```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean_squared_error, r2_score, classification_report
import matplotlib.pyplot as plt
# Load the dataset
file_path = '/content/crimes_cleaned.csv'  # Upload this CSV file to Colab or provide the co
data = pd.read_csv(file_path)

# Display the first few rows to inspect the data
data.head()
```

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KI A
0	ANDHRA PRADESH	ADILABAD	2013	96	72	13	61	0	61	
1	ANDHRA PRADESH	ANANTAPUR	2013	156	149	3	28	0	28	
2	ANDHRA PRADESH	CHITTOOR	2013	72	61	2	31	0	31	
3	ANDHRA PRADESH	CUDDAPAH	2013	93	107	7	19	0	19	
4	ANDHRA PRADESH	CYBERABAD	2013	162	123	16	138	0	138	

5 rows × 33 columns

```
#####the results of Linear Regression are shown

# Drop non-numeric columns like STATE/UT and DISTRICT, if these are not needed
data = data.drop(columns=['STATE/UT', 'DISTRICT'])

# Check for missing values
data.isnull().sum()

# Fill or drop missing values if necessary
# data = data.fillna(data.mean())  # or data.dropna()

# Splitting data into features (X) and target variable (y)
```

```
# For this example, let's predict 'TOTAL IPC CRIMES' as the target
X = data.drop(columns=['TOTAL IPC CRIMES'])
y = data['TOTAL IPC CRIMES']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Linear Regression
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Predictions
y_pred_lr = linear_regressor.predict(X_test)

# Evaluation
print("Linear Regression R2 Score:", r2_score(y_test, y_pred_lr))
print("Linear Regression MSE:", mean_squared_error(y_test, y_pred_lr))
```

```
Linear Regression R2 Score: 0.9999999999991817
Linear Regression MSE: 1.1713681407303513e-05
```

```
#####the results of KNN and Gradient Boosting are shown

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor

models = {
    'KNN': KNeighborsRegressor(),

    'Gradient Boosting': GradientBoostingRegressor(),

}

# Train and evaluate each model
results = {}

for model_name, model in models.items():
    # Fit the model
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

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

    # Store the results
    results[model_name] = {'MSE': mse, 'R2': r2}

# Print out the results
for model_name, metrics in results.items():
    print(f"{model_name}: MSE = {metrics['MSE']:.4f}, R2 = {metrics['R2']:.4f}")
```

```
KNN: MSE = 98954.7008, R2 = 0.9931
Gradient Boosting: MSE = 61927.1496, R2 = 0.9957
```

```
#####ML Models Usage
#####Machine Learning Models
#####Aim 1: Predicting the Total Number of Crimes Based on All Features


import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, classification_report
import matplotlib.pyplot as plt
# Load the dataset
file_path = 'crimes_cleaned.csv'
data = pd.read_csv(file_path)

# Display the first few rows to inspect the data
data.head()
```

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KI A
0	ANDHRA PRADESH	ADILABAD	2013	96	72	13	61	0	61	
1	ANDHRA PRADESH	ANANTAPUR	2013	156	149	3	28	0	28	
2	ANDHRA PRADESH	CHITTOOR	2013	72	61	2	31	0	31	
3	ANDHRA PRADESH	CUDDAPAH	2013	93	107	7	19	0	19	
4	ANDHRA PRADESH	CYBERABAD	2013	162	123	16	138	0	138	

5 rows × 33 columns

```
# Drop non-numeric columns like STATE/UT and DISTRICT, if these are not needed
data = data.drop(columns=['STATE/UT', 'DISTRICT'])

# Check for missing values
data.isnull().sum()
```

```
# For this example, let's predict 'TOTAL IPC CRIMES' as the target
X = data.drop(columns=['TOTAL IPC CRIMES'])
y = data['TOTAL IPC CRIMES']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Linear Regression
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Predictions
y_pred_lr = linear_regressor.predict(X_test)

# Evaluation
print("Linear Regression R2 Score:", r2_score(y_test, y_pred_lr))
print("Linear Regression MSE:", mean_squared_error(y_test, y_pred_lr))
```

```
Linear Regression R2 Score: 0.9999999999991817
Linear Regression MSE: 1.1713681407303513e-05
```

```
#####Aim 2: Predicting the Total Number of Crimes Using Top 5 Features



# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv('crimes_cleaned.csv')


label_encoder = LabelEncoder()
df['STATE/UT'] = label_encoder.fit_transform(df['STATE/UT'])
df['DISTRICT'] = label_encoder.fit_transform(df['DISTRICT'])

#  3: Calculate the variance-covariance matrix between features
# Exclude the target column `TOTAL IPC CRIMES` and categorical columns like 'STATE/UT', 'DI
features = df.drop(columns=['TOTAL IPC CRIMES', 'STATE/UT', 'DISTRICT'])
cov_matrix = features.cov()

# Calculate covariance of each feature with the target variable
cov_with_target = df.cov()['TOTAL IPC CRIMES'].drop(['TOTAL IPC CRIMES', 'STATE/UT', 'DISTR

# Select the top 5 features with the highest covariance (absolute value) with the target va
top_5_features = cov_with_target.abs().sort_values(ascending=False).head(5).index
print("\nTop 5 Features based on Covariance with Target:")
print(top_5_features)
```

```
#  4: Split the dataset using the selected features
X = df[top_5_features]
y = df['TOTAL IPC CRIMES']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#  5: Train and evaluate models
models = {
    'Linear Regression': LinearRegression(),
    'K-Nearest Neighbors': KNeighborsRegressor(),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42)
}

for model_name, model in models.items():
    print(f"\nTraining {model_name}...")
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Model: {model_name}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R²: {r2}")
```

```
Top 5 Features based on Covariance with Target:
Index(['OTHER IPC CRIMES', 'THEFT', 'OTHER THEFT', 'AUTO THEFT',
       'HURT/GREVIOUS HURT'],
      dtype='object')

Training Linear Regression...
Model: Linear Regression
Mean Squared Error (MSE): 190673.6910986067
R²: 0.9866794599801313

Training K-Nearest Neighbors...
Model: K-Nearest Neighbors
Mean Squared Error (MSE): 157902.23678723403
R²: 0.9889688868336671

Training Gradient Boosting...
Model: Gradient Boosting
Mean Squared Error (MSE): 161932.86414667498
R²: 0.9886873056006336
```

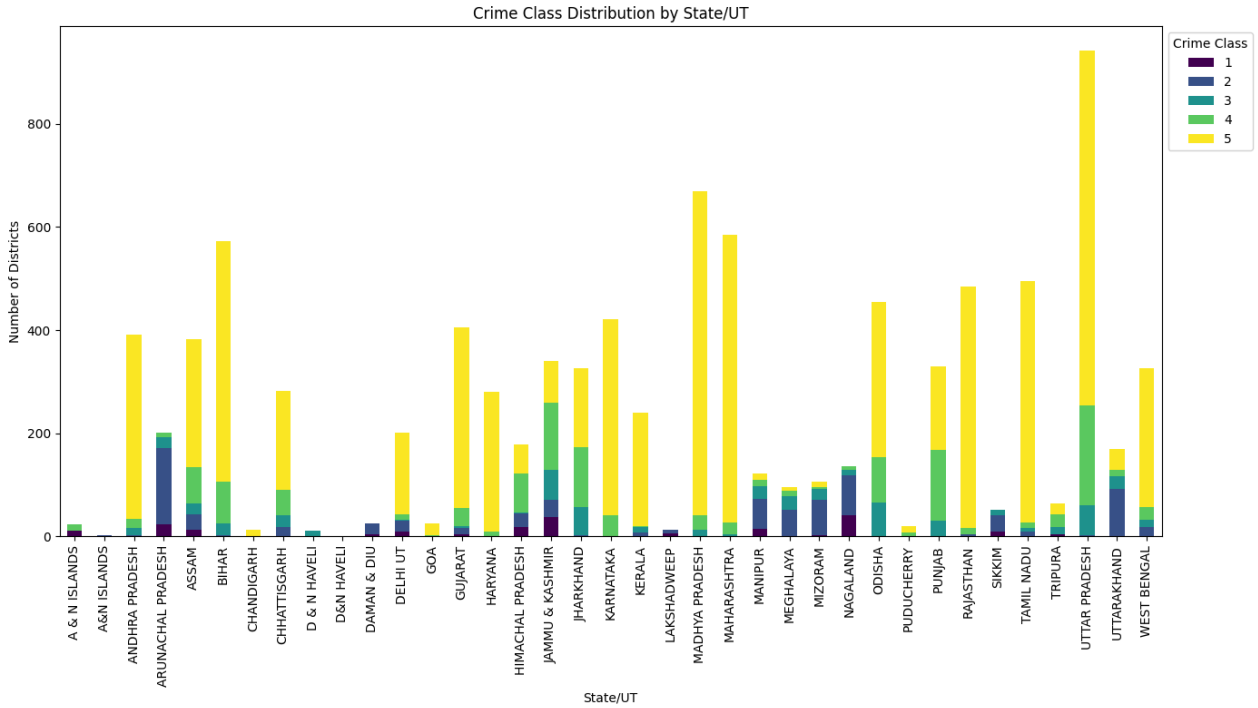
```
##### Aim 3: Categorizing Districts Based on Crime Levels



import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import cross_val_score

# Load the dataset
data = pd.read_csv("crimes_cleaned.csv")

# Check the first few rows
print(data.head())

# Data Preprocessing
# Convert 'YEAR' to categorical (it is not useful as a numeric feature for this task)
data['YEAR'] = data['YEAR'].astype(str)

# Handle missing values (if any)
data.fillna(0, inplace=True)

# Create a new feature 'TOTAL_CRIMES' by summing crime-related columns
crime_columns = ['MURDER', 'ATTEMPT TO MURDER', 'CULPABLE HOMICIDE NOT AMOUNTING TO MURDER'
                 'CUSTODIAL RAPE', 'OTHER RAPE', 'KIDNAPPING & ABDUCTION',
                 'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS', 'KIDNAPPING AND ABDUCTION O
                 'DACOITY', 'PREPARATION AND ASSEMBLY FOR DACOITY', 'ROBBERY', 'BURGLARY',
                 'AUTO THEFT', 'OTHER THEFT', 'RIOTS', 'CRIMINAL BREACH OF TRUST', 'CHEATIN
                 'COUNTERFIETING', 'ARSON', '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 NEGLIGENC
                 'OTHER IPC CRIMES']

data['TOTAL_CRIMES'] = data[crime_columns].sum(axis=1)

# Label the districts into 5 crime classes based on the 'TOTAL_CRIMES'
crime_bins = [0, 50, 300, 600, 1200, np.inf]  # Define the ranges for crime classes
crime_labels = [1, 2, 3, 4, 5]  # The labels for the five crime classes

# Apply the binning process
data['CRIME_LABEL'] = pd.cut(data['TOTAL_CRIMES'], bins=crime_bins, labels=crime_labels)

# Check if there are any NaN values in CRIME_LABEL
print(f"Missing values in 'CRIME_LABEL': {data['CRIME_LABEL'].isnull().sum()}")

# If there are NaN values in CRIME_LABEL, fill them with the most common crime label (or ar
data['CRIME_LABEL'].fillna(data['CRIME_LABEL'].mode()[0], inplace=True)

# Now, encode categorical columns ('State/UT' and 'District') using One-Hot Encoding
data_encoded = pd.get_dummies(data, columns=['STATE/UT', 'DISTRICT'], drop_first=True)

# Feature selection (excluding 'CRIME_LABEL', 'YEAR' for simplicity)
X = data_encoded.drop(columns=['CRIME_LABEL', 'YEAR', 'TOTAL_CRIMES'])
y = data_encoded['CRIME_LABEL']

# 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)
```

```
# Feature scaling (optional but often recommended for better model performance)
scaler = StandardScaler()

# Fit the scaler to the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)

# Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Cross-validation (optional)
cv_scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-validation accuracy: {cv_scores.mean() * 100:.2f}%")

# Feature importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("Feature Importance:")
print(feature_importance)

# Visualizing Crime Class Distribution by State
state_crime_class = data.groupby(['STATE/UT', 'CRIME_LABEL']).size().unstack(fill_value=0)

# Plotting the crime class distribution by state
plt.figure(figsize=(12, 8))
state_crime_class.plot(kind='bar', stacked=True, colormap='viridis', figsize=(14, 8))
plt.title('Crime Class Distribution by State/UT')
plt.xlabel('State/UT')
plt.ylabel('Number of Districts')
plt.legend(title='Crime Class', loc='upper left', bbox_to_anchor=(1, 1))
plt.xticks(rotation=90)
plt.tight_layout()

# Show the plot
plt.show()
```

```
          STATE/UT   DISTRICT  YEAR  MURDER  ATTEMPT TO MURDER  \
0  ANDHRA PRADESH   ADILABAD  2013      96                 72
1  ANDHRA PRADESH  ANANTAPUR  2013     156                149
2  ANDHRA PRADESH   CHITTOOR  2013      72                 61
3  ANDHRA PRADESH   CUDDAPAH  2013      93                107
4  ANDHRA PRADESH  CYBERABAD  2013     162                123

   CULPABLE HOMICIDE NOT AMOUNTING TO MURDER  RAPE  CUSTODIAL RAPE  \
0                                         13    61               0
1                                          3    28               0
2                                          2    31               0
3                                          7    19               0
4                                         16   138               0

   OTHER RAPE  KIDNAPPING & ABDUCTION  ...  ARSON  HURT/GREVIOUS HURT  \
0          61                      65  ...     30                2394
1          28                     110  ...     29                2537
2          31                      52  ...     18                 937
3          19                      84  ...     34                2310
4         138                     192  ...     40                4284

   DOWRY DEATHS  ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY  \
0            12                                                 197
1            23                                                 337
2            13                                                 119
3             9                                                 318
4            43                                                 350

   INSULT TO MODESTY OF WOMEN  CRUELTY BY HUSBAND OR HIS RELATIVES  \
0                         138                                 464
1                          43                                 161
2                          84                                 435
3                         163                                 207
4                         338                                1526

   IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES  CAUSING DEATH BY NEGLIGENCE  \
0                                            0                          376
1                                            0                          573
2                                            0                          546
3                                            0                          464
4                                            0                         1104

   OTHER IPC CRIMES  TOTAL IPC CRIMES
0              1390              6381
1              1634              6913
2              2239              5610
3              1741              7048
4              3139             19992

[5 rows x 33 columns]
Missing values in 'CRIME_LABEL': 4
<ipython-input-7-b1e7ad9dfd8d>:53: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(


  data['CRIME_LABEL'].fillna(data['CRIME_LABEL'].mode()[0], inplace=True)
Classification Report:
              precision    recall  f1-score   support
```

```
              1       0.97      0.95      0.96        62
              2       0.95      0.98      0.96       219
              3       0.94      0.84      0.89       178
              4       0.94      0.95      0.95       400
              5       0.99      1.00      1.00      1961

    accuracy                           0.98      2820
   macro avg       0.96      0.94      0.95      2820
weighted avg       0.98      0.98      0.98      2820

Accuracy: 97.91%
Cross-validation accuracy: 97.85%
Feature Importance:
                 Feature  Importance
29      TOTAL IPC CRIMES    0.128891
28      OTHER IPC CRIMES    0.079248
13                 THEFT    0.069631
15           OTHER THEFT    0.064857
21    HURT/GREVIOUS HURT    0.044726
..                   ...         ...
834    DISTRICT_TIRUPPUR    0.000000
38   STATE/UT_D&N HAVELI    0.000000
881      DISTRICT_WARDHA    0.000000
871  DISTRICT_VILUPPURAM    0.000000
870  DISTRICT_VILLUPURAM    0.000000

[891 rows x 2 columns]
<ipython-input-7-b1e7ad9dfd8d>:101: FutureWarning: The default of observed=False is
  state_crime_class = data.groupby(['STATE/UT', 'CRIME_LABEL']).size().unstack(fill_
<Figure size 1200x800 with 0 Axes>
```



```
##### Aim 4 : Add ON : Categorizing Districts Based on Crime Levels using TOP 6 Features


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import cross_val_score

# Load the dataset
data = pd.read_csv("crimes_cleaned.csv")

# Check the first few rows
print(data.head())

# Data Preprocessing
# Convert 'YEAR' to categorical (it is not useful as a numeric feature for this task)
data['YEAR'] = data['YEAR'].astype(str)

# Handle missing values (if any)
data.fillna(0, inplace=True)

# Create a new feature 'TOTAL_CRIMES' by summing crime-related columns
crime_columns = ['MURDER', 'ATTEMPT TO MURDER', 'CULPABLE HOMICIDE NOT AMOUNTING TO MURDER'
                 'CUSTODIAL RAPE', 'OTHER RAPE', 'KIDNAPPING & ABDUCTION',
                 'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS', 'KIDNAPPING AND ABDUCTION O
                 'DACOITY', 'PREPARATION AND ASSEMBLY FOR DACOITY', 'ROBBERY', 'BURGLARY',
                 'AUTO THEFT', 'OTHER THEFT', 'RIOTS', 'CRIMINAL BREACH OF TRUST', 'CHEATIN
                 'COUNTERFIETING', 'ARSON', '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 NEGLIGENC
                 'OTHER IPC CRIMES']

data['TOTAL_CRIMES'] = data[crime_columns].sum(axis=1)

# Label the districts into 5 crime classes based on the 'TOTAL_CRIMES'
crime_bins = [0, 50, 300, 600, 1200, np.inf]  # Define the ranges for crime classes
crime_labels = [1, 2, 3, 4, 5]  # The labels for the five crime classes

# Apply the binning process
data['CRIME_LABEL'] = pd.cut(data['TOTAL_CRIMES'], bins=crime_bins, labels=crime_labels)

# Check if there are any NaN values in CRIME_LABEL
print(f"Missing values in 'CRIME_LABEL': {data['CRIME_LABEL'].isnull().sum()}")

# If there are NaN values in CRIME_LABEL, fill them with the most common crime label (or an
data['CRIME_LABEL'].fillna(data['CRIME_LABEL'].mode()[0], inplace=True)

# Select only the desired features and target variable
```

```
selected_features = ['MURDER', 'HURT/GREVIOUS HURT', 'KIDNAPPING & ABDUCTION', 'THEFT', 'Bl
X = data[selected_features]
y = data['CRIME_LABEL']

# 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)

# Feature scaling (optional but often recommended for better model performance)
scaler = StandardScaler()

# Fit the scaler to the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)

# Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Cross-validation (optional)
cv_scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-validation accuracy: {cv_scores.mean() * 100:.2f}%")

# Feature importance
feature_importance = pd.DataFrame({
    'Feature': selected_features,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("Feature Importance:")
print(feature_importance)

# Visualizing Crime Class Distribution by State
state_crime_class = data.groupby(['STATE/UT', 'CRIME_LABEL']).size().unstack(fill_value=0)

# Plotting the crime class distribution by state
plt.figure(figsize=(12, 8))
state_crime_class.plot(kind='bar', stacked=True, colormap='viridis', figsize=(14, 8))
plt.title('Crime Class Distribution by State/UT')
plt.xlabel('State/UT')
plt.ylabel('Number of Districts')
plt.legend(title='Crime Class', loc='upper left', bbox_to_anchor=(1, 1))
plt.xticks(rotation=90)
plt.tight_layout()
```

```
    # Show the plot
    plt.show()
```