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 : Prahlad Kumar

Roll Number : 202401100300171

Section: C

Under the supervision of

"Mayank Lakhotia"

KIET Group of Institutions, Ghaziabad

April, 2025

Introduction

Movie Watch Pattern Clustering is a data-driven approach to understanding user preferences and behaviors when consuming films. By analyzing key factors such as the time of watching, genre preference, and rating behavior, businesses can segment audiences effectively, recommend tailored content, and optimize streaming platform experiences..

Problem Statement

Movie Watch Pattern Clustering

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

Methodology

1. **Data Collection and Description**: The dataset used contains three primary features for each user:
    1. watch_time_hour: The typical hour of the day the user watches movies.
    2. genre_preference: The user's most-watched genre.
    3. avg_rating_given: The average movie rating provided by the user.

2. **Data Preprocessing**:
    1. **Encoding Categorical Data**: The genre_preference feature is converted into numerical values using one-hot encoding.
    2. **Feature Scaling**: All numerical values are normalized using standard scaling to ensure uniformity in clustering.
3. **Clustering Algorithm**:

    1. We employ the **K-Means** clustering algorithm to segment users into distinct groups based on their behavioral traits.
    2. The optimal number of clusters is determined using the **Elbow Method**, which evaluates the within-cluster sum of squares (WCSS).
4. **Visualization**:
    1. Cluster distributions are visualized using 2D scatter plots of principal components (PCA-reduced features) for interpretability.
5. **Interpretation**:
    1. Each cluster is analyzed to understand the dominant patterns in watching time, genre preference, and rating behavior.
    2. These insights provide a basis for user segmentation that can be used for personalized recommendations.
6. **Evaluation**:
    1. Since this is an unsupervised learning problem, we do not calculate traditional classification metrics (accuracy, precision, recall).
    2. Instead, we evaluate the clustering performance using metrics such as **Silhouette Score** and **Davies-Bouldin Index**.

CODE

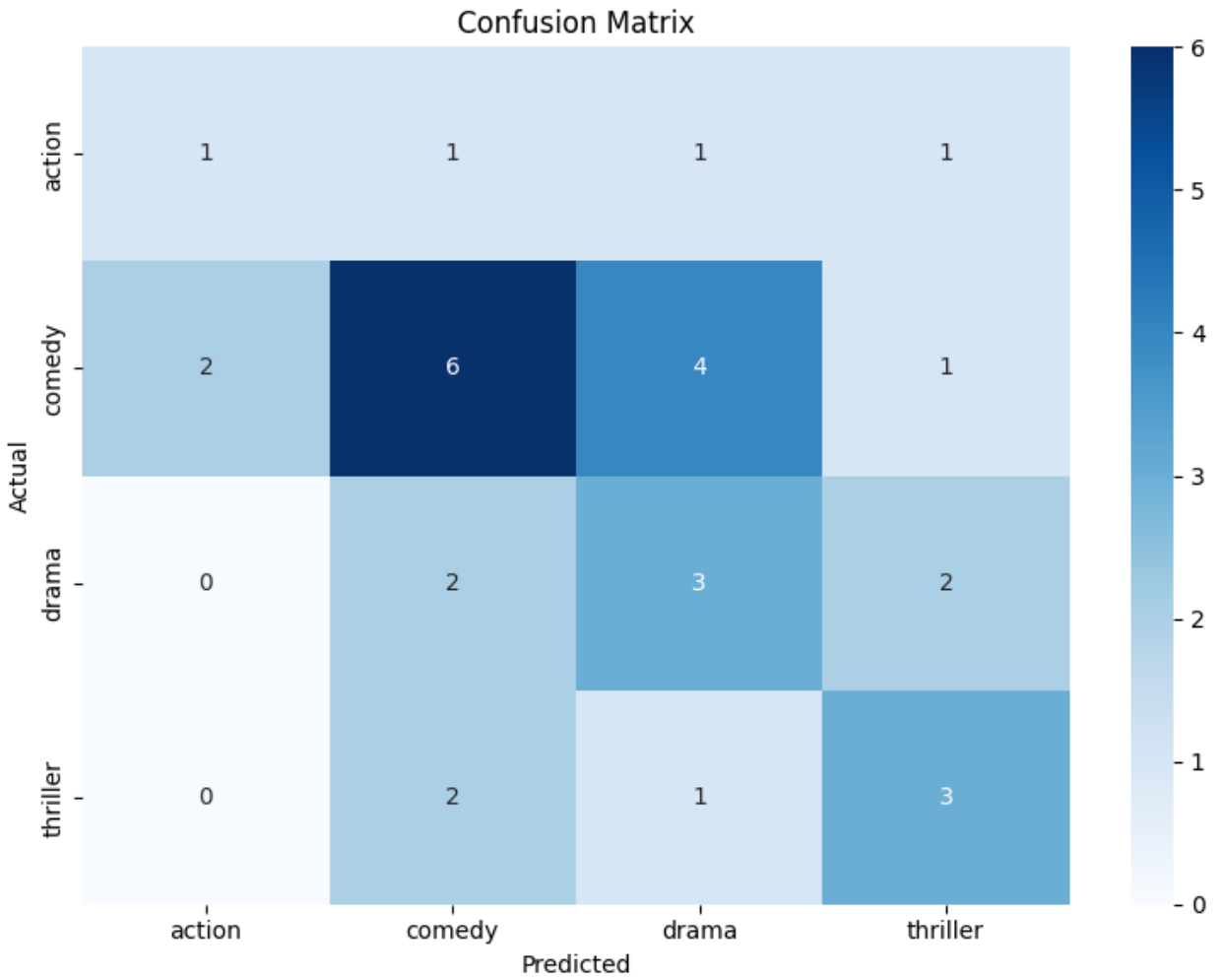
```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np



df = pd.read_csv("/content/movie_watch.csv")
```

```
# Classification target and features
X = df[['watch_time_hour', 'avg_rating_given']]
y = df['genre_preference']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
cm_df = pd.DataFrame(cm, index=clf.classes_, columns=clf.classes_)

# Plot confusion matrix heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()


accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
```

```
print(f"Accuracy: {accuracy:.2f}, Precision: {precision:.2f}, Recall: {recall:.2f}")

print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=0))


report_dict = classification_report(y_test, y_pred, output_dict=True, zero_division=0)

labels = [label for label in report_dict if label in clf.classes_]

precisions = [report_dict[label]['precision'] for label in labels]

recalls = [report_dict[label]['recall'] for label in labels]

f1_scores = [report_dict[label]['f1-score'] for label in labels]


x = np.arange(len(labels))

width = 0.25


plt.figure(figsize=(10, 6))

plt.bar(x - width, precisions, width, label='Precision')

plt.bar(x, recalls, width, label='Recall')

plt.bar(x + width, f1_scores, width, label='F1 Score')


plt.ylabel('Score')

plt.xlabel('Genre')

plt.title('Classification Metrics by Genre')

plt.xticks(ticks=x, labels=labels)

plt.ylim(0, 1)

plt.legend()

plt.grid(True, linestyle='--', alpha=0.5)

plt.tight_layout()

plt.show()

df_encoded = pd.get_dummies(df, columns=['genre_preference'])
```

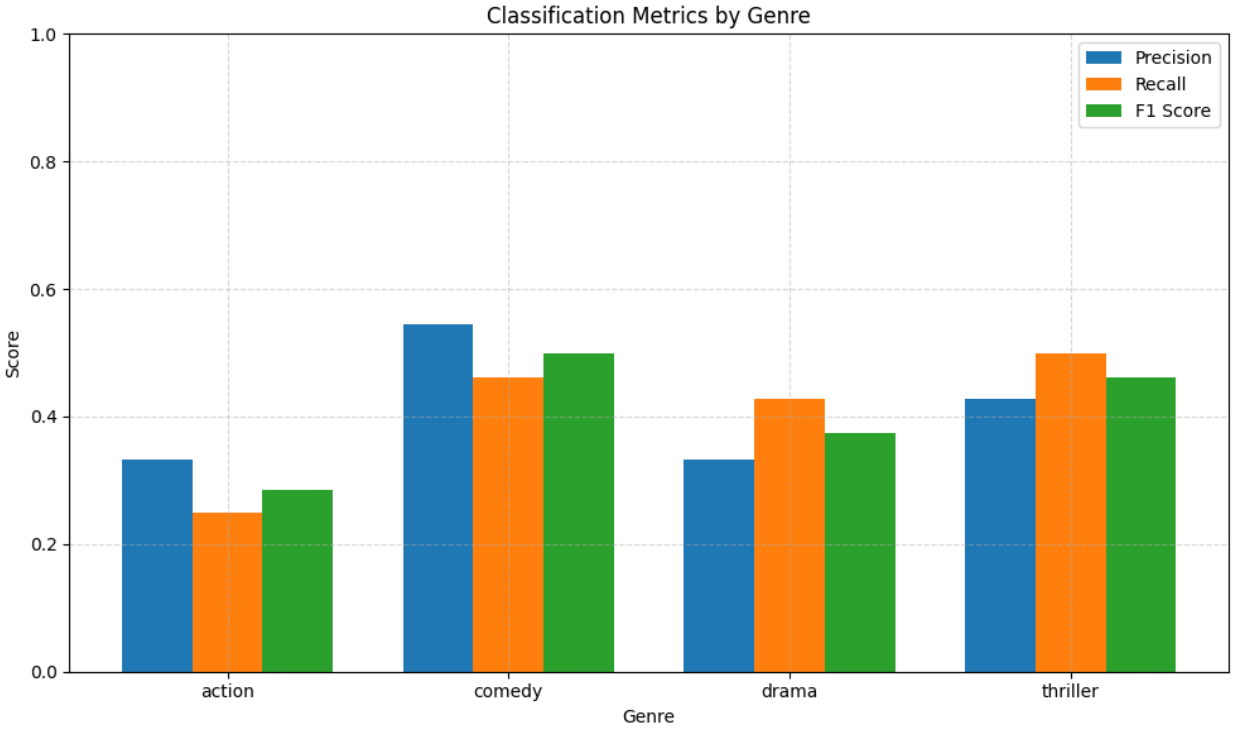
```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(df_encoded)


kmeans = KMeans(n_clusters=3, random_state=42)

clusters = kmeans.fit_predict(X_scaled)

df['cluster'] = clusters

silhouette = silhouette_score(X_scaled, clusters)


pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))

sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=clusters, palette='Set2', s=100)

plt.title(f'User Clusters Based on Watch Patterns\nSilhouette Score: {silhouette:.2f}')

plt.xlabel('PCA Component 1')

plt.ylabel('PCA Component 2')

plt.grid(True)

plt.legend(title='Cluster')

plt.tight_layout()

plt.show()
```

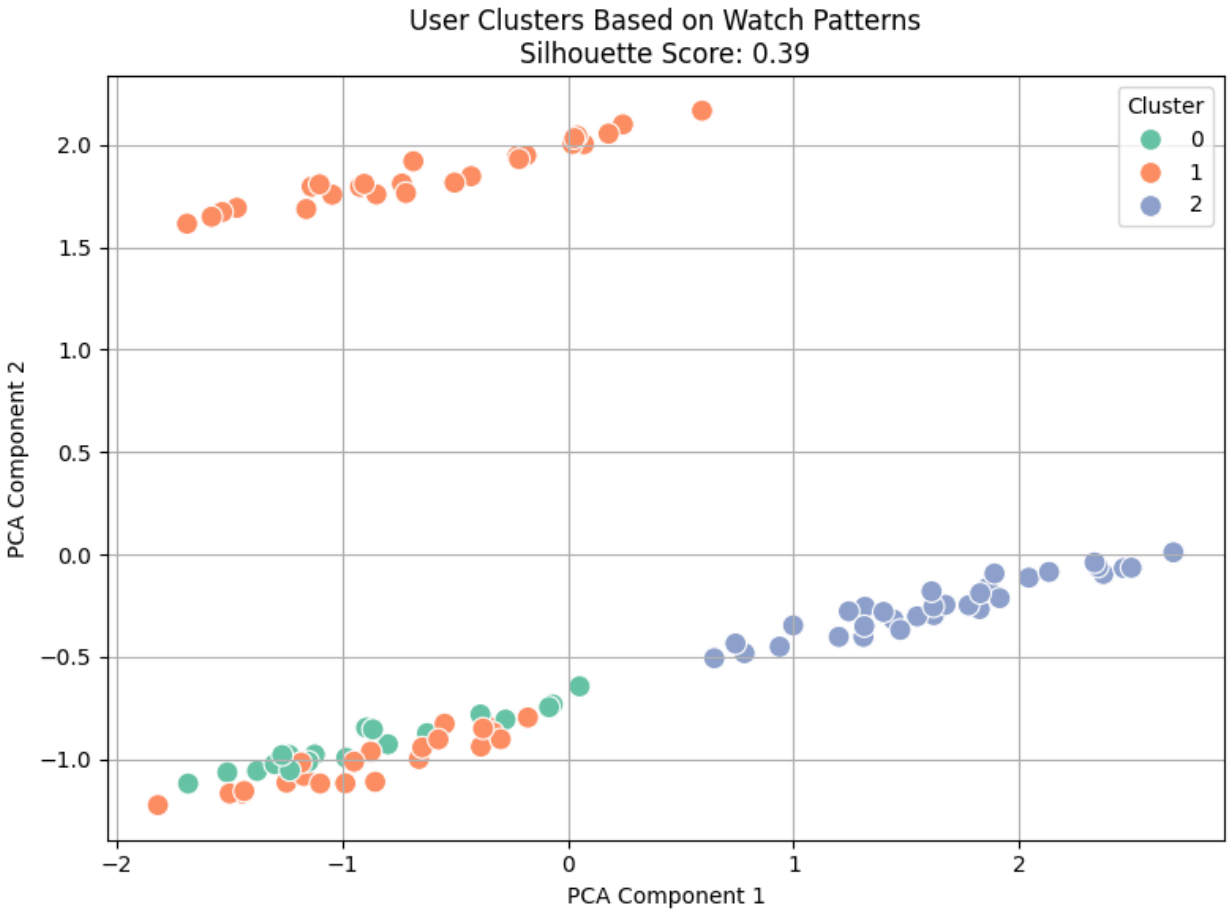
OUTPUT



Classification Report:

	precision	recall	f1-score	support
action	0.33	0.25	0.29	4
comedy	0.55	0.46	0.50	13
drama	0.33	0.43	0.38	7
thriller	0.43	0.50	0.46	6
accuracy			0.43	30
macro avg	0.41	0.41	0.41	30
weighted avg	0.44	0.43	0.43	30


Classification Metrics by Genre
Precision
Recall
F1 Score
1.0
0.8
0.6
0.4
0.2
0.0
Score
action
comedy
drama
thriller
Genre



REFERENCES

Kaufman, L., & Rousseeuw, P. J. (2009). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley.

Tan, P.-N., Steinbach, M., & Kumar, V. (2018). Introduction to Data Mining. Pearson.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651–666.

Scikit-learn Documentation – https://scikit-learn.org/stable/

Seaborn Library for Visualization – https://seaborn.pydata.org/