# Movie Recommendation System for Netflix

Phase 4: Group 13

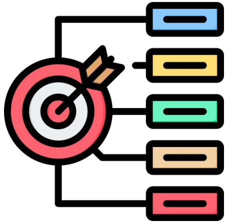
**Members:**
Sylvia Manono
Amos Kipngetich
Angela Maina
Charles Ndegwa
Sandra Koech
Gloria Tisnanga
Alex Miningwa

# Overview

## Problem

- Netflix is a global movie streaming platform.
- Due to a large movie catalogue, customers face difficulty in finding suitable content that matches their preferences.
- This leads to decision fatigue and lower engagement.

## Objectives



- To develop a personalized movie recommendation system.
- To address the cold start issue for new users by implementing content-based filtering and recommending movies based on user preferences.
- To enhance system precision and user feedback integration.

# Outline

- Business and Data Understanding
- Data Exploration
- Modelling Techniques
- Evaluation and Model Performance
- Conclusion
- Recommendations
- Next Steps

# 1.0 Business and Data Understanding

**Business Context:**

- Netflix, with its abundant options, often leaves users with decision fatigue.
- This leads to low engagement, which is detrimental to customer retention.
- A content-based recommendation system enhances Netflix's ability to deliver personalized content to users, boosting engagement levels.

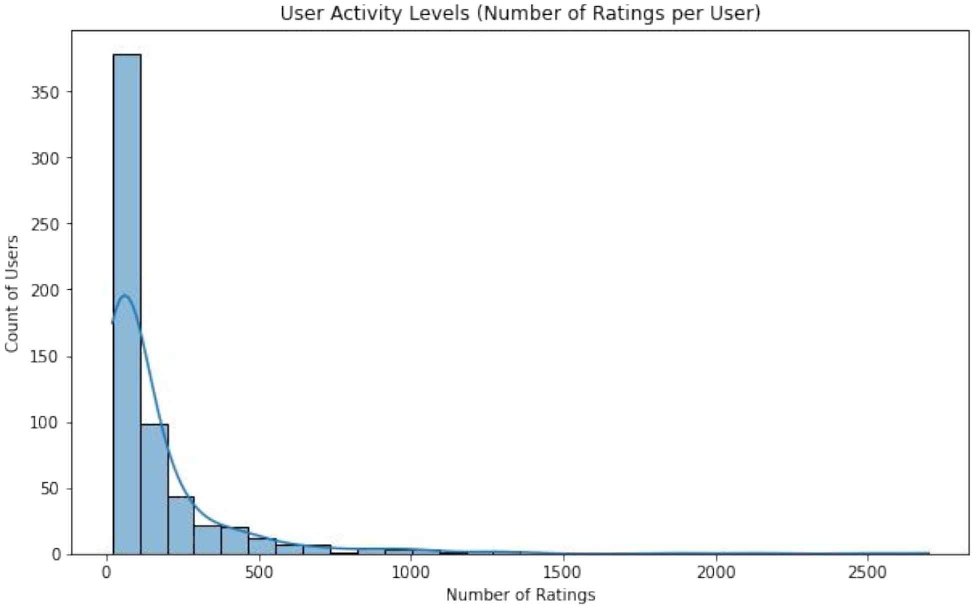
**Data Overview:**

- The dataset consists of **4 csv files.**
- Key features include movie titles, genres, user IDs, movie IDs and ratings.
- This data allows for both content-based and collaborative filtering approaches.

# 2.0 Data Exploration

1. **User Activity Levels: Which users are the most active, and how does their activity compare to less active users?**

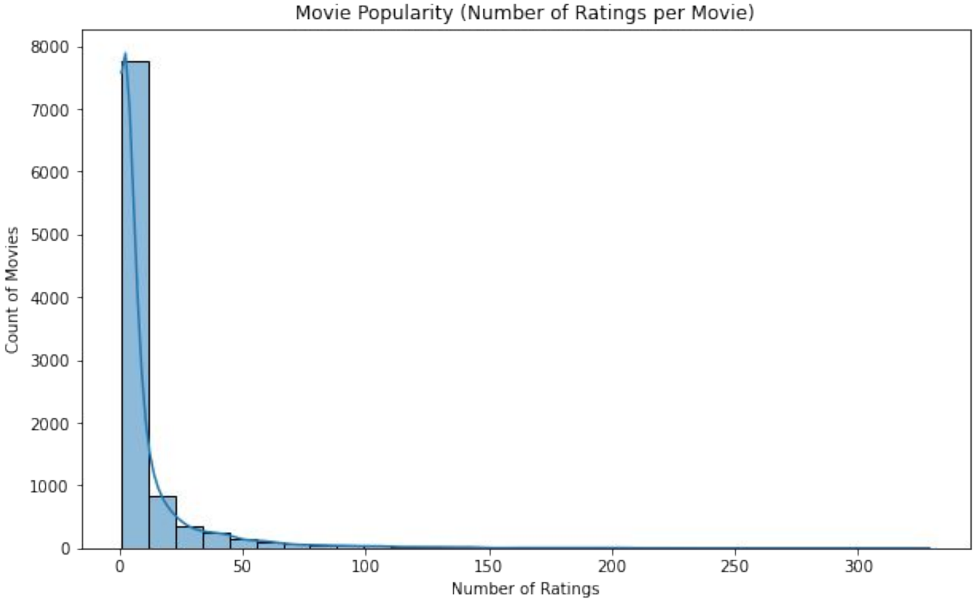
- **Most Active:** Most users have very few ratings, with a large majority rating fewer than 100 movies. However, there is a small group of highly active users who have rated over 500 movies.
- **Less Active Users:** The most active users contribute a disproportionate number of ratings than the average user. This indicates that while most users provide minimal input, a few power users contribute extensively to the rating dataset.



# 2.1 Data Exploration

2. **Item Popularity: Which movies are rated most frequently, and are there patterns in terms of genre or release year?**

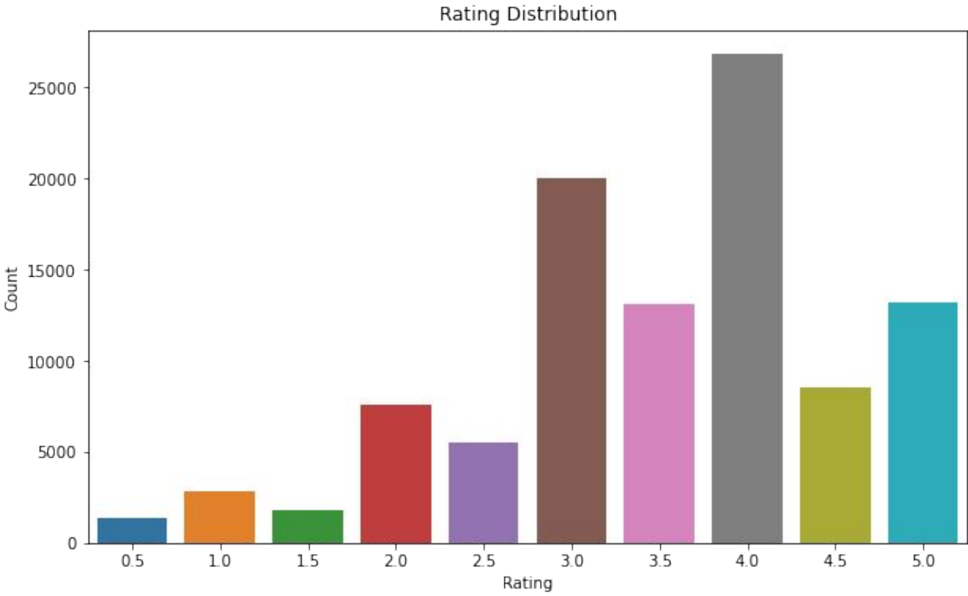
- **Most Frequent:** Most movies receive very few ratings, with a sharp drop-off after around 20 ratings per movie. However, a few movies are highly popular, receiving almost 100 ratings.
- **Patterns:** We can hypothesize that the most frequently rated movies are likely popular or mainstream films, possibly clustered around certain genres (e.g., blockbusters, action, comedy) or more recent release years when movie engagement is higher.



# 2.2 Data Exploration

3. **Rating Trends Over Time: How do average ratings and the number of ratings submitted change over time, and are there any seasonal or genre-specific trends?**

- **Rating Distribution:** Most ratings are clustered around 3.0 to 4.0, with a sharp increase at 4.0, indicating a tendency for users to give relatively high ratings. Very few ratings fall below 2.0.
- **Trends:** The plot suggests that users are more likely to give positive ratings. Seasonal or genre-specific trends could be analyzed by associating rating patterns with release dates.



# 3.0 Modelling Techniques

**Why classification?**

- Classification models help us predict whether a customer will stay with SyriaTel or leave (churn). Customers are sorted into two groups: those who are likely to stick around and those who might be at risk of leaving.
- Different models help to see which one can best predict customer behavior. This helps us understand which customers need attention and what actions SyriaTel can take to keep them.

# 3.1 Modelling Techniques

- We employed 3 collaborative filtering algorithms:

| # | Model | Scores | Description |
|---|---|---|---|
| 1 | KNNBasic | • RMSE = **1.1131**<br>• MAE = **1.0969** | • This is the baseline model. This algorithm uses a straightforward similarity-based approach, recommending items based on the preferences of users with similar tastes.<br>• KNN Basic does not adjust for user biases, leading to higher errors compared to KNN With Means, indicating less accurate predictions. |
| 2 | SVD (Singular Value Decompositio n) | • RMSE = **0.7342**<br>• MAE = **0.5546** | • Known for handling large, sparse datasets effectively, SVD uncovers hidden patterns (latent factors) in the way users interact with items, which improves its ability to make recommendations.<br>• The model scores as follows: SVD outperforms both KNN methods by achieving the lowest error rates, making it the most accurate model among the three. |

# 3.2 Modelling Techniques

- We employed 3 collaborative filtering algorithms:

| # | Model | Scores | Description |
|---|---|---|---|
| 3 | KNNWith Means | • RMSE = **0.8294**<br>• MAE = **0.7602** | • A more refined version of KNNBasic, this model adjusts predictions by considering the average ratings of users, which often leads to more accurate results.<br>• This algorithm performs well in predicting accurate ratings. |

# 3.3 Hyperparameter Tuning

- We evaluated RMSE, and MAE of algorithm KNNWithMeans on 5 split(s).
- Computing the Pearson similarity matrix:

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2748  1.1180  1.4731  1.4167  2.0000  1.4565  0.2983
MAE (testset)     1.2500  1.0000  1.2917  1.4167  2.0000  1.3917  0.3329
Fit time          0.00    0.00    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
KNNWithMeans Results: {'test_rmse': array([1.27475488, 1.11803399, 1.47313913, 1.41666667, 2.        ]),
```

- Computing the cosine similarity matrix:

```
MAE (testset)     0.3907  1.3316  1.9812  0.7075  0.0383  0.8899  0.6919
Fit time          0.00    0.00    0.00    0.00    0.01    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
SVD Results: {'test_rmse': array([0.39821108, 1.33441439, 2.07247333, 0.707491  , 0.03834371]),
```

# 3.4 Model 2: Evaluating KNNWithMeans Model After Tuning

- Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0607  1.7520  1.2022  0.1628  2.0000  1.2355  0.6377
MAE (testset)     0.7500  1.5833  1.0625  0.1628  2.0000  1.1117  0.6396
Fit time          0.00    0.01    0.00    0.00    0.01    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
KNNWithMeans Results:  {'test_rmse': array([1.06066017, 1.751983  , 1.2022115 , 0.16278261, 2.        ])
```

**Output after tuning:** The best parameters are as follows:

- **k = 10:** The optimal number of neighbors is 10.
- **sim_options:** Uses the Pearson correlation for similarity calculation between items rather than users. The Pearson similarity matrix was computed for each fold in the cross-validation process.

# 3.5 Model 3: Evaluating KNNBasic Model After Tuning

- Computing the coastline similarity matrix

| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Mean | Std |
|---|---|---|---|---|---|---|---|
| RMSE (testset) | 1.1859 | 1.1180 | 1.1180 | 1.2574 | 0.7523 | 1.0863 | 0.1748 |
| MAE (testset) | 1.1250 | 1.0000 | 1.0000 | 1.2574 | 0.7523 | 1.0269 | 0.1671 |
| Fit time | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Test time | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

```
KNNBasic Results:  {'test_rmse': array([1.18585412, 1.11803399, 1.11803399, 1.25735365, 0.75233043])
```

**Output after tuning:** The best parameters are as follows:

- **k = 10:** similar to KNNWithMeans.
- **sim_options:** Uses cosine similarity (instead of Pearson), also with user_based set to False (item-based similarity). **T**he similarity matrix was computed for each fold.

# 3.6 Model 4: Evaluating SVD Model After Tuning

- Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.3091  1.3643  0.7510  2.0550  1.2213  1.1401  0.5892
MAE (testset)     0.2648  1.3637  0.7305  2.0550  1.2213  1.1271  0.6044
Fit time          0.00    0.01    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
SVD Results:  {'test_rmse': array([0.3090748 , 1.36425977, 0.7509978 , 2.05502439, 1.22128296]),
```

# 4.0 Evaluation

- The KNNWIthMeans model was evaluated on 5 different splits (folds) of the data, reporting metrics for each fold.
- RMSE varies from **1.1180** to **2**, with an average of **1.4565** and a standard deviation of **0.2983.**
- MAE values range from **1** to **2**, with an average of **1.3917** and a standard deviation of **0.3329**.

# 4.1 Model Performance

- KNNWithMeans (KWM) generally performs **better** than KNNBasic based on the RMSE and MAE values across the folds.
- SVD results show varying RMSE and MAE values across folds, suggesting the model's performance is **inconsistent** across different data splits.
- KWM with **Pearson similarity** shows **moderate** performance, with an RMSE around 1.45.

# 4.2 Hybrid Recommendation Approach

**User-Movie pairing key takeaways:**

- **content_based_pred:** Ratings derived from the characteristics of movies, where 0 indicates no relevant features were found.
- **collab_pred:** Ratings based on user behavior and comparisons with similar users, showing a wide range of predicted values.
- **hybrid_pred:** The average of content-based and collaborative predictions, aiming for a balanced forecast.

| | userId | movieId | content_based_pred | collab_pred | hybrid_pred |
|---|---|---|---|---|---|
| 67037 | 432 | 77866 | 0.000000 | 0.091708 | 0.045854 |
| 42175 | 288 | 474 | 0.301741 | 2.026938 | 1.164340 |
| 93850 | 599 | 4351 | 0.376228 | 3.073397 | 1.724812 |
| 6187 | 42 | 2987 | 0.539974 | 3.378978 | 1.959476 |
| 12229 | 75 | 1610 | 0.307364 | 0.946193 | 0.626779 |
| ... | ... | ... | ... | ... | ... |
| 57416 | 380 | 5048 | 0.540689 | 1.945774 | 1.243231 |
| 67290 | 434 | 54272 | 0.000000 | 1.696063 | 0.848032 |
| 33423 | 226 | 5989 | 0.402719 | 4.177620 | 2.290170 |
| 98552 | 607 | 1320 | 0.163991 | 1.384325 | 0.774158 |
| 87803 | 567 | 750 | 0.650370 | 1.160052 | 0.905211 |

[20168 rows x 5 columns]

# 4.3 Model Validation and Deployment

**Print rmse_deployed: 1.1689114983876199**

- On average, the estimated ratings from the deployed model deviate from the actual ratings by approximately 1.17 units.
- This suggests that the predictions are relatively close to the actual ratings, a better performance compared to the RMSE of 2.5133 from the hybrid model.
- The plot shows a **strong positive correlation** between actual and estimated ratings, and tight clustering of data points.

# 5.0 Conclusion

- The movie recommendation system reveals promising results.
- **Deploy the model:** the refined approach effectively captures user preferences, contributing to a more personalized experience. The hexbin plot analysis further highlights the model's reliability and consistency in predicting user ratings.
- **Monitor and improve:** Regularly check how the model is performing and update it with new customer data to keep it accurate and relevant.
- **Customer engagement:** Use the insights from the model to develop new strategies to increase customer engagement, improving overall customer retention and satisfaction.

# 6.0 Recommendations

- **Refine the collaborative filtering approach:** Experiment with different matrix factorization techniques and tuning hyperparameters to further reduce RMSE and improve accuracy.
- **Leverage content-based filtering:** Use features such as genres, directors, and actors to recommend movies aligned with their preferences.
- **User feedback mechanisms:** Introduce a feedback loop that allows users to rate recommended movies. Use this data to refine the model continually.
- **Evaluate Additional Metrics:** In addition to RMSE, consider using metrics likePrecision@K, Recall, and F1 Score to assess the system’s effectiveness.
- **Continuous learning:** Explore advanced techniques like reinforcement learning or deep learning approaches to enhance the model’s ability.

# 7.0 Next Steps

- **Conduct A/B Testing:** Implement A/B testing for different recommendation strategies to determine the most effective model. This will provide insights into user engagement and satisfaction levels.
- **Regular model updates:** Regularly incorporate new data and user feedback to maintain the system's relevance and accuracy.
- **Expand data sources:** Use diverse data sources, e.g. social media trends to enrich the recommendation process and improve user engagement.
- **User interface enhancement:** Improve the user interface to make the recommendations more visible and accessible. Clear communication of why specific movies are recommended can enhance user trust and satisfaction.

# Thank You!