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 : Praveer Singh

Roll Number : 202401100300181

Section: C

1. Introduction

This project analyzes user behavior in movie-watching habits by clustering viewers based on three key features:

1. **Time of Watching** – The hour of the day when users watch movies (0-23).
2. **Genre Preference** – The most frequently watched genre (action, comedy, drama, thriller).
3. **Rating Behavior** – The average rating given by users (1-5 scale

2. Problem Statement

Movie Watch Pattern Clustering

Cluster users based on time of watching, genre preference, and rating behavior.

4. Methodoloy

. Data Understanding

We started with a dataset containing:

- watch_time_hour: The hour of the day when a movie was watched.
- genre_preference: The user's preferred genre (text).
- avg_rating_given: Average rating the user gives to movies.

---

a. Convert watch_time_hour to Time Blocks

To make analysis more meaningful, the 24-hour format is grouped into:

- **Morning** (5–11)
- **Afternoon** (12–17)
- **Evening** (18–22)
- **Night** (23–4)

This captures *viewing behavior* better than using raw hour values.

b. Encode Categorical Features

- genre_preference and time_block are categorical.
- Used **Label Encoding** to convert them into numeric format for clustering.

c. Scale Features

- Standardized the features (mean = 0, std = 1) using StandardScaler.
- KMeans is distance-based, so scaling ensures fair contribution from all features.

---

3. Clustering (KMeans)

- Applied **KMeans Clustering** with 3 clusters (n_clusters=3) to group users.
- The algorithm tries to minimize intra-cluster variance and maximize inter-cluster separation.

---

4. Dimensionality Reduction for Visualization (PCA)

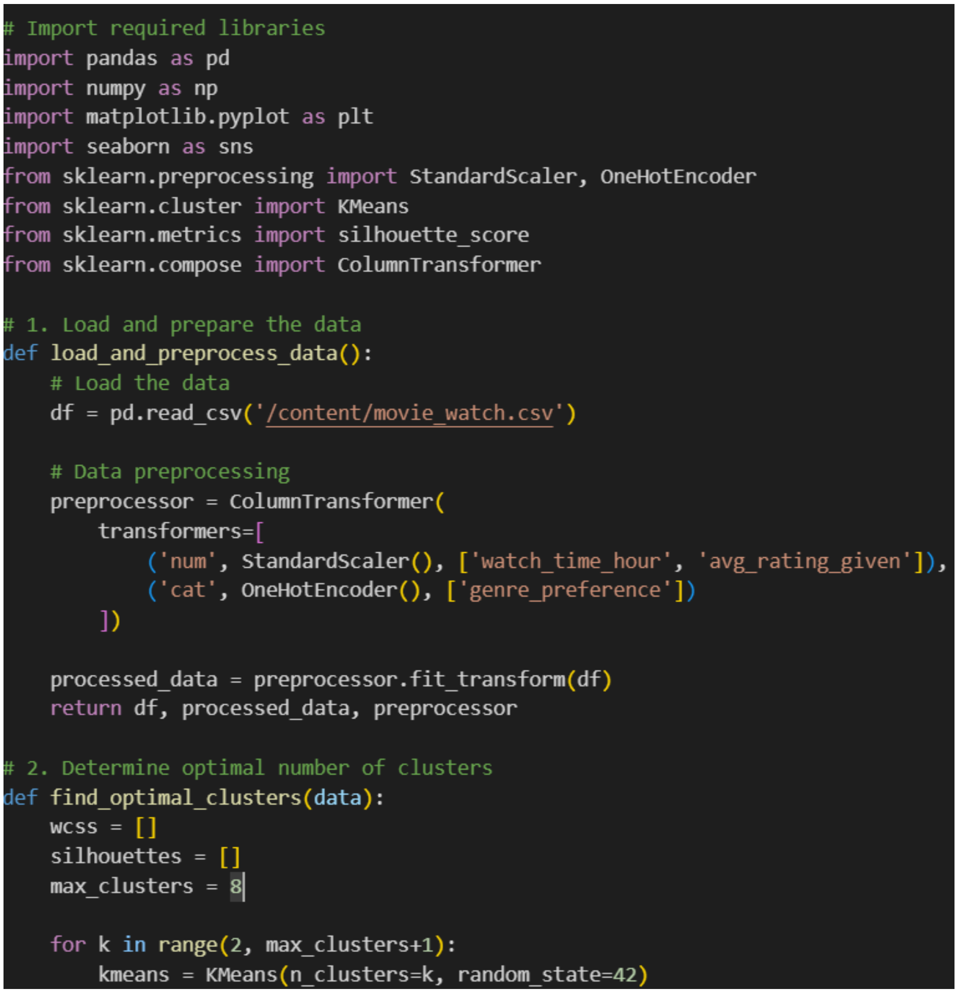
- Since we had 3 features, we used **PCA (Principal Component Analysis)** to reduce them to 2D.
- This helps visualize the clusters clearly in a 2D scatter plot.

---

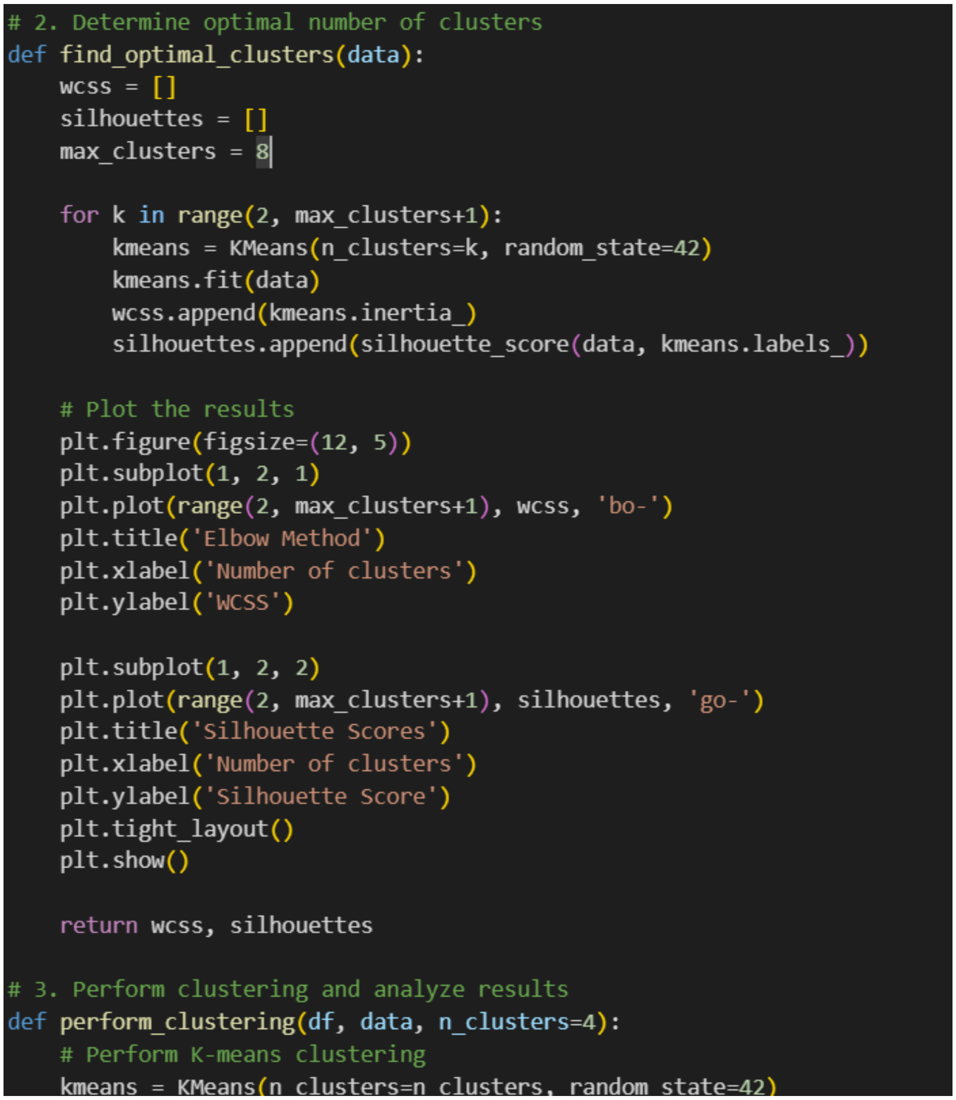
5. Visualization

- Used **Seaborn** to create a colored scatter plot showing clusters.
- Each color represents a group of

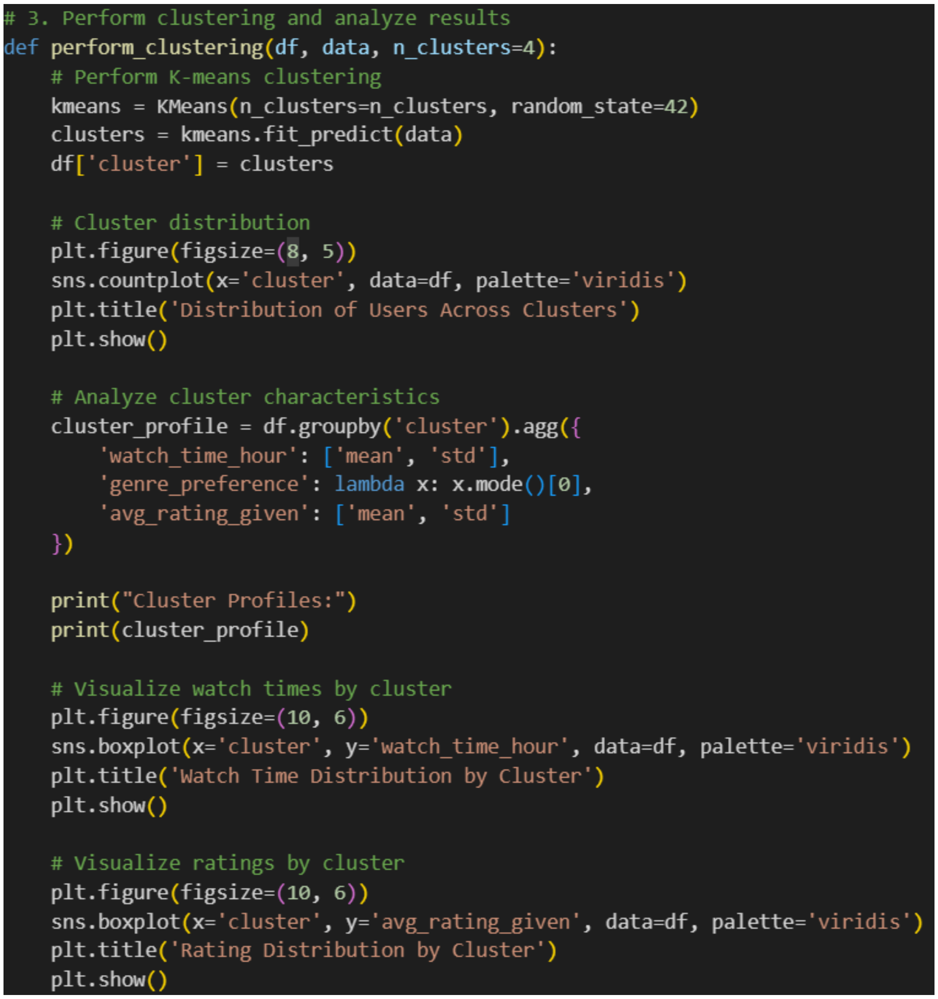
CODE:



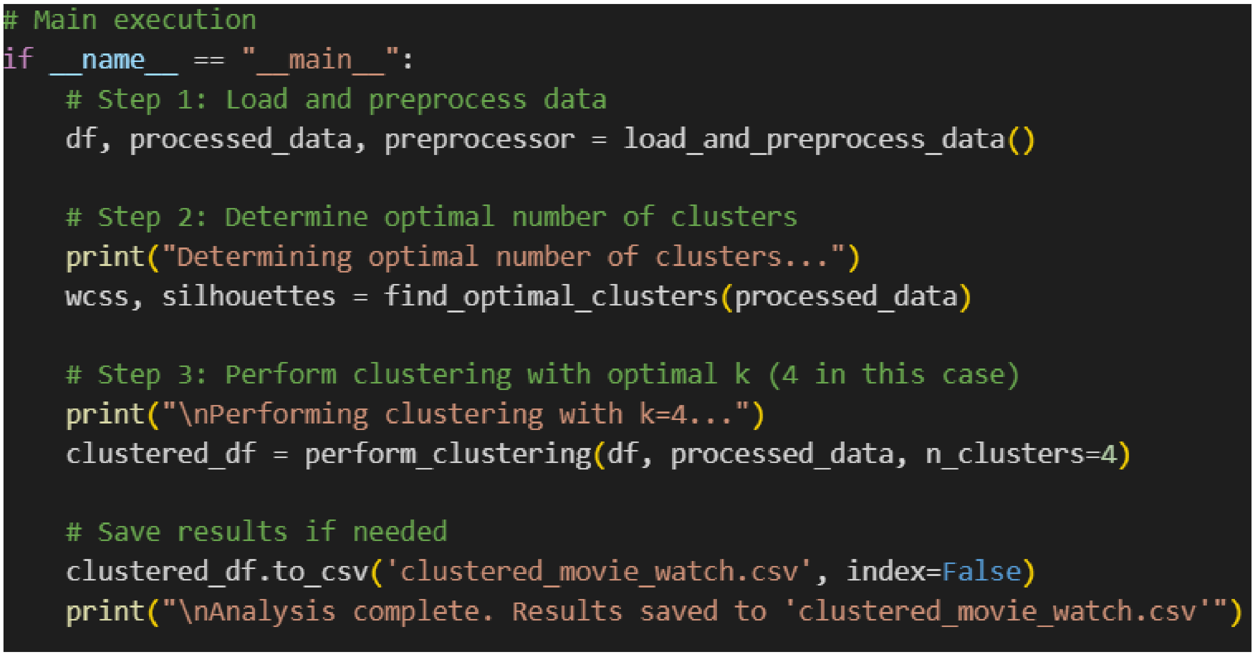
```python
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.compose import ColumnTransformer

# 1. Load and prepare the data
def load_and_preprocess_data():
    # Load the data
    df = pd.read_csv('/content/movie_watch.csv')

    # Data preprocessing
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), ['watch_time_hour', 'avg_rating_given']),
            ('cat', OneHotEncoder(), ['genre_preference'])
        ])

    processed_data = preprocessor.fit_transform(df)
    return df, processed_data, preprocessor

# 2. Determine optimal number of clusters
def find_optimal_clusters(data):
    wcss = []
    silhouettes = []
    max_clusters = 8

    for k in range(2, max_clusters+1):
        kmeans = KMeans(n_clusters=k, random_state=42)
```



```
# 2. Determine optimal number of clusters
def find_optimal_clusters(data):
    wcss = []
    silhouettes = []
    max_clusters = 8

    for k in range(2, max_clusters+1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        wcss.append(kmeans.inertia_)
        silhouettes.append(silhouette_score(data, kmeans.labels_))

    # Plot the results
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(range(2, max_clusters+1), wcss, 'bo-')
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')

    plt.subplot(1, 2, 2)
    plt.plot(range(2, max_clusters+1), silhouettes, 'go-')
    plt.title('Silhouette Scores')
    plt.xlabel('Number of clusters')
    plt.ylabel('Silhouette Score')
    plt.tight_layout()
    plt.show()

    return wcss, silhouettes

# 3. Perform clustering and analyze results
def perform_clustering(df, data, n_clusters=4):
    # Perform K-means clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
```

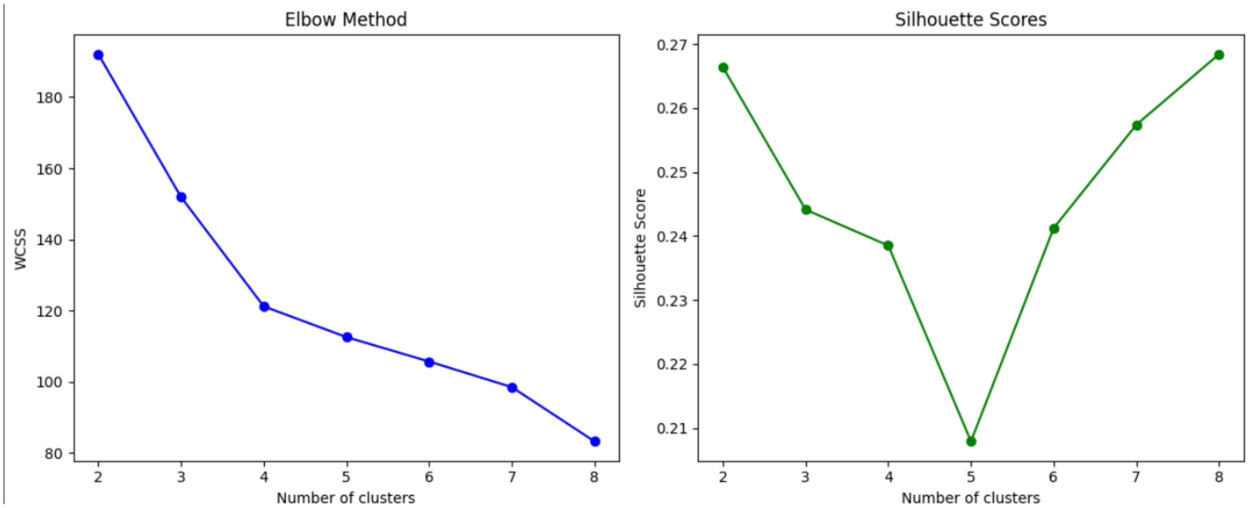


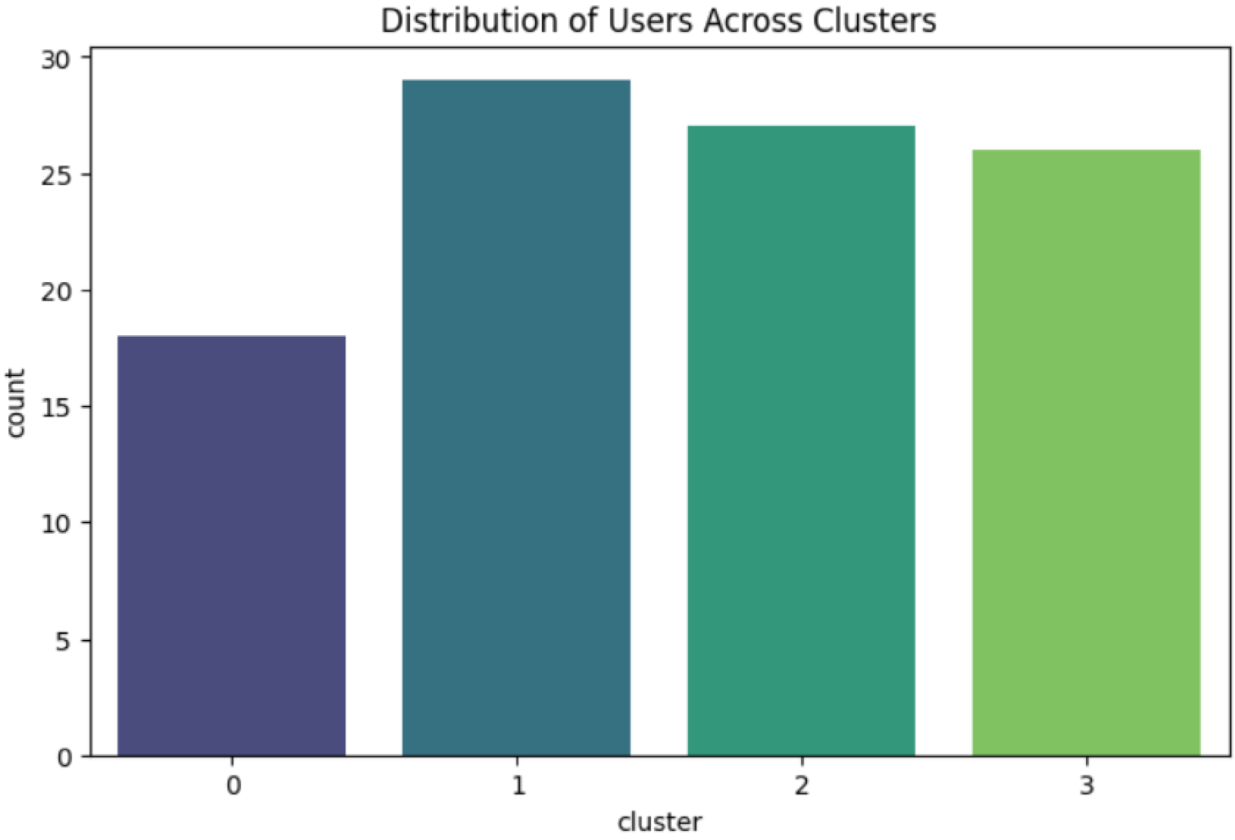
```python
# 3. Perform clustering and analyze results
def perform_clustering(df, data, n_clusters=4):
    # Perform K-means clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(data)
    df['cluster'] = clusters

    # Cluster distribution
    plt.figure(figsize=(8, 5))
    sns.countplot(x='cluster', data=df, palette='viridis')
    plt.title('Distribution of Users Across Clusters')
    plt.show()

    # Analyze cluster characteristics
    cluster_profile = df.groupby('cluster').agg({
        'watch_time_hour': ['mean', 'std'],
        'genre_preference': lambda x: x.mode()[0],
        'avg_rating_given': ['mean', 'std']
    })

    print("Cluster Profiles:")
    print(cluster_profile)

    # Visualize watch times by cluster
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='cluster', y='watch_time_hour', data=df, palette='viridis')
    plt.title('Watch Time Distribution by Cluster')
    plt.show()

    # Visualize ratings by cluster
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='cluster', y='avg_rating_given', data=df, palette='viridis')
    plt.title('Rating Distribution by Cluster')
    plt.show()
```

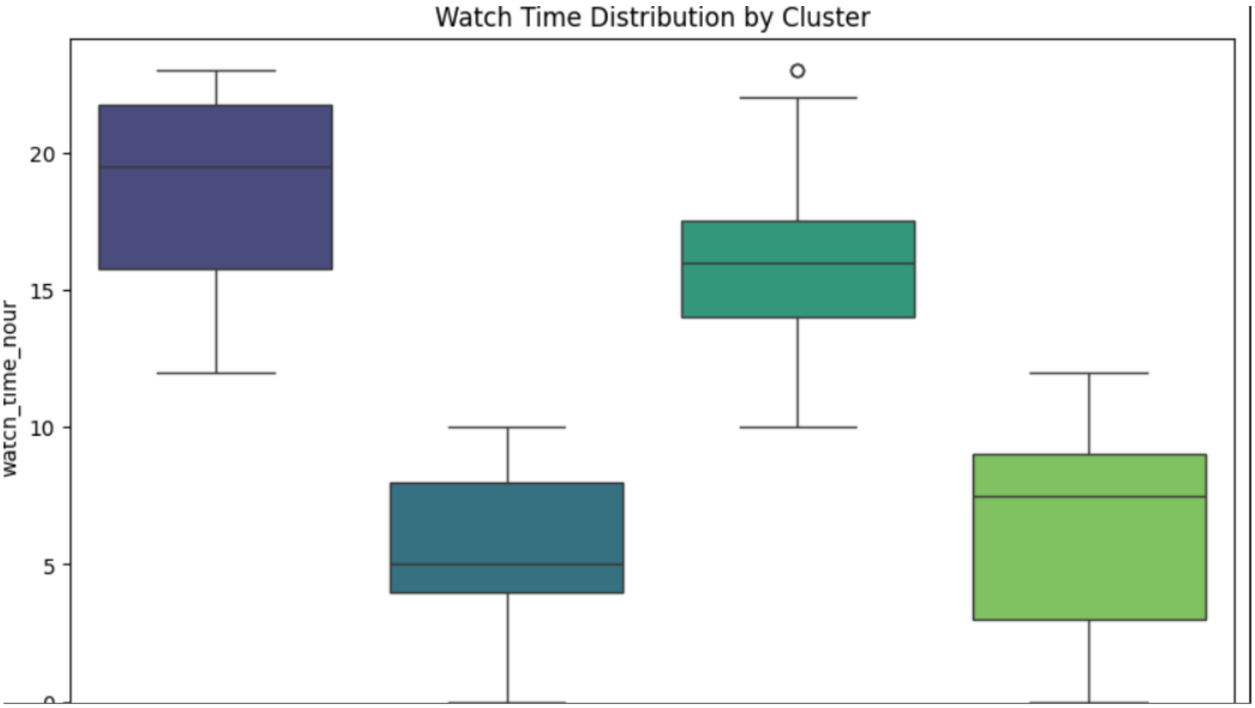


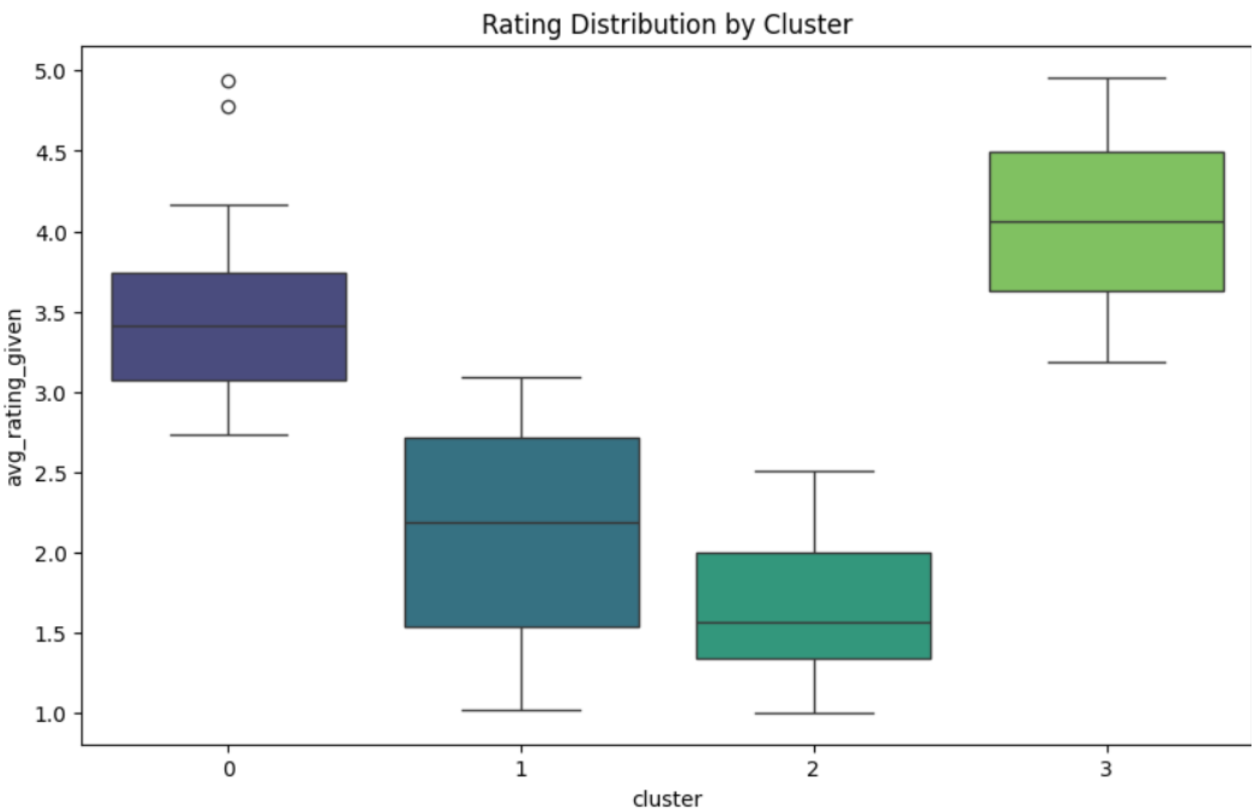
```python
# Main execution
if __name__ == "__main__":
    # Step 1: Load and preprocess data
    df, processed_data, preprocessor = load_and_preprocess_data()

    # Step 2: Determine optimal number of clusters
    print("Determining optimal number of clusters...")
    wcss, silhouettes = find_optimal_clusters(processed_data)

    # Step 3: Perform clustering with optimal k (4 in this case)
    print("\nPerforming clustering with k=4...")
    clustered_df = perform_clustering(df, processed_data, n_clusters=4)

    # Save results if needed
    clustered_df.to_csv('clustered_movie_watch.csv', index=False)
    print("\nAnalysis complete. Results saved to 'clustered_movie_watch.csv'")
```

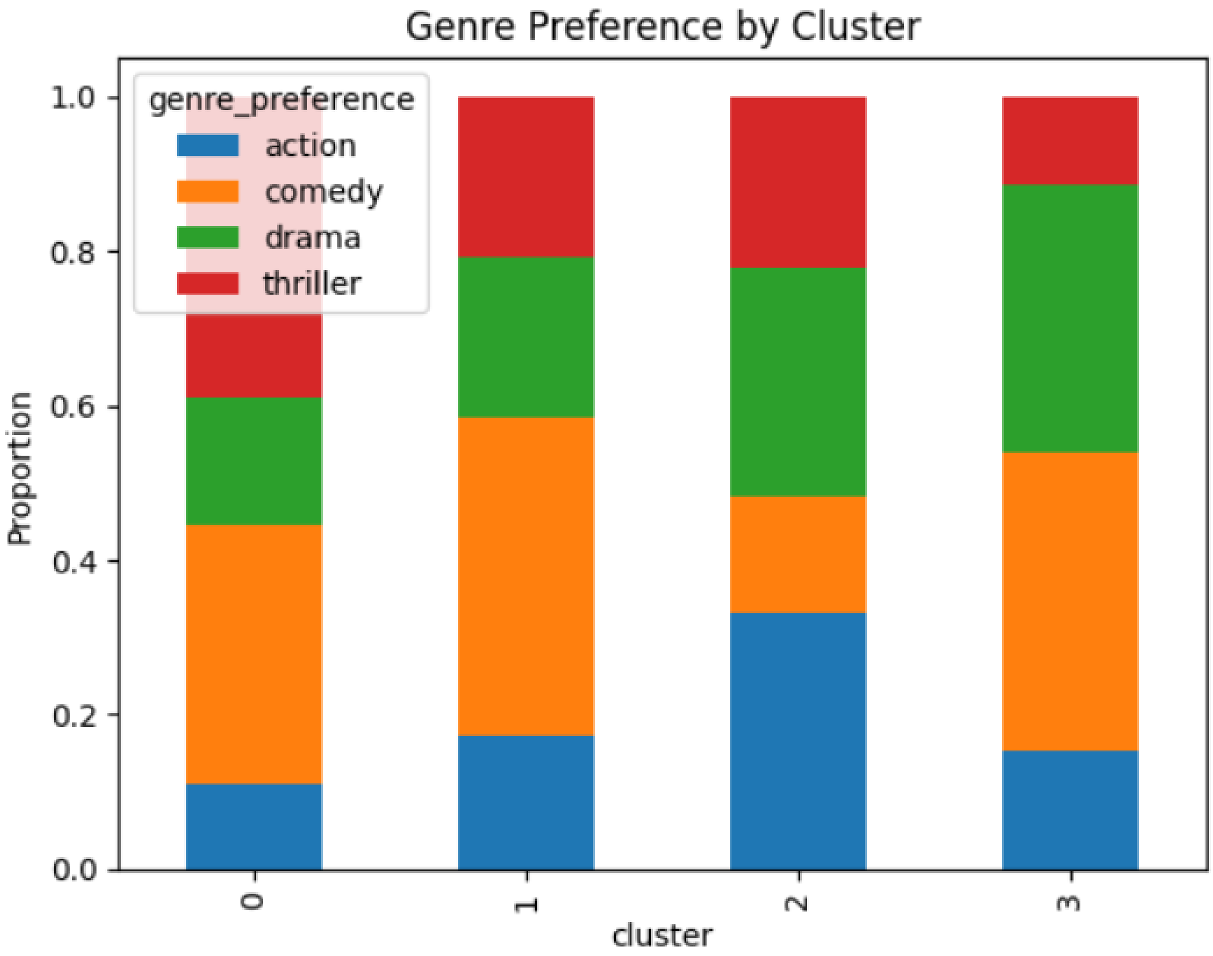
OUTPUT:


Elbow Method
WCSS
180
160
140
120
100
80
2
3
4
5
6
7
8
Number of clusters
Silhouette Scores
Silhouette Score
0.27
0.26
0.25
0.24
0.23
0.22
0.21
2
3
4
5
6
7
8
Number of clusters


Distribution of Users Across Clusters
count
30
25
20
15
10
5
0
0
1
2
3
cluster


Watch Time Distribution by Cluster
watch_time_hour
20
15
10
5
0


Rating Distribution by Cluster
avg_rating_given
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
2
3
cluster



REFRENCES:

1. **Dataset**
   Custom dataset named movie_watch.csv containing user watch patterns (watch time, genre preference, and rating behavior). If sourced or simulated, specify origin or tool used (e.g., self-generated,etc.).

2. Scikit-learn: Machine Learning in Python
   https://scikit-learn.org/
   Used for KMeans clustering, PCA, data preprocessing, and evaluation metrics.