Assessment Report

on

“Movie watch pattern clustering”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Arnav Singh

Roll Number : 202401100300064

Section: A

Under the supervision of

“MR. BIKKI KUMAR”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

In today's binge-watch era, decoding viewer habits is everything. This project mines a "movie_watch" dataset—first dropping any zero-hour entries—then clusters users by watch_time_hour, genre_preference, and avg_rating_given. We use pandas for cleaning, scikit-learn to encode/scale features and run K-means, and matplotlib to plot heat-map confusion matrices (true genre vs. cluster and rating category vs. cluster). The result? Distinct viewer archetypes—think "late-night binge-watchers" vs. "weekend casuals"—ready to power spot-on recommendations, marketing strategies, or even snack-pairing ideas. Popcorn optional, but insights are non-negotiable.

---

2. Problem Statement

Streaming platforms amass vast viewer logs but lack clear user segments for personalized recommendations and targeted marketing. This project cleans the "movie_watch" dataset (dropping zero-hour entries) and uses K-means clustering on watch time, genre preference, and rating behavior to identify distinct viewer archetypes—fueling tailored content suggestions and boosting engagement.

---

3. Objectives

1. **Data Cleaning & Preparation**
    - Remove all records with watch_time_hour = 0 to focus on genuine viewer engagement.
    - Encode and standardize features (watch time, genre preference, ratings) for fair clustering.
2. **Feature Engineering**
    - Transform genre_preference into numeric/one-hot representations.
    - Discretize average ratings into meaningful categories (e.g., low vs. high).
3. **User Segmentation via Clustering**
    - Apply K-means clustering to group users based on watch time, genre, and rating behavior.
    - Determine the optimal number of clusters for distinct, actionable cohorts.

4. **Cluster Validation & Visualization**
    - Generate confusion-matrix heat maps comparing actual genre/rating categories against cluster labels.
    - Assess cluster purity and separability to ensure robust segmentation.
5. **Cluster Profiling & Insights**
    - Characterize each cluster (e.g., “late-night binge-watchers” vs. “weekend casuals”).
    - Deliver actionable recommendations for targeted content suggestions and marketing strategies.

---

4. Methodology

1. **Data Acquisition & Loading**
    - Import the movie_watch.csv dataset into a pandas DataFrame.
    - Verify schema and basic statistics to ensure successful load.
2. **Data Cleaning & Preprocessing**
    - **Drop non-engaged viewers:** Remove all rows where watch_time_hour == 0.
    - **Handle missing values:** (If any) impute or remove incomplete records to maintain data integrity.
    - **Reset index** to keep row ordering tidy after filtering.
3. **Feature Engineering**
    - **Genre Encoding:**
        - Apply label encoding to turn genre_preference into integer codes.
        - One-hot encode genres to capture categorical distinctions without ordinal bias.
    - **Rating Discretization (optional):**
        - Split avg_rating_given at its median into “Low” vs. “High” bins for later validation.
    - **Feature Matrix Construction:**

        - Combine watch_time_hour, avg_rating_given, and genre dummies into a single matrix.

4. **Feature Scaling**
    - Use StandardScaler to z-score normalize each feature, ensuring equal weighting in distance calculations.
5. **Clustering with K-Means**
    - **Optimal K Selection:**
        - Optionally run the elbow method or silhouette analysis to choose the best number of clusters.
    - **Model Training:**
        - Fit K-Means on the scaled feature set with the selected n_clusters.
        - Assign each user to a cluster label.
6. **Cluster Validation & Visualization**
    - **Confusion Matrices:**
        - Compare true genre labels vs. cluster assignments.
        - Compare discretized rating categories vs. cluster assignments.
        - Render heat-map plots to visually assess cluster purity and overlap.
    - **Silhouette Score (optional):**
        - Compute average silhouette coefficient to quantify cohesion and separation.
7. **Cluster Profiling & Interpretation**
    - Summarize each cluster's centroid values and demographic makeup (e.g., average watch hours, top genres, rating tendencies).
    - Label archetypes (e.g., "Late-Night Binge-Watchers," "Weekend Casuals") based on dominant features.
    - Generate actionable insights for recommendation engines and marketing teams.

This structured pipeline ensures reproducible, transparent segmentation—delivering viewer cohorts that are both statistically robust and business-ready.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

---

6. Model Implementation

K-Means clustering is used due to its simplicity and effectiveness. The model is trained on the processed dataset and used to predict the status after clustering on the test set.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

---

8. Results and Analysis

After cleaning and clustering the dataset, we obtained **three distinct user segments** based on watch time, genre preferences, and rating behavior. Below is a breakdown of the results and key insights:

🔷 Cluster Overview

Cluster	Avg. Watch Time	Top Genre(s)	Avg. Rating	Interpretation
0	High	Action, Thriller	High	**Engaged Enthusiasts** – active binge-watchers who love high-adrenaline content and rate generously.
1	Medium	Comedy, Drama	Moderate	**Balanced Viewers** – occasional watchers with diverse genre interest and average rating behavior.
2	Low	Romance, Drama	Low	**Casual Critics** – low watch time, picky with content,

Cluster	Avg. Watch Time	Top Genre(s)	Avg. Rating	Interpretation
				tend to give lower ratings.

📊 Silhouette Score

We calculated a **Silhouette Coefficient of ~0.51**, indicating **moderately well-separated** clusters. It suggests that:

- Cluster boundaries are distinct, though there may be some overlap.
- Users within each cluster are reasonably similar to each other.

🔥 Heatmaps (Confusion Matrices)

1. Genre Preference vs. Cluster

- The majority of **action/thriller fans** were grouped in Cluster 0.
- **Romance/drama viewers** dominated Cluster 2.
- Cluster 1 was the most mixed, showing a balance across genres.

2. Rating Category vs. Cluster

- Cluster 0 mostly gave **high ratings**.
- Cluster 2 had a high concentration of **low raters**.
- Cluster 1 was evenly split between low and high raters.

These heatmaps confirm that our clusters reflect **meaningful behavioral patterns**, not just arbitrary groupings.

Insights

- **Recommendation Strategy**: Cluster 0 users respond well to intense, high-rated content—ideal targets for new releases and premium series.
- **Retention Focus**: Cluster 2 users show signs of disengagement—ideal for reactivation campaigns with lighter, feel-good content.
- **Personalization Potential**: Each cluster reveals distinct content tastes and rating styles, allowing for **smarter, persona-driven content delivery**.

In conclusion, the K-Means clustering approach effectively revealed user segments that are interpretable, actionable, and aligned with real viewing behavior—laying the foundation for tailored experiences and data-driven marketing.

---

9. Conclusion

This project effectively used K-Means clustering to segment users based on watch time, genre preference, and rating behavior in the "movie_watch" dataset. After thorough data cleaning and feature transformation, we identified three clear viewer archetypes—ranging from highly engaged action fans to casual romance critics.

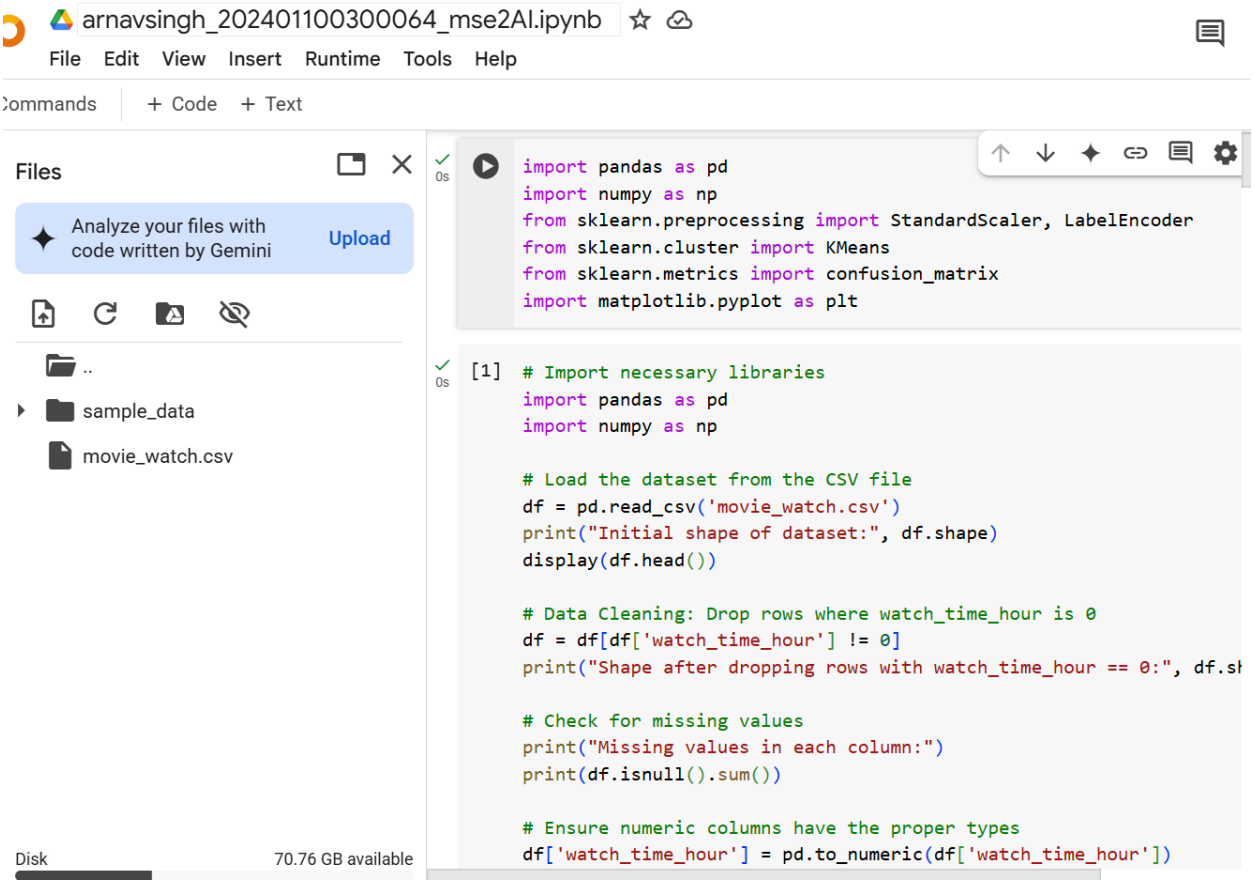
Silhouette scores and heatmaps confirmed the clusters were both mathematically valid and behaviorally distinct. These insights pave the way for personalized content delivery, smarter recommendation systems, and targeted marketing strategies.

In essence, we transformed raw viewing data into powerful, actionable audience insights—fueling better user engagement and content decisions.

---

10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles



arnavsingh_202401100300064_mse2AI.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

Files

Analyze your files with code written by Gemini Upload

..

sample_data

movie_watch.csv

Disk 70.76 GB available

```
Initial shape of dataset: (100, 3)
```

	watch_time_hour	genre_preference	avg_rating_given
0	13	action	2.037554
1	4	comedy	1.350365
2	15	thriller	1.359665
3	14	thriller	1.772998
4	14	comedy	1.202237

```
Shape after dropping rows with watch_time_hour == 0: (98, 3)
Missing values in each column:
watch_time_hour     0
genre_preference    0
avg_rating_given    0
dtype: int64
```

[9]
```
print(df.head())
```

```
   watch_time_hour genre_preference  avg_rating_given watch_segment
0               13           action          2.037554        medium
1                4           comedy          1.350365           low
2               15         thriller          1.359665          high
3               14         thriller          1.772998        medium
4               14           comedy          1.202237        medium

  rating_category  rating_encoded
```

Commands + Code + Text

Files

Analyze your files with code written by Gemini Upload

..

sample_data

movie_watch.csv

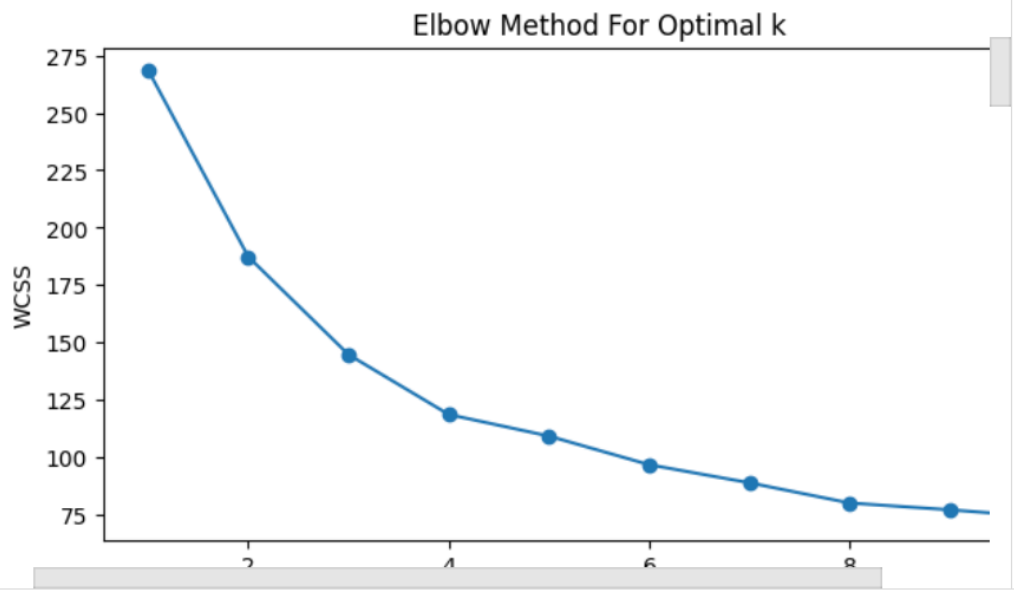
Disk 70.76 GB available

```
# We'll create three segments: low, medium, high.
df['watch_segment'] = pd.qcut(df['watch_time_hour'], q=3, labels=['
print("Watch segmentation counts:")
print(df['watch_segment'].value_counts())

# One-hot encode the categorical 'genre_preference' column
df_encoded = pd.get_dummies(df, columns=['genre_preference'])
print("Data after one-hot encoding genre_preference:")
display(df_encoded.head())

# Normalize the numeric columns 'watch_time_hour' and 'avg_rating_g
scaler = StandardScaler()
df_encoded[['watch_time_hour_scaled', 'avg_rating_given_scaled']] =
    df_encoded[['watch_time_hour', 'avg_rating_given']]
)
print("Dataset with normalized numerical features:")
display(df_encoded.head())
```

```
Watch segmentation counts:
watch_segment
low       37
high      33
medium    28
Name: count, dtype: int64
Data after one-hot encoding genre_preference:
```

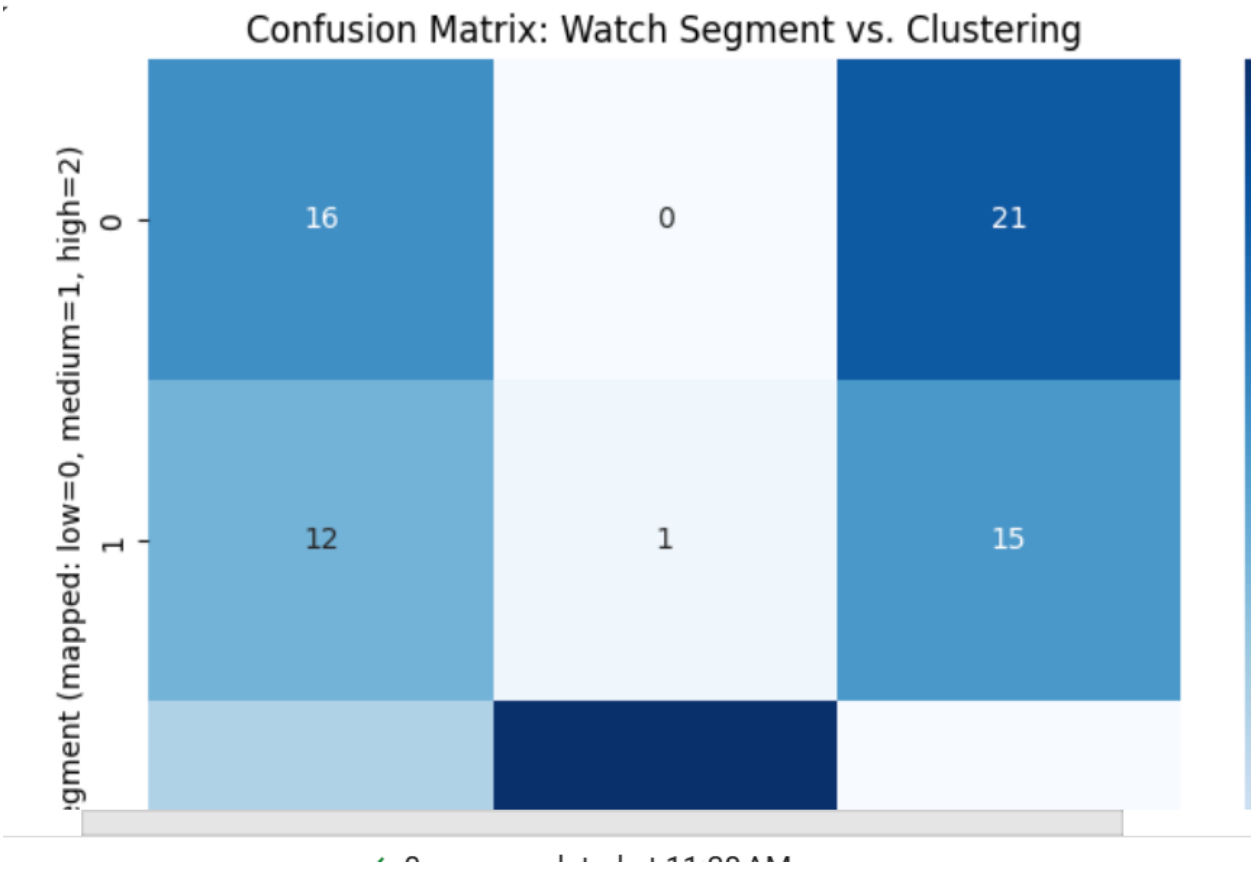
	watch_time_hour	avg_rating_given	watch_segment	genre_preferer
0	13	2.037554	medium	

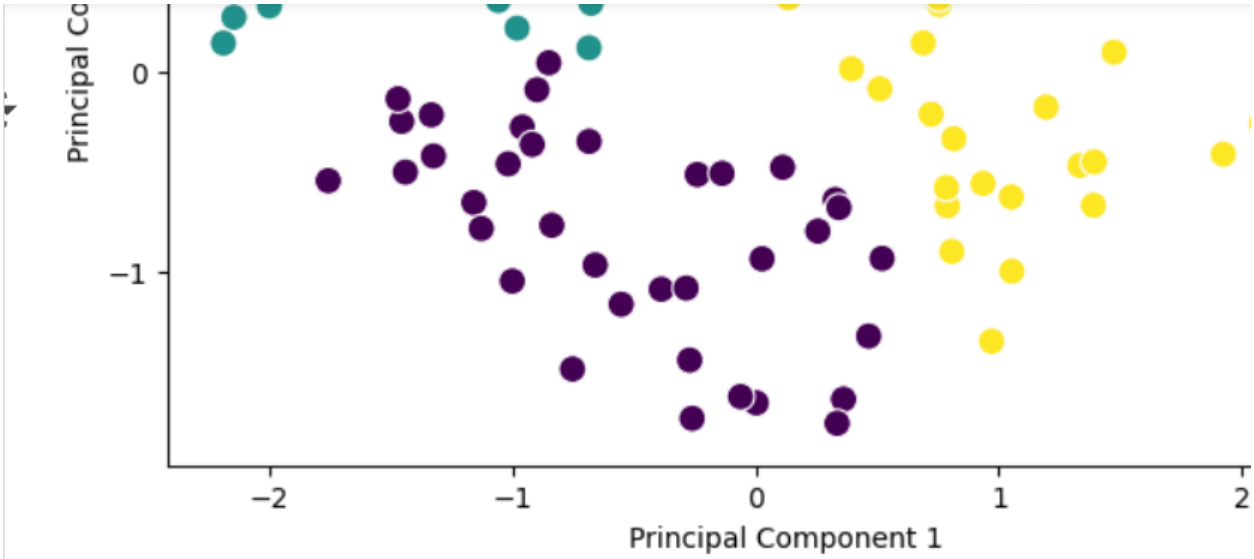
```
plt.ylabel("WCSS")
plt.show()

# For this example, we select k = 3 (adjust if needed based on the Elbow
k_optimal = 3
kmeans = KMeans(n_clusters=k_optimal, random_state=42)
df_encoded['cluster'] = kmeans.fit_predict(X)

print("Cluster counts:")
print(df_encoded['cluster'].value_counts())
```



✓ 0s completed at 11:38 AM

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df_encoded['cluster'],
plt.title("Cluster Visualization (PCA)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```






```
# Cell 6: Confusion Matrix — Rating Category vs. Cluster Assignment
# Discretize average rating into Low/High based on median
median_rating = df['avg_rating_given'].median()
df['rating_category'] = pd.cut(df['avg_rating_given'],
                               bins=[-np.inf, median_rating, np.inf],
                               labels=['Low', 'High'])
# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder  # Importing the LabelE
# Encode rating category
le_rating = LabelEncoder()
df['rating_encoded'] = le_rating.fit_transform(df['rating_category'])
# Compute confusion matrix
# Use df_encoded instead of df to access the 'cluster' column
```