Project 5: Crime Data Analysis

Introduction

Crime rate analysis and prediction play a crucial role in understanding the patterns and trends of criminal activities within a region. By leveraging data-driven approaches, law enforcement agencies and policymakers can gain valuable insights into the underlying factors contributing to crime rates and devise effective strategies for crime prevention and resource allocation.

This project aims to analyze and predict crime rates across different states and union territories (UTs) in India. Crime rate data is a valuable resource for government agencies, researchers, and stakeholders to identify areas with high crime rates and develop targeted interventions. By employing machine learning techniques and statistical analysis, this project seeks to uncover meaningful patterns and relationships within the crime rate data.

The primary objectives of this project are as follows:

1. Data Preprocessing: Clean and preprocess the crime rate dataset, handling missing data and encoding categorical variables to prepare the data for analysis and modelling.

2. Exploratory Data Analysis: Conduct an in-depth exploration of the crime rate data, visualizing trends and patterns across different states/UTs, types of crimes, and time periods.

3. Clustering: Employ clustering algorithms to group states/UTs based on their crime rate patterns, identifying safe and unsafe regions.

4. Predictive Modelling: Develop and evaluate various machine learning models, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), K-Nearest Neighbors (K-NN), Logistic Regression, Random Forest Regression, Decision Tree Regression, and Support Vector Regression (SVR), to predict crime rates based on the given features.

5. Model Evaluation: Assess the performance of the developed models using appropriate evaluation metrics, such as confusion matrices and accuracy scores, to determine their effectiveness in predicting crime rates.

6. Results and Analysis: Analyze and interpret the results obtained from the exploratory data analysis and predictive modelling, providing insights into the crime rate patterns and the performance of the developed models.

By leveraging the power of data analysis and machine learning, this project aims to contribute to a better understanding of crime rates in India and potentially assist in formulating effective strategies for crime prevention and public safety.

2. Project Prerequisites

To successfully execute this project and reproduce the results, the following prerequisites are necessary:

- Programming Language and Environment**
  - Python 3.x
  - Jupyter Notebook or any Python Integrated Development Environment (IDE)

- Python Libraries and Packages
  - NumPy: A fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices.
  - Pandas: A powerful data manipulation and analysis library for structured data, enabling efficient data handling and preprocessing.
  - Matplotlib: A plotting library for creating static, animated, and interactive visualizations in Python.
  - Seaborn: A data visualization library based on Matplotlib, providing a high-level interface for creating attractive and informative statistical graphics.
  - Scikit-learn: A machine learning library that features various classification, regression, and clustering algorithms, as well as data preprocessing tools.
  - TensorFlow: A popular open-source library for machine learning and deep learning, used for building and training the Artificial Neural Network (ANN) model in this project.

- Data
  - The crime rate dataset used in this project should be available in a compatible format (e.g., CSV, Excel) and accessible to the Python environment. The dataset contains district-wise crime data across various states and union territories in India, spanning multiple years and including different types of crimes.

- Hardware Requirements
  - While the specific hardware requirements may vary depending on the size of the dataset and the complexity of the models, it is recommended to have a computer with a reasonably powerful processor and sufficient RAM (at least 8GB) to ensure smooth execution of the code.

- 5. Knowledge Prerequisites
  - Familiarity with Python programming language and its libraries mentioned above.
  - Understanding of data preprocessing techniques, such as handling missing data and encoding categorical variables.
  - Foundational knowledge of exploratory data analysis and data visualization techniques.

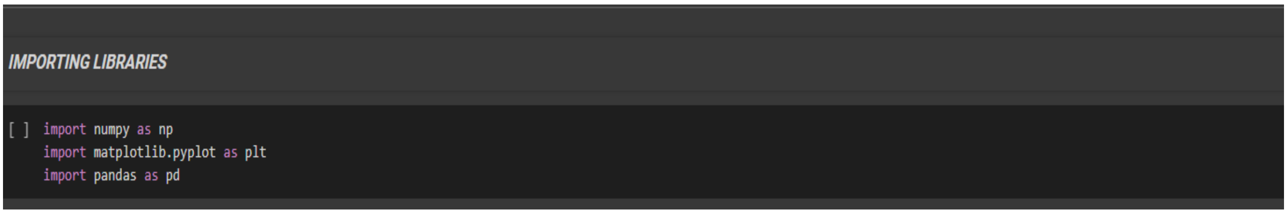
- Concepts of machine learning algorithms, including supervised and unsupervised learning methods (e.g., clustering, classification, and regression).
- Familiarity with model evaluation metrics, such as confusion matrices and accuracy scores.

By ensuring that these prerequisites are met, individuals interested in reproducing or extending this project will have the necessary tools and resources to work with the crime rate data, preprocess it, perform exploratory analysis, build and train machine learning models, and evaluate their performance.

3. Steps to build the project

1. Import Required Libraries

a. The project starts by importing the necessary Python libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn.



IMPORTING LIBRARIES

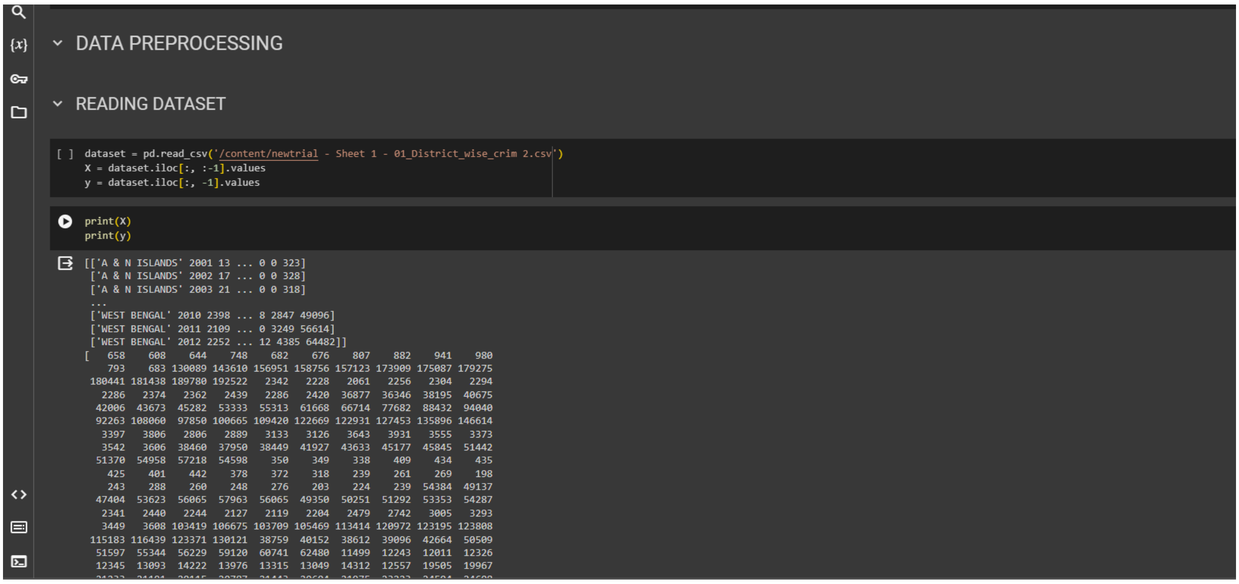
```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
```

1. Data Preprocessing

a. The crime rate dataset is read from a CSV file using Pandas.

b. Missing data is handled by replacing missing entries with the mean value of the respective column using SimpleImputer from Scikit-learn.

c. One-hot encoding is applied to the 'STATE/UT' column using OneHotEncoder from Scikit-learn.



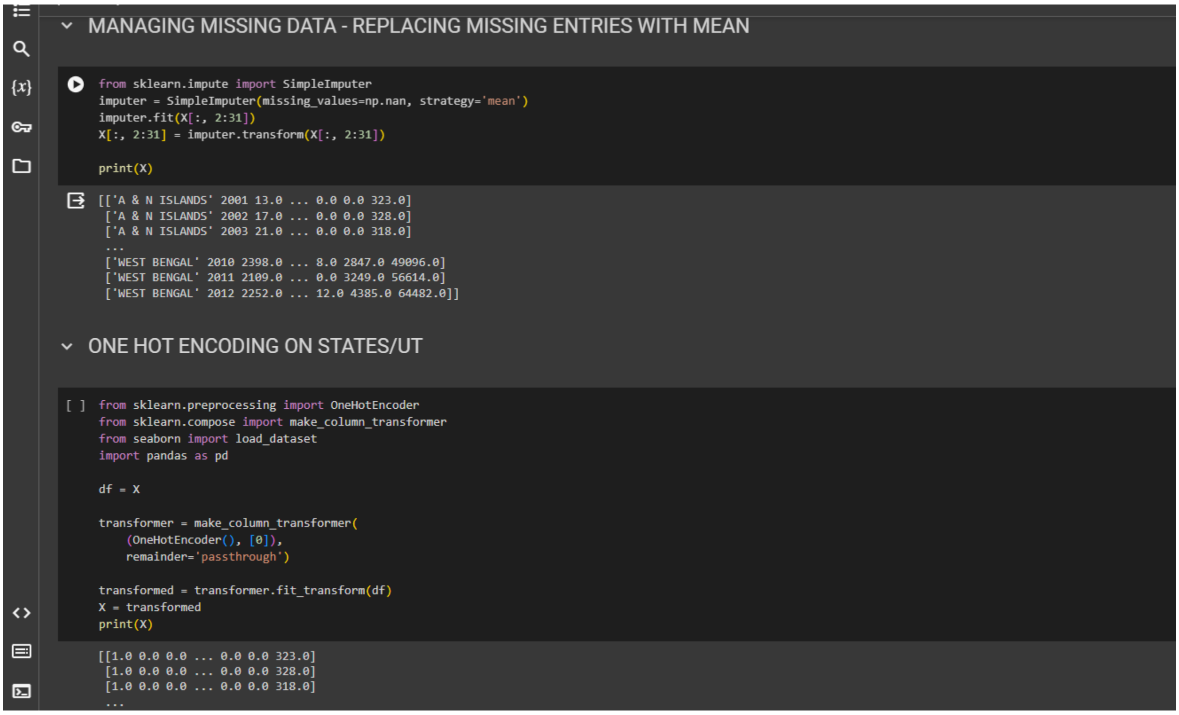
DATA PREPROCESSING

READING DATASET

```
[ ] dataset = pd.read_csv('/content/newtrial - Sheet 1 - 01_District_wise_crim 2.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
```

```
print(X)
print(y)
```

```
[['A & N ISLANDS' 2001 13 ... 0 0 323]
 ['A & N ISLANDS' 2002 17 ... 0 0 328]
 ['A & N ISLANDS' 2003 21 ... 0 0 318]
 ...
 ['WEST BENGAL' 2010 2398 ... 8 2847 49096]
 ['WEST BENGAL' 2011 2109 ... 0 3249 56614]
 ['WEST BENGAL' 2012 2252 ... 12 4385 64482]]
[   658    608    644    748    682    676    807    882    941    980
    793    683 130089 143610 156951 158756 157123 173909 175087 179275
 180441 181438 189780 192522   2342   2228   2061   2256   2304   2294
   2286   2374   2362   2439   2286   2420  36877  36346  38195  40675
  42006  43673  45282  53333  55313  61668  66714  77682  88432  94040
  92263 108060  97850 100665 109420 122669 122931 127453 135896 146614
   3397   3806   2806   2889   3133   3126   3643   3931   3555   3373
   3542   3606  38460  37950  38449  41927  43633  45177  45845  51442
  51370  54958  57218  54598    350    349    338    409    434    435
    425    401    442    378    372    318    239    261    269    198
    243    288    260    248    276    203    224    239  54384  49137
  47404  53623  56065  57963  56065  49350  50251  51292  53353  54287
   2341   2440   2244   2127   2119   2204   2479   2742   3005   3293
   3449   3608 103419 106675 103709 105469 113414 120972 123195 123808
 115183 116439 123371 130121  38759  40152  38612  39096  42664  50509
  51597  55344  56229  59120  60741  62480  11499  12243  12011  12326
  12345  13093  14222  13976  13315  13049  14312  12557  19505  19967
```



MANAGING MISSING DATA - REPLACING MISSING ENTRIES WITH MEAN

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 2:31])
X[:, 2:31] = imputer.transform(X[:, 2:31])

print(X)
```

```
[['A & N ISLANDS' 2001 13.0 ... 0.0 0.0 323.0]
 ['A & N ISLANDS' 2002 17.0 ... 0.0 0.0 328.0]
 ['A & N ISLANDS' 2003 21.0 ... 0.0 0.0 318.0]
 ...
 ['WEST BENGAL' 2010 2398.0 ... 8.0 2847.0 49096.0]
 ['WEST BENGAL' 2011 2109.0 ... 0.0 3249.0 56614.0]
 ['WEST BENGAL' 2012 2252.0 ... 12.0 4385.0 64482.0]]
```

ONE HOT ENCODING ON STATES/UT

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from seaborn import load_dataset
import pandas as pd

df = X

transformer = make_column_transformer(
    (OneHotEncoder(), [0]),
    remainder='passthrough')

transformed = transformer.fit_transform(df)
X = transformed
print(X)
```

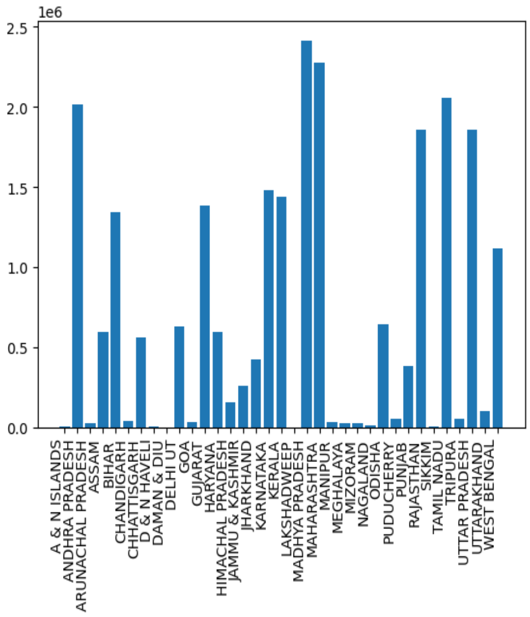
```
[[1.0 0.0 0.0 ... 0.0 0.0 323.0]
 [1.0 0.0 0.0 ... 0.0 0.0 328.0]
 [1.0 0.0 0.0 ... 0.0 0.0 318.0]
 ...
```

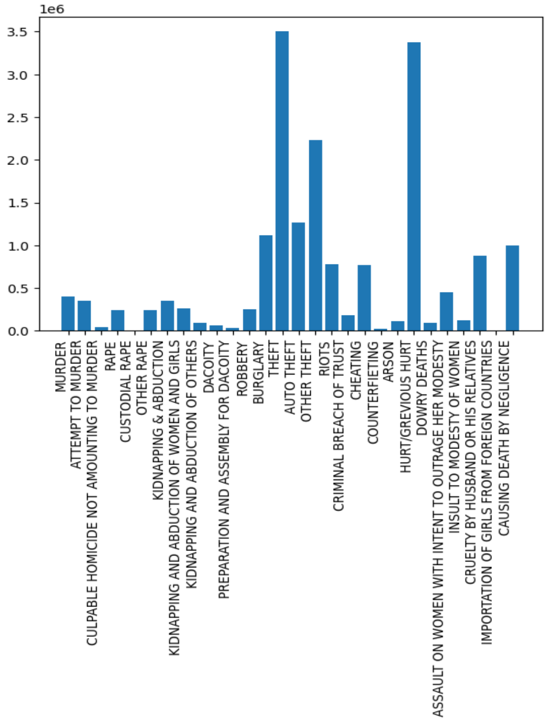
2. Exploratory Data Analysis (EDA)

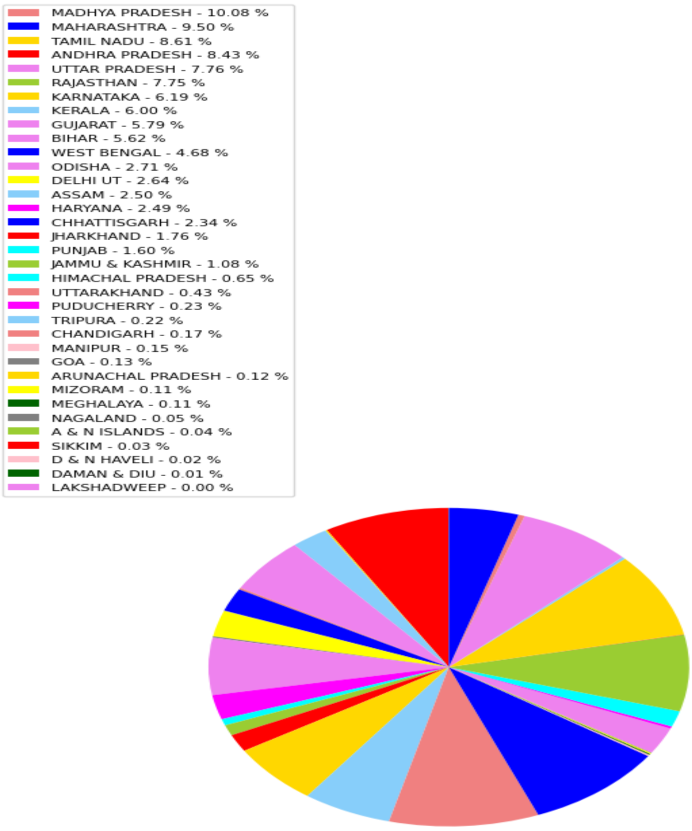
a. The total crime rate for each state/UT over 10 years is visualized using a bar plot.

b. A bar plot is created to represent the rate of different types of crimes.

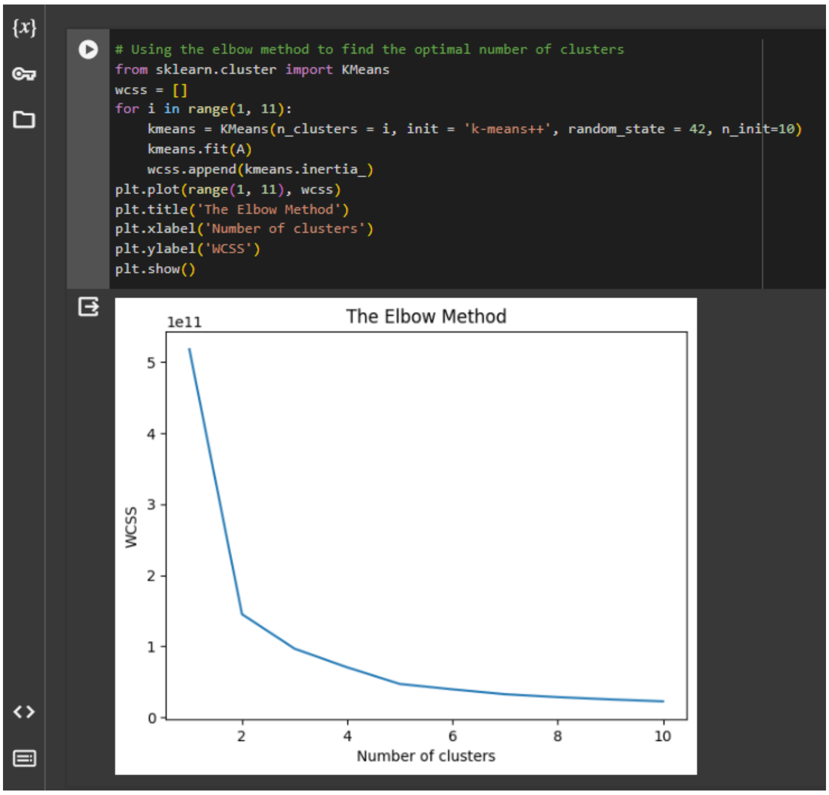
c. A pie chart is generated to show the crime rate distribution across states/UTs.







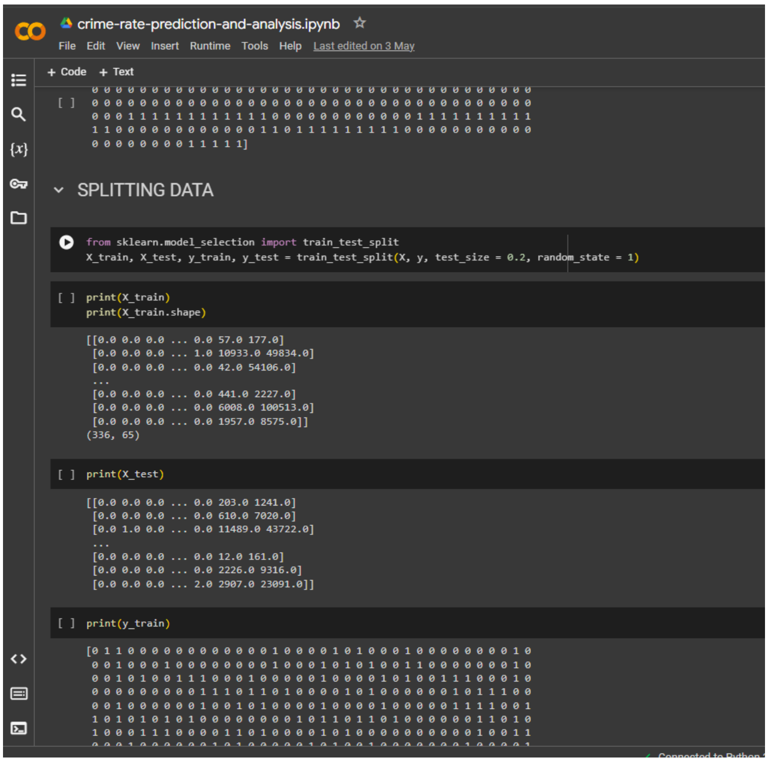
3. Clustering
a. The elbow method is used to determine the optimal number of clusters (k=2) for the crime rate data.
b. K-means clustering is applied to group states/UTs based on their crime rate patterns.
c. States/UTs are classified as safe or unsafe based on the clustering results.



```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42, n_init=10)
    kmeans.fit(A)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

4. Data Splitting

a. The preprocessed dataset is split into training and testing sets using train_test_split from Scikit-learn.



crime-rate-prediction-and-analysis.ipynb
File Edit View Insert Runtime Tools Help Last edited on 3 May
+ Code + Text

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1]
```

SPLITTING DATA

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

```
print(X_train)
print(X_train.shape)
```

```
[[0.0 0.0 0.0 ... 0.0 57.0 177.0]
 [0.0 0.0 0.0 ... 1.0 10933.0 49834.0]
 [0.0 0.0 0.0 ... 0.0 42.0 54106.0]
 ...
 [0.0 0.0 0.0 ... 0.0 441.0 2227.0]
 [0.0 0.0 0.0 ... 0.0 6008.0 100513.0]
 [0.0 0.0 0.0 ... 0.0 1957.0 8575.0]]
(336, 65)
```

```
print(X_test)
```

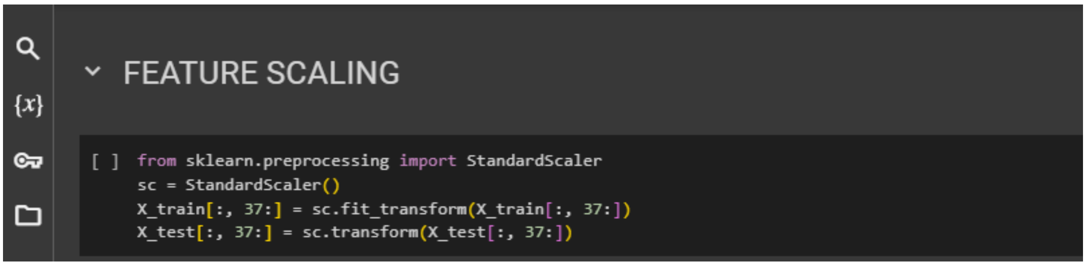
```
[[0.0 0.0 0.0 ... 0.0 203.0 1241.0]
 [0.0 0.0 0.0 ... 0.0 610.0 7020.0]
 [0.0 1.0 0.0 ... 0.0 11489.0 43722.0]
 ...
 [0.0 0.0 0.0 ... 0.0 12.0 161.0]
 [0.0 0.0 0.0 ... 0.0 2226.0 9316.0]
 [0.0 0.0 0.0 ... 2.0 2907.0 23091.0]]
```

```
print(y_train)
```

```
[0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0
 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0
 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0
 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1
 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
```

5. Feature Scaling

a. The numerical features in the training and testing datasets are scaled using StandardScaler from Scikit-learn.

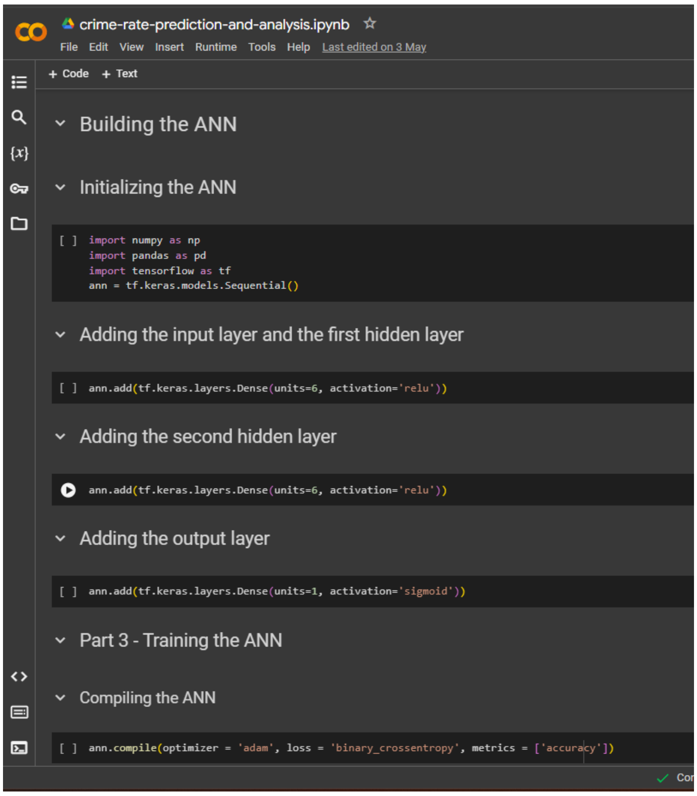


FEATURE SCALING

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 37:] = sc.fit_transform(X_train[:, 37:])
X_test[:, 37:] = sc.transform(X_test[:, 37:])
```

6. Model Building

a. An Artificial Neural Network (ANN) model is built and trained using TensorFlow's Keras API.
b. The following models are trained using Scikit-learn:
c. Support Vector Machine (SVM) with linear and RBF kernels
d. K-Nearest Neighbors (K-NN)
e. Logistic Regression
f. Random Forest Regression
g. Decision Tree Regression
h. Support Vector Regression (SVR)


crime-rate-prediction-and-analysis.ipynb
File Edit View Insert Runtime Tools Help Last edited on 3 May
+ Code + Text

Building the ANN

Initializing the ANN

```
import numpy as np
import pandas as pd
import tensorflow as tf
ann = tf.keras.models.Sequential()
```

Adding the input layer and the first hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the second hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

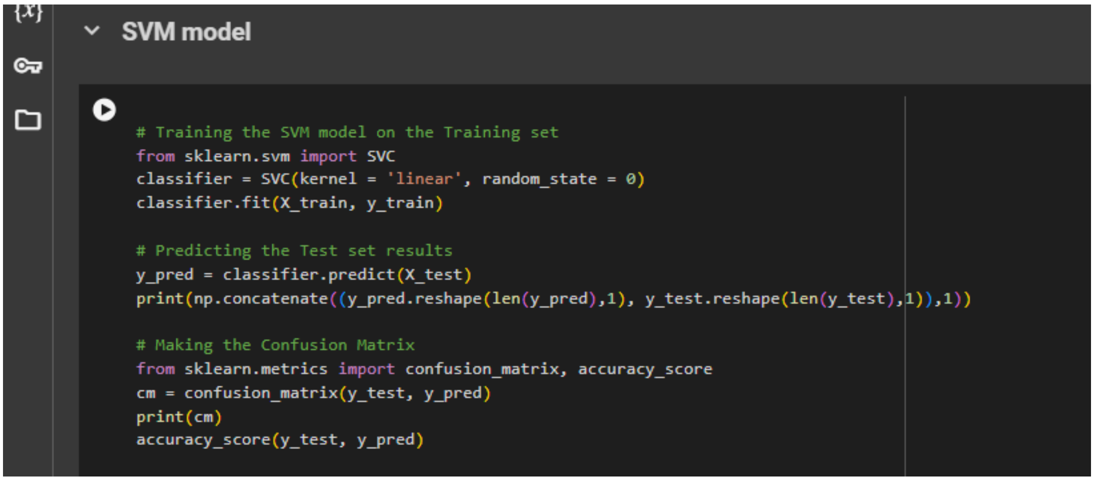
Adding the output layer

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Part 3 - Training the ANN

Compiling the ANN

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

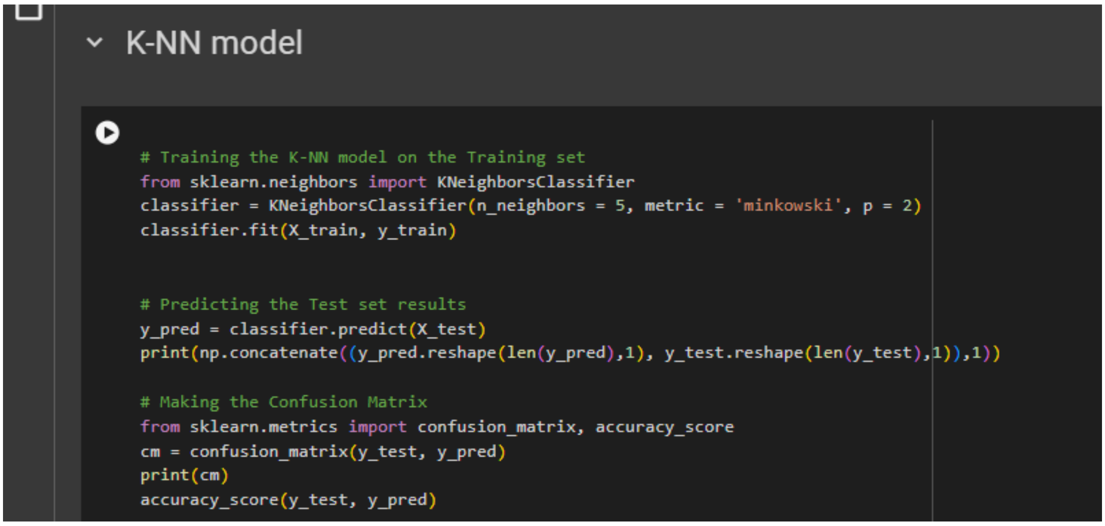


SVM model

```
# Training the SVM model on the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

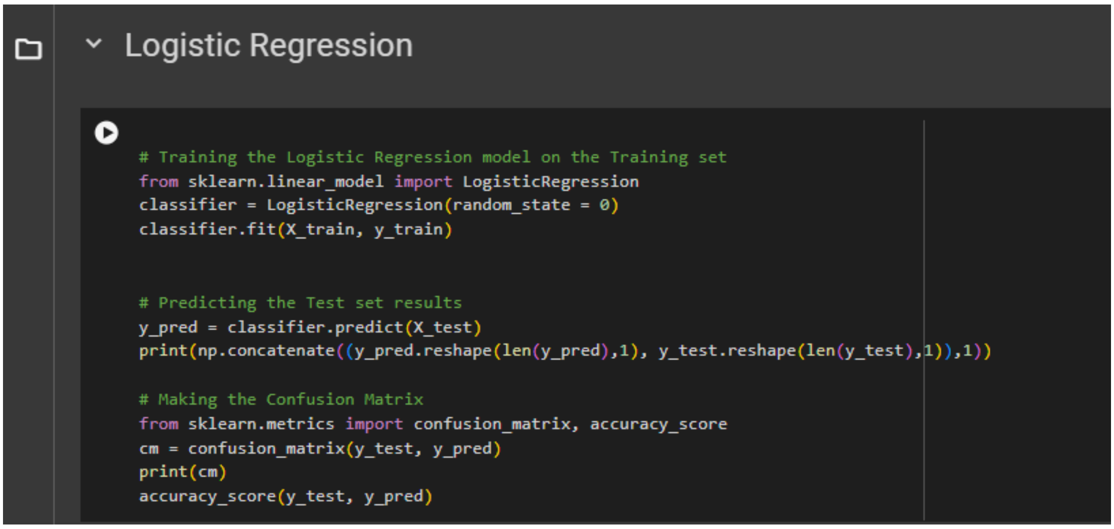


K-NN model

```
# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)


# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

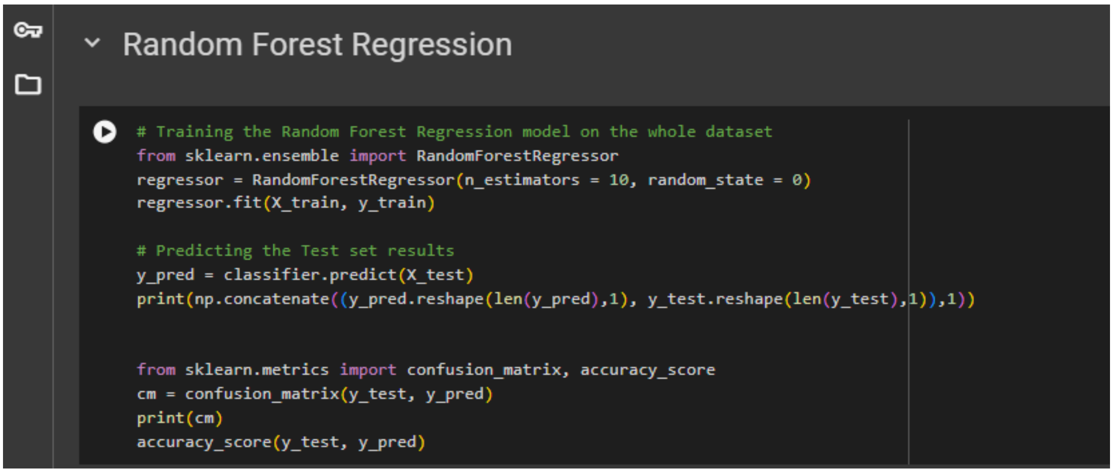


Logistic Regression

```
# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)


# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

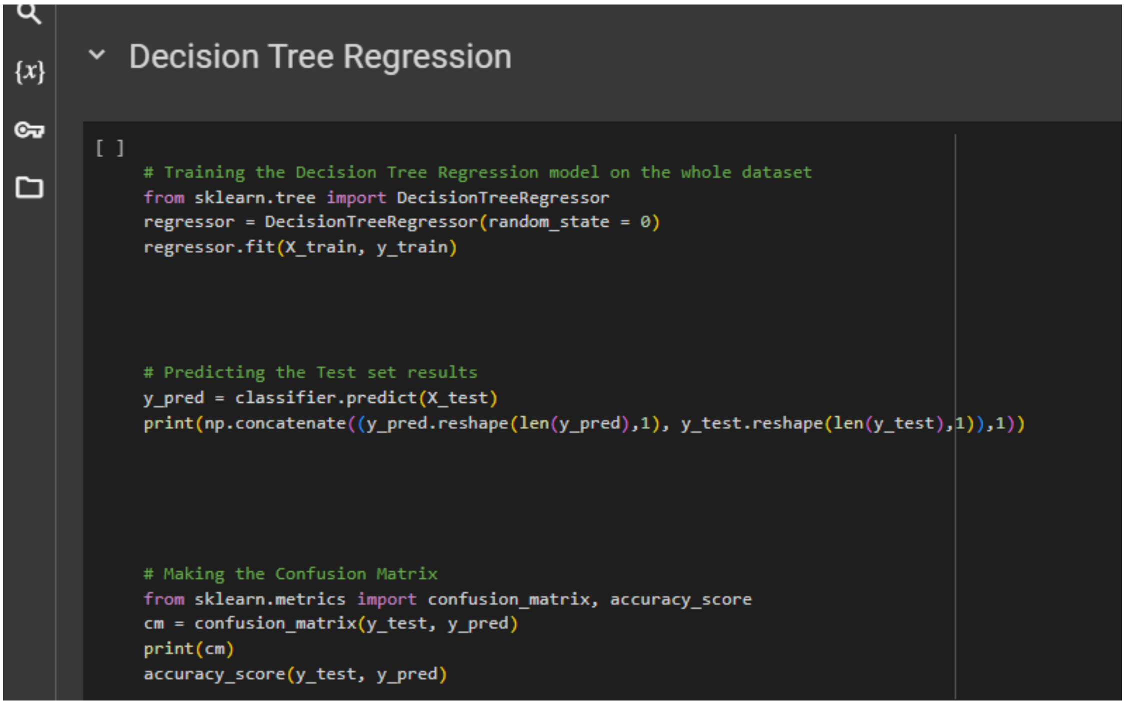


Random Forest Regression

```
# Training the Random Forest Regression model on the whole dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))


from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```



Decision Tree Regression

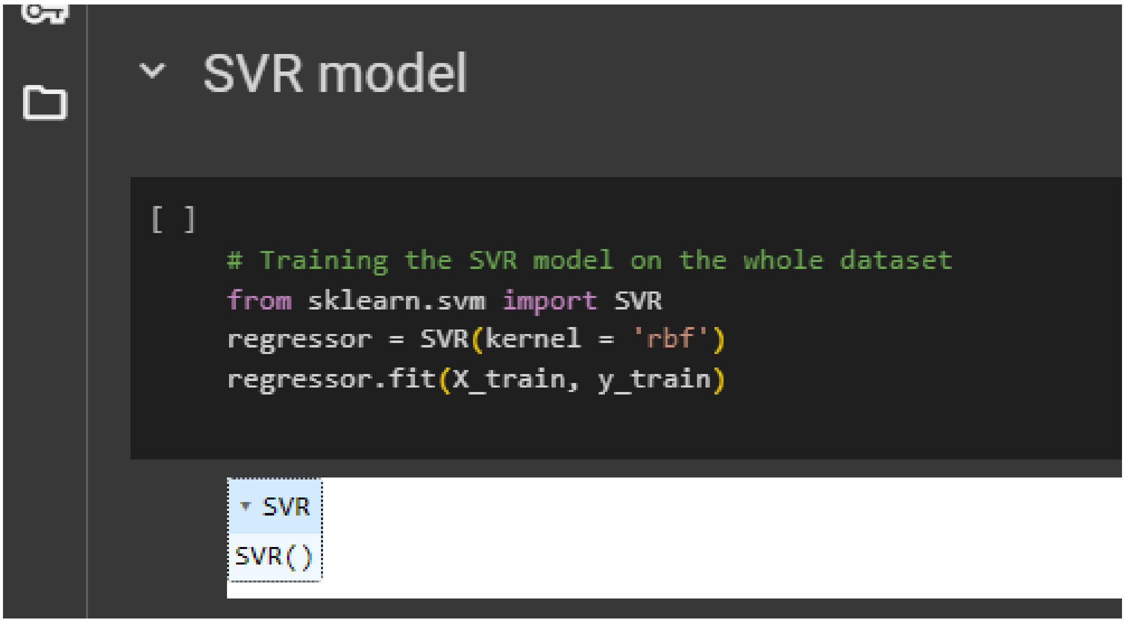
```
# Training the Decision Tree Regression model on the whole dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)



# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))



# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```



SVR model

```
# Training the SVR model on the whole dataset
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)
```

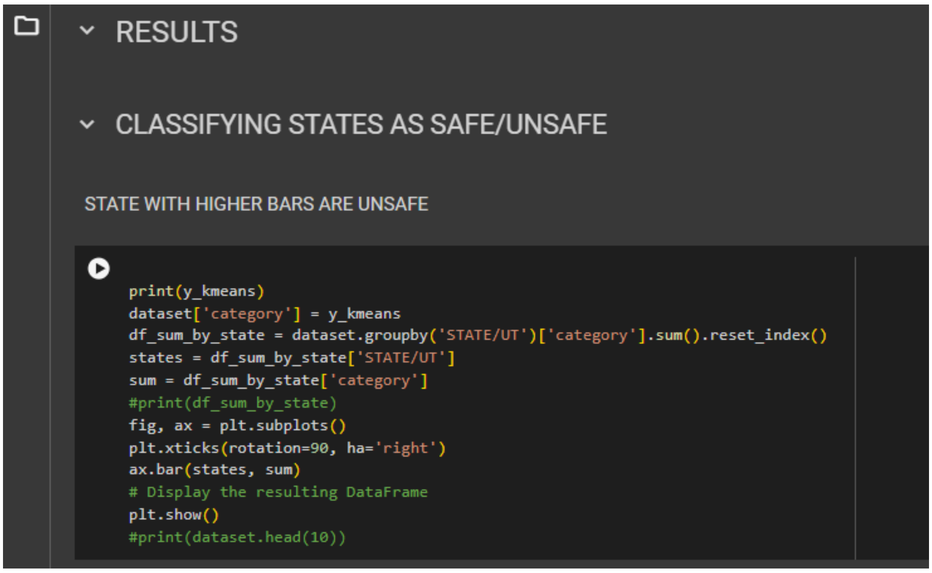
```
▾ SVR
SVR()
```

7. Model Evaluation
a. For each model:
b. Predictions are made on the testing set.
c. The confusion matrix and accuracy score are calculated and printed.

9. Results and Discussion

- The clustering results are visualized to classify states/UTs as safe or unsafe based on their crime rates.
- Time series plots are generated to show the crime rate trends for individual states/UTs over the years.

4. Output

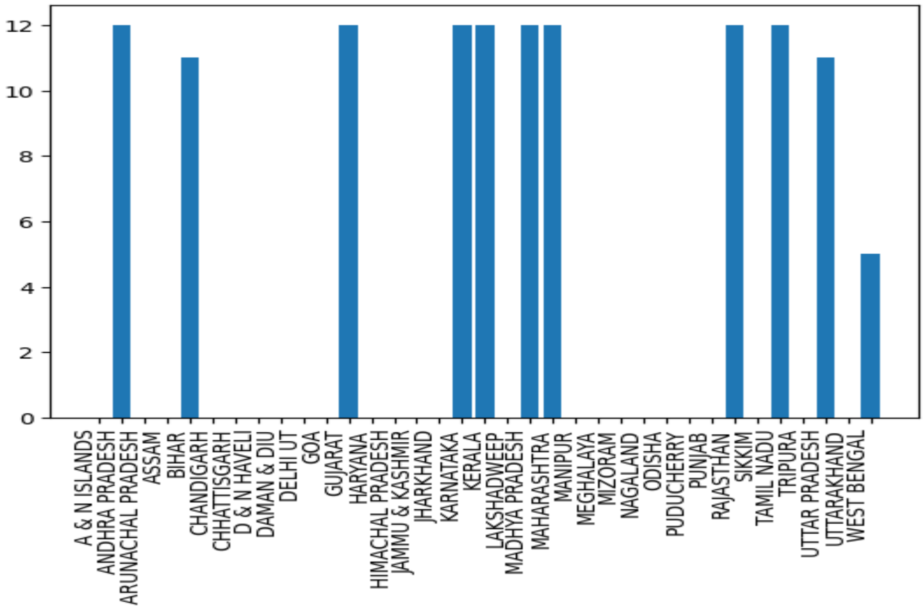


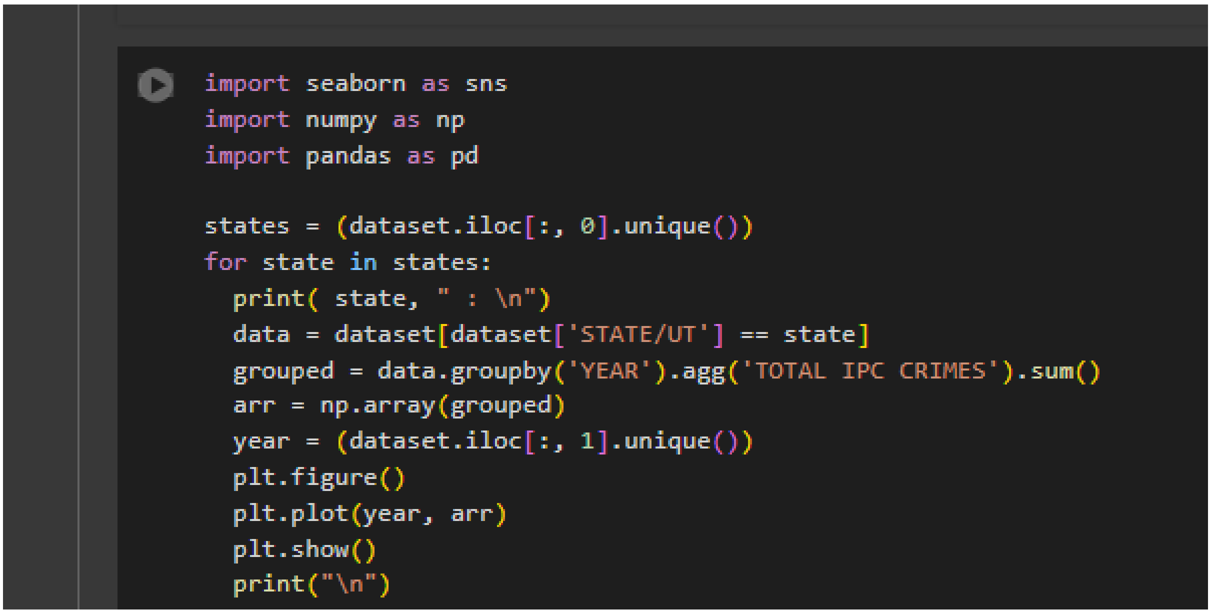
RESULTS

CLASSIFYING STATES AS SAFE/UNSAFE

STATE WITH HIGHER BARS ARE UNSAFE

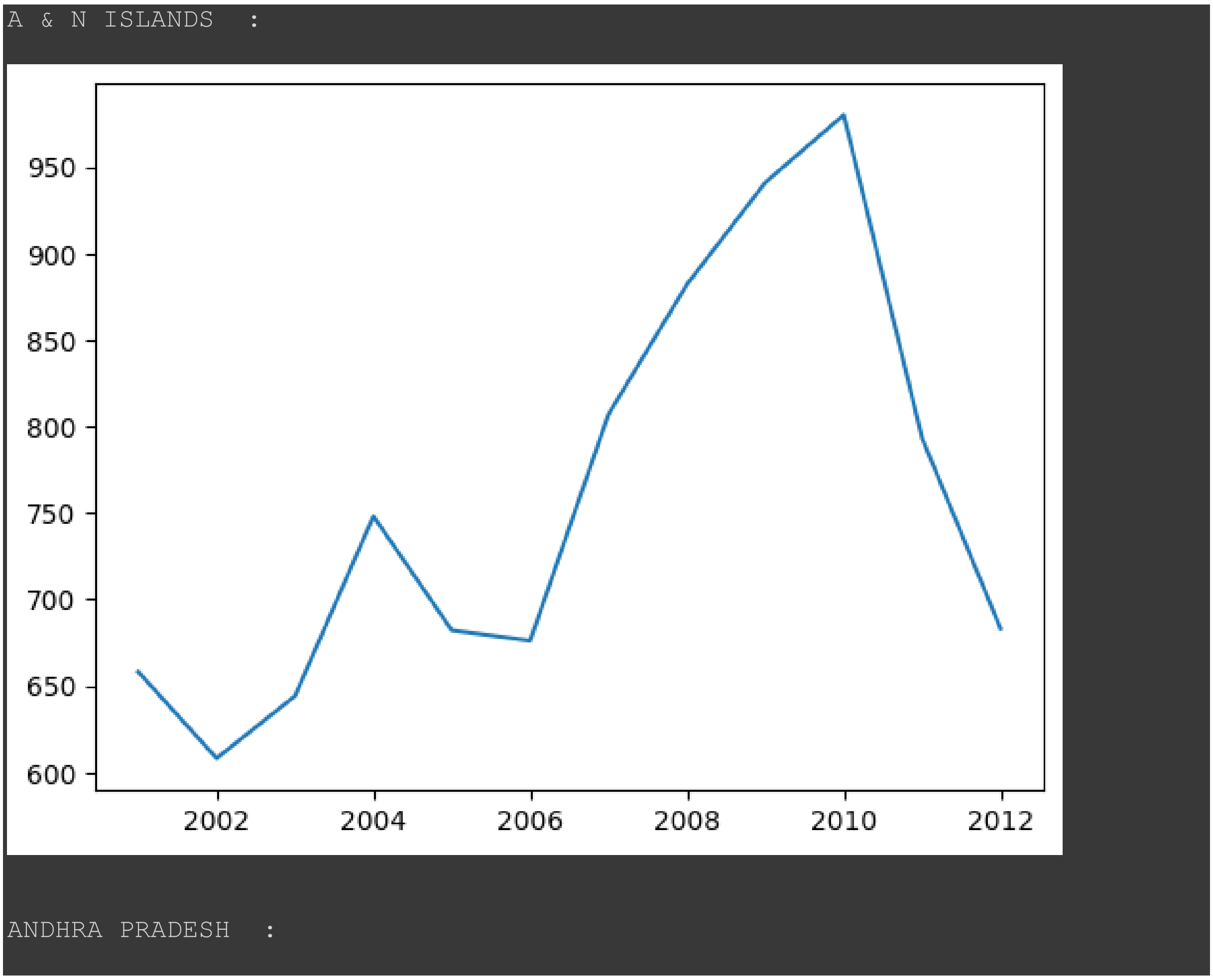
```
print(y_kmeans)
dataset['category'] = y_kmeans
df_sum_by_state = dataset.groupby('STATE/UT')['category'].sum().reset_index()
states = df_sum_by_state['STATE/UT']
sum = df_sum_by_state['category']
#print(df_sum_by_state)
fig, ax = plt.subplots()
plt.xticks(rotation=90, ha='right')
ax.bar(states, sum)
# Display the resulting DataFrame
plt.show()
#print(dataset.head(10))
```





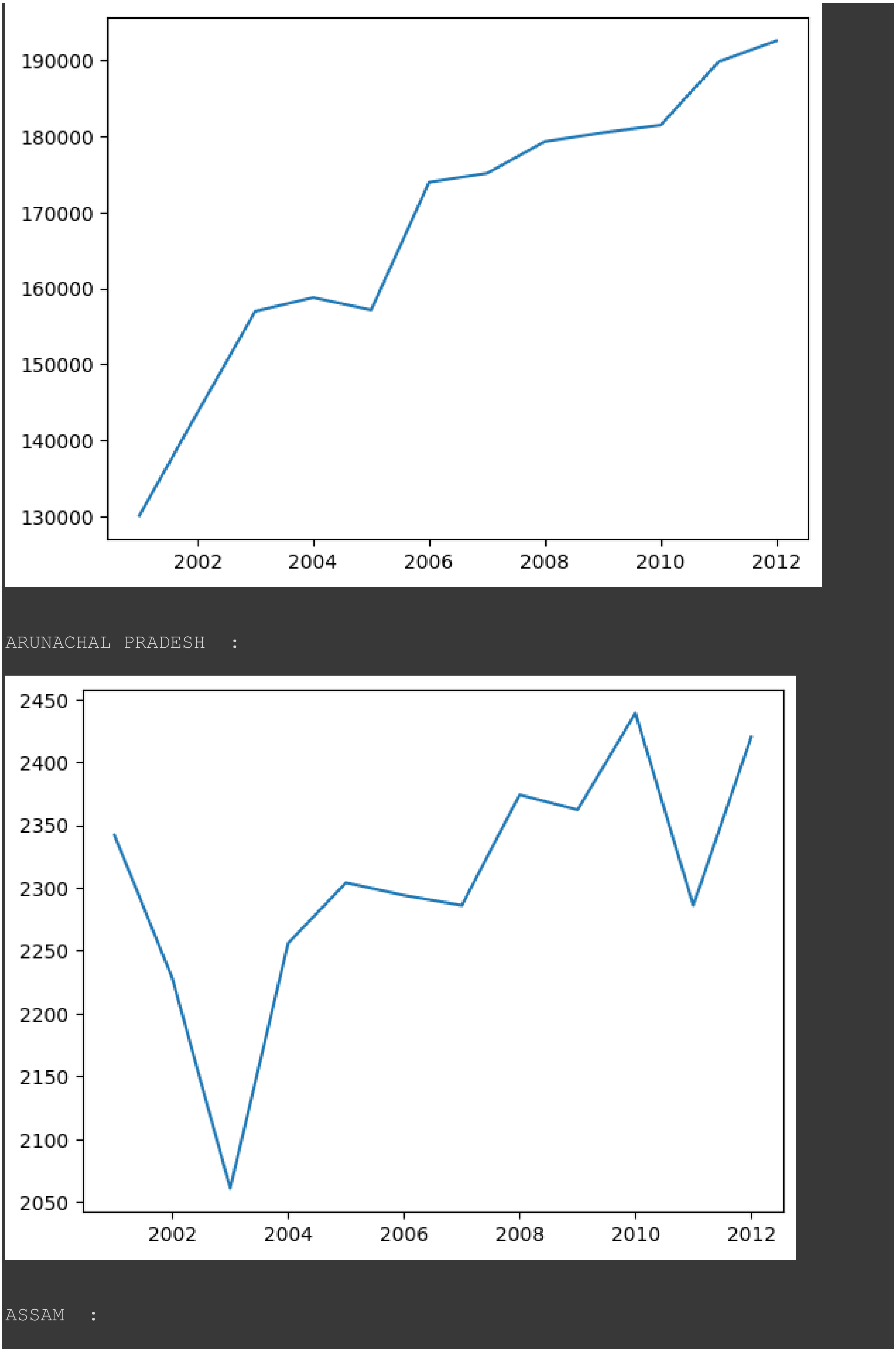
```python
import seaborn as sns
import numpy as np
import pandas as pd

states = (dataset.iloc[:, 0].unique())
for state in states:
  print( state, " : \n")
  data = dataset[dataset['STATE/UT'] == state]
  grouped = data.groupby('YEAR').agg('TOTAL IPC CRIMES').sum()
  arr = np.array(grouped)
  year = (dataset.iloc[:, 1].unique())
  plt.figure()
  plt.plot(year, arr)
  plt.show()
  print("\n")
```



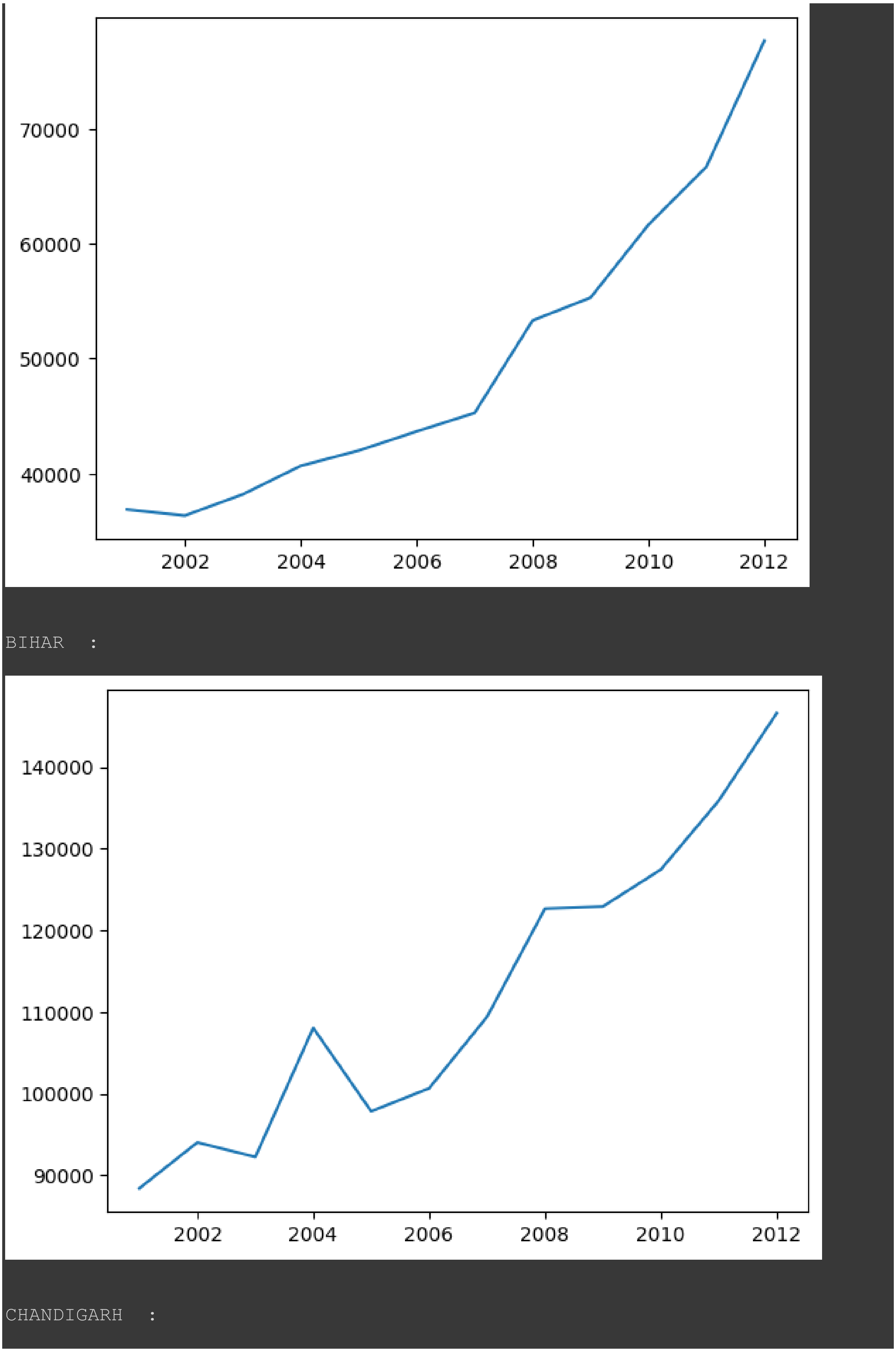
```
A & N ISLANDS  :
```

```
ANDHRA PRADESH  :
```



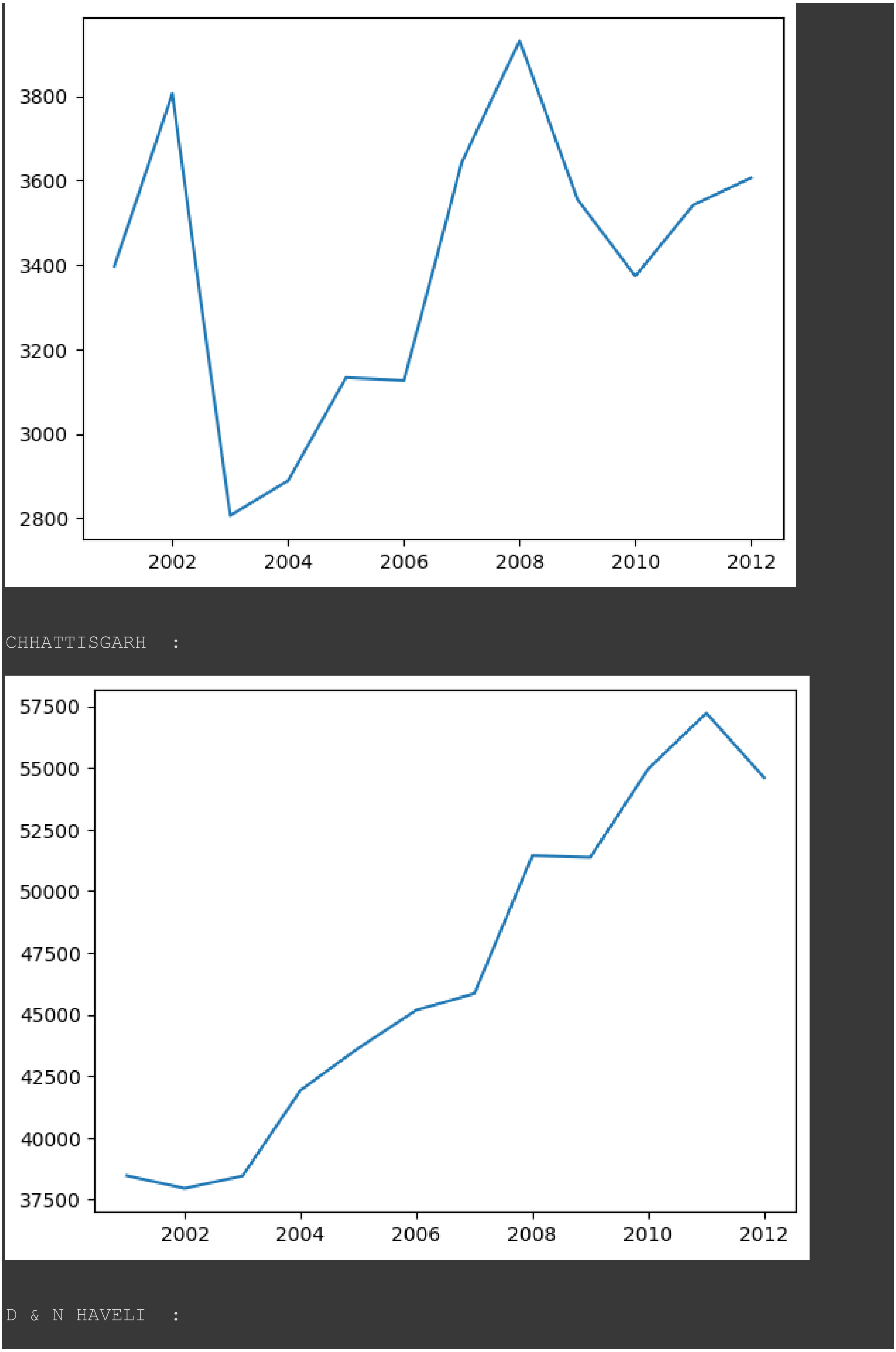
```
ARUNACHAL PRADESH  :
```

```
ASSAM  :
```



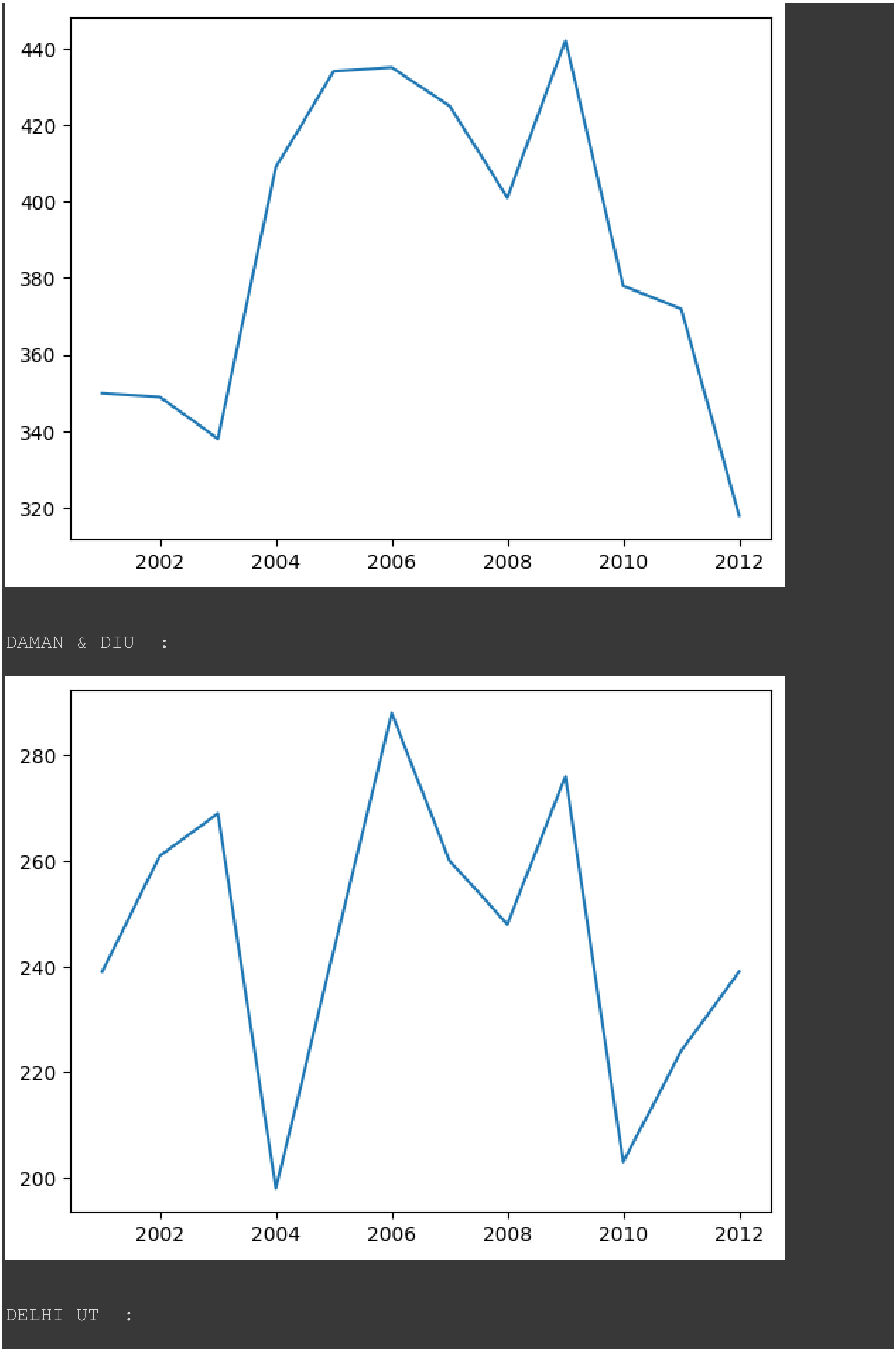
```
BIHAR  :
```

```
CHANDIGARH  :
```



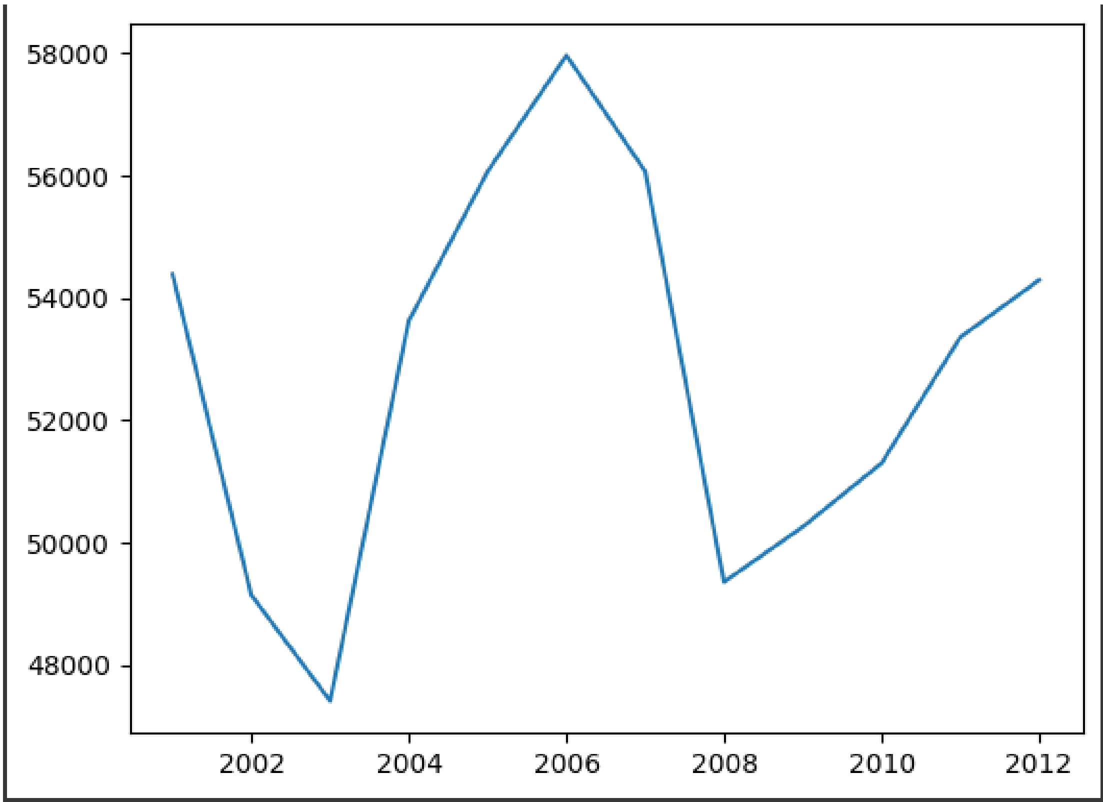
```
CHHATTISGARH   :
```

```
D & N HAVELI   :
```

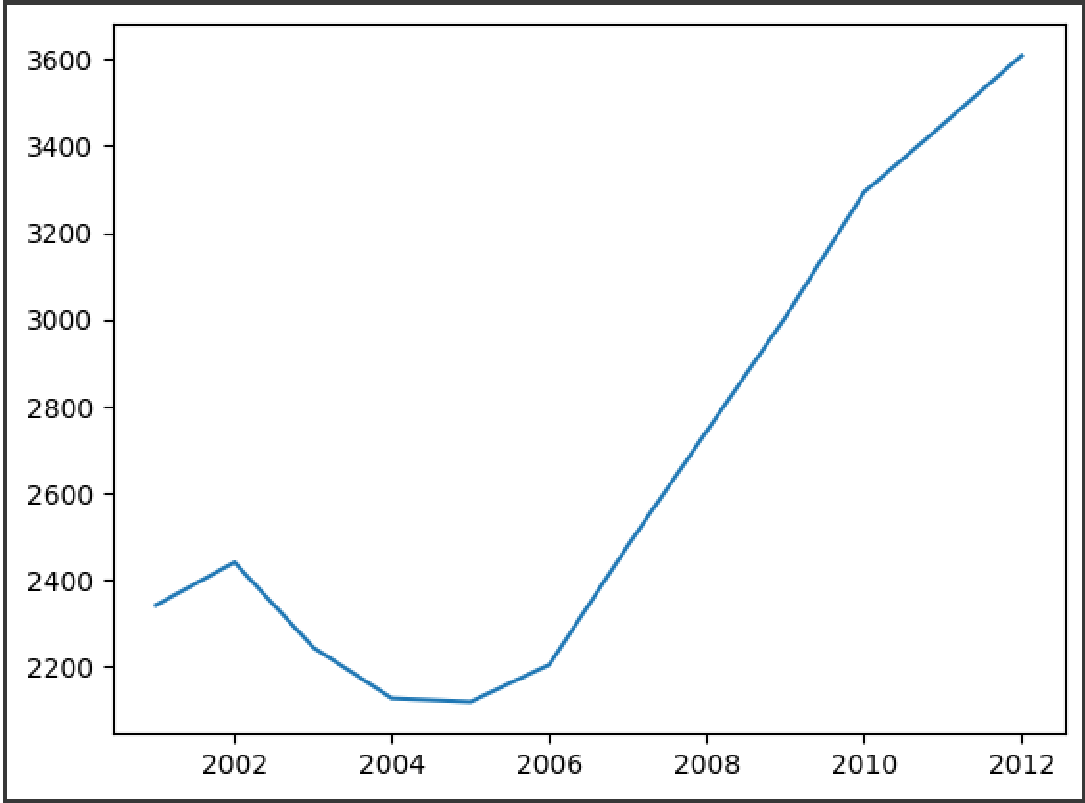


```
DAMAN & DIU  :
```

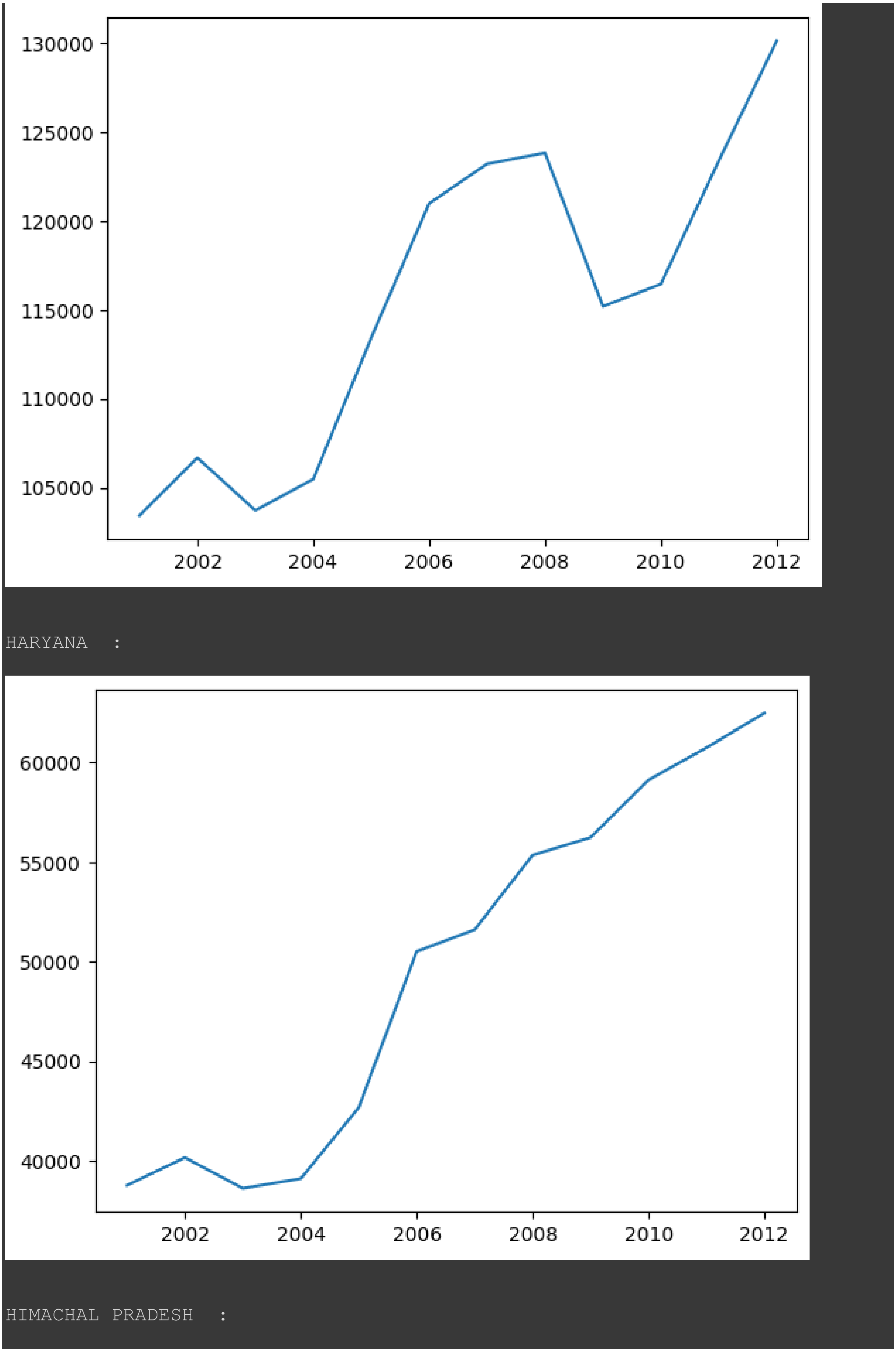
```
DELHI UT  :
```


58000
56000
54000
52000
50000
48000
2002
2004
2006
2008
2010
2012

```
GOA   :
```

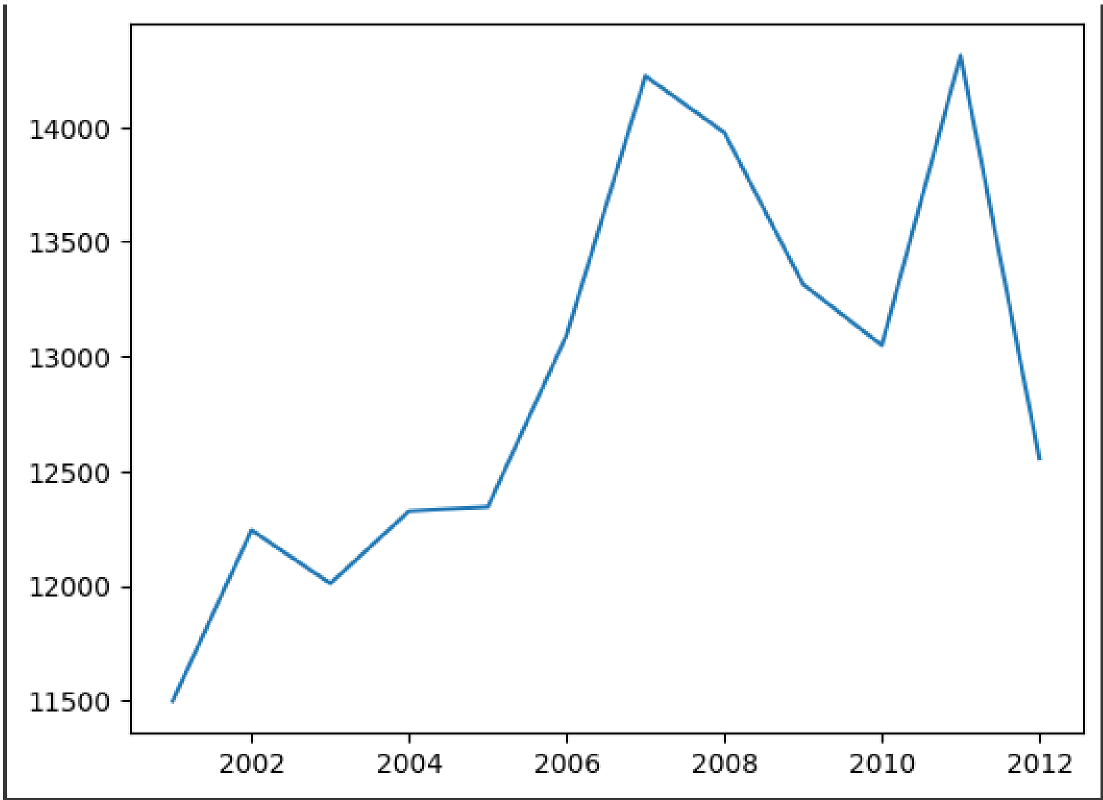

3600
3400
3200
3000
2800
2600
2400
2200
2002
2004
2006
2008
2010
2012

```
GUJARAT   :
```

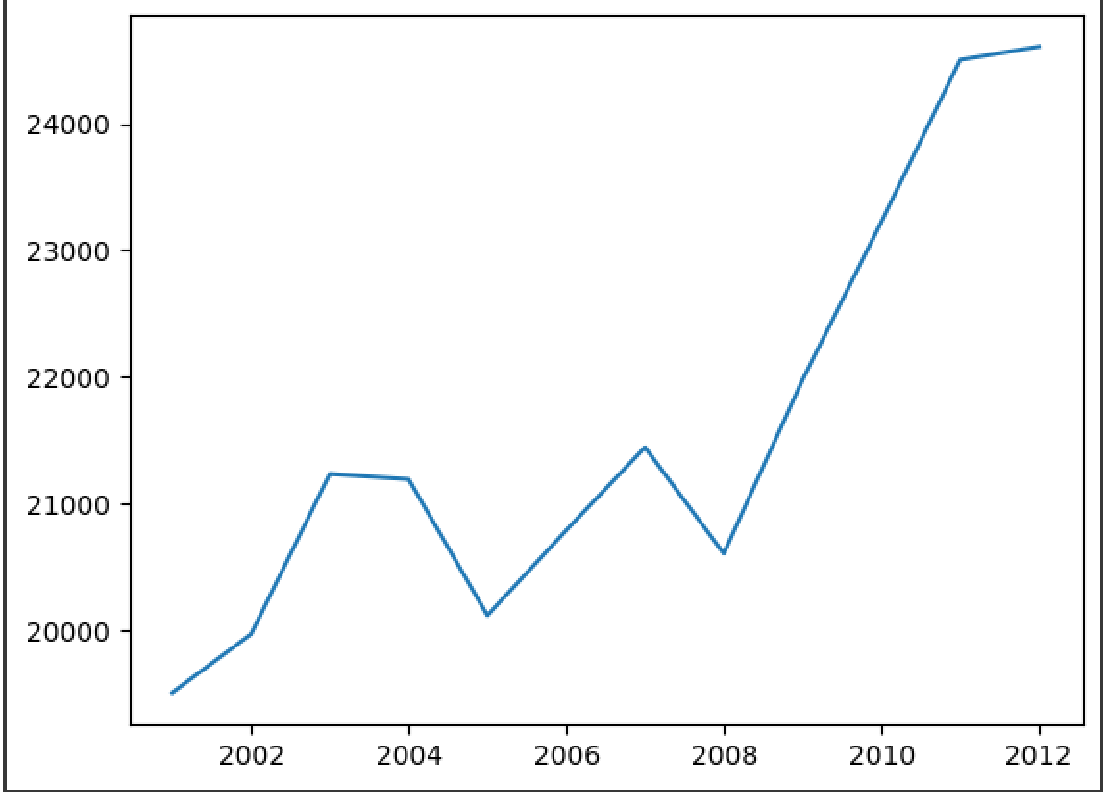


```
HARYANA  :
```

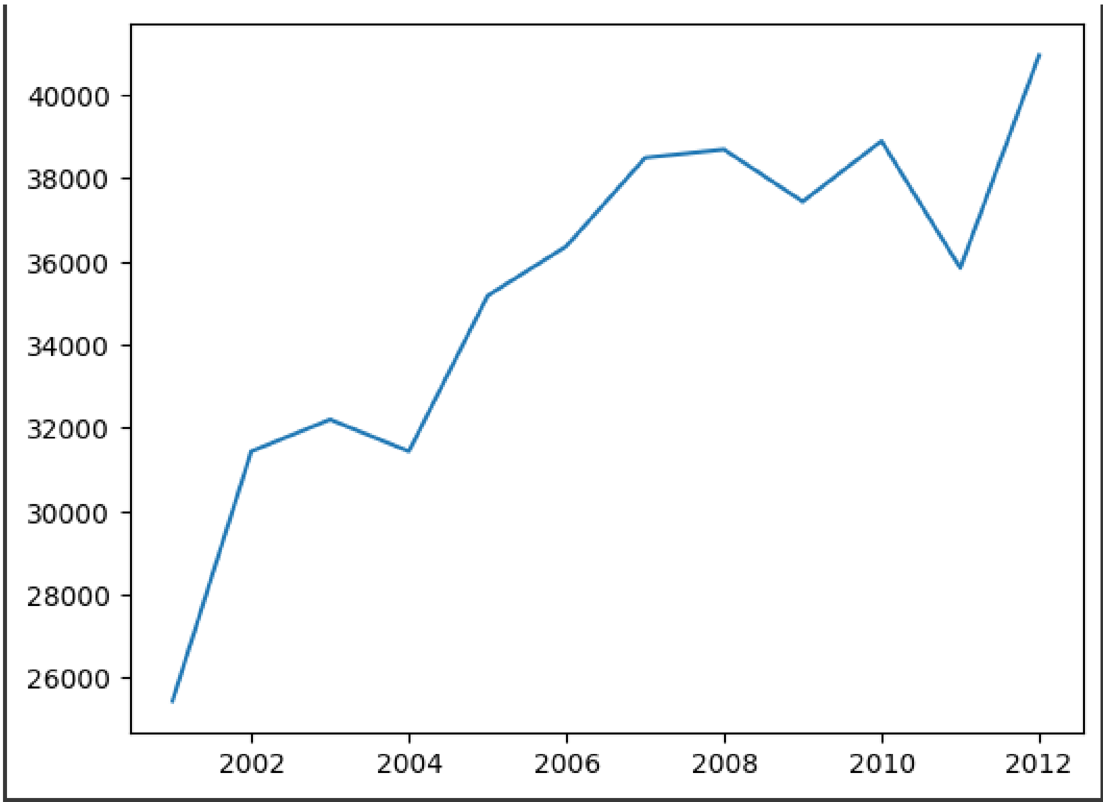
```
HIMACHAL PRADESH  :
```



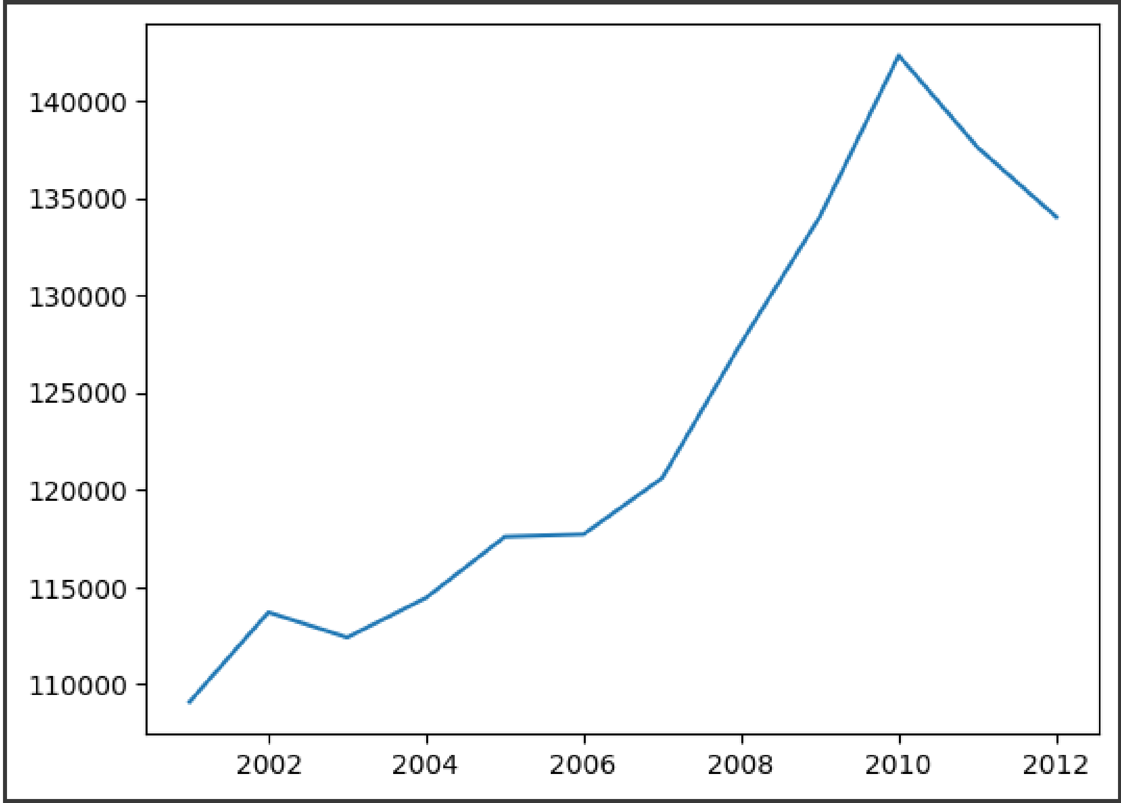
```
JAMMU & KASHMIR  :
```



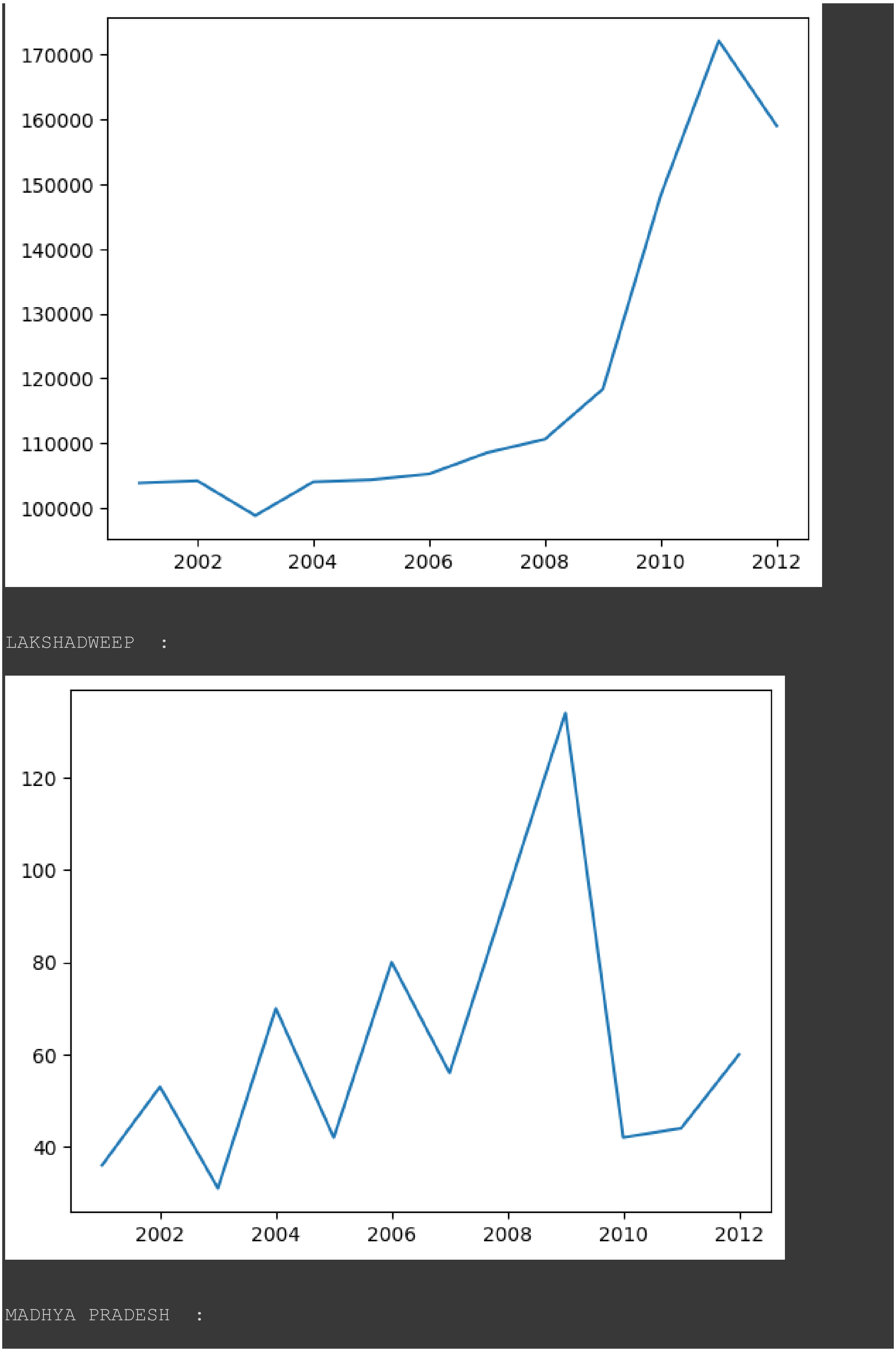
```
JHARKHAND  :
```



```
KARNATAKA   :
```

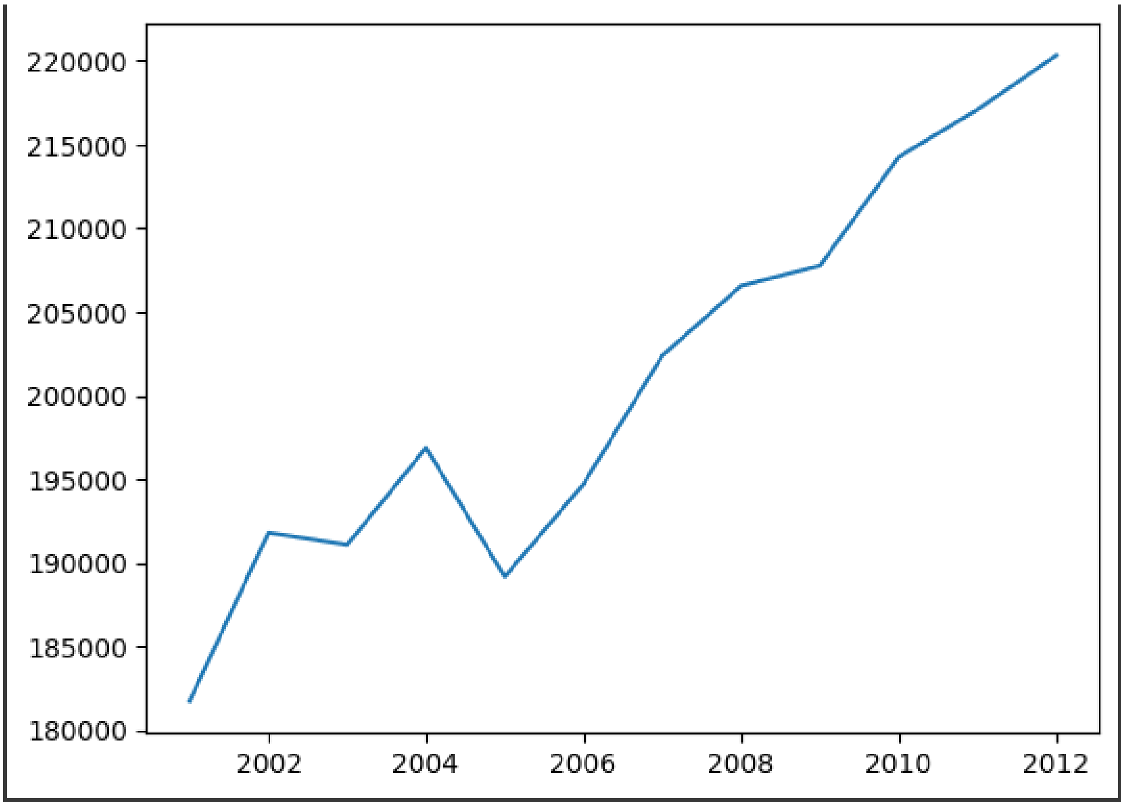


```
KERALA   :
```

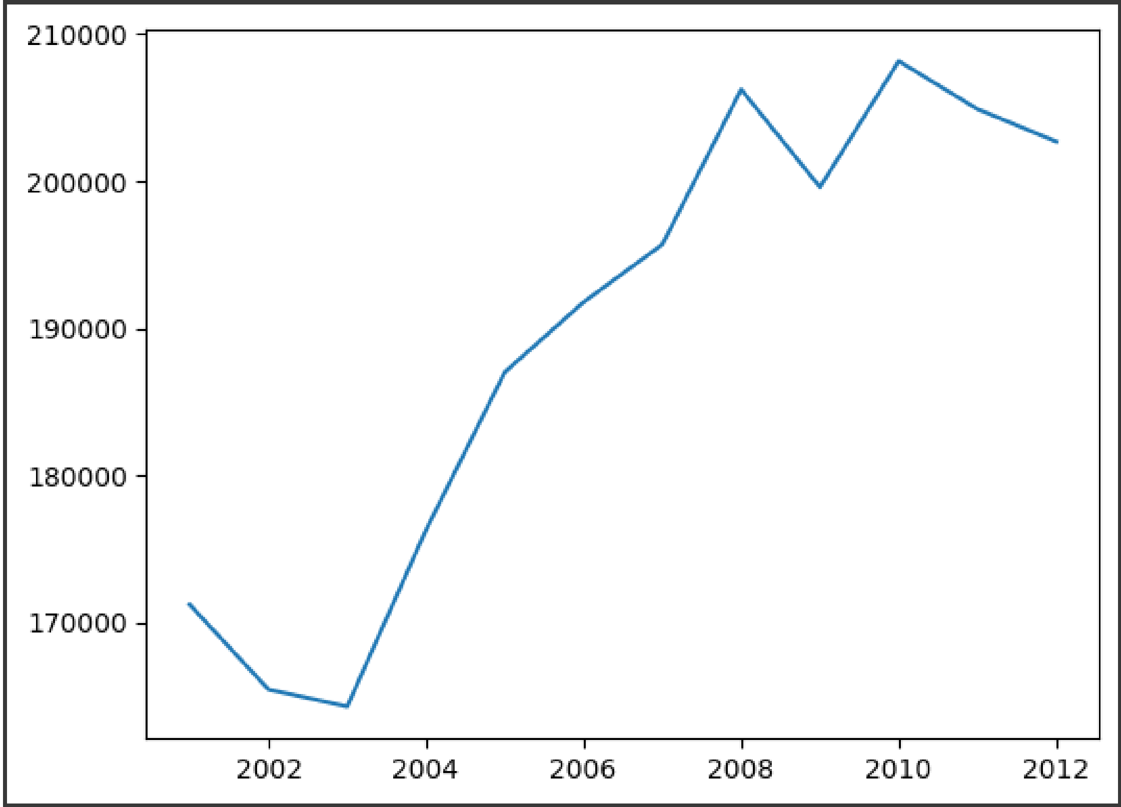


```
LAKSHADWEEP   :
```

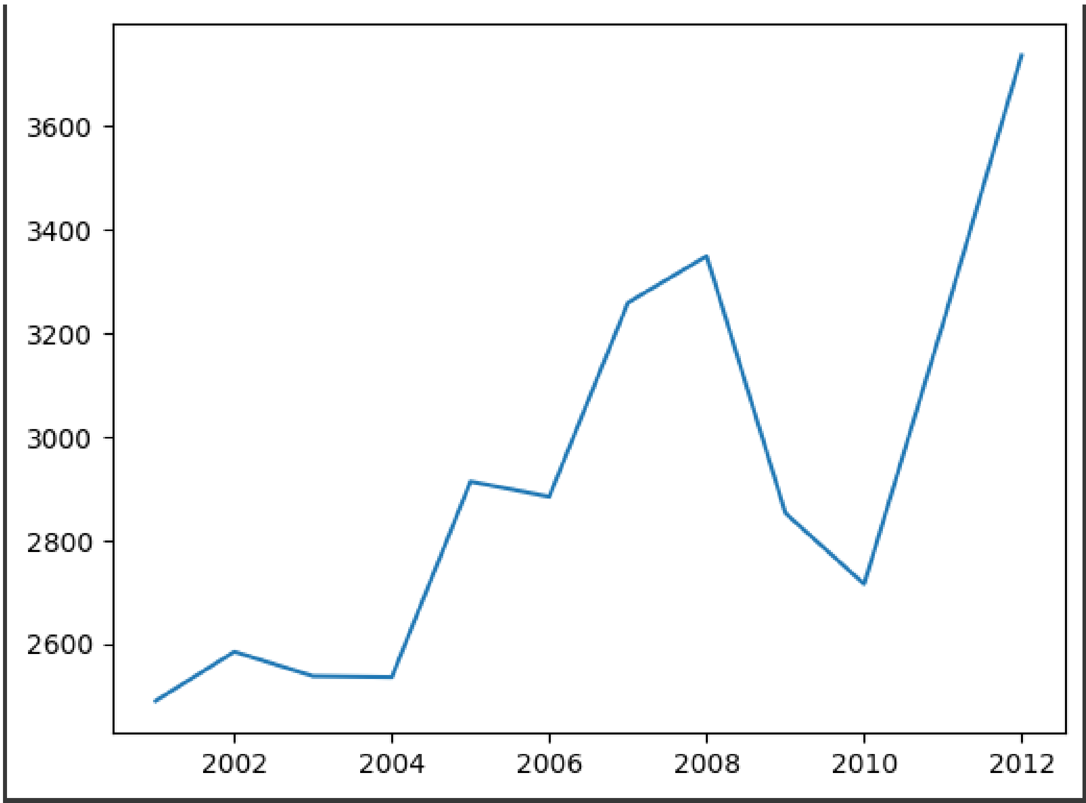
```
MADHYA PRADESH   :
```



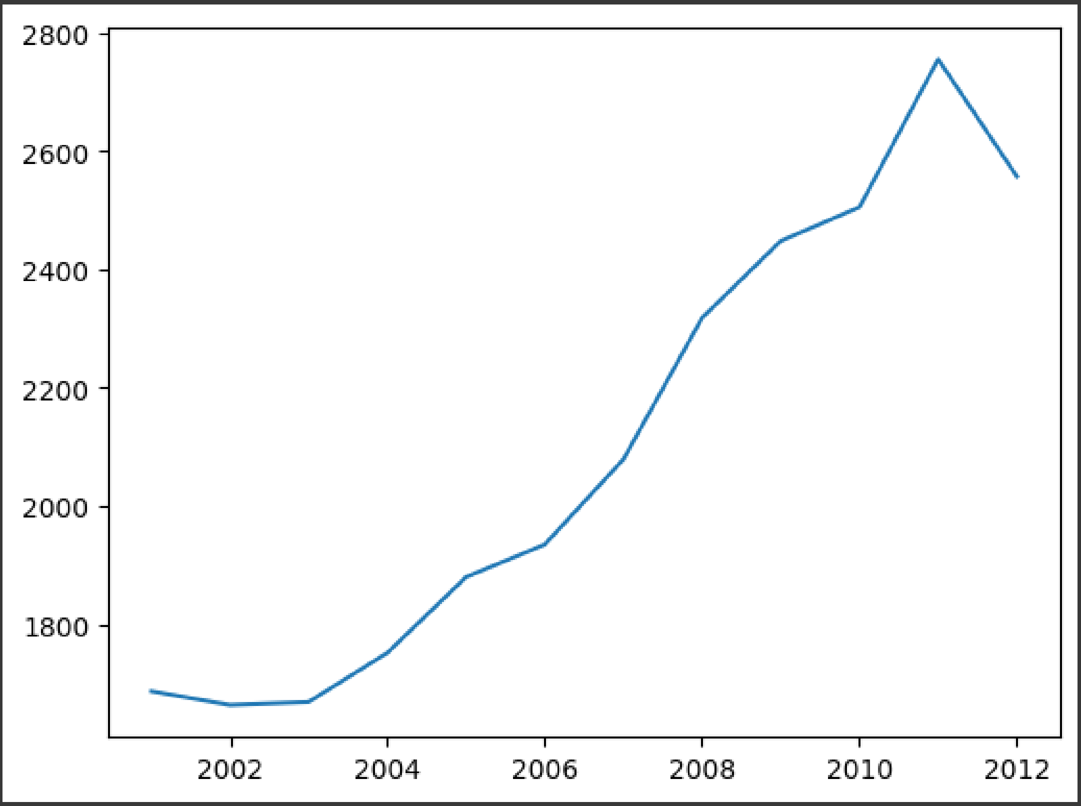
```
MAHARASHTRA  :
```



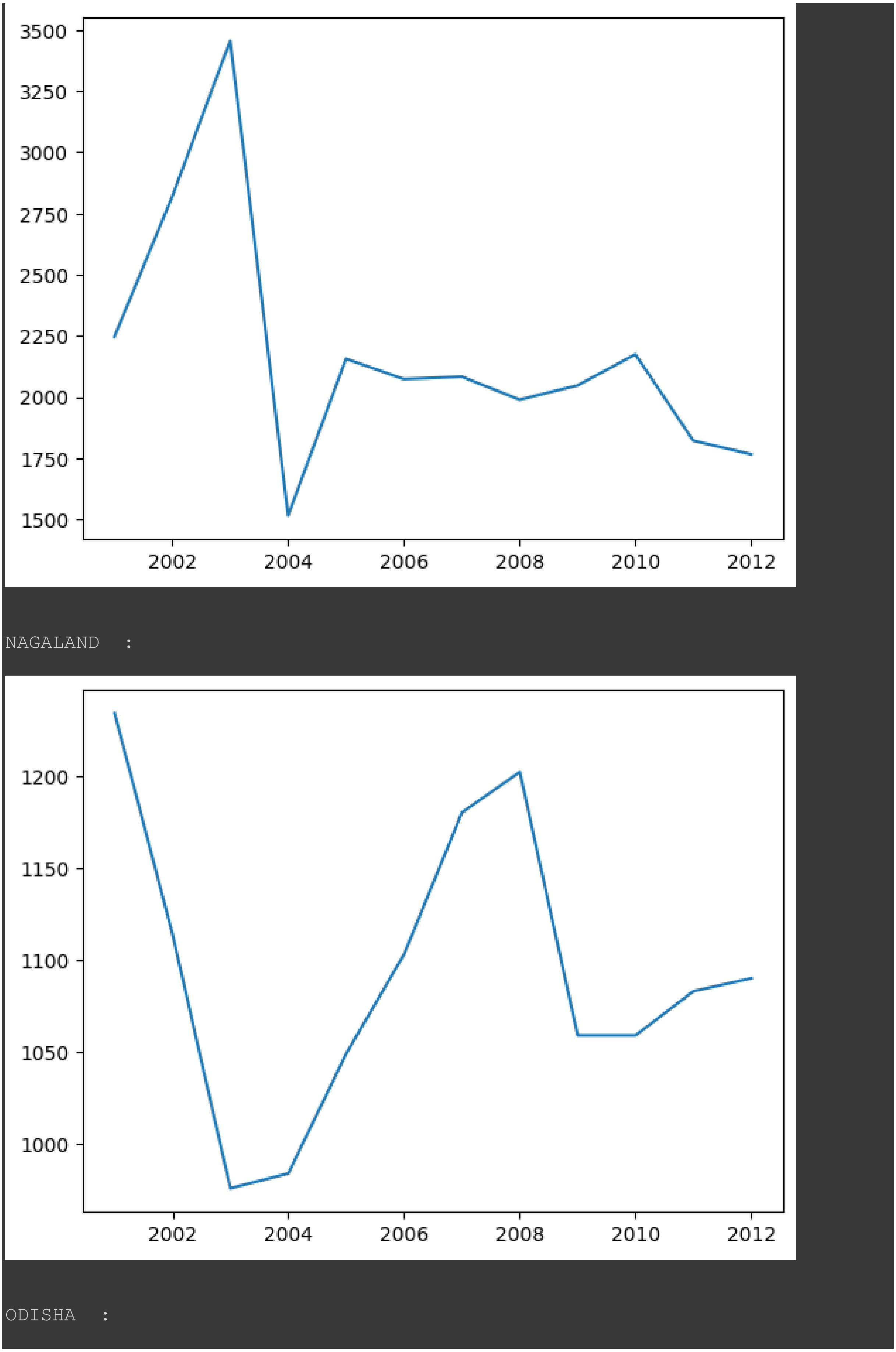
```
MANIPUR  :
```



```
MEGHALAYA  :
```

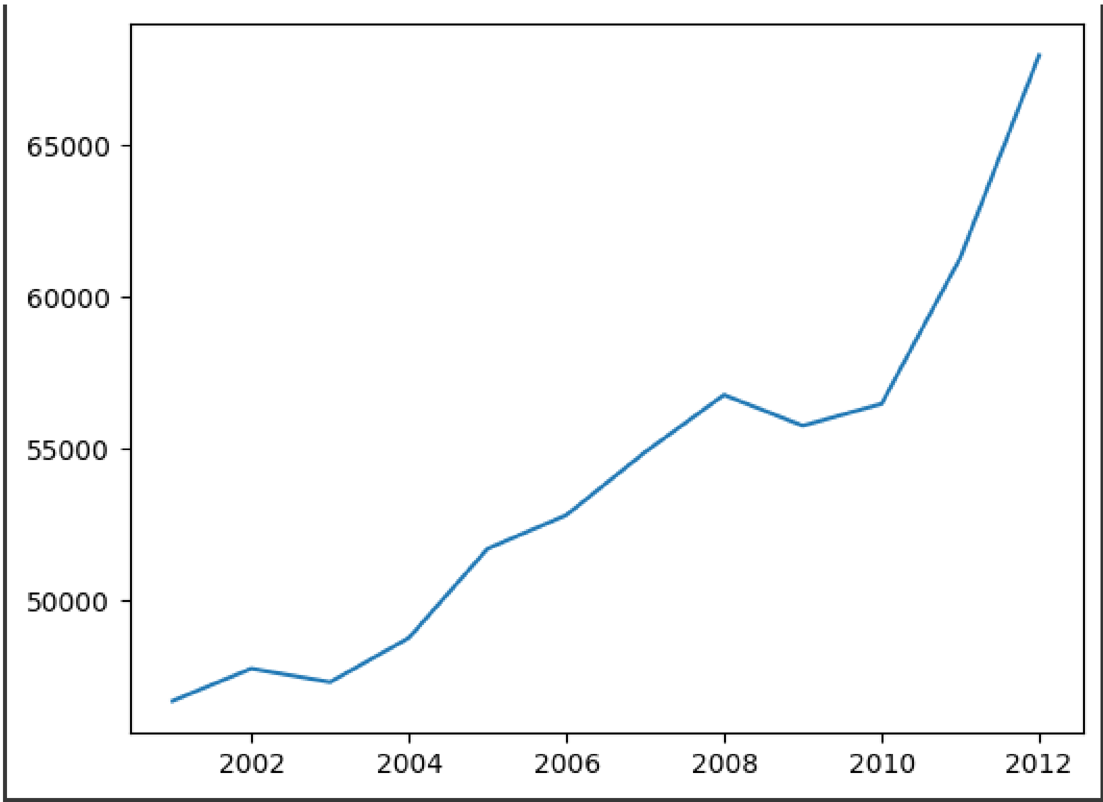


```
MIZORAM  :
```

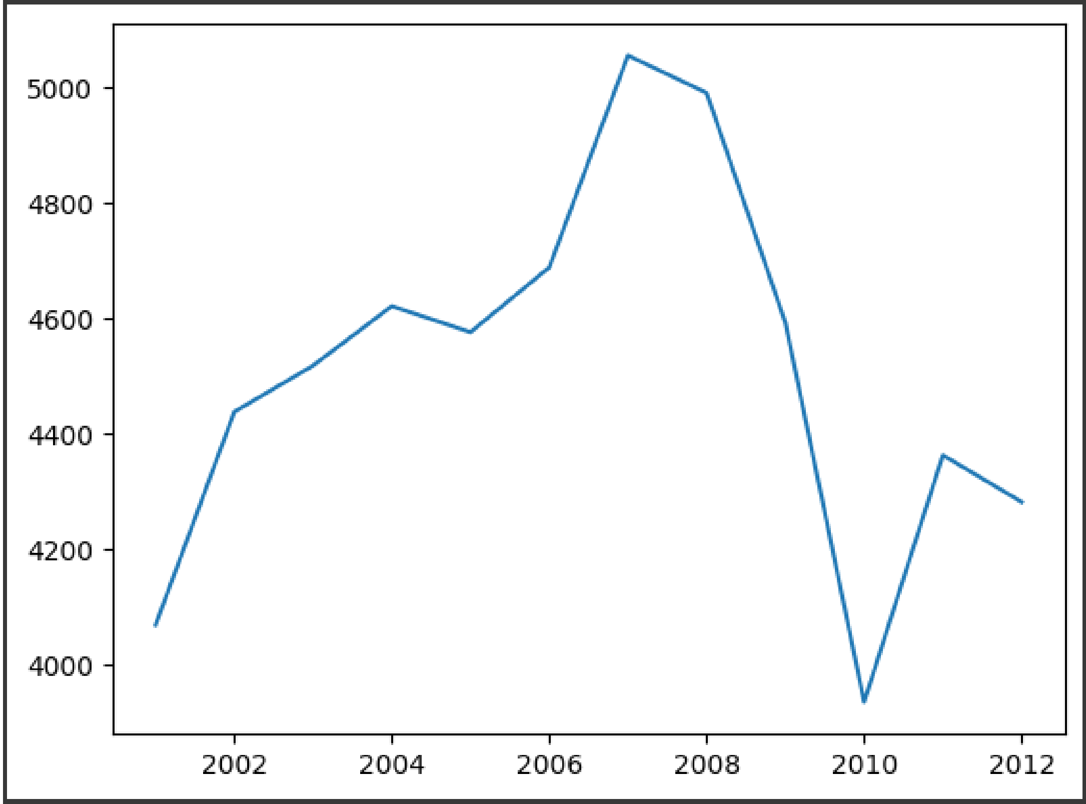


```
NAGALAND   :
```

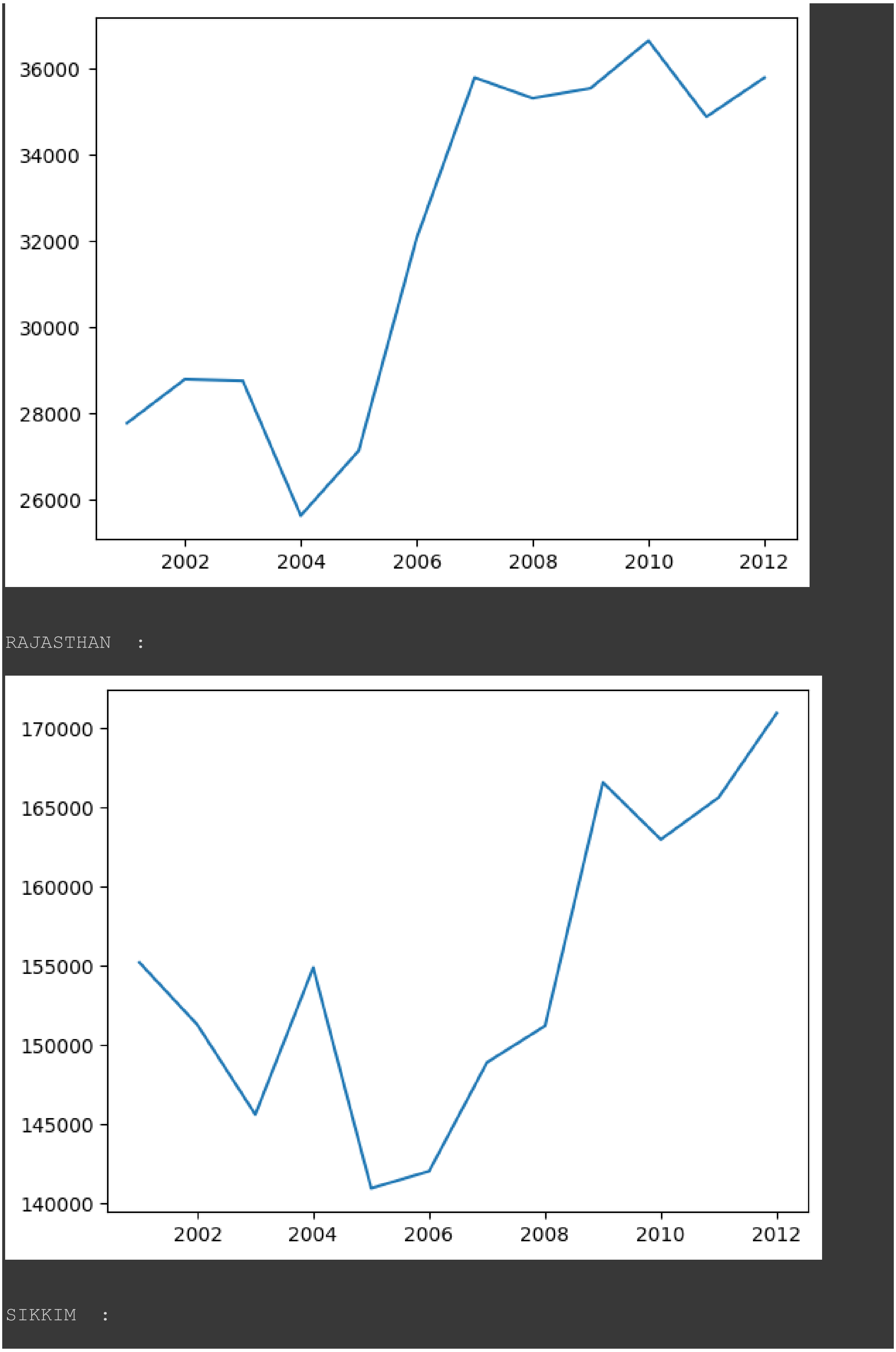
```
ODISHA   :
```



```
PUDUCHERRY  :
```

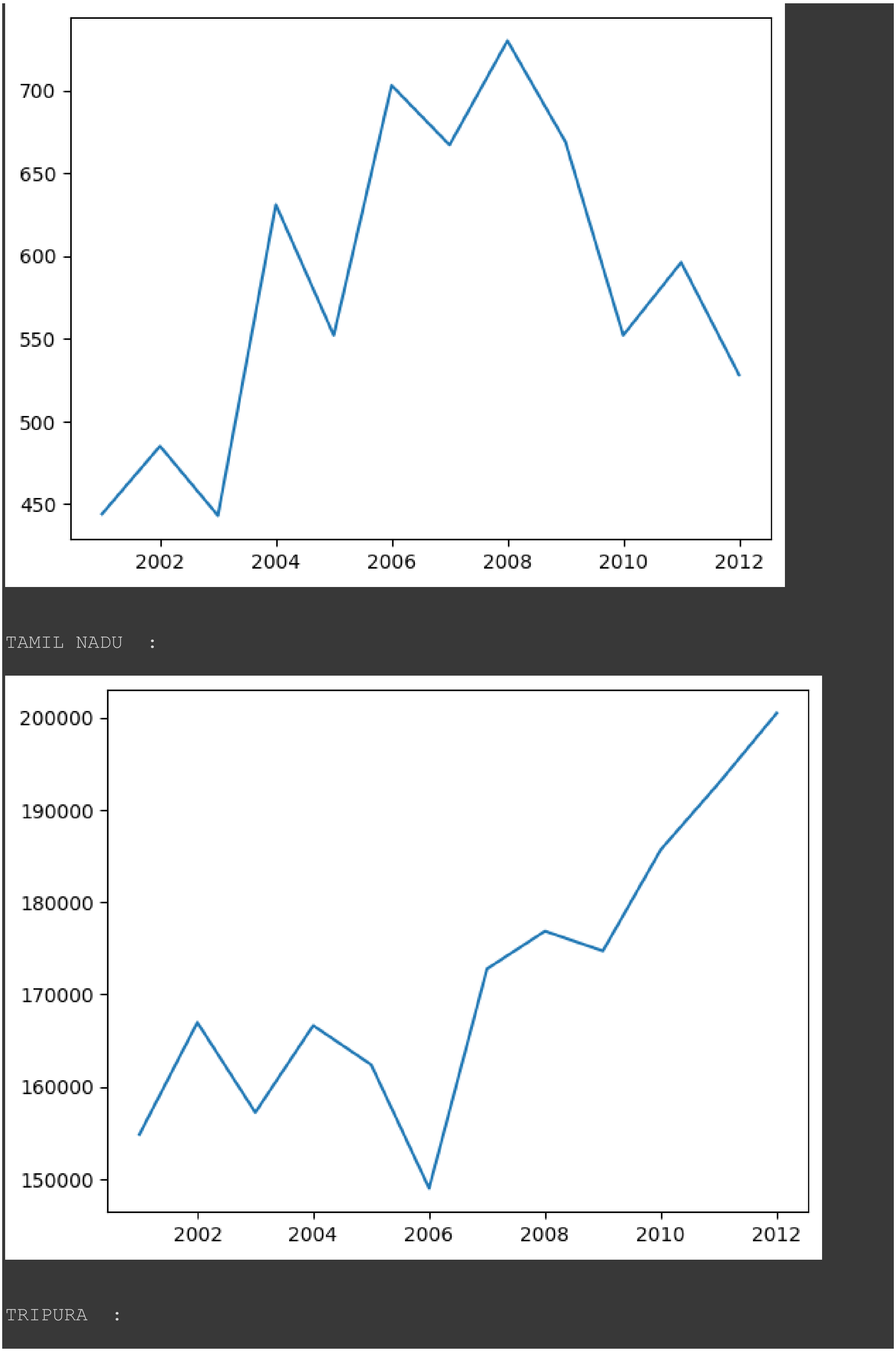


```
PUNJAB  :
```



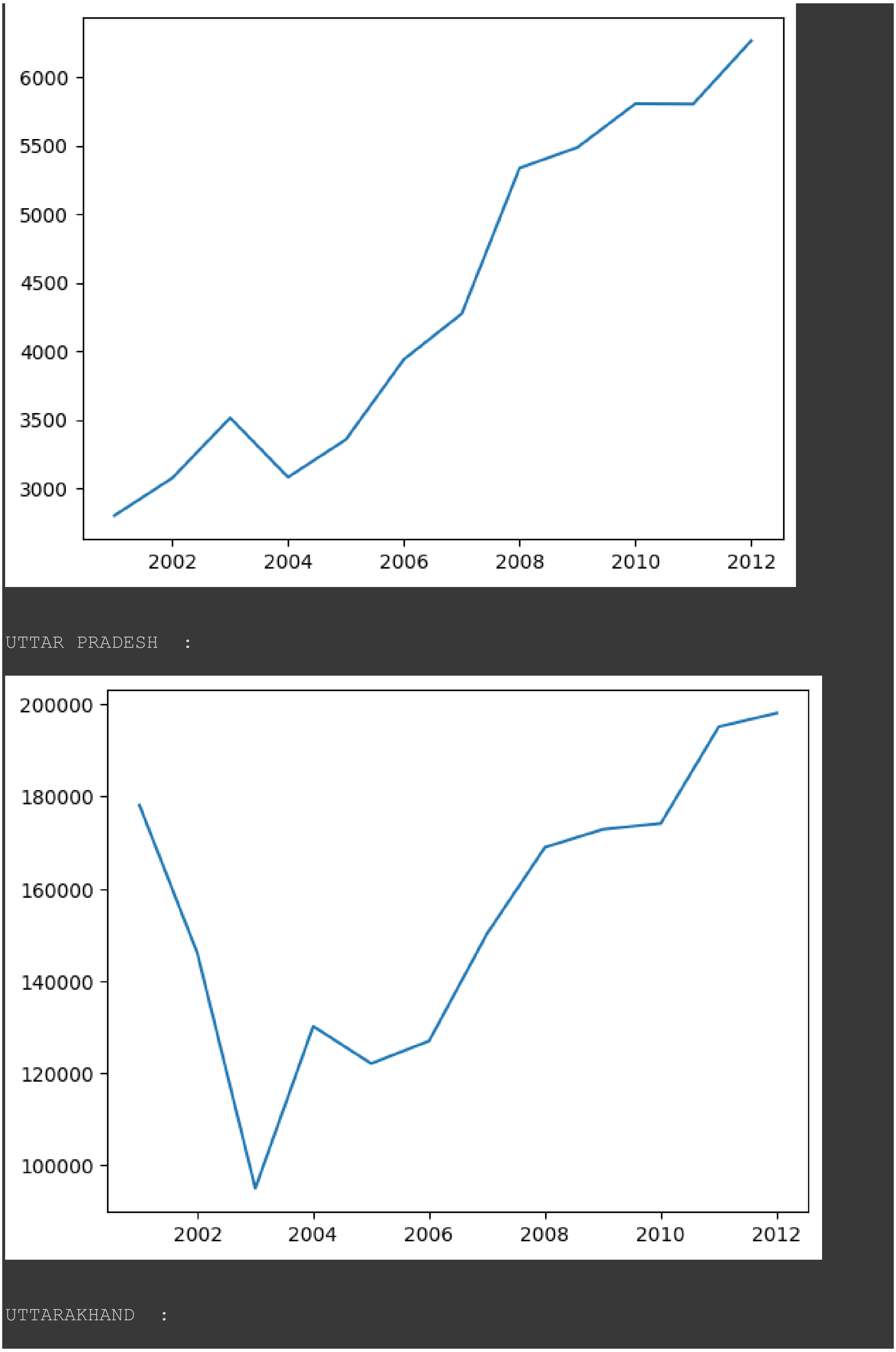
```
RAJASTHAN  :

SIKKIM  :
```



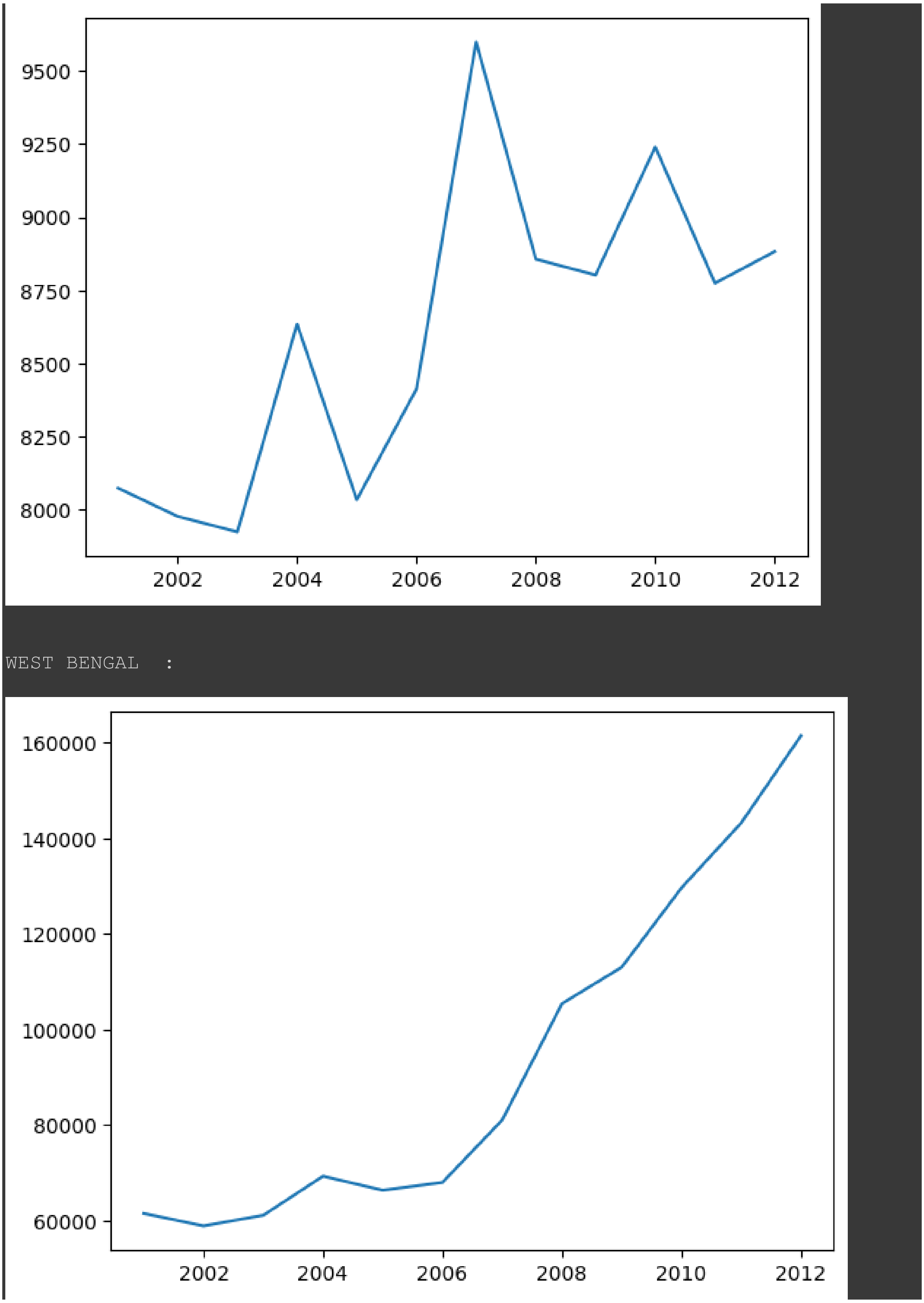
```
TAMIL NADU  :
```

```
TRIPURA  :
```



```
UTTAR PRADESH  :
```

```
UTTARAKHAND  :
```



9500
9250
9000
8750
8500
8250
8000
2002
2004
2006
2008
2010
2012

```
WEST BENGAL  :
```

160000
140000
120000
100000
80000
60000
2002
2004
2006
2008
2010
2012

5. Summary

The crime rate prediction and analysis project aimed to leverage machine learning techniques and statistical analysis to gain insights into crime patterns across different states and union territories (UTs) in India. The project involved several key stages, including data preprocessing, exploratory data analysis, clustering, predictive modeling, and model evaluation.

- Data Preprocessing:

- The crime rate dataset, containing district-wise crime data across various states/UTs spanning multiple years, was preprocessed to handle missing data and encode categorical variables.
- Missing entries were replaced with the mean value of the respective column using the SimpleImputer from Scikit-learn.
- One-hot encoding was applied to the 'STATE/UT' column using the OneHotEncoder from Scikit-learn to convert it into numerical form.

- Exploratory Data Analysis (EDA):

- Visualizations were created to explore the crime rate data:
- A bar plot showcased the total crime rate for each state/UT over 10 years.
- Another bar plot represented the rate of different types of crimes.
- A pie chart illustrated the crime rate distribution across states/UTs.

- Clustering:

- The elbow method was employed to determine the optimal number of clusters (k=2) for the crime rate data.
- K-means clustering was applied to group states/UTs based on their crime rate patterns.
- States/UTs were classified as safe or unsafe based on the clustering results.

- Predictive Modeling:

- Various machine learning models were built and trained for crime rate prediction:
- Artificial Neural Network (ANN) using TensorFlow's Keras API
- Support Vector Machine (SVM) with linear and RBF kernels
- K-Nearest Neighbors (K-NN)
- Logistic Regression
- Random Forest Regression
- Decision Tree Regression
- Support Vector Regression (SVR)

- Model Evaluation:

- For each model, predictions were made on the testing set.
- Confusion matrices and accuracy scores were calculated to assess the performance of the models.

- The results provided insights into the strengths and limitations of different models in predicting crime rates.

The project leveraged powerful data analysis and machine learning techniques to uncover meaningful patterns and relationships within the crime rate data. The findings and insights gained from this project can potentially assist law enforcement agencies and policymakers in developing effective strategies for crime prevention and resource allocation.