Name: Viven Hotwani Class:D15C/17

Experiment No 8

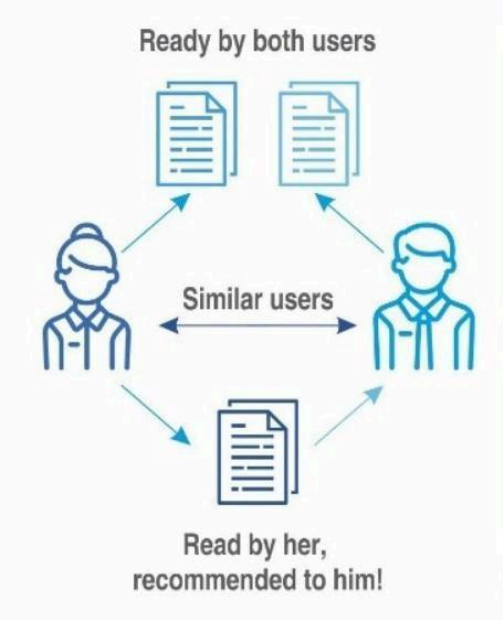
Aim: To implement a recommendation system on your dataset using the following machine learning techniques.

- Regression
- Classification
- Clustering
- Decision tree
- Anomaly detection
- Dimensionality Reduction
- Ensemble Methods

Theory: Recommendation systems predict user preferences to enhance engagement and are vital for content-rich platforms like Amazon and Netflix. They range from simple models to complex AI-driven systems using structured and unstructured data. This post introduces recommendation systems and outlines their business impact, forming the basis for my Master's Thesis work.

Types of Recommendation System

1) **Collaborative Filtering**
   Collaborative filtering generates recommendations by leveraging the behaviors, preferences, and interactions of multiple users. Instead of focusing solely on the actions of a specific user, it finds patterns across users to suggest relevant items. There are two common variations:

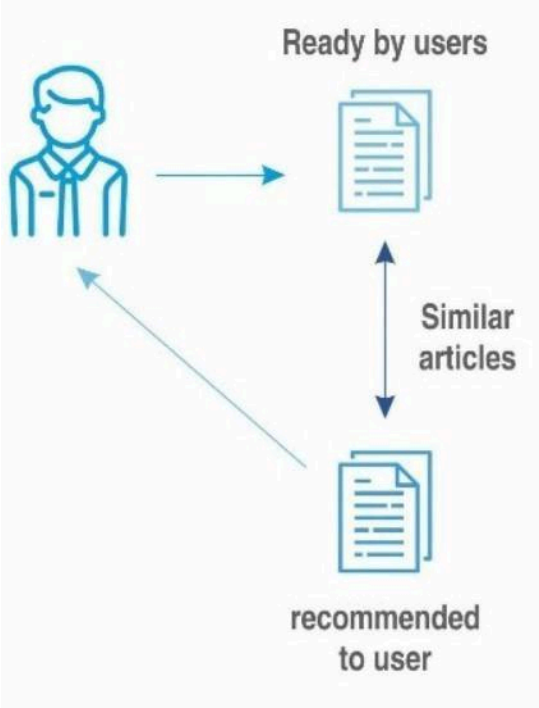


- **User Similarity:** This method groups users with similar behaviors and recommends items liked by others in the group. It's especially useful when a new user has limited data, as it can rely on similar users' preferences to generate suggestions.

- **Association-Based:** Often framed as "Users who viewed/bought X also viewed/bought Y," this technique analyzes purchasing patterns or viewing sequences. It's effective for recommending complementary items or content that aligns with a user's journey or current context.

2) **Content-Based**

Content-based systems focus on the user's individual history—such as items viewed, purchased, or rated—and recommend similar items. These systems become more accurate with increased user interaction and input. There are several nuanced techniques within this category:



- **Content Similarity:** Recommendations are based on matching metadata attributes (e.g., genre, price, author) between products. It's ideal for platforms with detailed item descriptions and limited user activity.

- **Topic Modeling:** This technique analyzes unstructured text (like news articles or reviews) to extract topics of interest, making it valuable for domains rich in textual data.

- **Popular Content Promotion:** Highlights widely appealing items based on features like popularity, recency, or price. It helps surface trending content and is commonly used when new content is frequently introduced.

The 6 steps to build a recommendation system:

1. Understand the Business
Start by clearly defining the goals of your recommendation system. These could include increasing sales, improving user engagement, reducing time to purchase, or promoting under-consumed content. Discuss with both data and business teams to understand what success looks like, where recommendations will appear in the user journey, and whether recommendations are the best solution compared to alternatives like curated lists. Consider what data is available, what product changes might be needed, and how to segment users based on similar preferences.

2. Get the Data
Gather as much relevant data as possible. Use both explicit feedback (like ratings and reviews) and implicit feedback (like browsing behavior, clicks, and watch time). Each has its strengths and weaknesses, so using both can provide a more accurate understanding of user preferences. For new or anonymous users, consider using demographic data or general behavior trends to make recommendations.

3. Explore, Clean, and Augment the Data
Explore and clean the data to ensure quality and relevance. Focus on recent interactions, as user preferences can change over time. Consider assigning more weight to newer data and possibly removing outdated interactions. Handle missing values carefully, especially in high-dimensional datasets where many features may be sparse, and look for opportunities to enhance the dataset with additional useful information.

4. Predict the Ranking
Use the prepared data to generate recommendations. A simple ranking system might be enough in some cases, but more sophisticated machine learning approaches can improve performance. You can use hybrid systems that combine different techniques, maintain multiple models in parallel, or apply machine learning to optimize which model is used. Tailor your recommendation approach depending on where the user is in their journey before or after they've interacted with content.

5. Visualize the Data
Visualization plays a key role in both understanding the dataset and communicating insights. During exploration, use visualizations to uncover trends and patterns. After deployment, visual dashboards can help business teams identify underperforming content, user behavior clusters, and opportunities for improvement. Good visualizations make large, complex datasets understandable and actionable.

6. Iterate and Deploy Models
Deploy the model into production to start impacting real users and business outcomes. Monitor its performance continuously and refine the model as more data becomes available. Set up feedback loops to understand whether recommendations are effective, and adapt the model as user preferences evolve. Recommendation systems are dynamic—they need ongoing updates, experimentation, and optimization to stay effective over time.

Implementation

Step 1: Load dataset and select Features

```
import pandas as pd

df = pd.read_csv("products.csv")

df['description'] = df['description'].fillna('')
df['ingredients'] = df['ingredients'].fillna('')

df['combined_text'] = df['ingredients'] + " " + df['description']
```

The code loads products.csv into a pandas DataFrame.
It fills missing values in the description and ingredients columns with empty strings.
Then, it creates a new column combined_text by combining ingredients and description for each row.

Step 2: Import and perform TF-IDF Vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')

tfidf_matrix = vectorizer.fit_transform(df['combined_text'])
```

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used in natural language processing and information retrieval to evaluate the importance of a word in a document relative to a collection of documents (corpus).

TF-IDF combines two components: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF): Measures how often a word appears in a document. A higher frequency suggests greater importance. If a term appears frequently in a document, it is likely relevant to the document's content. **Formula**:

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Inverse Document Frequency (IDF): Reduces the weight of common words across multiple documents while increasing the weight of rare words. If a term appears in fewer documents, it is more likely to be meaningful and specific. **Formula**:

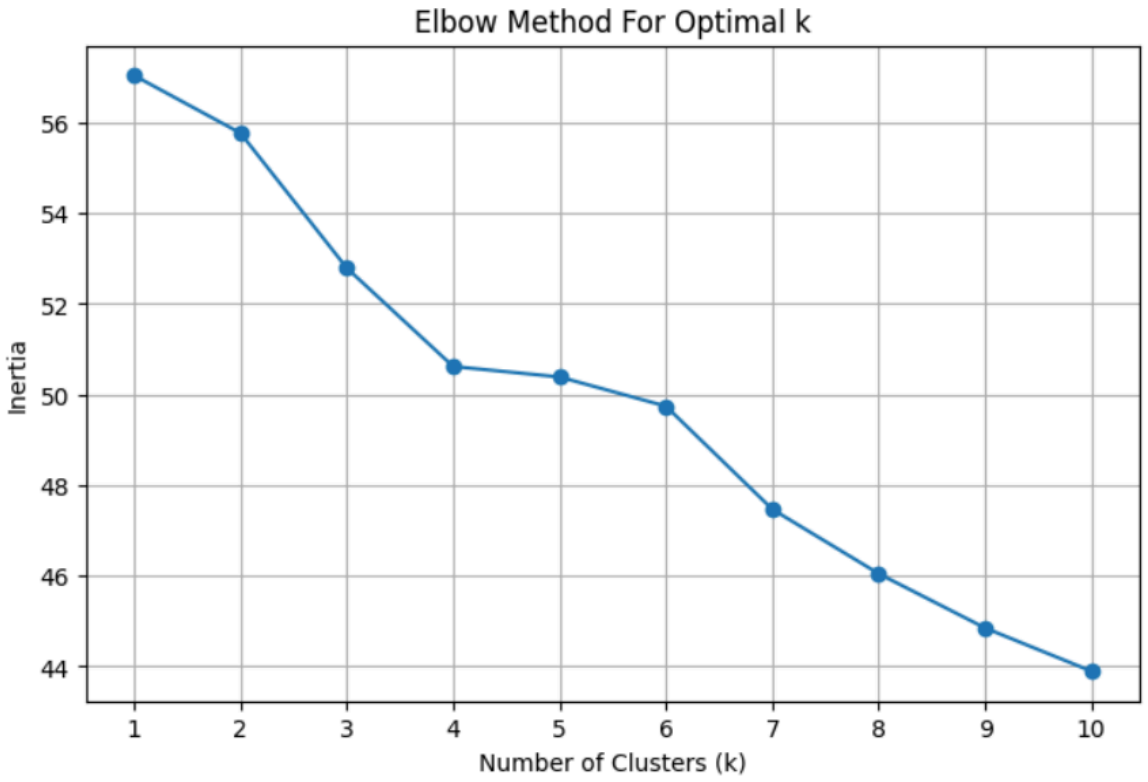
$$IDF(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

Step 3: Clustering

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Range of cluster numbers to try
cluster_range = range(1, 11)
inertias = []

# Calculate KMeans inertia for each cluster count
for k in cluster_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(tfidf_matrix)
    inertias.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(cluster_range, inertias, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(cluster_range)
plt.grid(True)
plt.show()
```

✓ 0.1s

Name: Viven Hotwani Class:D15C/17



Selected 4 as no of cluster with help of elbow curve

```
tfidf_matrix.shape
```
✓ 0.0s

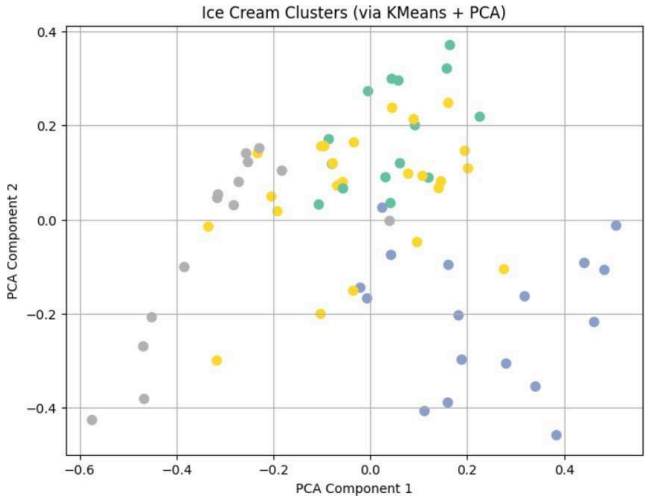
```
(70, 464)
```

```
from sklearn.cluster import KMeans

num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
df['cluster'] = kmeans.fit_predict(tfidf_matrix)
```
✓ 0.0s

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(tfidf_matrix.toarray())
```
✓ 0.0s

It works by transforming high-dimensional data into a lower-dimensional space while maximizing the variance (or spread) of the data in the new space. This helps preserve the most important patterns and relationships in the data.

```
plt.figure(figsize=(8, 6))
scatter = plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=df['cluster'], cmap='Set2', s=50)
plt.title("Ice Cream Clusters (via KMeans + PCA)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)

plt.show()
```



Step 4: Recommendation function

```
def recommend_by_cluster(user_input, top_n=5):
    user_input_tfidf = vectorizer.transform([user_input])

    user_cluster = kmeans.predict(user_input_tfidf)[0]

    cluster_df = df[df['cluster'] == user_cluster]

    # Compute similarity within the cluster
    cluster_tfidf = tfidf_matrix[cluster_df.index]
    similarities = cosine_similarity(user_input_tfidf, cluster_tfidf).flatten()

    # Sort and return top matches
    top_indices = similarities.argsort()[::-1][:top_n]
    recommendations = cluster_df.iloc[top_indices][['name', 'rating', 'rating_count']].copy()
    recommendations['similarity'] = similarities[top_indices]
    return recommendations
```

✓ 0.0s

In this code user inputs are classified and matched with relevant options. After transforming the input into numerical data (using TF-IDF), the system predicts the cluster to which the input belongs. This clustering can represent groups of similar items, users, or behaviors in your experiment. Next, the cosine similarity calculation ensures that recommendations are highly relevant and tailored to the user's input, making the results more precise and impactful.

Step 5: Result

```
user_query = "chocolate "
results = recommend_by_cluster(user_query, top_n=10)
results
```

✓ 0.0s

	name	rating	rating_count	similarity
58	Vanilla Cookie Squares	4.3	32	0.452555
69	White Chocolate Raspberry Ice Cream Bar	3.9	11	0.447600
13	Chocolate Dark Chocolate Ice Cream Bar	4.5	22	0.427577
28	Dulce de Leche Cookie Squares	3.9	35	0.374704
39	Peanut Butter Cookie Squares	4.4	14	0.371822
42	Peanut Butter Chocolate Fudge Non-Dairy Bar	4.8	32	0.301602
14	Chocolate Fudge Non-Dairy Bar	5.0	22	0.300940
44	Peppermint Bark Ice Cream Bar	5.0	8	0.269398
32	Irish Cream Cookie Squares	4.9	14	0.252807
20	Coconut Caramel Dark Chocolate Non-Dairy Bar	4.6	31	0.236645

It starts by transforming the user's input (like "chocolate") into a **numerical vector** through TF-IDF. Then it predicts the **most relevant cluster** of data based on this input using the k-means clustering algorithm, which essentially groups similar items together beforehand.

Conclusion:

This experiment demonstrated the successful implementation of a recommendation system using various machine learning techniques. By applying TF-IDF and K-Means clustering, we grouped similar items and matched user input to relevant products using cosine similarity. The system effectively provided accurate and personalized recommendations, showcasing a practical approach to real-world recommendation engines.