusters showed distinct genre preferences with cluster 0 focused on Action and thriller, cluster 1 on drama and comedy and cluster 2 heavily on comedy. Clusters 3 and 4 had a mix of adventure, drama, thriller and comedy (Fig 4-5.). Ratings varied slightly with clusters 1 and 3 having the highest ratings, while cluster 0 had the lowest (Fig 6). drama and romance consistently earned higher ratings while Action and horror showed wider variance.

I also saw more distinct movie preferences (Fig. 8-12), cluster 0 leaned more towards action and sci-fi classics like Fight Club and The Matrix while clusters 1 and 3 preferred critically acclaimed dramas like Pulp Fiction and The Shawshank Redemption. Cluster 2 focused more so on blockbusters like Inception and The Dark Knight and cluster 4 had a mix including The Lord of the Rings: The Fellowship of the Ring.

A graph of blue bars

Description automatically generated

Figure 3: Distribution of Clusters

A graph of different colored bars

Description automatically generated with medium confidence

Figure 4: Genre Distribution of Clusters

A graph of different colored squares

Description automatically generated

Figure 5: Top 3 Genres Proportion per Cluster

A bar graph with different colored rectangles

Description automatically generated

Figure 6: Average rating by cluster

A chart with multiple colored squares

Description automatically generated with medium confidence

Figure 7: Ratings by Genre

A graph with blue bars

Description automatically generated with medium confidence

Figure 8: Top movies for cluster 0

A graph with blue rectangular bars

Description automatically generated

Figure 9: Top movies for cluster 1

A graph with blue rectangular bars

Description automatically generated

Figure 10: Top movies for cluster 2

A graph with blue bars

Description automatically generated

Figure 11: Top movies for cluster 3

A graph with blue bars

Description automatically generated with medium confidence

Figure 12: Top movies for cluster 4

### HDBSCAN

*Silhouette score + Cluster analysis*

The next model I chose was DBSCAN (Density-Based Spatial Clustering of Applications with Noise), an algorithm that clusters points based on density which made it ideal for datasets with noise or clusters of different shapes and sizes (Ester et al., 1996). DBSCAN groups points based on a min density threshold, making clusters where points are closely packed together and labelling those in sparser areas as noise. I went with DBSCAN because it doesn’t need a predefined number of clusters, making it adaptable to the dataset’s structure. I used a k-distance plot to find that the optimal number of neighbours was 5. I found that DBSCAN struggled with the dataset due to its sensitivity to key parameters like EPS and Min Samples. Table 4 dhows how the model had a high number of noise points and fragmented clusters, showing me that the algorithm wasn’t able to effectively identify any meaningful groups.

To fix this then, I turned to HDBSCAN which is a hierarchical extension of DBSCAN that dynamically adjusts density thresholds to handle noise more effectively and find more stable clusters (McInnes et al., 2017). With HDBSCAN I also don’t need to manually tune parameters like EPS which makes it a more flexible choice for this dataset. Through experimentation (Table 5) I found ideal parameter ranges between min\_cluster\_size=150–200 and min\_samples=10–15. In my initial experiments I over segmented the data with 146 clusters with poor separation and a silhouette score of -0.279. A second experiment reduced the number of clusters to 6 and improved the silhouette score to -0.117. Finally then, the third experiment simplified the structure even more with 3 clusters having a positive silhouette score of 0.144. This did show improved separation but it might have over generalized the data, leaving room for further experimenting in future iterations.

|  |  |  |  |
| --- | --- | --- | --- |
| EPS | Min Samples | Number of Clusters | Number of noise points |
| 0.5 | 5 | 17098 | 679877 |
| 0.48 | 6 | 10848 | 802539 |
| 0.46 | 6 | 9842 | 847363 |
| 0.45 | 8 | 3616 | 934115 |
| 0.44 | 5 | 8473 | 885433 |
| 0.44 | 5 | 13725 | 831018 |
| 0.4 | 10 | 611 | 990121 |

Table 4: DBSCAN results

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment | 1 | 2 | 3 |
| min\_cluster\_size | 50 | 150 | 200 |
| min\_samples | 6 | 10 | 15 |
| Silhouette | -0.279 | -0.117 | 0.144 |
| Clusters | 146 | 6 | 2 |
| Noise Points | 317,054 | 274,579 | 269,936 |
| Top three cluster count | 637,003  7,366  3,747 | 635,028  89,344  389 | 632,401  97,396  267 |

Figure 13: HDBSCAN experiment results

The final clusters showed distinct patterns again in size, genres and ratings. Cluster 0 dominated with 635,000 points, cluster 5 the second-largest had 89,000 points, but the rest being very small in comparison (Fig 14-15). Clusters specialized in genres like thriller, action and drama, with cluster 5 showing strong genre specific preferences and cluster 0 having more balance (Fig16-19). Average ratings ranged from about 3.8 to 4.2, with cluster 2 having the highest rating of 4.2 (Fig 18). Top movies in clusters showed user preferences even more, with cluster 0 favouring critically acclaimed films like Pulp Fiction and Fight Club, while cluster 5 focused more so on visually stunning blockbusters like Inception and The Dark Knight (also directed by the same guy Nolan, and scored by Zimmer). These patterns give some actionable insights into user preferences and potential roads for personalized recommendations.

A graph with a bar graph

Description automatically generated with medium confidence

Figure 14: Cluster distribution

A graph of a bar chart

Description automatically generated with medium confidence

Figure 15: Top Ten Cluster distribution Zoomed In

A graph of different colored squares

Description automatically generated

Figure 16: Genre Distribution

A graph of different colored bars

Description automatically generated

Figure 17: Genre Proportion

A graph showing a number of bars

Description automatically generated

Figure 18: Average Rating per cluster

A screenshot of a graph

Description automatically generated

Figure 19: Ratings by genre

A graph with blue rectangles

Description automatically generated

Figure 20: Top movie titles for cluster 0

A graph with a number of text

Description automatically generated with medium confidence

Figure 21:Top movie titles for cluster 1

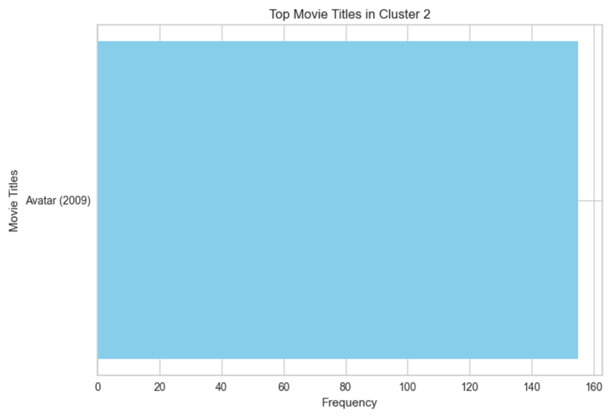


Figure 22Top movie titles for cluster 2

A graph with blue rectangles

Description automatically generated

Figure 23: Top movie titles for cluster 3

A graph with blue squares

Description automatically generated

Figure 24: Top movie titles for cluster 4

A graph with blue rectangles

Description automatically generated

Figure 25: Top movie titles for cluster 5

### 

### FUZZY-C MEANS

For my final model, I decided to go with Fuzzy C-Means (FCM) to try out soft clustering, which lets data points belong to multiple clusters with different membership degrees. This works for datasets with overlapping boundaries or muddy cluster assignments (Bezdek, 1981). FCM cuts the weighted sum of distances between points and cluster centres with weights being determined by membership probabilities which then leads to flexible and nuanced clustering (Ross, 2010).

To test out FCM I ran five experiments with varying parameters. In experiment 1 I used 10 clusters (c=10) and fuzziness (m=2) the silhouette score was -0.0121 showing poor separation. Increasing clusters to 15 in experiment 2 improved the score to 0.1201. Experiment 3 had c=15 and I reduced m=1.5 which improved the score to 0.1644. Experiment 4 had the highest score (0.1892) with c=20 and m=1.5 and experiment 5 had c=15 and m=2.5, showed reduced clarity (0.1024). I ended up going with c=20 and m=1.5 for my cluster analysis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Experiment | 1 | 2 | 3 | 4 | 5 |
| Params  error = 0.005  maxiter = 1000 init = None | c=10  m=2 | c=15  m=2 | c = 15  m = 1.5 | c=20  m = 1.5 | c = 15  m = 2.5 |
| Silhouette | -0.0121 | 0.1201 | 0.1644 | 0.1892 | 0.1024 |

Table 5: Fuzzy C Means results

The clusters again showed distinct preferences. Cluster 15 the largest with 175,000 points favoured ci-fi and adventure while cluster 11 had more variety. Cluster 0 leaned more so toward action and drama, and cluster 1 like thriller and mystery. Average ratings ranged from around 3.5 to 4.0, showing strong satisfaction. Genre based trends showed me thriller, sci-fi, and romance kept rating higher than genres like documentary and musical showing FCM’s ability to capture more nuanced user preferences.

A graph of different colored bars

Description automatically generated

Figure 26: Distribution of Cluster

A chart of different colored bars

Description automatically generated with medium confidence

Figure 27: Genre Distribution of Clusters

A graph of different colored bars

Description automatically generated

Figure 28: Top 3 Genres per cluster

A graph of a bar chart

Description automatically generated with medium confidence

Figure 29: Average Ratings per Cluster

A diagram of different colored bars

Description automatically generated with medium confidence

Figure 30: Ratings per cluster (1st half)

A diagram of different colored bars

Description automatically generated with medium confidence

Figure 31: Ratings per cluster (2nd half)

### RESULTS

In conclusion I effectively used and evaluated three clustering algorithms, KNN, HDBSCA and Fuzzy C-Means. I explored and generated some actionable insights from a df representing user behaviour and preferences.

Each algorithm had distinct benefits and limitations. KNN was simple and efficient but showed poor cluster separation, with a low silhouette score of 0.095. HDBSCAN identified fewer but more stable clusters with clear genre preferences but struggled with parameter sensitivity, with a best silhouette score of 0.144 after multiple iterations. Fuzzy C-Means had nicher insights with its soft clustering approach showing overlapping boundaries with the highest silhouette score of 0.189 with 20 clusters and a fuzziness parameter of 1.5 .

The results showed actionable insights with distinct audience preferences for genres and movies across clusters. For example clusters highlighted preferences for classics like Pulp Fiction and The Shawshank Redemption, blockbusters like Inception, and niche films like Old Boy. Average ratings across clusters remained high, indicating user satisfaction. These results show clustering can effectively capture user preferences and provide meaningful trends for business applications even on subsampled data. Overall, HDBSCAN and mainly Fuzzy C-Means came out as the most effective techniques for this analysis, showing robust and actionable insights while showcasing the power of clustering in a business context. Although further experimentation may be needed on larger samples of the dataset with further parameter tuning.

# DASHBOARD

For my dashboard, I chose a dark theme (#2D2D2D) with neon (#00CED1) and white (#FFFFFF) font colours to appeal to younger adults (Fig 32). The dark background reduces eye strain while the neon accents give a modern tech savvy aesthetic associated with younger demographics. This colour scheme adds to readability and draws the eye to insights giving an engaging user experience. I found that younger users prefer vibrant and visually striking designs that are more like popular apps and gaming interfaces (Norman, 2004) and this aesthetic keeps the dashboard both functional and visually appealing for our age group.

A screenshot of a movie dashboard

Description automatically generated

Figure 32: Dashboard zoomed in

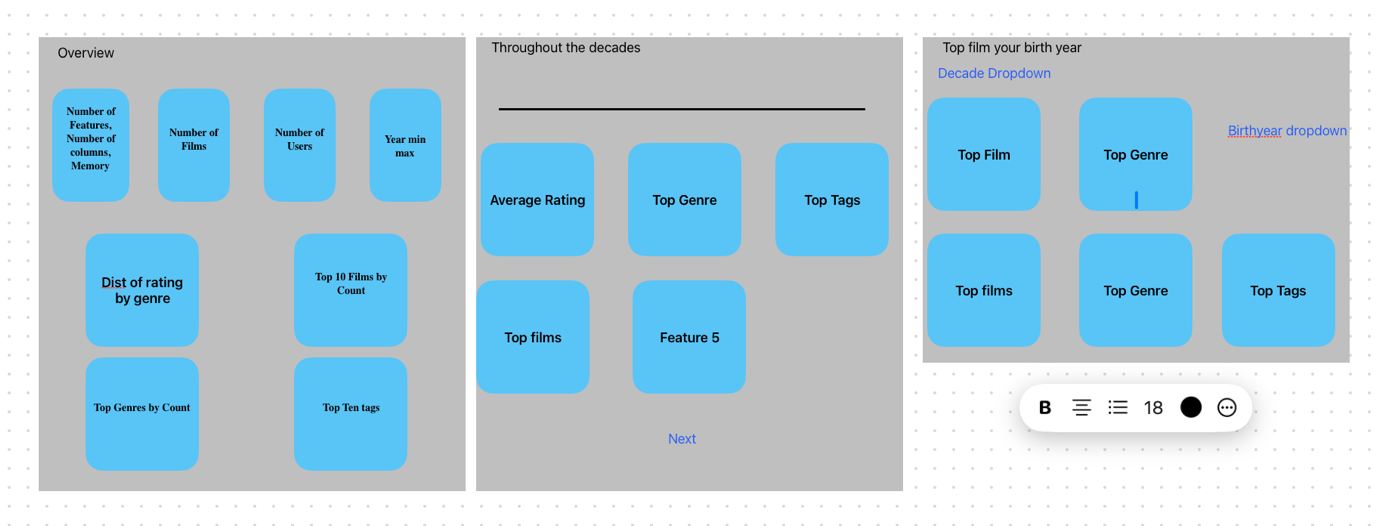
To create the layout I started by brainstorming different tabs and ideas using a freeform tool on my laptop. After looking at different possibilities I decided on the design in figure 33. But due to time constraints I decided to simplify the dashboard by focusing on two tabs instead of three. The first tab is the familiar “homepage/overview” showing key info in a static format. The second tab is the interactive one with animated graphs under the theme “Through the Decades,” using the popularity of nostalgia and historical trend analysis, which is particularly engaging for younger audiences like myself (Stern, 1992).

Figure 33: Brainstorming layout

For my first static tab I defined my four graphs (Fig 34-37). The first of the four is a box plot of ratings across genres showing variation in each category. The second is a bar chart of the number of movies per genre, the third ranks top rated films by the number of ratings, the fourth is also a bar chart of the most popular tags. Veridic gradients emphasise trends and make it colour blind accessible. For my four cards that I defined (fig 38), I made sure to use emojis and roboto sans serif to keep the modern design as emojis add an appealing, intuitive element which helps users quickly identify purpose and roboto offers clean readability. I got this idea from research showing the effectiveness of emojis in improving user engagement and how typography improves user experience (Cyr, 2013).

A graph with colorful squares

Description automatically generated with medium confidence

Figure 34: Tab 1 Graph 1

A graph showing a number of movies

Description automatically generated

Figure 35: Tab 1 Graph 2

A graph with numbers and a number of rats

Description automatically generated with medium confidence

Figure 36: Tab 1 Graph 3

A graph of a number of colored bars

Description automatically generated with medium confidence

Figure 37: Tab 1 Graph 4

A screenshot of a video game

Description automatically generated

Figure 38: Tab 1 cards

For tab 2 I created an app with a synced, animated dashboard to visualize decade based movie trends (Fig 39). The layout has a slider for selecting decades and interactive graphs showing average ratings, top genres, films, and tags for each decade. A “Next” button automates going through decades, and callback functions update the figures based on the selected decade, filtering data to keep the relevant insights with the timeline at the top of the app. Bar charts are used with the same styling, but the overall dark theme isn’t used as I was using this more so as a trial to try get the code right as getting the animations to sync up became really tricky.

A screenshot of a graph

Description automatically generated

Figure 39: Tab 2

I then combined the two processes of tab 1 and 2 into one big cell, and corrected the aesthetics to make sure it all was matching. Throughout this I also transformed the df by breaking down the genres column to analyse individual genres separately and worked out counts and percentages for films, tags, and genres. I also took release years from the titles, converted them into numbers and grouped them into decades. On top of that, I got the average ratings by decade and found the top three genres, films, and tags. These steps were crucial to creating the graphs.

# REFERENCES

## Word Count – 2747

1. Bezdek, J.C. (1981) Pattern recognition with fuzzy objective function algorithms. Chapter 2. New York: Springer.
2. Cyr, D. (2013) ‘Website design, trust and culture: An eight country investigation’, *Electronic Commerce Research and Applications*, 12(6), pp. 373–385. Available at: <https://doi.org/10.1016/j.elerap.2013.03.007>.
3. Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), pp. 226–231.
4. Jolliffe, I.T. and Cadima, J. (2016) ‘Principal component analysis: a review and recent developments’, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), pp. 20150202. Available at: https://doi.org/10.1098/rsta.2015.0202 (Accessed: 28 November 2024).
5. Koren, Y., Bell, R. & Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer, 42(8), pp.30-37.
6. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J., 1994. GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175-186). ACM.
7. McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density-based clustering. Journal of Open Source Software, 2(11), p. 205.
8. Norman, D.A. (2004) *Emotional Design: Why We Love (or Hate) Everyday Things*. New York: Basic Books.
9. Python Software Foundation, 2024. chardet: The Universal Character Encoding Detector. Available at: https://chardet.readthedocs.io [Accessed 27 Nov. 2024].
10. Ross, T.J. (2010) Fuzzy logic with engineering applications. 3rd edn. Chapter 1. Hoboken, NJ: Wiley.
11. Rocke, D.M. and Dai, J. (2003) ‘Sampling and subsampling for cluster analysis in data mining: With applications to sky survey data’, Data Mining and Knowledge Discovery, 7(3), pp. 215–232. Available at: https://helios2.mi.parisdescartes.fr/~lomn/Cours/DM/Material/Articles/Biologie/Skysurvey.pdf (Accessed: 28 November 2024).
12. Stern, B.B. (1992) ‘Historical and personal nostalgia in advertising text: The Fin de siècle effect’, Journal of Advertising, 21(4), pp. 11–22. Available at: <https://doi.org/10.1080/00913367.1992.10673382>.
13. Thorndike, R.L., 1953. Who belongs in the family?. Psychometrika, 18(4), pp.267-276. Available at: https://osf.io/fq6e9/download
14. Zhang, Z., Cao, L. and Wang, Y., 2017. Data Clustering Techniques. Springer.