Analysis

on

Crime against women in India

using

Tableau and Machine Learning

From,
Aashish Minson (23030141001)
Ayush Pinge (23030141013)
Priyanshu Maski (23030141045)
Suraj Shah (23030141083)
MBA-IT (Div-A)
SICSR (2023-25)

To,
Prof. Prajakta Soman
Business Intelligence - II
SICSR

Table of Contents

Project Overview

1. Project scope

The project titled "Crime Analysis against women in India" aims to analyze and derive actionable insights from data related to crimes against women in the country India. The analysis will focus on understanding crime trends, identifying hotspots, and suggesting potential solutions to address and mitigate these issues. The project uses a combination of tools and techniques including Excel, MySQL, MS SQL, Tableau, Python programming, and data mining algorithms to perform comprehensive data analysis.

2. Problem statement

This project is to analyze the crime statistics available in the dataset from year via:

- visually using Tableau which will help decision makers to make data-driven decisions
- machine learning algorithms using Python programming which will to perform analysis using linear regression, clustering and classification which will help decision makers to take data-driven decisions.

3. Tools and Technologies

Below are some the examples of tools and technologies used in this project:

1. **Microsoft Excel:**
    - **Purpose:** Used for preliminary data exploration, basic statistical analysis, and visualization.
    - **Features:** Excel provides features like pivot tables, charts, and basic functions to summarize and visualize data quickly.

2. **MySQL:**
    - **Purpose:** Utilized for managing and querying structured data in a relational database environment.
    - **Features:** Supports complex queries, joins, and indexing to efficiently retrieve and manipulate data.

3. **MS SQL Server:**
    - **Purpose:** Employed for advanced data management, querying, and reporting.
    - **Features:** Provides robust data integration, storage, and advanced analytical capabilities, including the use of MS SQL Server and SSMS (SQL Server Management Studio).

4. **Tableau:**
    - **Purpose:** Used for interactive data visualization and business intelligence reporting.
    - **Features:** Allows users to create dashboards and visualizations that help in understanding trends and patterns in the data.

5. **Python Programming:**
    - **Purpose:** Applied for data preprocessing, advanced data analysis, and implementation of data mining algorithms.
    - **Libraries:** Utilizes libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn for data manipulation, visualization, and machine learning.

6. **Data Mining Algorithms:**
    - **Purpose:** To extract meaningful patterns and insights from the dataset.
    - **Algorithms:** Includes clustering algorithms (e.g., K-Means), classification algorithms (e.g., random forest classification), and association rule mining.

4. Project Phases

1. Data Collection and Preparation:

- **Data Acquisition:** Import the dataset into Excel, MySQL, and MS SQL Server for preliminary review and manipulation such as data pre-processing.
- **Data Cleaning:** Address missing values, outliers, and inconsistencies in the dataset. Use Python for data preprocessing and transformation.
- **Data Transformation:** Normalize and aggregate data as needed to prepare it for analysis.

2. Exploratory Data Analysis (EDA):

- **Statistical Analysis:** Use Excel and Python to perform basic statistical analysis, such as mean, median, and standard deviation.
- **Visualization:** Create initial visualizations using Excel and Tableau to understand distributions and trends in the data.

3. Advanced Data Analysis:

- **Trend Analysis:** Analyze crime trends over time and across different states and districts using Python and SQL queries.
- **Pattern Identification:** Apply data mining algorithms to identify patterns and correlations between different types of crimes and their geographical distribution.

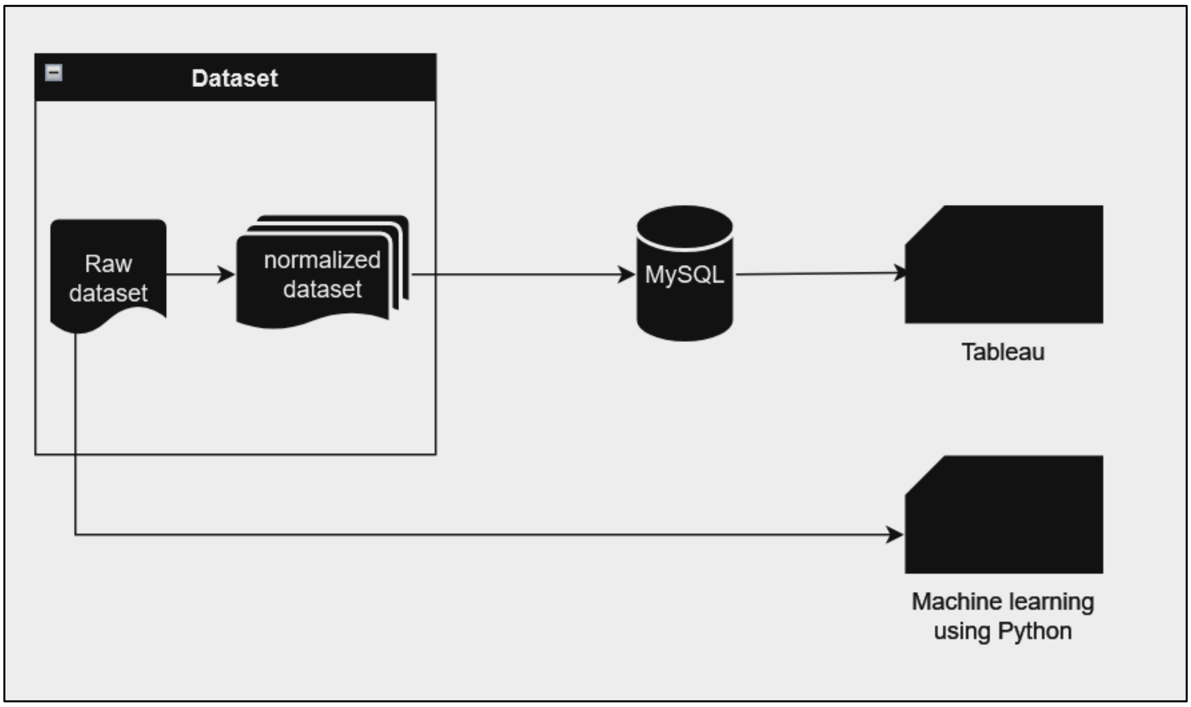
4. Data Mining and Modeling:

- **Clustering:** Use clustering algorithms (e.g., K-Means) to group districts with similar crime patterns.
- **Classification:** Implement classification algorithms (e.g., Decision Trees, Logistic Regression) to predict crime rates based on various features.
- **Association Rule Mining:** Identify frequent patterns and associations between different types of crimes.

5. Visualization:

- **Dashboards:** Create interactive dashboards in Tableau to visualize key metrics and trends.

<u>5. 3- tier BI project architecture</u>



Screenshot of 3-tier BI project architecture

Above given screenshot represents a 3-tier architecture used for data analysis and machine learning workflows, integrating a dataset, MySQL database, and tools like Tableau and Python.

Dataset Layer

The raw dataset is collected and then transformed into a normalized form, where the data is cleaned and structured to be compatible for further analysis. This step ensures that the data is standardized and optimized for querying.

Database Layer (MySQL)

The normalized dataset is stored in a MySQL database. MySQL serves as the centralized storage, allowing efficient data retrieval for analysis and modeling. It supports both visualization and machine learning processes.

Application Layer:

Tableau

Tableau connects to the MySQL database to create interactive visualizations and dashboards. It helps analysts and stakeholders visualize trends, patterns, and insights from the data.

Machine Learning using Python

Python is used for advanced analytics and machine learning on the dataset. Python tools access the MySQL-stored data, allowing machine learning models to be built and evaluated.

This architecture is scalable and modular and it helps to facilitate both business intelligence (Tableau) and data science (Python) workflows.

Dataset: "crimes_against_women.csv"

1. Dataset Overview

The dataset "crimes_against_women.csv" based on country India contains data on various types of crimes against women across different states and districts over several years. It includes 10 columns and 10,050 rows.

2. Dataset structure

1. STATE/UT:
- **Description:** Represents the state or Union Territory where the crime occurred.
- **Data Type:** Categorical.

2. DISTRICT:
- **Description:** Indicates the district within the state where the crime took place.
- **Data Type:** Categorical.

3. Year:
- **Description:** The year in which the crime was reported.
- **Data Type:** Numerical (typically in YYYY format).

4. Rape:
- **Description:** Number of reported rape cases in the given district and year.
- **Data Type:** Numerical.

5. Kidnapping and Abduction:
- **Description:** Number of cases involving kidnapping and abduction of women.
- **Data Type:** Numerical.

6. Dowry Deaths:
- **Description:** Number of deaths due to dowry-related violence.
- **Data Type:** Numerical.

7. Assault on Women with Intent to Outrage Her Modesty:
- **Description:** Number of cases involving assault with the intent to outrage a woman's modesty.
- **Data Type:** Numerical.

8. Insult to Modesty of Women:
- **Description:** Number of incidents where women's modesty was insulted.
- **Data Type:** Numerical.

9. Cruelty by Husband or His Relatives:

- **Description:** Number of cases of cruelty inflicted by a woman’s husband or his relatives.
- **Data Type:** Numerical.

10. Importation of Girls:

- **Description:** Number of cases related to the illegal importation of girls.
- **Data Type:** Numerical.

<u>Dataset Components:</u>

1. **Crime Types:**
    - Includes various categories of crimes against women such as
        - Rape,
        - Kidnapping and Abduction,
        - Dowry Deaths,
        - Assault on Women with Intent to Outrage Modesty,
        - Insult to Modesty of Women, Cruelty by Husband or Relatives, and
        - Importation of Girls.
    - Each crime type is identified by a unique Crime_ID and described with a Crime_name.

2. **Geographical Information:**
    - **States:** The dataset covers all Indian states and union territories, each identified by a unique State_ID.
    - **Districts:** The dataset includes districts within each state, with each district assigned a unique District_ID and linked to a specific State_ID.

3. **Crime Data:**
    - Records of crimes include attributes such as Year, Crime_ID_FK (foreign key linking to the crime type), State_ID_FK (foreign key linking to the state), District_ID_FK (foreign key linking to the district), and Count (number of occurrences of the crime).
    - This data is structured to allow for detailed analysis of crime trends by state, district, and year.

Please find below for sample screenshot of the dataset:

	A	B	C	D	E	F	G	H	I	J	K
1		STATE/UT	DISTRICT	Year	Rape	Kidnapping and Abduction	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
2	0	ANDHRA PRADESH	ADILABAD	2001	50	30	16	149	34	175	0
3	1	ANDHRA PRADESH	ANANTAPUR	2001	23	30	7	118	24	154	0
4	2	ANDHRA PRADESH	CHITTOOR	2001	27	34	14	112	83	186	0
5	3	ANDHRA PRADESH	CUDDAPAH	2001	20	20	17	126	38	57	0
6	4	ANDHRA PRADESH	EAST GODAVARI	2001	23	26	12	109	58	247	0
7	5	ANDHRA PRADESH	GUNTAKAL RLY.	2001	0	0	0	1	0	0	0
8	6	ANDHRA PRADESH	GUNTUR	2001	54	51	7	139	129	378	0
9	7	ANDHRA PRADESH	HYDERABAD CITY	2001	37	39	24	118	27	746	0
10	8	ANDHRA PRADESH	KARIMNAGAR	2001	56	49	62	414	81	224	0
11	9	ANDHRA PRADESH	KHAMMAM	2001	47	30	17	180	336	172	0
12	10	ANDHRA PRADESH	KRISHNA	2001	37	21	10	208	72	265	0
13	11	ANDHRA PRADESH	KURNOOL	2001	29	47	13	141	107	92	0
14	12	ANDHRA PRADESH	MAHABOOBNAGAR	2001	59	27	14	176	41	69	0
15	13	ANDHRA PRADESH	MEDAK	2001	35	20	26	100	25	192	0
16	14	ANDHRA PRADESH	NALGONDA	2001	35	19	31	188	59	214	0
17	15	ANDHRA PRADESH	NELLORE	2001	46	80	10	207	228	287	0
18	16	ANDHRA PRADESH	NIZAMABAD	2001	21	21	19	55	15	228	0

3. Dataset cleaning

Data cleaning is the process of identifying and rectifying errors, inconsistencies, and inaccuracies in the dataset to ensure that the data is accurate, complete, and suitable for analysis. This process involves several steps to prepare the data for further analysis and modeling.

Steps in Data Cleaning:

1. Handling Missing Values:

- **Identification:** Identify missing or null values in the dataset.
- **Imputation:** Fill missing values with appropriate data, such as mean, median, mode, or by using more advanced imputation techniques. For categorical data, missing values might be filled with the most frequent category or a placeholder value.

2. Removing Duplicates:

- **Identification:** Detect duplicate records that may have been entered more than once.
- **Removal:** Eliminate duplicate rows to ensure each record is unique and avoid redundancy in the dataset.

3. Correcting Inconsistencies:

- **Standardization:** Ensure consistency in data formats, such as date formats (e.g., YYYY-MM-DD), capitalization in text fields, and standardized state and district names.
- **Validation:** Cross-check data against known standards or reference datasets to correct inconsistencies, such as incorrect state or district codes.

4. Outlier Detection and Handling:

- **Identification:** Use statistical methods or visualizations to identify outliers or anomalies in numerical data.

- **Handling:** Investigate the cause of outliers and decide whether to remove them or transform them based on their relevance and impact on analysis.

5. Data Transformation:

- **Normalization:** Apply normalization techniques to ensure numerical values fall within a specific range or scale, which can be crucial for certain analyses and algorithms.
- **Encoding:** Convert categorical variables into numerical formats using techniques like one-hot encoding or label encoding for use in machine learning models.

6. Data Validation:

- **Accuracy Checks:** Verify the accuracy of the data by comparing it with source documents or external datasets.
- **Consistency Checks:** Ensure that the data adheres to predefined rules and constraints, such as valid ranges for numerical values or proper relationships between tables.

Tools and Techniques:

- **Python:** Libraries like pandas and NumPy are used for data cleaning tasks, including handling missing values, removing duplicates, and transforming data.
- **Excel:** Used for preliminary data cleaning and quick validation of data.
- **SQL:** Queries in SQL are used to identify and rectify data issues, such as duplicates and inconsistencies.

4. Dataset normalization

Please find below for tables which were created after normalization. Here tables are created for duplicate data like states, crimes etc., where dimension tables are created and related primary is used as foreign key in fact table

- **Dimension tables:**
    - *dim_crime_tbl*: Lists types of crimes.
    - *dim_state_tbl*: Lists states and union territories.
    - *subdim_district_tbl*: Lists districts within states.

- **Fact table:**
    - *fact_crimedetails_tbl*: Contains factual data about crime occurrences.

Please find below for sample screenshot of normalized dataset:

Crime_ID	Crime_name
CID001	Rape
CID002	Kidnapping and Abduction
CID003	Dowry Deaths
CID004	Assault on women with intent to outrage her modesty
CID005	Insult to modesty of Women
CID006	Cruelty by Husband or his Relatives
CID007	Importation of Girls

Dim_state_tbl | Subdim_district_tbl | Dim_crime_tbl | fact_crimeDetails_tbl

State_ID	State_name
ST001	A & N ISLANDS
ST002	ANDHRA PRADESH
ST003	ARUNACHAL PRADESH
ST004	ASSAM
ST005	BIHAR
ST006	CHANDIGARH
ST007	CHHATTISGARH
ST008	D & N HAVELI

Dim_state_tbl | Sub

District_ID	State_ID_FK	District_name
DT00001	ST001	ANDAMAN
DT00002	ST001	NICOBAR
DT00003	ST001	A and N ISLANDS
DT00004	ST001	NORTH
DT00005	ST001	SOUTH
DT00006	ST001	South Andaman
DT00007	ST001	North & Middle Andaman
DT00008	ST002	ADILABAD

Dim_state_tbl | Subdim_district_tbl

State_ID_FK	District_ID_FK	Year	Crime_ID_FK	Count
ST001	DT00001	2001	CID001	3
ST001	DT00001	2001	CID002	2
ST001	DT00001	2001	CID003	0
ST001	DT00001	2001	CID004	18
ST001	DT00001	2001	CID005	1
ST001	DT00001	2001	CID006	9
ST001	DT00001	2001	CID007	0
ST001	DT00002	2001	CID001	0

Dim_crime_tbl | fact_crimeDetails_tbl

Dataset into Datawarehouse

1. Dataset import into SQL Server as a Database

Data import involves transferring data from various sources into an SQL Server database, where it can be stored, managed, and queried efficiently. This process includes creating database schemas, loading data into tables, and ensuring data integrity during the import process.

Steps for Data Import:

1. Creating Database Schema:

- **Database Creation:** Use SQL Server Management Studio (SSMS) or SQL commands to create a new database. Define the database schema, including tables, columns, data types, and relationships.

```
CREATE DATABASE dw_crime_against_women_india;
USE dw_crime_against_women_india;
```

SQL code for creating database

2. Defining Tables:

- **Table Creation:** Create tables according to the schema defined in the dataset. Use SQL DDL (Data Definition Language) commands to define tables and their structures.

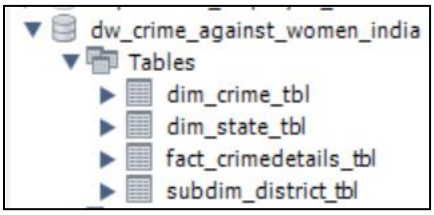
```
CREATE TABLE Dim_state_tbl (
State_ID VARCHAR(10) PRIMARY KEY,
State_name VARCHAR(255) NOT NULL
   );

   CREATE TABLE Subdim_district_tbl (
District_ID VARCHAR(20) PRIMARY KEY,
State_ID_FK VARCHAR(10),
District_name VARCHAR(255) NOT NULL,
FOREIGN KEY (State_ID_FK) REFERENCES Dim_state_tbl(State_ID)
   );

   CREATE TABLE Dim_crime_tbl (
Crime_ID VARCHAR(20) PRIMARY KEY,
Crime_name VARCHAR(255) NOT NULL
   );

   CREATE TABLE fact_crimeDetails_tbl (
State_ID_FK VARCHAR(10),
District_ID_FK VARCHAR(20),
Year INT NOT NULL,
Crime_ID_FK VARCHAR(20),
Count INT NOT NULL,
FOREIGN KEY (State_ID_FK) REFERENCES dim_state_tbl(State_ID),
```

```
FOREIGN KEY (District_ID_FK) REFERENCES subdim_district_tbl(District_ID),
FOREIGN KEY (Crime_ID_FK) REFERENCES dim_crime_tbl(Crime_ID)
   );
```



Screenshot of database and related tables in MySQL

3. Data Import:

- **Import Data:** Use SQL Server's import tools or SQL commands to load data into the tables. Data can be imported from CSV files, Excel spreadsheets, or other sources.
    - **Using SSMS:** Navigate to the "Import Data" option, select the source file, and map the columns to the database schema.
    - **Using SQL Commands:** Use the BULK INSERT or OPENROWSET commands to import data from files.

```
BULK INSERT dim_crime_tbl
FROM 'C:\path\to\dim_crime_tbl.csv'
WITH (FIELDTERMINATOR = ',', ROWTERMINATOR = '\n');
```

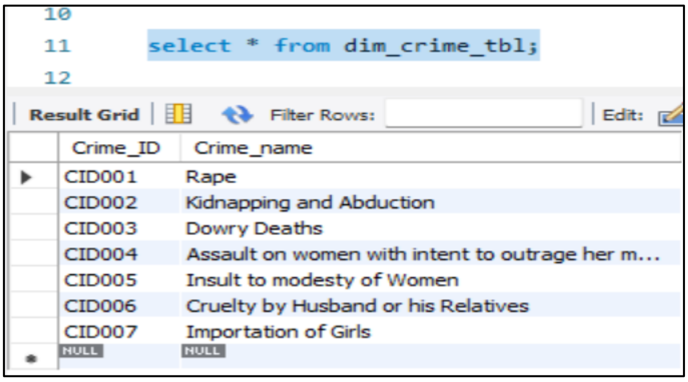
Screenshot of bulk insert

4. Data Validation:

- **Check Data Integrity:** Verify that the data has been imported correctly by running queries to check data counts, sample records, and ensure no data loss or corruption has occurred.

```
SELECT COUNT(*) FROM dim_crime_tbl;
SELECT * FROM dim_crime_tbl WHERE Crime_ID = 'CID001';
```

Screenshot of SQL query



Crime_ID	Crime_name
CID001	Rape
CID002	Kidnapping and Abduction
CID003	Dowry Deaths
CID004	Assault on women with intent to outrage her m...
CID005	Insult to modesty of Women
CID006	Cruelty by Husband or his Relatives
CID007	Importation of Girls
NULL	NULL

Screenshot of SQL query

5. Testing and Validation:

- **Data Queries:** Run sample queries to ensure the imported data is accessible and correctly structured.
- **Consistency Checks:** Validate relationships between tables using joins and referential integrity constraints.

Tools and Techniques used for data import:

- **SQL Server Management Studio (SSMS):** Provides graphical interfaces for creating databases, tables, and importing data.
- **SQL Scripts:** Used for automated and batch processing of data import tasks.
- **Data Import Wizards:** Tools within SQL Server that assist in importing data from various file formats.

2. Datawarehouse schema

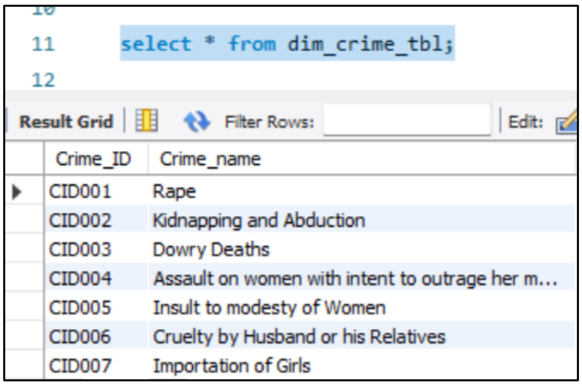
Database Name: *dw_crime_against_women_india*

The database is designed to store and analyze crime data specific to crimes against women in India. The schema includes several tables to capture crime details, states, districts, and the relationships between these entities.

Tables and Their Structures:

1. dim_crime_tbl

- **Purpose:** Contains the types of crimes.
- **Columns:**
  - Crime_ID (varchar(20)): Unique identifier for each type of crime.
  - Crime_name (varchar(255)): Description of the crime.
- **Data:** Includes types of crimes like Rape, Kidnapping and Abduction, Dowry Deaths, etc.



```
11    select * from dim_crime_tbl;
12
```

Crime_ID	Crime_name
CID001	Rape
CID002	Kidnapping and Abduction
CID003	Dowry Deaths
CID004	Assault on women with intent to outrage her m...
CID005	Insult to modesty of Women
CID006	Cruelty by Husband or his Relatives
CID007	Importation of Girls

Screenshot of dim_crime_tbl

2. dim_state_tbl

- **Purpose:** Contains the states and union territories in India.
- **Columns:**
    - State_ID (varchar(10)): Unique identifier for each state or territory.
    - State_name (varchar(255)): Name of the state or territory.
- **Data:** Lists all Indian states and union territories such as Andhra Pradesh, Bihar, Delhi, etc.

```
select * from dim_state_tbl;
```

Result Grid Filter Rows:

State_ID	State_name
ST001	A & N ISLANDS
ST002	ANDHRA PRADESH
ST003	ARUNACHAL PRADESH
ST004	ASSAM
ST005	BIHAR
ST006	CHANDIGARH
ST007	CHHATTISGARH
ST008	D & N HAVELI

Screenshot of dim_state_tbl

3. subdim_district_tbl

- **Purpose:** Contains details about districts within states.
- **Columns:**
    - District_ID (varchar(20)): Unique identifier for each district.
    - State_ID_FK (varchar(10)): Foreign key linking to dim_state_tbl.
    - District_name (varchar(255)): Name of the district.
- **Data:** Represents various districts within the states, linked to their respective states.

```
select * from subdim_district_tbl;
```

Result Grid Filter Rows: Edit:

District_ID	State_ID_FK	District_name
DT00001	ST001	ANDAMAN
DT00002	ST001	NICOBAR
DT00003	ST001	A and N ISLANDS
DT00004	ST001	NORTH
DT00005	ST001	SOUTH
DT00006	ST001	South Andaman
DT00007	ST001	North & Middle Andaman
DT00008	ST002	ADILABAD

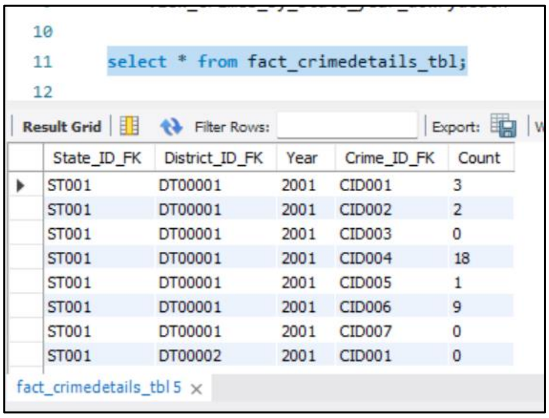
Screenshot of subdim_district_tbl

4. fact_crimedetails_tbl

- **Purpose:** Contains factual data about crimes.
- **Columns:**
    - State_ID_FK (varchar(10)): Foreign key linking to dim_state_tbl.

- District_ID_FK (varchar(20)): Foreign key linking to subdim_district_tbl.
- Year (int): Year in which the crime occurred.
- Crime_ID_FK (varchar(20)): Foreign key linking to dim_crime_tbl.
- Count (int): Number of occurrences of the crime.

- **Data:** Provides detailed crime statistics by state, district, and year.



```
select * from fact_crimedetails_tbl;
```

Result Grid | Filter Rows: | Export:

State_ID_FK	District_ID_FK	Year	Crime_ID_FK	Count
ST001	DT00001	2001	CID001	3
ST001	DT00001	2001	CID002	2
ST001	DT00001	2001	CID003	0
ST001	DT00001	2001	CID004	18
ST001	DT00001	2001	CID005	1
ST001	DT00001	2001	CID006	9
ST001	DT00001	2001	CID007	0
ST001	DT00002	2001	CID001	0

fact_crimedetails_tbl 5

Screenshot of fact_crimedetails_tbl

Data Management:

- **Character Set and Collation:** utf8mb4 character set and utf8mb4_0900_ai_ci collation are used to support a wide range of characters, including special symbols and emojis.
- **Foreign Keys:** Ensures referential integrity between tables, linking crime details to specific crimes, states, and districts.

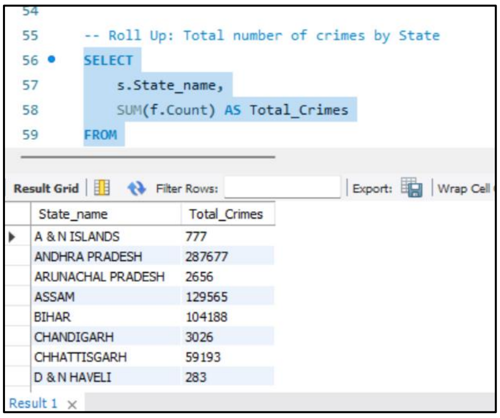
3. OLAP Operations

OLAP (Online Analytical Processing) operations are used for multidimensional data analysis, allowing users to quickly retrieve complex analytical queries. OLAP is typically applied in data warehouses to help in decision-making processes. The four primary OLAP operations are:

1. Roll-up (Aggregation): This operation summarizes or aggregates data along a hierarchy. For example, crime data can be rolled by based on district or country level, providing an aggregated view at a higher granularity.

```
-- Roll Up: Total number of crimes by State
SELECT
    s.State_name,
    SUM(f.Count) AS Total_Crimes
FROM
```

```
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
GROUP BY
    s.State_name
WITH ROLLUP;
```

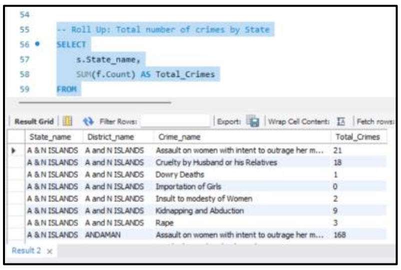


State_name	Total_Crimes
A & N ISLANDS	777
ANDHRA PRADESH	287677
ARUNACHAL PRADESH	2656
ASSAM	129565
BIHAR	104188
CHANDIGARH	3026
CHHATTISGARH	59193
D & N HAVELI	283

Screenshot of roll-up OLAP operation

2. Drill-down: This is the reverse of the roll-up operation, where data is explored in more detail by moving down a hierarchy. It allows the user to view more granular data.

```
  -- Drill Down: Number of crimes per State, District, and Crime Type
SELECT
    s.State_name,
    d.District_name,
    c.Crime_name,
    SUM(f.Count) AS Total_Crimes
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
GROUP BY
    s.State_name, d.District_name, c.Crime_name
ORDER BY
    s.State_name, d.District_name, c.Crime_name;
```

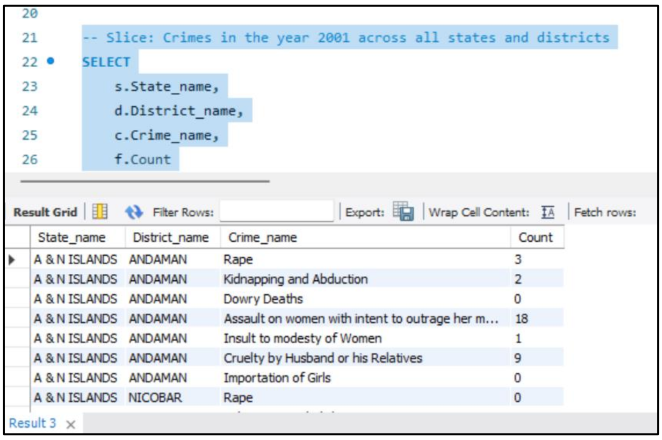


State_name	District_name	Crime_name	Total_Crimes
A & N ISLANDS	A and N ISLANDS	Assault on women with intent to outrage her m...	21
A & N ISLANDS	A and N ISLANDS	Cruelty by Husband or his Relatives	18
A & N ISLANDS	A and N ISLANDS	Dowry Deaths	1
A & N ISLANDS	A and N ISLANDS	Importation of Girls	0
A & N ISLANDS	A and N ISLANDS	Insult to modesty of Women	2
A & N ISLANDS	A and N ISLANDS	Kidnapping and Abduction	9
A & N ISLANDS	A and N ISLANDS	Rape	3
A & N ISLANDS	ANDAMAN	Assault on women with intent to outrage her m...	168

Screenshot of drill-down OLAP operation

3. Slice: This operation allows users to select a specific dimension from the OLAP cube, reducing the complexity by focusing on one particular set of data.

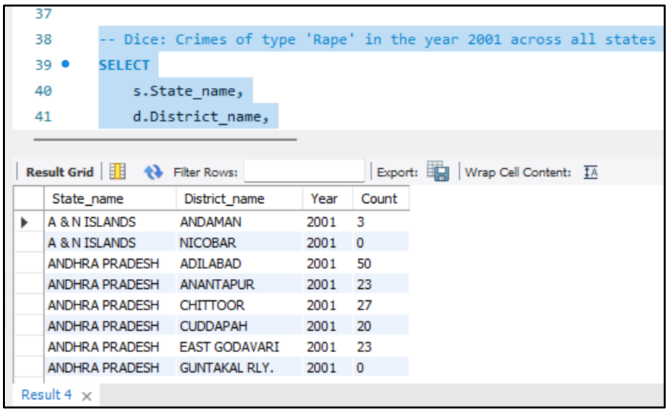
```sql
-- Slice: Crimes in the year 2001 across all states and districts
SELECT
    s.State_name,
    d.District_name,
    c.Crime_name,
    f.Count
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    f.Year = 2001;
```



Screenshot of slice OLAP Operation

4. Dice: This operation allows users to select a sub-cube by specifying values for multiple dimensions.

```sql
  -- Dice: Crimes of type 'Rape' in the year 2001 across all states
SELECT
    s.State_name,
    d.District_name,
    f.Year,
    f.Count
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    f.Year = 2001 AND c.Crime_name = 'Rape';
```

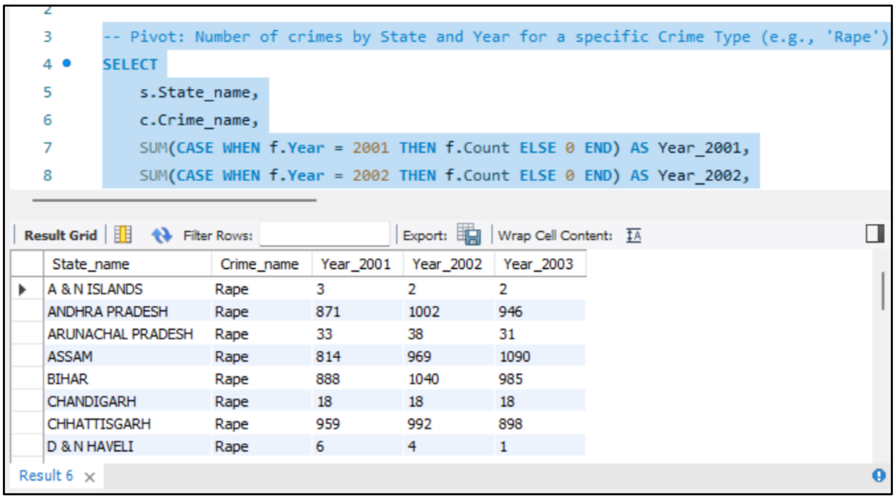


State_name	District_name	Year	Count
A & N ISLANDS	ANDAMAN	2001	3
A & N ISLANDS	NICOBAR	2001	0
ANDHRA PRADESH	ADILABAD	2001	50
ANDHRA PRADESH	ANANTAPUR	2001	23
ANDHRA PRADESH	CHITTOOR	2001	27
ANDHRA PRADESH	CUDDAPAH	2001	20
ANDHRA PRADESH	EAST GODAVARI	2001	23
ANDHRA PRADESH	GUNTAKAL RLY.	2001	0

Screenshot of Dice OLAP Operation

5. Pivot (Rotation): This operation rotates the data axes to offer a different view or perspective of the data.

```
-- Pivot: Number of crimes by State and Year for a specific Crime Type (e.g., 'Rape')
SELECT
    s.State_name,
    c.Crime_name,
    SUM(CASE WHEN f.Year = 2001 THEN f.Count ELSE 0 END) AS Year_2001,
    SUM(CASE WHEN f.Year = 2002 THEN f.Count ELSE 0 END) AS Year_2002,
    SUM(CASE WHEN f.Year = 2003 THEN f.Count ELSE 0 END) AS Year_2003
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    c.Crime_name = 'Rape'
GROUP BY
    s.State_name, c.Crime_name;
```



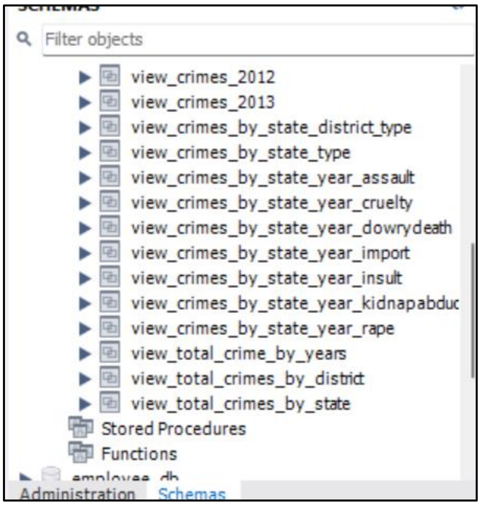
State_name	Crime_name	Year_2001	Year_2002	Year_2003
A & N ISLANDS	Rape	3	2	2
ANDHRA PRADESH	Rape	871	1002	946
ARUNACHAL PRADESH	Rape	33	38	31
ASSAM	Rape	814	969	1090
BIHAR	Rape	888	1040	985
CHANDIGARH	Rape	18	18	18
CHHATTISGARH	Rape	959	992	898
D & N HAVELI	Rape	6	4	1

Screenshot of Pivot OLAP Operation

These OLAP operations allow users for:

- Efficient data analysis
- Multi-dimensional analysis
- Aggregated Views and Hierarchies
- Interactive data exploration
- Faster query response
- Trend analysis and forecasting
- Decision support

4. OLAP SQL Views



Screenshot of views

4.1 Slice OLAP Views

Definition: This view isolates crime data for the years 2001-2013, focusing on the count of different types of crimes across various states and districts.

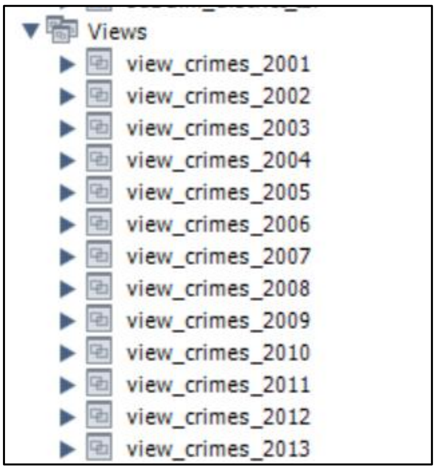
Sample SQL Query:

```
CREATE VIEW view_crimes_2001
AS
SELECT
    s.State_name,
    d.District_name,
    c.Crime_name,
    f.Count
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
```

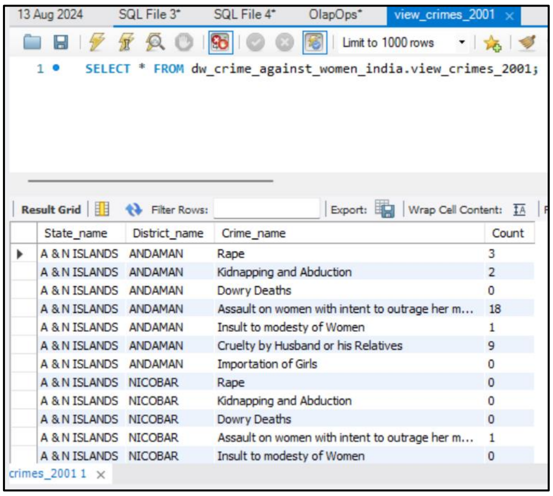
```sql
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    f.Year = '2001';
```

Description:

- **Tables Involved:**
    - *fact_crimeDetails_tbl***:** Contains detailed records of crime counts.
    - *Dim_state_tbl***:** Provides state-level information.
    - *Subdim_district_tbl***:** Details on district-level information.
    - *Dim_crime_tbl***:** Lists different types of crimes.
- **Purpose:**
    - To provide a snapshot of crime data for the year 2001, including state, district, and crime type.
    - Useful for year-by-year comparison and trend analysis over multiple years.



Screenshot of views created on OLAP slice operation



Screenshot of a slice view table

4.2 Pivot OLAP views

Definition: This view extracts crime data on the basis of crime type by state crime type and year from year 2001-2013.

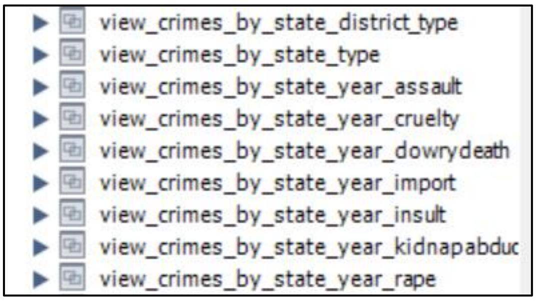
Sample SQL Query:

```
CREATE VIEW view_crimes_by_state_year_rape
AS
SELECT
    s.State_name,
    c.Crime_name,
    SUM(CASE WHEN f.Year = 2001 THEN f.Count ELSE 0 END) AS Year_2001,
    SUM(CASE WHEN f.Year = 2002 THEN f.Count ELSE 0 END) AS Year_2002,
    SUM(CASE WHEN f.Year = 2003 THEN f.Count ELSE 0 END) AS Year_2003
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    c.Crime_name = 'Rape'
GROUP BY
    s.State_name, c.Crime_name;
```

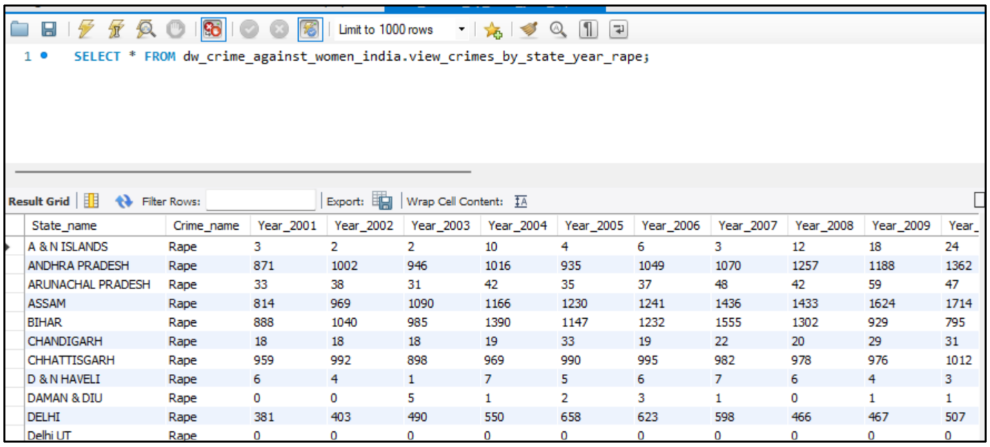
Description:

- **Tables involved:**
    - *fact_crimeDetails_tbl***:** Contains detailed records of crime counts.
    - *Dim_state_tbl***:** Provides state-level information.
    - *Dim_crime_tbl***:** Lists different types of crimes.

- **Purpose:**
    - Allows for comparison of crime data between states for a specific crime for given years.



Screenshot of Pivot OLAP views



```
SELECT * FROM dw_crime_against_women_india.view_crimes_by_state_year_rape;
```

State_name	Crime_name	Year_2001	Year_2002	Year_2003	Year_2004	Year_2005	Year_2006	Year_2007	Year_2008	Year_2009	Year_
A & N ISLANDS	Rape	3	2	2	10	4	6	3	12	18	24
ANDHRA PRADESH	Rape	871	1002	946	1016	935	1049	1070	1257	1188	1362
ARUNACHAL PRADESH	Rape	33	38	31	42	35	37	48	42	59	47
ASSAM	Rape	814	969	1090	1166	1230	1241	1436	1433	1624	1714
BIHAR	Rape	888	1040	985	1390	1147	1232	1555	1302	929	795
CHANDIGARH	Rape	18	18	18	19	33	19	22	20	29	31
CHHATTISGARH	Rape	959	992	898	969	990	995	982	978	976	1012
D & N HAVELI	Rape	6	4	1	7	5	6	7	6	4	3
DAMAN & DIU	Rape	0	0	5	1	2	3	1	0	1	1
DELHI	Rape	381	403	490	550	658	623	598	466	467	507
Delhi UT	Rape	0	0	0	0	0	0	0	0	0	0

Screenshot of view_crimes_by_state_year_rape

4.3 Rollup OLAP Views

Definition: This view isolates crime data for the basis of year or state or district which is rolled up or aggregated.

Sample SQL Query:

```
CREATE VIEW view_crimes_by_state_district_type
AS
SELECT
    s.State_name,
    d.District_name,
    c.Crime_name,
    SUM(f.Count) AS Total_Crimes
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
GROUP BY
    s.State_name, d.District_name, c.Crime_name
ORDER BY

    s.State_name, d.District_name, c.Crime_name;
```

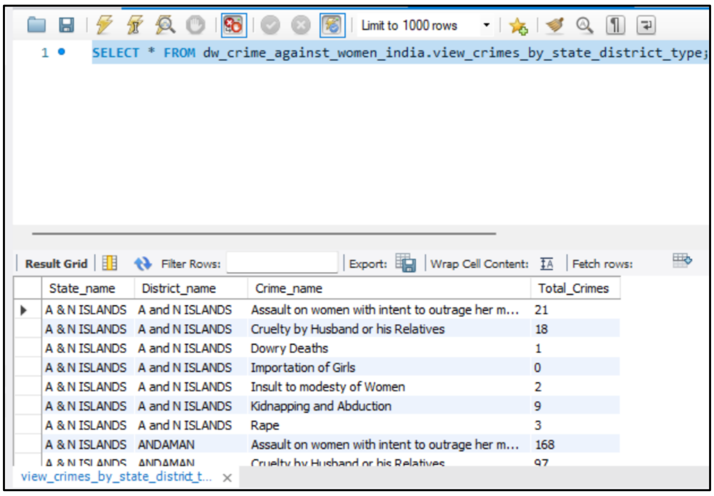
Description:

- **Tables Involved:**
    - *fact_crimeDetails_tbl*: Contains detailed records of crime counts.
    - *Dim_state_tbl*: Provides state-level information.
    - *Subdim_district_tbl*: Details on district-level information.
    - *Dim_crime_tbl*: Lists different types of crimes.

- **Purpose:**
    - To provide a snapshot of detailed crime data on the basis of state or district

- Useful for r comparison and trend analysis over district or state.

Screenshot of views created on OLAP rollup operation



State_name	District_name	Crime_name	Total_Crimes
A & N ISLANDS	A and N ISLANDS	Assault on women with intent to outrage her m...	21
A & N ISLANDS	A and N ISLANDS	Cruelty by Husband or his Relatives	18
A & N ISLANDS	A and N ISLANDS	Dowry Deaths	1
A & N ISLANDS	A and N ISLANDS	Importation of Girls	0
A & N ISLANDS	A and N ISLANDS	Insult to modesty of Women	2
A & N ISLANDS	A and N ISLANDS	Kidnapping and Abduction	9
A & N ISLANDS	A and N ISLANDS	Rape	3
A & N ISLANDS	ANDAMAN	Assault on women with intent to outrage her m...	168

Screenshot of a rollup view table

4.4 Drill through OLAP Views

Definition: This view isolates crime data granularly on the basis of state or district_type.

Sample SQL Query:

```
CREATE VIEW view_crimes_by_state_district_delhi_type
AS
SELECT
    f.Year,
    s.State_name,
    d.District_name,
    c.Crime_name,
    f.Count AS Crime_Count
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE
    s.State_name = 'DELHI' -- Replace with the selected state
ORDER BY

    f.Year, d.District_name;
```

Description:

- **Tables Involved:**
  - *fact_crimeDetails_tbl*: Contains detailed records of crime counts.

- *Dim_state_tbl*: Provides state-level information.
- *Subdim_district_tbl*: Details on district-level information.
- *Dim_crime_tbl*: Lists different types of crimes.

4.5 Dice OLAP Views

Definition: This view isolates crime data for a year and the type of crime across various states and districts.

Sample SQL Query:

```
CREATE VIEW view_crimes_by_state_district_Rape_2001
AS
SELECT
    s.State_name,
    d.District_name,
    f.Year,
    f.Count
FROM
    fact_crimeDetails_tbl f
JOIN
    Dim_state_tbl s ON f.State_ID_FK = s.State_ID
JOIN
    Subdim_district_tbl d ON f.District_ID_FK = d.District_ID
JOIN
    Dim_crime_tbl c ON f.Crime_ID_FK = c.Crime_ID
WHERE

    f.Year = '2001' AND c.Crime_name = 'Rape';
```

Description:

- **Tables Involved:**
    - *fact_crimeDetails_tbl*: Contains detailed records of crime counts.
    - *Dim_state_tbl*: Provides state-level information.
    - *Subdim_district_tbl*: Details on district-level information.
    - *Dim_crime_tbl*: Lists different types of crimes.

Tableau

Tableau is a powerful data visualization and business intelligence (BI) tool that allows users to analyze and visualize large sets of data interactively. It transforms raw data into easily understandable, shareable, and actionable insights through visual dashboards, charts, graphs, and maps. Tableau is widely used for creating visual analytics that can be explored interactively to find patterns, trends, and insights.

It is used primarily for:

- Data visualization
- Business intelligence
- Real-time data analytics
- Ease of use
- Integration with multiple sources
- Interactive dashboards
- Collaboration sharing

In summary, Tableau stands out for its intuitive interface, advanced visualization capabilities, and wide data source compatibility, making it the preferred tool for users needing robust, interactive data visualizations.

1. Tableau: Connection with MySQL database

Tableau can easily connect to MySQL databases, allowing users to visualize and analyze data stored in MySQL.

Below are the steps which will guide to connect with MySQL.

Step 1: Open Tableau

- Launch Tableau Desktop or Tableau Public on our system.

Step 2: Choose MySQL as the Data Source

- In Tableau's "Connect" pane, we'll see options for different data sources.
- Under **To a Server**, select **MySQL** as the data connection option.

Step 3: Provide MySQL Credentials

- A new window will appear where we need to enter MySQL connection details:
  - **Server**: The hostname or IP address of the MySQL server (e.g., localhost, 127.0.0.1, or a specific server address).
  - **Port**: The MySQL port number (default is 3306).

    - **Username**: The username for accessing the MySQL database.
    - **Password**: The password associated with the MySQL user account.
    - **Database**: (Optional) Specify the particular database name we want to connect to.

Step 4: Connect to MySQL

- After entering the credentials, click **Sign In**. Tableau will attempt to establish a connection to the MySQL server.
    - If the MySQL connector is not installed, Tableau will prompt you to download and install the MySQL ODBC connector (driver). Follow the instructions to install it if necessary.

Step 5: Select Database and Tables

- Once connected, we will see a list of available databases in your MySQL instance on the left pane.
- Select the database you want to work with that is *dw_crime_against_women_india.*
- Drag and drop the tables you want to analyze into the canvas area for visualization and analysis.

Step 6: Data Preparation (Joins, Filtering, etc.)

- After selecting the needed tables, we can define relationships (JOINs) between them, filter data, or add calculated fields directly in Tableau's data pane before visualizing it.

Step 7: Start Creating Visualizations

- Once the data is imported and prepared, we can begin creating visualizations using Tableau's drag-and-drop interface.
- Use the **Sheets** and **Dashboard** tabs to create different charts, graphs, maps, and other visual representations.

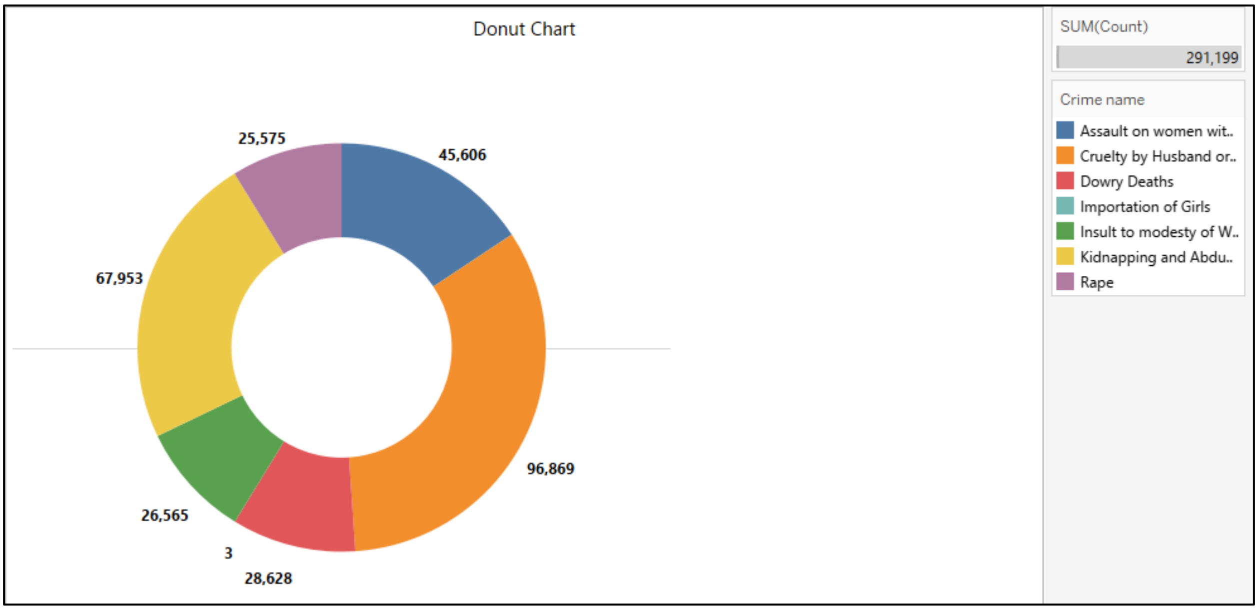
Types of Connections in Tableau

- **Live Connection**: When you select a live connection, Tableau will query the MySQL database in real-time. Any updates to the data in MySQL will be reflected instantly in Tableau dashboards. This is useful for real-time or dynamic datasets but can slow down performance if the MySQL database is large or complex.
- **Extract (In-memory)**: An extract is a snapshot of the data from MySQL, stored locally in Tableau's in-memory engine. This method improves performance, especially for large datasets, by allowing faster querying and visualization. However, updates to the MySQL database won't be reflected in Tableau until the extract is refreshed.

2. Tableau: worksheet

2.1 Donut Chart:

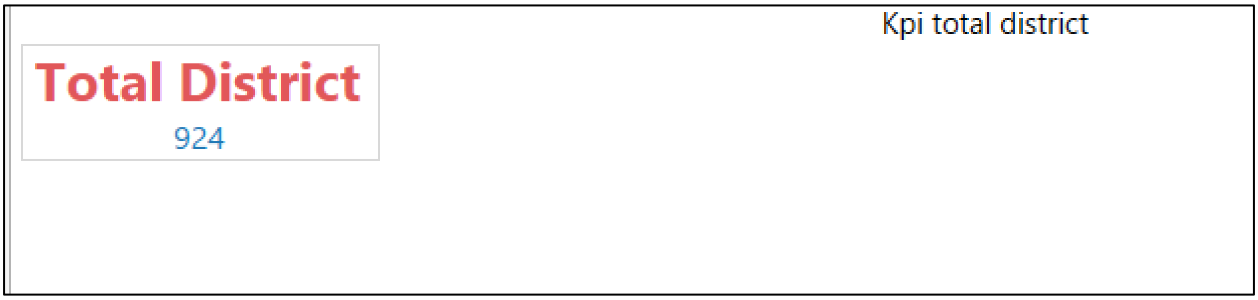
It provides the count of total crime based on their crime type. It helps to analyze the crime which is high and low. Filters like state names can show the count of crimes according to the states.



Screenshot of donut chart

2.2 Key Performance Indicator:

It shows the total number of districts in India. The filter state name shows the total number of districts in every state.

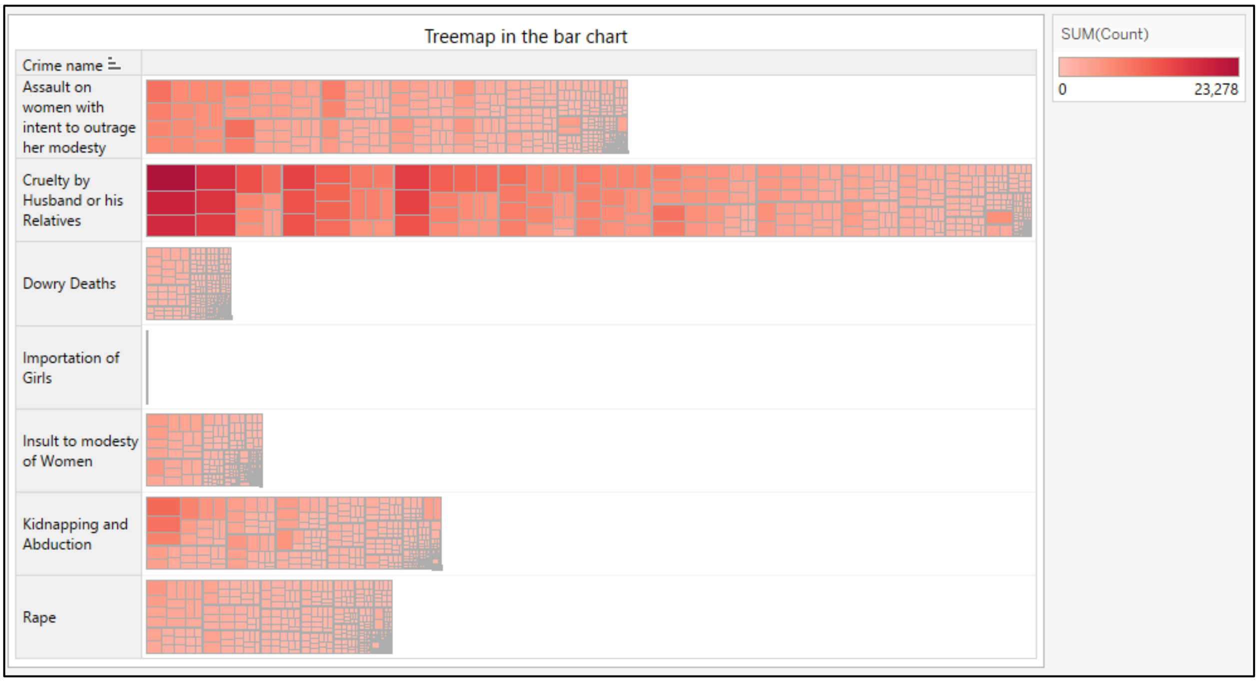


Screenshot of KPI of Total district

2.3 Tree map in the bar chart:

This graph shows the total number of crimes according to type in each bar form. We can see that the cruelty by the husband or his relatives is very high. The treemap inside the bar chart shows the saturation of the crime year-wise from 2001-2014.

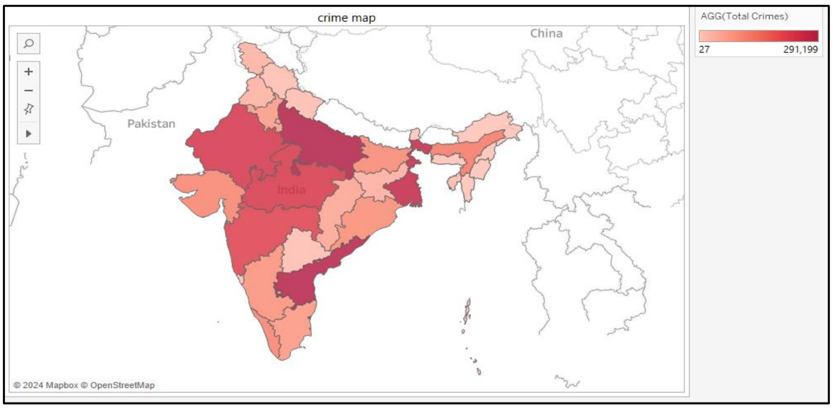
This graph has filters like state name and crime type which upon selection shows the crime rate and saturation of the crime over period of time.



Screenshot of Treemap

2.4 Geographical distribution of the crime:

This graph shows the number of crimes happening in each state, a color filter is used which says the darker the color higher is the crime count.

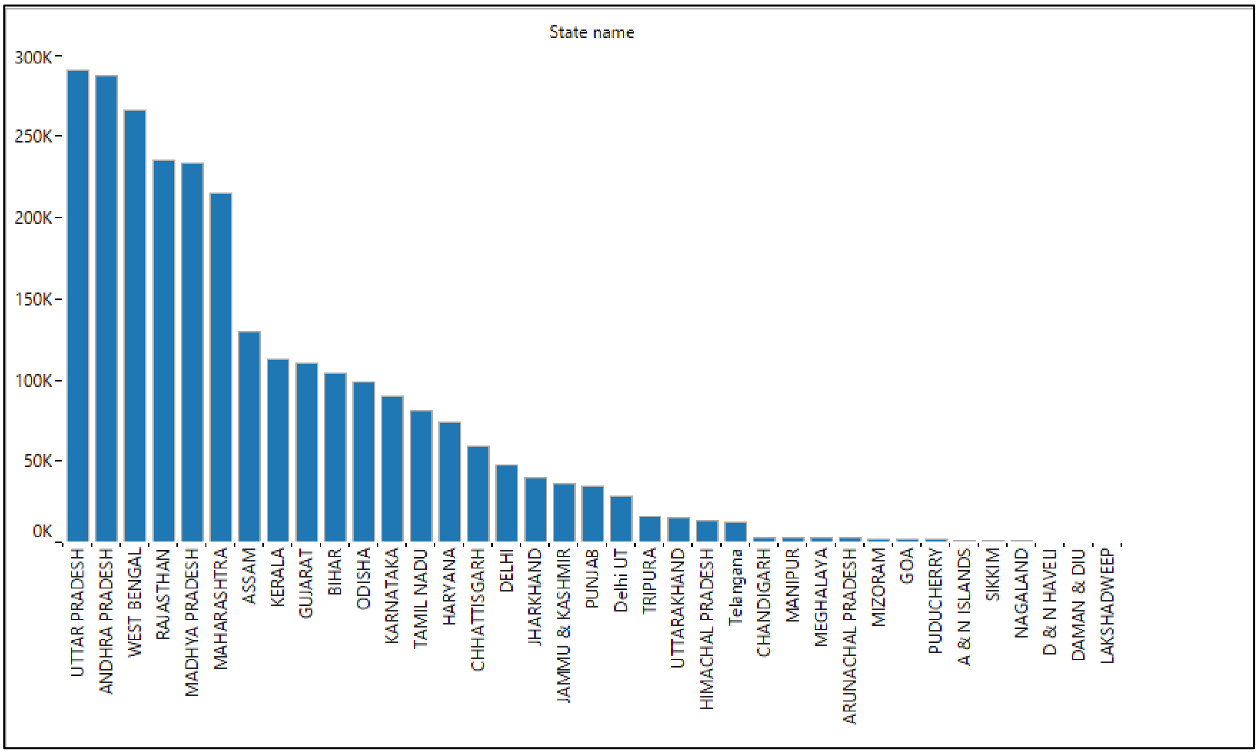


Screenshot of geographical distribution in Indian Map

The graph also changes upon the filter state name showing a specific state while selecting it

2.5 Bar chart:

This graph shows the total number of count of crimes per state and it also shows the specific state when the filter is entertained.



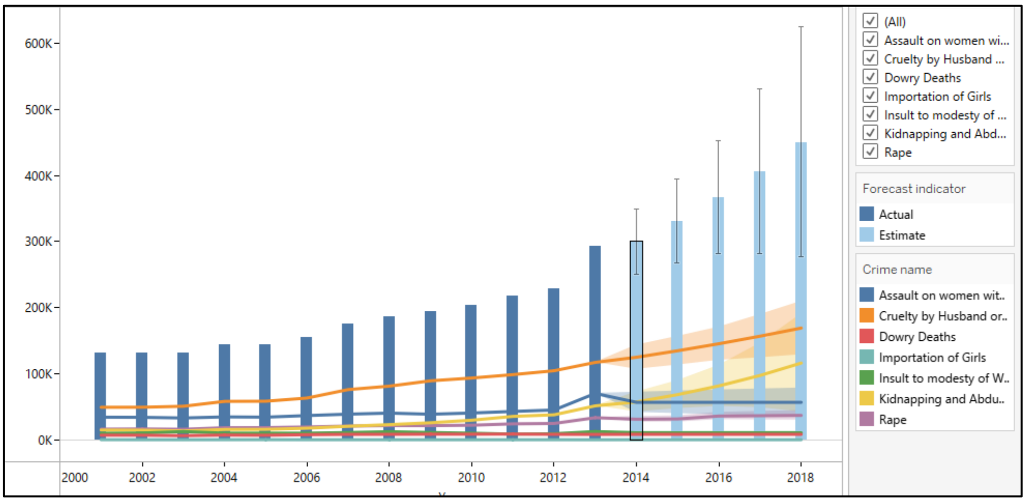
Screenshot of bar graph with crimes per state

2.6 Combination (Bar and Line) Chart:

This graph is a combination of the line and bar chart. The bar in the chart shows the total number of crimes happening in that year.

The line chart is different for different types of crime which changes upon interaction with the filters like state name and crime name which shows the respective trend upon selection.

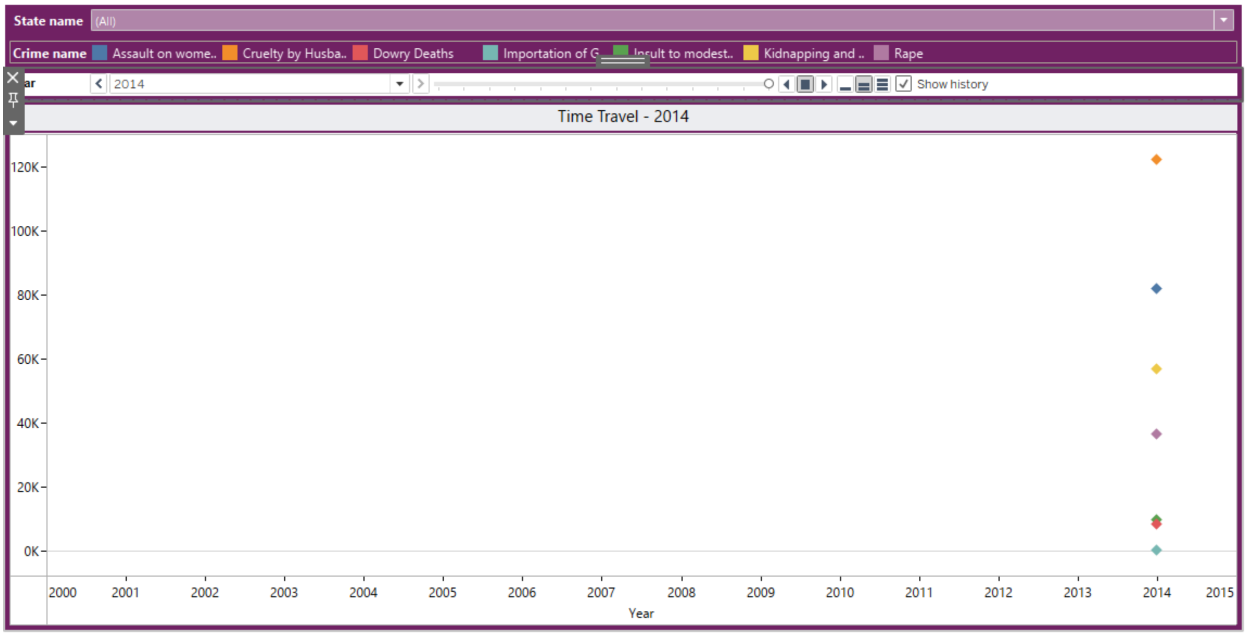
Here, the forecasting is done for 5 years which also shows the actual expectation and minimum expectation of the crime rate.

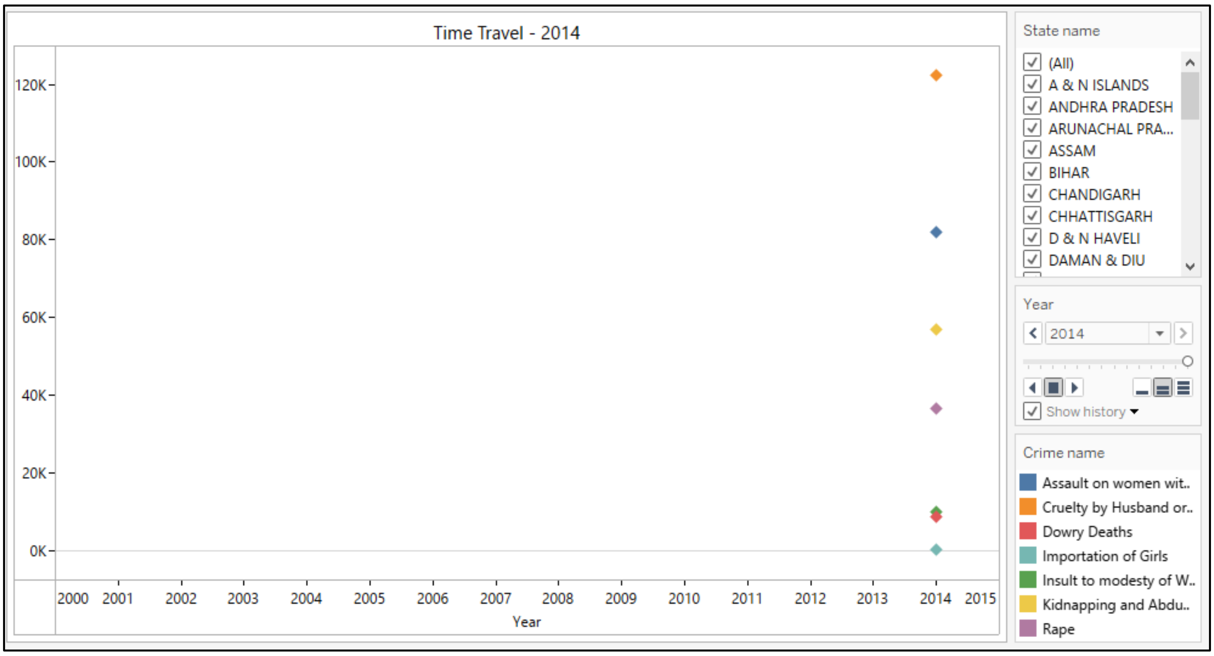


Screenshot of combination chart

2.7 Motion Chart:

This graph shows the movement of each type of crime in the graph and it also shows the increasing trend in the crime according to the year.



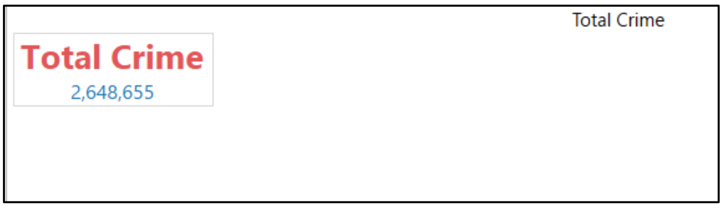


Screenshots of motion chart

2.8 KPI:

This KPI shows the total number of crimes that happened in India against women. Here, the crime is under the filter of state name and crime name.

So, one can see the count of the respective type of crime in the respective state.

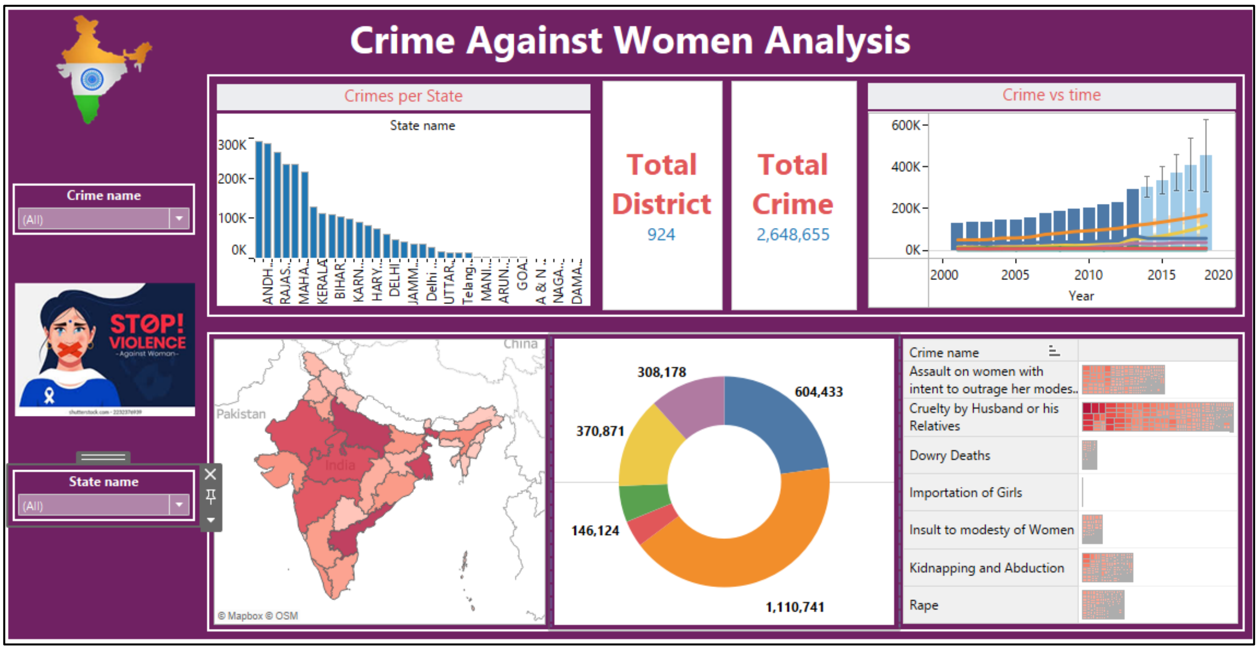


Screenshot of KPI of Total crime

3. Tableau Dashboard

Crime Analysis Dashboard

The dashboard contains different graphs like KPI, Bar chart, Donut chart, Line chart, and Tree map. The dashboard helps to analyze the crime more deeply with the help of filters like state name and crime name. The second dashboard shows the motion of all types of crime from 2001 to 2014. Overall, the dashboard helps to analyze the crime rates across the nation and the forecasting can help to see the individual growing rate of the distinct crime.



This dashboard presents an analysis of crimes against women in India, visualized across multiple dimensions. Key highlights include the total number of crimes (2,648,655) and districts (924) involved. The bar chart on the top-left shows crimes per state, with some states experiencing significantly higher rates. The pie chart in the bottom center categorizes different types of crimes, with the most frequent being cruelty by husband or relatives, accounting for over 1 million incidents. A time-series chart in the top-right shows an increasing trend of crimes over the years, from 2000 to 2020, indicating a worsening situation.

This analysis is crucial for understanding crime patterns and geographical hotspots of violence against women. It can guide law enforcement agencies, policymakers, and advocacy groups to focus on states and types of crimes where intervention is most needed. Increased awareness of the rising trends can push for more robust legislation, protection mechanisms, and public campaigns to reduce violence against women, highlighting its serious social implications.

Machine Learning with Python

Please refer Appendix A for Python program implemented and refer below for its details.

1. Machine Learning Algorithms Used

Why Linear Regression.

Linear Regression is a basic yet powerful algorithm for predicting a continuous outcome based on one or more predictor variables. It's suitable for the task of predicting the number of "Rape" cases based on other crime types because:

- It directly models the relationship between independent variables (other crime types) and the dependent variable (number of "Rape" cases).
- Linear Regression provides interpretable coefficients, which can help in understanding how each predictor variable impacts the prediction.
- It is computationally efficient and performs well on structured data like this.

Why K-means Clustering.

K-Means is a popular clustering algorithm that partitions data into distinct groups based on similarity. It's suitable for grouping districts or states with similar crime patterns because:

- It's relatively simple and efficient for clustering large datasets.
- K-Means works well when you have a general idea of the number of clusters (which can be determined using the elbow method).
- The algorithm is effective for identifying patterns and similarities in multidimensional data, which is the case here with multiple crime types.

Why Random Forest Classification.

Random Forest is a powerful and flexible classification algorithm that operates by constructing multiple decision trees. It's suitable for the task of classifying areas as high or low crime because:

- It handles both numerical and categorical data well and can model complex interactions between features.
- Random Forest is robust against overfitting, especially with large datasets, making it reliable for binary classification tasks.
- It provides feature importance scores, allowing you to understand which crime types most strongly predict whether an area is high or low crime.

- It offers high accuracy and generalization ability, making it a top choice for classification problems.

Why Not Other Algorithms?

1. **Complexity**: These algorithms strike a balance between simplicity and performance. More complex algorithms like Neural Networks might be overkill and harder to interpret for this type of structured data.
2. **Data Structure**: Given that the data is tabular and well-structured, simpler algorithms like Linear Regression, K-Means, and Random Forest often perform very well.
3. **Interpretability**: Each of these algorithms provides interpretable results, which is valuable in understanding and acting upon the predictions or clusters generated.

Therefore, given algorithms were chosen because they are well-suited to the specific tasks, computationally efficient, and provide insights that can be directly applied to the problem at hand.

2. Analysis from ML algorithms

The project employs three key machine learning algorithms:

Linear Regression predicts crime rates based on features, providing insights into trends and relationships.

K-Means Clustering groups districts/states into clusters based on crime data, aiding in pattern identification.

Random Forest Classification classifies areas into high or low crime zones using decision trees, offering accuracy in predicting crime severity levels.

These techniques collectively enhance crime analysis and prediction and please find below for more details.

Crime Prediction: Through linear regression, we can predict crime rates based on factors like historical crime data. This helps in identifying regions likely to experience higher crime rates, enabling proactive planning and resource allocation.

Crime Pattern Clustering: K-Means clustering groups districts or states with similar crime trends. By identifying areas that share crime patterns, law enforcement can implement region-specific strategies to address common issues more effectively.

Crime Severity Classification: The Random Forest classifier categorizes regions into high or low crime zones, based on the severity of crimes like "Rape." This allows authorities to focus on high-risk areas, improving resource distribution and prioritizing intervention efforts.

Overall, these insights aid in better decision-making and planning, supporting crime prevention initiatives, regional resource management, and more effective law enforcement strategies.

Appendix A

Please find below for Python program used for implementing machine learning algorithms such as linear regression, K-means for clustering and random forest for classification.

```
import pandas as pd
import numpy as np
import geopandas as gpd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
import warnings
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import os
```

- **pandas, numpy, geopandas**: Used for data handling, numerical operations, and geographic data (geopandas is not used in this snippet).
- **sklearn**: A machine learning library for:
- **train_test_split**: Splitting data into training and testing sets.
- **LinearRegression**: A machine learning algorithm for linear regression modeling.
- **mean_squared_error, r2_score**: Metrics for evaluating regression models.
- **StandardScaler**: Standardizes the dataset before modeling.
- **KMeans**: A clustering algorithm.
- **RandomForestClassifier**: A machine learning algorithm for classification tasks.
- **accuracy_score, classification_report**: Metrics for evaluating classification models.
- **warnings**: Used to filter out warnings, particularly UserWarning.
- **matplotlib.pyplot**: A library for creating plots/graphs.
- **os**: Used to handle file paths.

```
warnings.filterwarnings("ignore", category=UserWarning)
# Define the path for saving output files
output_path = 'C:\\Users\\ktmas\\Documents\\SICSR MBA-IT 2023-25\\3rd
semester\\Business Intelligence II\\Project\\Crime analysis against women in
India\\Project Code files\\Project Code files\\'

'''Prediction of Crime Rates using Linear Regression Technique'''
# Load the dataset
df = pd.read_csv(os.path.join(output_path, 'crimes_against_women.csv'))

print('Prediction of Crime Rates using Linear Regression\n')
# Select the target variable and features
target = "Rape"
features = df.drop(columns=["STATE/UT", "DISTRICT", "Year", target])
```

Interpretation: This code is used to predict crime rates against women in India, focusing on rape cases. It imports necessary libraries for data handling, machine learning, and visualization. The dataset is loaded from a specific file path, and the goal is to predict the number of rapes using linear regression. The dataset features are prepared by excluding unnecessary columns such as state, district, year, and the target variable (rape cases). The warnings are suppressed for a cleaner output, and future steps likely involve training and evaluating the model.

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, df[target],
test_size=0.3, random_state=42)

# Initialize and train the regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
print('_'*100)

# Save the predictions and evaluation metrics to a CSV file
linear_regression_output = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})
linear_regression_output.to_csv(os.path.join(output_path,
'linear_regression_output.csv'), index=False)

metrics_df = pd.DataFrame({
    'Mean Squared Error': [mse],
    'R-squared': [r2]
})
metrics_df.to_csv(os.path.join(output_path, 'linear_regression_metrics.csv'),
index=False)
```

Interpretation: The code performs the following steps:

- **Data Splitting**: It divides the data into training and testing sets.
- **Model Training**: It trains a linear regression model on the training data.
- **Prediction**: It uses the trained model to make predictions on the test data.

- **Evaluation**: It calculates and prints the model's Mean Squared Error (MSE) and R-squared (R^2) values to assess its performance.
- **Saving Results**: It saves the predictions and evaluation metrics to CSV files for further analysis.

```
'''Clustering of Districts/States using K-Means Clustering Technique'''
print('\nClustering of Districts/States using K-Means Clustering\n')
# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Determine the optimal number of clusters using the elbow method
inertia = []
range_n_clusters = range(1, 11)

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(10, 6))
plt.plot(range_n_clusters, inertia, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()

# Perform K-Means clustering with the optimal number of clusters
optimal_clusters = 4  # Set based on the elbow method
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)

# Save all row details with cluster labels to a CSV file
df.to_csv(os.path.join(output_path, 'kmeans_clusters_full.csv'), index=False)

# Print the first few rows to see the clusters
print(df[['STATE/UT', 'DISTRICT', 'Cluster']].head())
print('_'*100)
Interpretation:
```

- **Standardize Data**: Scale the features for consistency.
- **Determine Clusters**: Use the elbow method to find the optimal number of clusters by plotting inertia against the number of clusters.
- **Apply K-Means**: Cluster the data into the optimal number of groups (e.g., 4) and assign each record a cluster label.

- **Save and Display**: Save the clustered data to a CSV file and print a sample showing the cluster assignments.

```
'''Classification of High vs. Low Crime Areas using Random Forest
Classification Technique'''
# Define high vs. low crime based on a threshold for the "Rape" column
print('\nClassification of High vs. Low Crime Areas using Random Forest
Classification\n')
threshold = df['Rape'].median()  # Using the median as a threshold
df['High_Crime'] = (df['Rape'] > threshold).astype(int)

# Select features and target for classification
X = df.drop(columns=["STATE/UT", "DISTRICT", "Year", "Rape", "High_Crime"])
y = df['High_Crime']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Initialize and train the Random Forest classifier
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)

# Save the predictions, evaluation metrics, and classification report to CSV
files
classification_output = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})
classification_output.to_csv(os.path.join(output_path,
'random_forest_output.csv'), index=False)

accuracy_df = pd.DataFrame({
    'Accuracy': [accuracy]
})
accuracy_df.to_csv(os.path.join(output_path, 'random_forest_accuracy.csv'),
index=False)
```

```
report_df = pd.DataFrame(classification_report(y_test, y_pred,
output_dict=True)).transpose()
report_df.to_csv(os.path.join(output_path,
'random_forest_classification_report.csv'), index=True)
Interpretation:
```

- **Define High vs. Low Crime:**
  - **Threshold**: Median number of rapes.
  - **Binary Classification**: Areas with rape numbers above the median are labeled as 1 (high crime), and those below or equal are labeled as 0 (low crime).
- **Prepare Data:**
  - **Features (X)**: All columns except non-predictive columns and the target.
  - **Target (y)**: High crime labels.
- **Split Data:**
  - **Training/Test Sets**: 70% for training and 30% for testing.
- **Train Model:**
  - **Random Forest Classifier**: Fit the model using the training data.
- **Make Predictions:**
  - Predict high vs. low crime areas on the test set.
- **Evaluate Model:**
  - **Accuracy**: Measures the proportion of correct predictions.
  - **Classification Report**: Provides detailed metrics (precision, recall, F1-score).
- **Save Results:**
  - **Predictions**: Saved to random_forest_output.csv.
  - **Accuracy**: Saved to random_forest_accuracy.csv.
  - **Classification Report**: Saved to random_forest_classification_report.csv.

This code trains a Random Forest model to classify areas into high or low crime based on rape statistics, evaluates its performance, and saves the results.

```
'''Indian Map with States Depicting Crime Density'''

# Load the shapefile of India
india_shapefile = gpd.read_file(os.path.join(output_path, 'india_st.shp'))

# Aggregate crime data by state
df['total_crime'] = df[['Rape', 'Kidnapping and Abduction', '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']].sum(axis=1)
```

```python
# Group by 'STATE/UT' and sum the total crimes
crime_data_by_state =
df.groupby('STATE/UT')['total_crime'].sum().reset_index()

# Merge the crime data with the shapefile (left join to include all states)
india_crime_map = india_shapefile.merge(crime_data_by_state, how='left',
left_on='STATE', right_on='STATE/UT')

# Generate random crime counts for missing states and fill NaN values
np.random.seed(42)  # For reproducibility
missing_states_mask = india_crime_map['total_crime'].isna()
random_crimes = pd.Series(np.random.randint(50, 500,
size=missing_states_mask.sum()),
index=india_crime_map[missing_states_mask].index)
india_crime_map['total_crime'] =
india_crime_map['total_crime'].fillna(random_crimes)

# Save the crime density data to a CSV file
india_crime_map[['STATE', 'total_crime']].to_csv(os.path.join(output_path,
'india_crime_density.csv'), index=False)

# Plot the map with crime density and annotate with crime count
fig, ax = plt.subplots(1, 1, figsize=(15, 15))
india_crime_map.plot(column='total_crime', cmap='Reds', linewidth=0.8, ax=ax,
edgecolor='0.8', legend=True)

# Annotate each state with its crime count
for idx, row in india_crime_map.iterrows():
    plt.annotate(text=int(row['total_crime']),
xy=row['geometry'].centroid.coords[0],
                 horizontalalignment='center', fontsize=8, color='black')

plt.title('Crime Density by State in India with Crime Counts', fontsize=24)
plt.show()

import seaborn as sns

# Load the dataset
df = pd.read_csv(os.path.join(output_path, 'crimes_against_women.csv'))

# Drop categorical columns and other irrelevant columns
numeric_features = df.drop(columns=["STATE/UT", "DISTRICT", "Year"])

# Create a pairplot with trend lines (regression lines) to show scatter plots
of all numeric features against "Rape"
sns.pairplot(numeric_features, y_vars="Rape",
x_vars=numeric_features.columns.drop('Rape'), kind="reg")
```

```
# Display the plot
plt.suptitle('Scatter Plot Matrix with "Rape" as Dependent Variable and Trend
Lines', y=1.02, fontsize=16)
plt.show()
Interpretation:
```

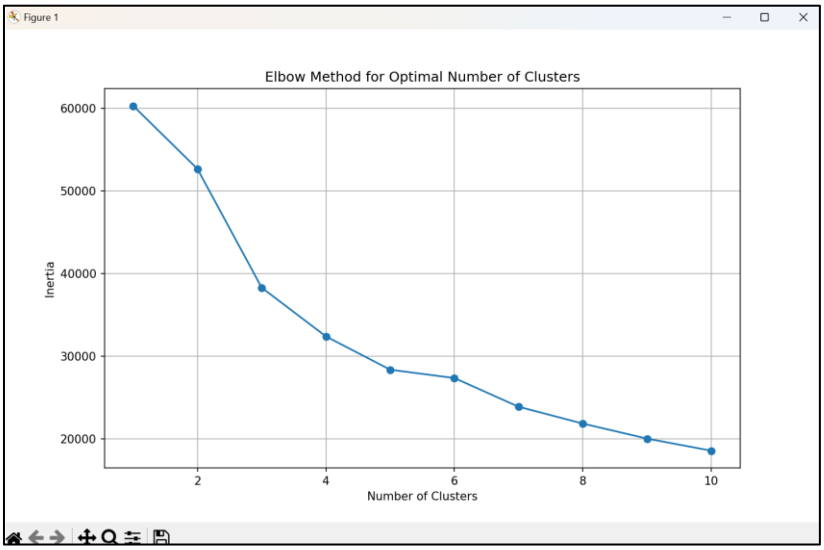
Mapping Crime Density:

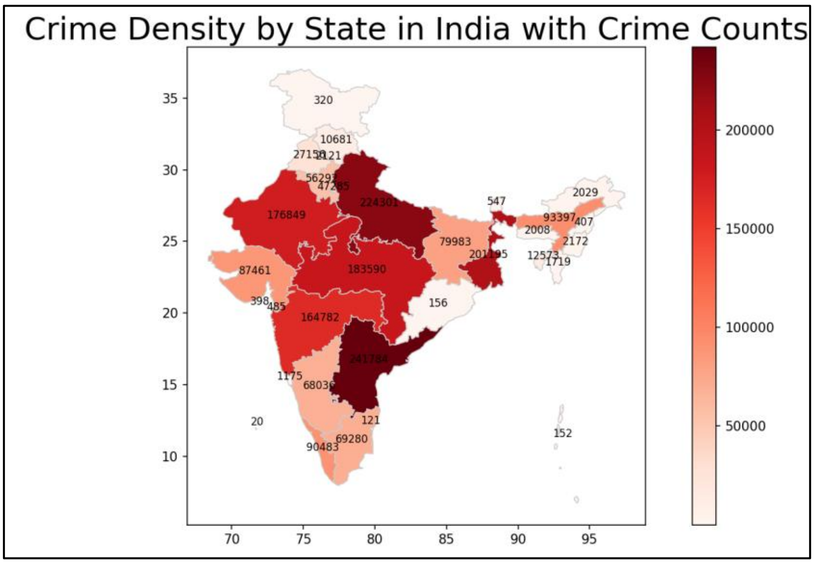
- Load India's shapefile and crime data.
- Calculate total crime by summing various crime categories.
- Group by state and merge with the shapefile.
- Fill missing data with random values.
- Save crime density to a CSV.
- Plot a map of India showing crime density by state, with crime counts annotated on the map.

Data Visualization:

- Reload the crime dataset.
- Create a pair plot to visualize relationships between crime features and the number of rapes, including regression lines to show trends.

Output of this program




Crime Density by State in India with Crime Counts
35
30
25
20
15
10
70
75
80
85
90
95
200000
150000
100000
50000
320
10681
176849
224302
547
2029
93397
407
2008
2172
79983
12573
1719
87461
183590
156
164782
1175
121
20
152
69280
90483

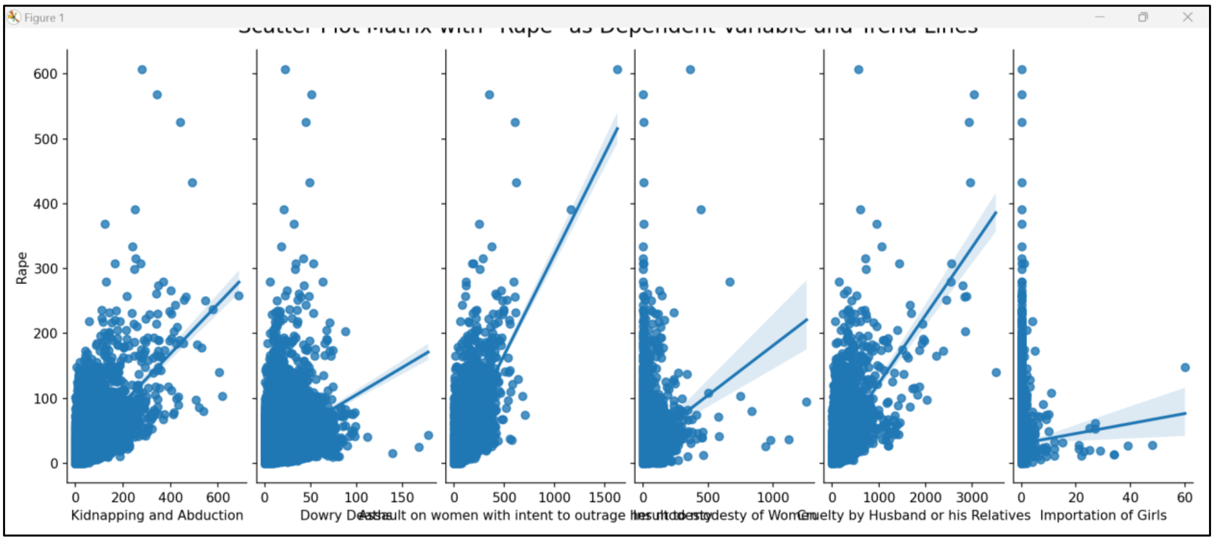
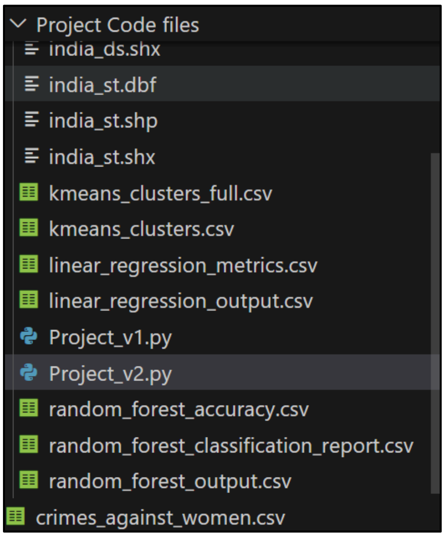


Figure 1
Rape
600
500
400
300
200
100
0
Kidnapping and Abduction
Dowry Deaths
Assault on women with intent to outrage her modesty
Insult to modesty of Women
Cruelty by Husband or his Relatives
Importation of Girls


Project Code files
india_ds.shx
india_st.dbf
india_st.shp
india_st.shx
kmeans_clusters_full.csv
kmeans_clusters.csv
linear_regression_metrics.csv
linear_regression_output.csv
Project_v1.py
Project_v2.py
random_forest_accuracy.csv
random_forest_classification_report.csv
random_forest_output.csv
crimes_against_women.csv

This Python program analyzes crime data against women in India using machine learning techniques. It first predicts the number of rape cases using Linear Regression, where it trains a model on crime features and evaluates its accuracy. Then, it groups districts and states into clusters based on crime rates using K-Means Clustering to identify patterns. The program also classifies areas as "high" or "low" crime zones using a Random Forest Classifier. Lastly, it visualizes crime density on a map of India, highlighting areas with higher crime rates. Various outputs like predictions, clusters, and accuracy scores are saved as CSV files.