Experiment No: 8

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

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

Theory:
A Recommendation System is a type of machine learning that suggests relevant items to users by analyzing past data and item features. In this experiment, we build a Content-Based Recommendation System using the K-Means clustering algorithm.

Clustering is an unsupervised learning technique that groups similar data points together. The K-Means algorithm divides the dataset into K distinct clusters, with each data point assigned to the cluster whose mean is closest. For this recommendation task, books are grouped based on features such as genres and ratings, allowing us to identify books with similar content and popularity.

Once we know the cluster a book belongs to, we can recommend other books from the same group, ensuring they have comparable attributes. This method is especially helpful when user interaction or preference data is not available.

Description about dataset:

The dataset used for building the Book Recommendation System is derived from **Goodreads** and contains various features related to books, their authors, and user interactions. The primary columns include:

- **Book**: The title of the book, which may sometimes include the series it belongs to.
- **Author**: The name(s) of the author(s) of the book.
- **Description**: A summary or synopsis of the book, useful for text-based analysis or classification.
- **Genres**: A list of genres assigned to each book (e.g., Fiction, Fantasy, Historical), enabling genre-based recommendations.

- **Avg_Rating**: The average rating given by users on Goodreads (out of 5), reflecting overall user satisfaction.

- **Num_Ratings**: The total number of ratings the book has received, indicating its popularity.

- **URL**: A direct link to the book's page on Goodreads for more information.

Step 1:

Data loading and preprocessing are critical initial steps in any machine learning pipeline. First, the dataset is loaded from a file (such as CSV) using tools like Pandas, which allows easy exploration and manipulation. Once loaded, preprocessing is performed to clean and prepare the data for modeling.

Preprocessing typically includes:

- **Handling missing or null values** to ensure consistency.

- **Converting data types** (e.g., converting rating counts from strings to integers).

- **Cleaning text columns** such as removing special characters from book descriptions or genres.

- **Encoding categorical data** like genres or authors using one-hot encoding or label encoding.

- **Normalizing numerical features** (like ratings or rating counts) to bring them on the same scale.

```python
# Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ast
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import accuracy_score, classification_report
from sklearn.neighbors import NearestNeighbors

# Step 1: Load Dataset
file_path = '/content/goodreads_data.csv'  # Replace with your file path
df = pd.read_csv(file_path)

# Step 2: Preprocessing
df = df.dropna(subset=['Description', 'Genres', 'Avg_Rating', 'Num_Ratings'])

# Clean ratings
df['Num_Ratings'] = df['Num_Ratings'].replace(',', '', regex=True).astype(int)
df['Avg_Rating'] = df['Avg_Rating'].astype(float)

# Convert Genres to list
df['Genres'] = df['Genres'].apply(ast.literal_eval)

# Combine text fields for content-based features
df['combined_text'] = df['Description'] + ' ' + df['Genres'].astype(str)
```

Step 2: Elbow Method

The Elbow Method is a popular technique used to determine the optimal number of clusters (k) in K-Means Clustering. It works by plotting the Within-Cluster Sum of Squares (WCSS) for different values of k (number of clusters), and selecting the k at which the WCSS starts to decrease slowly — forming an "elbow" shape in the plot.

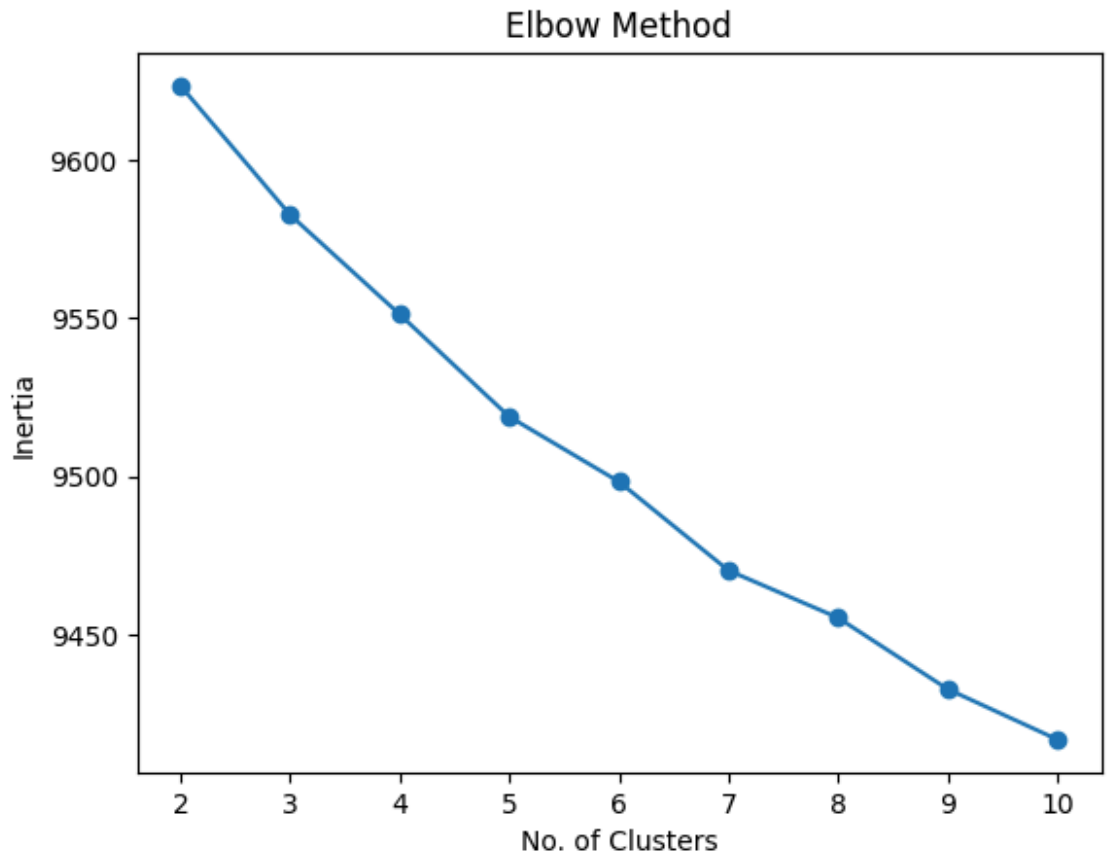
In the context of the book recommendation system, we applied the Elbow Method to cluster books based on features like:

- Average Rating (how much people liked the book)
- Number of Ratings (how many people rated it)
- Encoded Genres (genres as numerical vectors)

This helps group similar books together and can be used to recommend books from the same cluster.

```
# Step 4: KMeans Clustering with Elbow Method
inertia = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(tfidf_matrix)
    inertia.append(kmeans.inertia_)

# Plot Elbow
plt.plot(range(2, 11), inertia, marker='o')
plt.title("Elbow Method")
plt.xlabel("No. of Clusters")
plt.ylabel("Inertia")
plt.show()
```



Step 3: K-Means clustering and visualization:

In this step of the book recommendation system, we use K-Means Clustering to group similar books based on their genre, average rating, and number of ratings. K-Means is an unsupervised machine learning algorithm that partitions the dataset into k distinct clusters by minimizing the distance between the data points and their respective cluster centers.

- The scatter plot shows 5 distinct clusters, each representing a group of books with similar features (like genre and popularity).
- Books that are highly rated and belong to similar genres are grouped closer together in the same cluster.
- The clusters are relatively well-separated, indicating that K-Means was effective in categorizing the books.
- This visual representation makes it easier to interpret patterns and can be used to recommend books from the same cluster as the user's favorite book.

Step 4: Evaluation using Silhouette Score

The Silhouette Score is a metric used to evaluate the quality of clusters created by unsupervised learning algorithms like K-Means. It measures how similar each data point is to its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to 1, where a higher value indicates that the data points are well matched to their own cluster and distinct from other clusters. A value close to 1 implies that the clusters are dense and well-separated, while a value near 0 suggests overlapping clusters. A negative score indicates poor clustering.

```
sil_score = silhouette_score(final_features, kmeans.labels_)
print(f"Silhouette Score: {sil_score}")
```

```
Silhouette Score: 0.1309930761620488
```

Step 5: Recommendation using K-means clustering

The recommend_books_kmeans function is designed to recommend similar books based on K-Means clustering results. Here's how it works:

1. Book Search: The function first checks if the provided book_title exists in the dataset (df). If not, it returns an error message.
2. Identify the Book's Cluster: The function finds the cluster to which the given book belongs by referencing the K-Means labels (kmeans.labels_) assigned during the clustering phase. Each book is assigned a cluster label, indicating its membership in a specific group of similar books.
3. Cluster-Based Recommendation: Once the book's cluster is identified, the function filters the dataset to retrieve all books within the same cluster. Since K-Means aims to group similar books together, these books should share common characteristics like genres, ratings, and descriptions.
4. Sorting by Ratings: To provide more meaningful recommendations, the function sorts the books within the same cluster by their average ratings (Avg_Rating) in descending order, ensuring that books with higher ratings appear first.
5. Return Top-N Books: The function then returns the top N recommendations (default is 5) based on the highest ratings.

```python
# Step 7: Recommendation Function based on K-Means Clusters
def recommend_books_kmeans(book_title, top_n=5):
    if book_title not in df['Book'].values:
        print("Book not found in the dataset.")
        return []

    index = df[df['Book'] == book_title].index[0]
    cluster_label = kmeans.labels_[index]

    # Get all books in the same cluster
    cluster_books = df[kmeans.labels_ == cluster_label]

    # Sort books by rating and return top N
    cluster_books_sorted = cluster_books.sort_values(by='Avg_Rating', ascending=False).head(top_n)

    recommendations = cluster_books_sorted[['Book', 'Author', 'Avg_Rating']]
    return recommendations

# Example Usage
book_to_search = "To Kill a Mockingbird"  # Replace with a book in your dataset
recommendations = recommend_books_kmeans(book_to_search)

print(f"\n Recommended books similar to '{book_to_search}':")
for i, (title, author, rating) in enumerate(recommendations.values, 1):
    print(f"{i}. {title} by {author} (Rating: {rating})")
```

```
 Recommended books similar to 'To Kill a Mockingbird':
1. Mark of the Lion Trilogy by Francine Rivers (Rating: 4.77)
2. பொன்னியின் செல்வன், முழுத்தொகுப்பு by Kalki (Rating: 4.7)
3. An Echo in the Darkness (Mark of the Lion, #2) by Francine Rivers (Rating: 4.62)
4. The Nightingale by Kristin Hannah (Rating: 4.6)
5. The Complete Novels by Jane Austen (Rating: 4.57)
```

Conclusion:

In this project, we built a Content-Based Book Recommendation System using K-Means Clustering on book genres and ratings. We began with cleaning and transforming the data, especially converting genre labels into machine-readable format. Numerical features like average rating and number of ratings were scaled for uniformity. We used the Elbow Method to determine the optimal number of clusters and applied PCA for visualization in two dimensions.

Each cluster represents books with similar characteristics, and we visualized them with scatter plots. The recommendation system suggests books from the same cluster as the chosen one, ensuring they have similar genres and popularity