Name : Advik Hegde **Div : D15C** **Roll No. : 15**

AIDS Experiment 8

Aim: To implement a recommendation system on your dataset using the following machine learning techniques.
o Regression
o Classification
o Clustering
o Decision tree
o Anomaly detection
o Dimensionality Reduction
o Ensemble Methods.

Theory :

A recommendation system leverages various machine learning techniques to predict user preferences and suggest items. Techniques like regression and classification predict numerical ratings or categorize items for users, while clustering groups similar items for more relevant suggestions. Anomaly detection identifies outliers in user behavior, and dimensionality reduction simplifies data to improve efficiency. Ensemble methods combine multiple models to enhance prediction accuracy.

Decision trees play a crucial role in recommendation systems by modeling the relationship between user features and item preferences. These trees split data based on specific features, creating a clear path for predictions. In the context of recommendations, decision trees can classify or predict which items a user is most likely to prefer by analyzing patterns in user-item interactions. This method is highly interpretable and efficient, especially for datasets with numerous attributes, providing transparent decision-making for personalized suggestions.

Dataset Description :

The **Books.csv** dataset contains metadata for a collection of books, including information such as titles, authors, publication years, publishers, and cover images. This data provides a comprehensive view of the books in the system, which can be leveraged to categorize and recommend books based on various attributes.

The **Ratings.csv** dataset contains user interactions with the books, specifically user ratings for each book. This dataset reflects user preferences and allows for the creation of personalized recommendations by analyzing how users rate different books. By combining both datasets, a recommendation system can suggest books based on user ratings and book attributes.

Steps to perform :

1. First, missing or null values are handled by either removing the affected records or replacing them with statistical estimates like mean or median. Then, irrelevant or redundant attributes are eliminated to reduce noise and focus on important features. The data is also checked for inconsistent formats or incorrect entries, which are corrected or standardized. After that, normalization or scaling is applied to ensure all numerical features are within the same range, which helps algorithms treat each feature equally.

```
# Install if needed:
# pip install pandas scikit-learn matplotlib seaborn

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[ ] # Load datasets
books = pd.read_csv('Books.csv', encoding='latin-1')
ratings = pd.read_csv('Ratings.csv', encoding='latin-1')

# Merge books and ratings on ISBN
data = pd.merge(ratings, books, on='ISBN')

# Drop rows with missing values in critical columns
data.dropna(subset=['Book-Rating', 'Year-Of-Publication'], inplace=True)

# Preview merged data
data.head()
```

2. To develop a decision tree, we need to make sure that we establish a boundary of rating beyond which it can be said that a user does or does not like a book. In our case we decide that rating number oto be “7”.So , if a user rates a book 7 or above, we assume that it can be said that the user likes a book and for all ratings below it,we assume that the user does not like that book.

```
# Define like = 1 if rating >= 7, else dislike = 0
data['Liked'] = data['Book-Rating'].apply(lambda x: 1 if x >= 7 else 0)
```

3. We clip extreme years to ensure that the values fall within a valid range, between 1000 and 2025, to handle any outliers or invalid years. For the feature matrix X, we select the relevant columns (Year-Of-Publication, Author, and Publisher) and apply one-hot encoding to the categorical variables (Author and Publisher), converting them into numerical format. Finally, we set the target variable y as the Liked column, which indicates the user's preference.

```
# Convert year to numeric
data['Year-Of-Publication'] = pd.to_numeric(data['Year-Of-Publication'], errors='coerce')

# Replace missing values and clip extreme years
data['Year-Of-Publication'] = data['Year-Of-Publication'].fillna(0)
data['Year-Of-Publication'] = data['Year-Of-Publication'].clip(lower=1000, upper=2025)

# Feature matrix (X) and target (y)
X = data[['Year-Of-Publication', 'Author', 'Publisher']]
X = pd.get_dummies(X, drop_first=True)  # One-hot encoding for categorical

y = data['Liked']
```

4. After this, we split the dataset in a 20:80 split to train the model and after training , we evaluate the performance of the model .The evaluation metrics of the model are :

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(class_weight='balanced', max_depth=10, random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5250353977151502
              precision    recall  f1-score   support

           0       0.74      0.52      0.61    148106
           1       0.31      0.54      0.39     58122

    accuracy                           0.53    206228
   macro avg       0.52      0.53      0.50    206228
weighted avg       0.62      0.53      0.55    206228
```

5. We can then use this model to make predictions for whether or not a user will like a book from various years

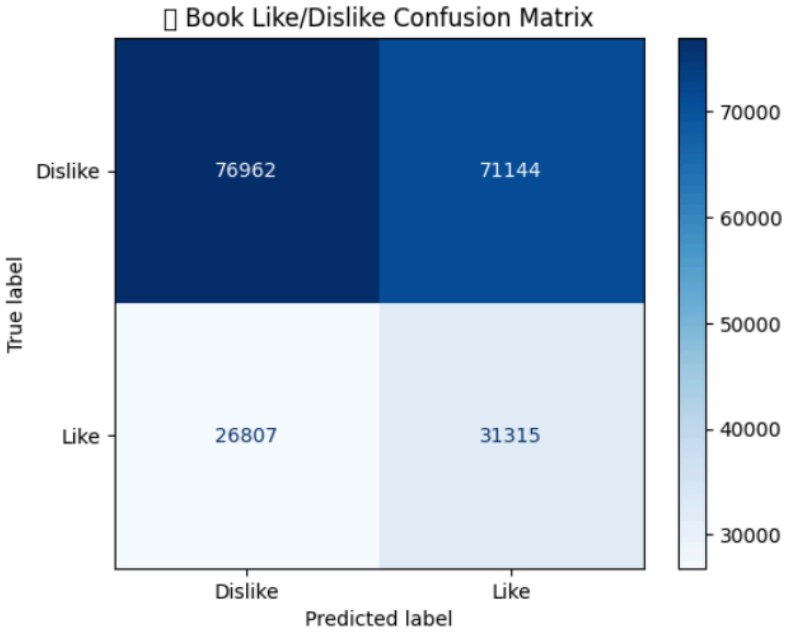
```
# Predict whether user might like a book from year 2010
new_books = pd.DataFrame({
    'Year-Of-Publication': [1985, 2000, 2010, 2015]
})

predictions = model.predict(new_books)

for year, pred in zip(new_books['Year-Of-Publication'], predictions):
    print(f"Book published in {year} → {'Like' if pred == 1 else 'Dislike'}")
```

```
Book published in 1985 → Dislike
Book published in 2000 → Like
Book published in 2010 → Like
Book published in 2015 → Dislike
```

The confusion matrix for the model can also be used to evaluate it's performance



Conclusion : The implementation of a recommendation system using machine learning techniques, specifically decision trees, has proven effective in predicting user preferences for books. By leveraging book metadata and user ratings, we created a model that classifies whether a user will like a book based on certain features like publication year, author, and publisher. The decision tree model provides transparency in its decision-making process, allowing for clear interpretations of user-item interactions. Through data preprocessing, feature encoding, and model evaluation using performance metrics like the confusion matrix, the system demonstrates how machine learning can be used to make personalized recommendations with accuracy.