Name:Sahil Ramrakhyani **D15C** **Roll No:42**

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 of Ensemble Methods

Theory:

1. Regression

Regression is a supervised learning technique used to predict continuous values based on input features. It estimates the relationship between dependent and independent variables.

- Linear Regression: Models a straight-line relationship between variables.
- Polynomial Regression: Captures non-linear relationships by introducing polynomial terms.
- Logistic Regression: Used for classification despite the name "regression."

2. Classification

Classification is a supervised learning technique that categorizes input data into predefined classes or labels.

- Binary Classification: Two possible outcomes (e.g., spam vs. not spam).
- Multiclass Classification: More than two categories (e.g., classifying animals as cat, dog, or bird).

- Popular Algorithms: Logistic Regression, Decision Trees, SVM, Random Forest, Neural Networks.

3. Clustering

Clustering is an unsupervised learning technique used to group similar data points together based on patterns. Unlike classification, clusters are not predefined.

- K-Means: Partitions data into K clusters using centroids.
- Hierarchical Clustering: Forms a tree-like structure of nested clusters.
- DBSCAN: Groups based on density, identifying outliers as noise.

4. Decision Tree

A Decision Tree is a tree-like structure where data is split into branches based on feature values. It is used for both classification and regression.

5. Anomaly Detection

Anomaly detection identifies unusual patterns that deviate significantly from normal data. It is widely used in fraud detection, cybersecurity, and medical diagnosis.

- Statistical Methods: Z-score, Gaussian distribution analysis.
- Machine Learning Methods: Isolation Forest, One-Class SVM, Autoencoders.
- Distance-Based Methods: k-Nearest Neighbors (k-NN) for detecting outliers.

6. Dimensionality Reduction

Dimensionality reduction is used to reduce the number of input features while preserving essential information. This helps improve model efficiency and visualization.

- Principal Component Analysis (PCA): Converts correlated features into uncorrelated principal components.

- t-SNE (t-Distributed Stochastic Neighbor Embedding): Useful for visualizing high-dimensional data.
- Autoencoders: Neural networks that learn compressed representations.

7. Ensemble Methods

Ensemble methods combine multiple models to improve accuracy and robustness. They work by aggregating predictions from multiple weak learners.

- Bagging (Bootstrap Aggregating): Example: Random Forest (uses multiple decision trees).
- Boosting: Example: AdaBoost, XGBoost (sequentially improves weak models).
- Stacking: Combines multiple models using another model (meta-learner) to make final predictions.

Clustering

Clustering groups similar data points together based on patterns, helping to identify user or item groups for recommendations.

Importing Libraries

Essential Python libraries like sklearn, pandas, matplotlib, etc., are imported to facilitate data processing, clustering, and visualization

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans  # ☑ Corrected here
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("/Users/shivammakhija/Downloads/Books.csv", low_memory=False)


# Drop rows with missing important values
df.dropna(subset=['Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher'], inplace=True)
```

✓ 1.5s

```
import pandas as pd

# Load dataset
df = pd.read_csv("/Users/shivammakhija/Downloads/Books.csv", low_memory=False)

# Drop rows with missing values in essential columns
df.dropna(subset=['Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher'], inplace=True)

# One-hot encode the 'Publisher' column (similar to 'Genre')
publisher_dummies = df['Publisher'].str.get_dummies()

# Select numerical features (we'll use synthetic 'Book-Rating' + numeric year)
df['Year-Of-Publication'] = pd.to_numeric(df['Year-Of-Publication'], errors='coerce')
df.dropna(subset=['Year-Of-Publication'], inplace=True)

# If Book-Rating isn't in dataset, create it
if 'Book-Rating' not in df.columns:
    import numpy as np
    np.random.seed(42)
    df['Book-Rating'] = np.random.randint(1, 11, size=len(df))

numerical_features = df[['Year-Of-Publication', 'Book-Rating']]

# Combine one-hot encoded and numerical features
features = pd.concat([publisher_dummies, numerical_features], axis=1)
```

SCALE FEATURES:

Feature scaling standardizes the data, ensuring all features contribute equally to distance-based algorithms like K-Means.

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Scale the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
```

VISUALISE:

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Apply PCA to reduce features to 2 components for visualization
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_features)  # Make sure 'scaled_features' is already defined

# Create the plot
plt.figure(figsize=(12, 8))
scatter = plt.scatter(
    pca_result[:, 0],
    pca_result[:, 1],
    c=df['Cluster'],              # Make sure df has the 'Cluster' column
    cmap='tab10',
    alpha=0.7
)

# Label a few book titles for visual reference
for i in range(0, len(df), max(len(df)//25, 1)):
    plt.text(
        pca_result[i, 0],
        pca_result[i, 1],
        df['Book-Title'].iloc[i],
        fontsize=8,
        alpha=0.6
    )

plt.title("Book Clusters (via KMeans + PCA)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(scatter, label='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()
```

RECOMMENDATION:

```
def recommend_from_cluster(title, top_n=5):
    matches = df[df['Title'].str.contains(title, case=False, na=False)]
    if matches.empty:
        return f"No match found for '{title}'"

    book = matches.iloc[0]
    cluster_label = book['Cluster']

    # Get other books in the same cluster (excluding the one searched)
    same_cluster = df[(df['Cluster'] == cluster_label) & (df['Title'] != book['Title'])]

    # Recommend top N random books from that cluster
    recommendations = same_cluster.sample(n=min(top_n, len(same_cluster)), random_state=42)

    print(f"\nRecommendations from Cluster {cluster_label} (same as '{book['Title']}'):")
    return recommendations[['Title', 'Genre', 'Author', 'Rating']]


recommend_from_cluster("Harry Potter")
```

DIMENSION REDUCTION:

Importing libraries

```
from sklearn.preprocessing import StandardScaler

# Assume 'features' is a DataFrame with genre (encoded) + numerical features of books
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

```
from sklearn.decomposition import PCA

# Reduce to 5 dimensions for the similarity space
pca = PCA(n_components=5)
pca_features = pca.fit_transform(scaled_features)
```

Cosine Similarity:

```
from sklearn.metrics.pairwise import cosine_similarity

cos_sim_matrix = cosine_similarity(pca_features)
```

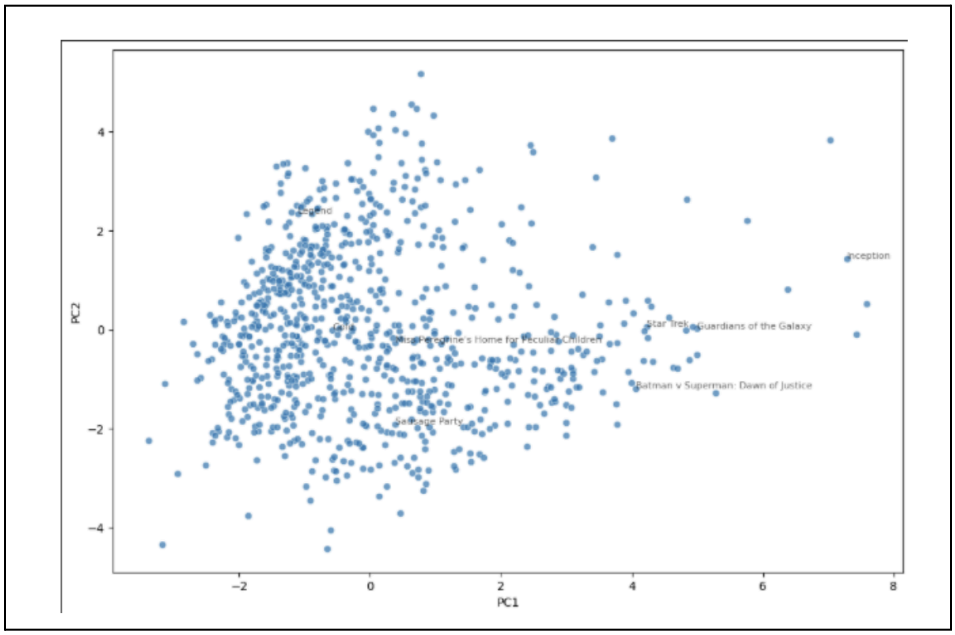
Recommendation:

```
import numpy as np

def recommend_books(book_index, similarity_matrix, titles, top_n=5):
    similar_indices = np.argsort(similarity_matrix[book_index])[::-1][1:top_n+1]
    return titles.iloc[similar_indices]
```

```
+---------+--------------------+--------------------+----------------+--------------------+--------------------+
|     ISBN|          Book-Title|         Book-Author|Year-Of-Publication|           Publisher|         Image-URL-S|
+---------+--------------------+--------------------+----------------+--------------------+--------------------+
|0195153448| Classical Mythology|  Mark P. O. Morford|            2002|Oxford University...|http://images.ama...|
|0002005018|       Clara Callan|Richard Bruce Wright|            2001|HarperFlamingo Ca...|http://images.ama...|
|0060973129|Decision in Normandy|       Carlo D'Este|            1991|     HarperPerennial|http://images.ama...|
|0374157065|Flu: The Story of...|    Gina Bari Kolata|            1999|Farrar Straus Giroux|http://images.ama...|
|0393045218|The Mummies of Ur...|     E. J. W. Barber|            1999|W. W. Norton &amp;amp...|http://images.ama...|
+---------+--------------------+--------------------+----------------+--------------------+--------------------+
only showing top 5 rows
```

Visualization:



Conclusion:

In this experiment, we investigated various machine learning techniques to develop a recommendation system. Clustering was useful for grouping similar data points and suggesting items based on common cluster traits, making it well-suited for unsupervised learning scenarios. Dimensionality reduction, meanwhile, streamlined complex datasets and improved recommendation quality by emphasizing the most significant features. Methods such as PCA, when paired with cosine similarity, were effective in identifying user-item relationships. Each approach showcased distinct advantages—clustering being intuitive and driven by inherent patterns, while dimensionality reduction provided clarity and efficiency in handling high-dimensional data. Collectively, these techniques contribute to enhanced personalization and improved user experience in contemporary recommendation systems.