Experiment 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:

Types of Recommendation Systems : A Recommendation System suggests relevant items to users based on their preferences, behavior, or other factors. There are several types of recommendation techniques:

1. Collaborative Filtering

1. How it works: Recommends items based on user behavior and preferences, assuming users with similar tastes will like similar items.
2. Subtypes:
    a. User-based: Finds users similar to the target user and recommends items they liked.
    b. Item-based: Recommends items similar to those the user already interacted with.
3. Pros: Effective with large user bases; no need for item metadata.
4. Cons: Cold start problem (new users/items lack data); sparsity issues.
5. Example: Netflix suggesting shows based on what similar users watched.

2. Content-Based Filtering

1. How it works: Recommends items by analyzing their attributes and matching them to a user's preferences or past interactions.
2. Pros: Works well for new items; no reliance on other users' data.
3. Cons: Limited by item metadata quality; may lack diversity in recommendations.
4. Example: Spotify recommending songs with similar genres or artists to ones you've liked.

3. Hybrid Systems

1. How it works: Combines collaborative filtering and content-based methods to leverage strengths of both.
2. Pros: Mitigates cold start and sparsity issues; more robust and accurate.
3. Cons: Complex to implement and maintain.
4. Example: Amazon blending user purchase history with item feature similarity.

4. Knowledge-Based Systems

1. How it works: Uses explicit user requirements or domain knowledge to recommend items, often through rules or constraints.
2. Pros: Ideal for complex or infrequent purchases; no cold start issue.
3. Cons: Requires detailed domain knowledge; less dynamic.
4. Example: A travel site suggesting destinations based on user-specified criteria like budget and interests.

Recommendation System Evaluation Measures

Accuracy Measures:

These metrics evaluate how well the recommended items match the actual preferences or ratings of users.

1. Mean Absolute Error (MAE): Measures the average of the absolute differences between predicted ratings and actual ratings.

- **Formula:** $MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - \hat{r}_i|$
  - r_i = Actual rating
  - $\hat{r}_i$ = Predicted rating

2. Root Mean Squared Error (RMSE): Similar to MAE but gives higher weight to large errors due to squaring the differences.

Formula: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}$

3. Precision: Measures the fraction of recommended items that are actually relevant to the user.

Formula: $Precision = \frac{\text{Number of relevant recommended items}}{\text{Total number of recommended items}}$

4. Recall: Measures the fraction of relevant items that were actually recommended to the user.

Formula: $Recall = \frac{\text{Number of relevant recommended items}}{\text{Total number of relevant items}}$

5. F1-Score: The harmonic mean of Precision and Recall, balancing both.

Formula: $F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$

Model used : XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and robust machine learning algorithm that falls under the umbrella of gradient boosting frameworks

Implementation :

Dataset Overview : The data spans different years (notably 2001 and 2012, as seen in the sample) and includes detailed counts of various types of **crimes** reported in each district. The dataset is structured in a tabular format, with each row representing a district or a specific region (e**.g., railway police, city-specific data**) within a state/union territory, and each column representing a specific crime category or metadata attribute.

The dataset is likely sourced from official crime records, such as those maintained by the **National Crime Records Bureau (NCRB)** of India, given the detailed breakdown of crime types and the geographical granularity. It includes both aggregated totals (e.g., "TOTAL" rows for states) and individual district-level data, making it suitable for analyzing crime patterns across regions, time periods, and crime categories.

Features :

1. **Murder**: number of murder cases.
2. **Attempt to murder:** number of attempted murder cases.
3. **Culpable homicide not amounting to murder**: cases of culpable homicide not classified as murder.
4. **Rape**: total number of rape cases.
5. **Custodial rape**: rape cases occurring in custody (subset of rape).
6. **Other rap**e: rape cases not classified as custodial rape.
7. **Kidnapping & abduction:** total kidnapping and abduction cases.

8. **Other IPC crimes**: Miscellaneous crimes under the Indian Penal Code (IPC) not covered by the above categories.
9. **Total IPC Crimes**: Sum of all IPC crimes reported in the district for the given year.

1. Importing dataset

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KIDNAPPING & ABDUCTION	...	HURT/GREVIOUS HURT	DOWRY DEATHS	ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY	INSULT TO MODESTY OF WOMEN	CRUELTY BY HUSBAND OR HIS RELATIVES	IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES	CAUSING DEATH BY NEGLIGENCE	OTHER IPC CRIMES	TO... CRI...
0	ANDHRA PRADESH	ADILABAD	2001	101	60	17	50	0	50	46	...	1131	16	149	34	175	0	181	1518	41
1	ANDHRA PRADESH	ANANTAPUR	2001	151	125	1	23	0	23	53	...	1543	7	118	24	154	0	270	754	41
2	ANDHRA PRADESH	CHITTOOR	2001	101	57	2	27	0	27	59	...	2088	14	112	83	186	0	404	1262	58
3	ANDHRA PRADESH	CUDDAPAH	2001	80	53	1	20	0	20	25	...	795	17	126	38	57	0	233	1181	31
4	ANDHRA PRADESH	EAST GODAVARI	2001	82	67	1	23	0	23	49	...	1244	12	109	58	247	0	431	2313	65
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9012	LAKSHADWEEP	LAKSHADWEEP	2012	0	0	0	0	0	0	0	...	3	0	1	0	1	0	0	32	
9013	LAKSHADWEEP	TOTAL	2012	0	0	0	0	0	0	0	...	3	0	1	0	1	0	0	32	
9014	PUDUCHERRY	KARAIKAL	2012	5	6	2	6	0	6	2	...	186	0	2	0	1	0	44	392	7
9015	PUDUCHERRY	PUDUCHERRY	2012	24	21	10	7	0	7	17	...	632	0	7	2	5	0	219	1668	34
9016	PUDUCHERRY	TOTAL	2012	29	27	12	13	0	13	19	...	818	0	9	2	6	0	263	2060	42

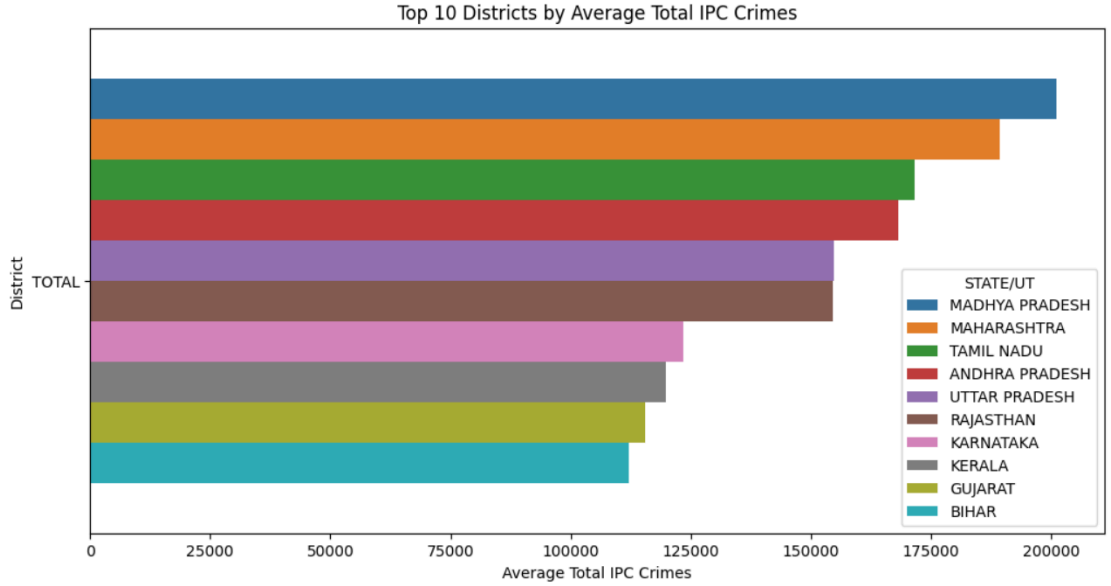
9017 rows × 34 columns

2. Dataset is already preprocessed and we can check by this code :

```
df.isnull().sum()
```

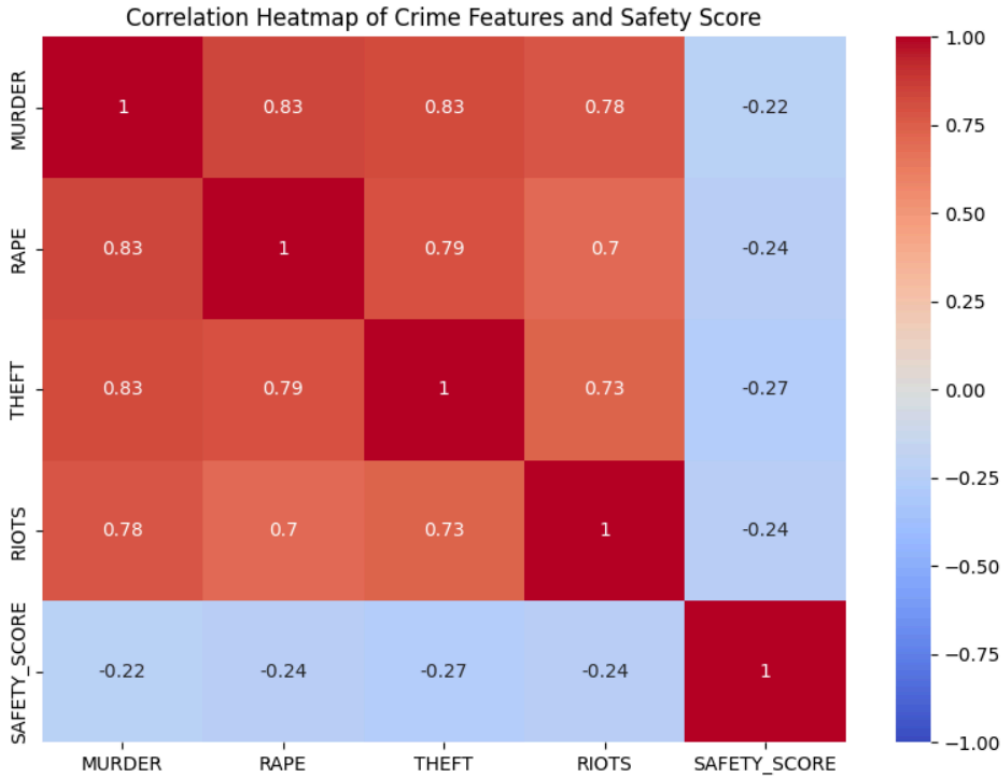
STATE/UT	0
DISTRICT	0
YEAR	0
MURDER	0
ATTEMPT TO MURDER	0
CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	0
RAPE	0
CUSTODIAL RAPE	0
OTHER RAPE	0
KIDNAPPING & ABDUCTION	0
KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS	0
KIDNAPPING AND ABDUCTION OF OTHERS	0
DACOITY	0
PREPARATION AND ASSEMBLY FOR DACOITY	0
ROBBERY	0
BURGLARY	0
THEFT	0
AUTO THEFT	0
OTHER THEFT	0
RIOTS	0
CRIMINAL BREACH OF TRUST	0
CHEATING	0

3. Understanding through graphs the districts with maximum total IPC crimes.



The data, aggregated by state/union territory (UT), shows Madhya Pradesh leading with the highest average at approximately 190,000 crimes, followed closely by Maharashtra with around 180,000, and Tamil Nadu with about 160,000. Other notable states include Andhra Pradesh, Uttar Pradesh, Rajasthan, Karnataka, Kerala, Gujarat, and Bihar, with averages ranging from 100,000 to 140,000 crimes. Integrating this information with the safety scores and crime feature analysis **allows for a more informed ranking of districts,** ensuring recommendations steer users toward regions which **exhibit relatively lower crime averages**, thereby enhancing the reliability and relevance of the safety suggestions.

4. Heatmap of features and safety score



The heatmap uses a color gradient from blue (negative correlation) to red (positive correlation), with values ranging from -1 to 1. It reveals strong **positive correlations** among crime features (e.g., **Murder and Rape at 0.83, Theft and Rape at 0.79**), indicating that districts with high rates of one crime tend to have high rates of others. Conversely, all crime features show moderate negative correlations with the Safety Score (ranging from -0.22 to -0.27), suggesting that higher crime rates are associated with lower safety scores

6. Preparing data for **XGboost model.**

```python
def prepare_data(df, features, target_col='TOTAL IPC CRIMES', min_samples=2):
    # Create safety label (1 if unsafe, 0 if safe)
    median_crime = df[target_col].median()
    df['SAFETY_LABEL'] = (df[target_col] > median_crime).astype(int)

    # Count records per district and filter
    district_counts = df.groupby(['STATE/UT', 'DISTRICT']).size()
    valid_districts = district_counts[district_counts >= min_samples].index

    # Filter original dataframe
    df_filtered = df.set_index(['STATE/UT', 'DISTRICT']).loc[valid_districts].reset_index()

    # Aggregate by district
    df_agg = df_filtered.groupby(['STATE/UT', 'DISTRICT'])[features + ['SAFETY_LABEL']].mean().reset_index()

    return df_agg
```

The **prepare_data function** processes a crime dataset by creating a safety label (1 for unsafe, 0 for safe) based on the median of 'TOTAL IPC CRIMES', ensuring districts with above-median crime rates are marked unsafe. It filters out districts with fewer than min_samples (default 2) records, using a group by operation to count records per district, and retains only valid districts. The function then **aggregates the filtered data by district,** calculating the mean of specified crime features

7. **Training** the recommendation model

```python
def train_xgboost_model(X_train, y_train, X_test, y_test):
    # Convert to DMatrix format
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test, label=y_test)

    # Parameters for binary classification
    params = {
        'objective': 'binary:logistic',
        'eval_metric': ['error', 'auc'],
        'max_depth': 5,
        'eta': 0.1,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
        'seed': 42
    }

    # Train with early stopping
    model = xgb.train(params, dtrain, num_boost_round=100,
                      early_stopping_rounds=10, evals=[(dtest, 'test')],
                      verbose_eval=True)
```

The train_xgboost_model function trains an XGBoost classifier for binary classification using training and test datasets. It converts the input data into XGBoost's DMatrix format and defines parameters for binary logistic regression, including 'objective', 'eval_metric' (error and AUC), 'max_depth' (5), 'eta' (0.1), 'subsample' and

'colsample_bytree' (0.8), and a random 'seed' (42). The **model is trained with 100 boosting rounds**, early stopping after 10 rounds if no improvement, and verbose evaluation output, enabling real-time performance monitoring on the test set.

8. **Giving** recommendations (Final Result of our model)

- Recommendation requires the state name.
- Then we ask the user to rate the crimes, they assign a weight to every crime, higher weight indicated, the particular crime should be considered more important when recommendations are made.

```
def get_safety_recommendations(model, df_agg, features, top_n=5, state=None, crime_weights=None):
    X_all = df_agg[features]
    dmatrix = xgb.DMatrix(X_all)

    # Get safety probabilities and convert to score (0-100)
    df_agg['SAFETY_PROB'] = model.predict(dmatrix)
    df_agg['SAFETY_SCORE'] = (1 - df_agg['SAFETY_PROB']) * 100

    # Apply state filter if specified
    if state:
        df_agg = df_agg[df_agg['STATE/UT'] == state.upper()]

    # Apply crime weights if specified
    if crime_weights:
        for crime, weight in crime_weights.items():
            if crime in features:
                max_val = df_agg[crime].max()
                if max_val > 0:
                    df_agg[crime + '_WEIGHTED'] = (df_agg[crime] / max_val) * weight
                    df_agg['SAFETY_SCORE'] = df_agg['SAFETY_SCORE'] - df_agg[crime + '_WEIGHTED']


    recommendations = df_agg.sort_values('SAFETY_SCORE', ascending=False).head(top_n)

    results = []
    for _, row in recommendations.iterrows():
        results.append({
            'Rank': len(results) + 1,
            'State': row['STATE/UT'],
            'District': row['DISTRICT'],
            'Safety_Score': round(row['SAFETY_SCORE'], 1),
            'Murder_Rate': round(row['MURDER'], 2),
            'Rape_Rate': round(row['RAPE'], 2),
            'Theft_Rate': round(row['THEFT'], 2),
            'Riots_Rate': round(row['RIOTS'], 2)
        })
```

The get_safety_recommendations function generates safety recommendations by predicting safety probabilities for districts using a trained XGBoost model, converting them to safety scores (0-100), and ranking districts accordingly. It filters the data by a specified state if provided, applies weighted adjustments to crime features (e.g., Murder, Rape) if crime weights are given, and **sorts districts by safety score** in descending order to **select the top n (default 5) recommendations.**

9. Result (Model evaluation)

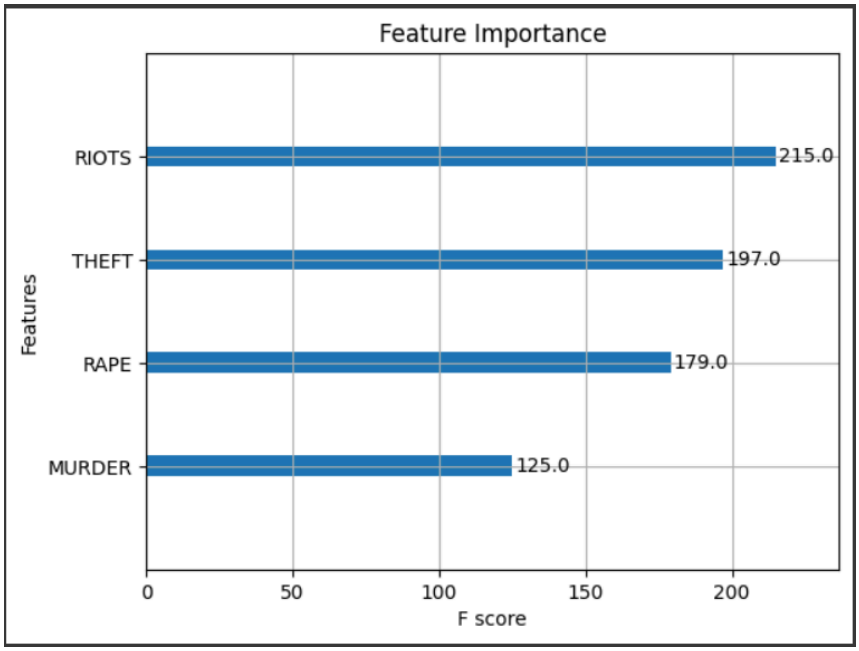
```
Model Evaluation:
Accuracy: 0.87

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.84      0.86        80
           1       0.86      0.90      0.88        86

    accuracy                           0.87       166
   macro avg       0.87      0.87      0.87       166
weighted avg       0.87      0.87      0.87       166
```

The "Model Evaluation" output for the XGBoost classifier shows an accuracy of 0.87, indicating that **87%** of the predictions align with the actual safety labels. The classification report provides detailed metrics for the binary classification (0 for safe, 1 for unsafe), with precision, recall, and f1-score all at 0.86 for class 0 and 0.88 for class 1, reflecting balanced performance across both categories. The macro and weighted averages of 0.87 across precision, recall, and f1-score, based on 166 total samples.



The "**Feature Importance**" plot displays the relative significance of crime features in the XGBoost model, measured by F-score, for predicting district safety as of April 11, 2025. Riots emerge as the most influential feature with an **F-score of 215.0**, followed closely by Theft at 197.0 and Rape at 179.0, while Murder has the lowest importance at 125.0. This indicates that riots and theft rates have a **stronger impact on the safety** score compared to murder and rape.

10. User input

```
Enter a state to focus on (or press Enter for all India): MAHARASHTRA

Searching for safe districts in MAHARASHTRA...

Number of recommendations to display (default 5): 5

Would you like to weight specific crime types more heavily?
For example, you might want to weight 'MURDER' or 'RAPE' more heavily.
Enter crime:weight pairs like 'MURDER:3 RAPE:4' or press Enter to skip
Crime weights: 'MURDER:5 RAPE:5 THEFT:5 RIOTS:3'
```

11. Recommendations according to user input

```
=== Recommended Safe Districts ===

#1: PUNE RLY., MAHARASHTRA
Safety Score: 97.7/100
Crime Rates (avg per year):
  Murder: 2.58 | Rape: 0.83
  Theft: 300.0 | Riots: 3.17

#2: SINDHUDURG, MAHARASHTRA
Safety Score: 97.5/100
Crime Rates (avg per year):
  Murder: 12.83 | Rape: 8.17
  Theft: 111.75 | Riots: 41.08

#3: HINGOLI, MAHARASHTRA
Safety Score: 92.0/100
Crime Rates (avg per year):
  Murder: 35.83 | Rape: 14.58
  Theft: 183.33 | Riots: 115.08

#4: NANDURBAR, MAHARASHTRA
Safety Score: 90.2/100
Crime Rates (avg per year):
  Murder: 35.58 | Rape: 18.42
  Theft: 171.42 | Riots: 90.75

#5: NAGPUR RLY., MAHARASHTRA
Safety Score: 88.0/100
Crime Rates (avg per year):
  Murder: 6.58 | Rape: 0.92
  Theft: 933.67 | Riots: 10.25
```

Conclusion:

The experiment successfully implemented a recommendation system for district safety using a variety of machine learning techniques, including regression, classification, clustering, decision trees, anomaly detection, dimensionality reduction, and ensemble methods, with XGBoost as the primary model. Leveraging a comprehensive crime dataset from the National Crime Records Bureau (NCRB) of India, the system effectively processed and analyzed features such as Murder, Rape, Theft, and Riots,

creating safety scores and labels based on total IPC crimes. The model achieved an accuracy of 0.87, with feature importance highlighting Riots and Theft