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&ML)

By

Name : MAKSUDURRAHMAN

Roll Number : 202401100400117

Section: B

Under the supervision of

“Mr ABHISHEK SHUKLA”

KIET Group of Institutions, Ghaziabad

May, 2025

🧩 Introduction

In the world of personalized recommendations, understanding a user's watching behavior is crucial. This project applies unsupervised machine learning (KMeans clustering) to group users into clusters based on:

- **Time of watching** (watch_time_hour)
- **Genre preference** (genre_preference)
- **Rating behavior** (avg_rating_given)

This allows service providers to identify patterns and tailor user experiences accordingly. PCA (Principal Component Analysis) was used to reduce the features into two dimensions to visualize the clusters meaningfully.

🔬 Methodology

We followed a systematic approach using Python and Google Colab:

1. **Data Upload:** A CSV file (`movie_watch.csv`) containing user behavior data was uploaded using Colab's file upload feature.
2. **Data Preprocessing:**
    - `StandardScaler` was used to normalize numeric features.
    - `OneHotEncoder` was used to convert categorical genre preferences into numerical vectors.
3. **Clustering:**
    - `KMeans` algorithm with 3 clusters (`n_clusters=3`) was applied.
4. **Dimensionality Reduction:**
    - `PCA` was used to reduce high-dimensional data to 2D for visualization.

5. **Visualization:**
    - A 2D scatter plot was generated to show how users are grouped into clusters.
6. **Result Export:** The final DataFrame with cluster labels was saved to a CSV file for further analysis.

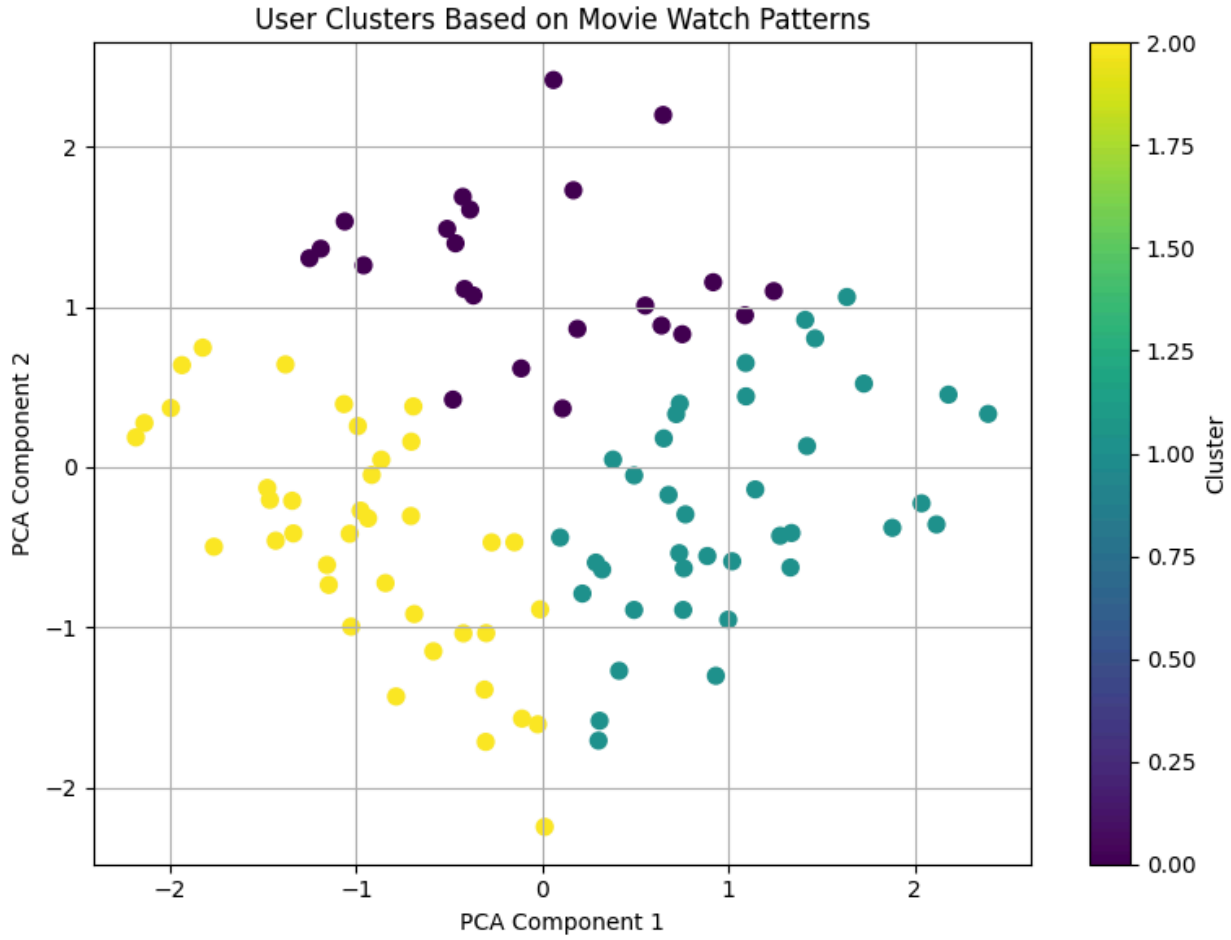
💻 Code

```
# Import all necessary libraries
import pandas as pd
from google.colab import files
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
import matplotlib.pyplot as plt

# Step 1: Upload the CSV file
print("Please upload your 'movie_watch.csv' file...")
uploaded = files.upload()

# Step 2: Load the uploaded file
filename = list(uploaded.keys())[0]
df = pd.read_csv(filename)
print("\nPreview of the dataset:")
print(df.head())

# Step 3: Define feature types
numeric_features = ['watch_time_hour', 'avg_rating_given']
categorical_features = ['genre_preference']

# Step 4: Preprocessing setup
numeric_transformer = StandardScaler()
```

```
categorical_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
# Step 5: KMeans clustering pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('kmeans', KMeans(n_clusters=3, random_state=42))
])
# Step 6: Fit the pipeline
pipeline.fit(df)

# Step 7: Add cluster labels
df['cluster'] = pipeline.named_steps['kmeans'].labels_

# Step 8: Dimensionality reduction for visualization
X_processed = preprocessor.fit_transform(df.drop('cluster', axis=1))
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_processed)

# Step 9: Visualize the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['cluster'], cmap='viridis',
s=50)
plt.title("User Clusters Based on Movie Watch Patterns")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.colorbar(label='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()
# Step 10: Save clustered data to CSV
df.to_csv("clustered_movie_watch.csv", index=False)
print("✅ Clustered data saved as 'clustered_movie_watch.csv'")
```

📸 Output/Result

📍 Below is the screenshot showing the cluster plot generated in Google Colab:



📚 References/Credits

- **Dataset:** Provided/created for academic project work.
- **Tools Used:**
    - Google Colab
    - `pandas`, `matplotlib`, `scikit-learn`
- **Libraries Referenced:**
    - `StandardScaler`, `OneHotEncoder`, `KMeans`, `PCA`, `Pipeline` from `sklearn`
- No external pre-trained models or proprietary tools were used.
