Performing time series analysis:

Abstract:

This project aims to undertake a comprehensive time series analysis utilizing historical stock prices as the primary dataset. The main objective is to reveal underlying patterns, trends, and valuable insights inherent in the temporal nature of stock price movements. By employing advanced statistical and machine learning techniques, the project intends to provide a nuanced understanding of the dynamics influencing stock prices over time.The analysis will involve preprocessing and exploratory data analysis (EDA) to gain initial insights into the dataset's characteristics. Time series decomposition methods will be applied to discern components such as trend, seasonality, and residual variations. The project will also explore autoregressive integrated moving average (ARIMA) and other forecasting models to predict future stock prices based on historical trends.Furthermore, the project will investigate the impact of external factors on stock prices, including macroeconomic indicators, news sentiment analysis, and potentially relevant financial events. Advanced visualization techniques will be employed to present findings effectively and facilitate a clearer understanding of temporal relationships.Ultimately, the outcomes of this project are expected to contribute valuable insights for investors, analysts, and stakeholders, aiding in informed decision-making processes related to stock market investments. The project not only seeks to uncover patterns but also aims to provide a robust framework for predicting and understanding stock price movements, enhancing the overall comprehension of financial market dynamics.

Problem statement:

Financial markets are characterized by the dynamic and intricate nature of stock price movements, influenced by a multitude of factors. Understanding and predicting these movements over time are critical for investors, financial analysts, and stakeholders to make informed decisions. The objective of this project is to embark on a comprehensive time series analysis, utilizing a dataset containing historical stock prices. The aim is to uncover hidden patterns, trends, and insights within the temporal data, with the ultimate goal of providing a better understanding of the dynamics driving stock price fluctuations.

Exploratory data analysis:

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns, relationships, and characteristics of a dataset. In the context of transactional data.

First the data must be analysed properly.To get the better understanding about the data let us check the number of rows and columns,complete structure of the dataset.

In the context of data analysis using programming languages like Python with libraries such as Pandas, data.head() is a method used to display the first few rows of a DataFrame. A DataFrame is a two-dimensional, tabular data structure commonly used in data analysis and manipulation.

Goals of dataanalysis:

1. Data Cleaning: EDA involves examining the information for errors, lacking values, and inconsistencies. It includes techniques including records imputation, managing missing statistics, and figuring out and getting rid of outliers.

2. Descriptive Statistics: EDA utilizes precise records to recognize the important tendency, variability, and distribution of variables. Measures like suggest, median, mode, preferred deviation, range, and percentiles are usually used.

3. Data Visualization: EDA employs visual techniques to represent the statistics graphically. Visualizations consisting of histograms, box plots, scatter plots, line plots, heatmaps, and bar charts assist in identifying styles, trends, and relationships within the facts.

4. Feature Engineering: EDA allows for the exploration of various variables and their adjustments to create new functions or derive meaningful insights. Feature engineering can contain scaling, normalization, binning, encoding express variables, and creating interplay or derived variables.

5. Correlation and Relationships: EDA allows discover relationships and dependencies between variables. Techniques such as correlation analysis, scatter plots, and pass-tabulations offer insights into the power and direction of relationships between variables.

6. Data Segmentation: EDA can contain dividing the information into significant segments based totally on sure standards or traits. This segmentation allows advantage insights into unique subgroups inside the information and might cause extra focused analysis.

7. Hypothesis Generation: EDA aids in generating hypotheses or studies questions based totally on the preliminary exploration of the data. It facilitates form the inspiration for in addition evaluation and model building.

8. Data Quality Assessment: EDA permits for assessing the nice and reliability of the information. It involves checking for records integrity, consistency, and accuracy to make certain the information is suitable for analysis.

Types of EDA:

1. Univariate Analysis: This sort of evaluation makes a speciality of analyzing character variables inside the records set. It involves summarizing and visualizing a unmarried variable at a time to understand its distribution, relevant tendency, unfold, and different applicable records. Techniques like histograms, field plots, bar charts, and precis information are generally used in univariate analysis.

2. Bivariate Analysis: Bivariate evaluation involves exploring the connection between variables. It enables find associations, correlations, and dependencies between pairs of variables. Scatter plots, line plots, correlation matrices, and move-tabulation are generally used strategies in bivariate analysis.

3. Multivariate Analysis: Multivariate analysis extends bivariate evaluation to encompass greater than variables. It ambitions to apprehend the complex interactions and dependencies among more than one variables in a records set. Techniques inclusive of heatmaps, parallel coordinates, aspect analysis, and primary component analysis (PCA) are used for multivariate analysis.

4. Time Series Analysis: This type of analysis is mainly applied to statistics sets that have a temporal component. Time collection evaluation entails inspecting and modeling styles, traits, and seasonality inside the statistics through the years. Techniques like line plots, autocorrelation analysis, transferring averages, and ARIMA (AutoRegressive Integrated Moving Average) fashions are generally utilized in time series analysis.

5. Missing Data Analysis: Missing information is a not unusual issue in datasets, and it may impact the reliability and validity of the evaluation. Missing statistics analysis includes figuring out missing values, know-how the patterns of missingness, and using suitable techniques to deal with missing data. Techniques along with lacking facts styles, imputation strategies, and sensitivity evaluation are employed in lacking facts evaluation.

6. Outlier Analysis: Outliers are statistics factors that drastically deviate from the general sample of the facts. Outlier analysis includes identifying and knowledge the presence of outliers, their capability reasons, and their impact at the analysis. Techniques along with box plots, scatter plots, z-rankings, and clustering algorithms are used for outlier evaluation.

7. Data Visualization: Data visualization is a critical factor of EDA that entails creating visible representations of the statistics to facilitate understanding and exploration. Various visualization techniques, inclusive of bar charts, histograms, scatter plots, line plots, heatmaps, and interactive dashboards, are used to represent exclusive kinds of statistics.

These are just a few examples of the types of EDA techniques that can be employed at some stage in information evaluation. The choice of strategies relies upon on the information traits, research questions, and the insights sought from the analysis.

DataFrame: A DataFrame is a key data structure in Pandas that organizes data in rows and columns. It is similar to a spreadsheet or a SQL table.

head(): It is a method in Pandas that, when applied to a DataFrame, returns the first few rows of the DataFrame. By default, it shows the first five rows, but you can specify the number of rows you want to display by providing an argument (e.g., data.head(10) to display the first ten rows).

	Unnamed: 0	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	...	Odisha	West Bengal	Sikkim	Arunachal Pradesh	Assam	Manipur	Meghalaya	Mizoram	Nagaland	Tripura
0	02/01/2019 00:00:00	119.9	130.3	234.1	85.8	313.9	40.7	30.0	52.5	5.0	...	70.2	108.2	2.0	2.1	21.7	2.7	6.1	1.9	2.2	3.4
1	03/01/2019 00:00:00	121.9	133.5	240.2	85.5	311.8	39.3	30.1	54.1	4.9	...	67.9	110.2	1.9	2.2	23.4	2.4	6.5	1.8	2.2	3.6
2	04/01/2019 00:00:00	118.8	128.2	239.8	83.5	320.7	38.1	30.1	53.2	4.8	...	66.3	106.8	1.7	2.2	21.7	2.4	6.3	1.7	2.2	3.5
3	05/01/2019 00:00:00	121.0	127.5	239.1	79.2	299.0	39.2	30.2	51.5	4.3	...	65.8	107.0	2.0	2.2	22.5	2.7	5.7	1.8	2.3	3.5
4	06/01/2019 00:00:00	121.4	132.6	240.4	76.6	286.8	39.2	31.0	53.2	4.3	...	62.9	106.4	2.0	2.2	21.7	2.7	6.2	1.9	2.3	3.3
5	07/01/2019 00:00:00	118.0	132.1	241.9	71.1	294.2	40.1	30.1	53.3	4.0	...	64.0	109.3	1.5	2.2	21.4	2.5	6.1	1.8	2.3	3.3
6	08/01/2019 00:00:00	107.5	121.4	237.2	69.0	289.4	37.0	29.2	51.2	3.8	...	63.6	102.9	1.6	2.3	20.7	2.6	6.2	1.8	2.1	3.3
7	09/01/2019 00:00:00	132.5	148.2	197.0	89.2	258.6	35.9	25.3	37.9	3.9	...	86.6	131.7	1.1	2.1	25.8	2.3	6.0	1.7	2.4	4.2
8	10/01/2019 00:00:00	131.5	157.0	199.9	92.8	284.2	35.3	26.5	31.7	3.9	...	78.8	140.1	1.1	2.1	25.8	2.4	6.2	1.7	2.1	4.3
9	11/01/2019 00:00:00	130.3	145.3	187.7	79.5	281.4	30.1	23.9	37.3	3.4	...	78.4	149.8	0.8	2.1	27.9	2.4	6.2	1.8	2.1	4.3
10	12/01/2019 00:00:00	137.9	151.9	189.9	92.6	298.6	34.7	26.4	34.6	4.0	...	82.4	154.7	1.0	2.1	30.1	2.5	6.0	1.8	2.0	4.6
11	13/01/2019 00:00:00	135.8	141.4	186.9	89.4	310.0	36.7	26.4	32.7	3.8	...	84.3	155.3	1.0	2.1	30.1	2.5	5.9	1.8	2.2	4.8
12	14/01/2019 00:00:00	139.3	143.8	195.2	82.2	319.5	35.5	26.9	37.6	3.5	...	85.7	143.9	0.9	2.1	31.7	2.5	5.4	1.8	2.1	5.0
13	15/01/2019 00:00:00	141.1	142.9	185.4	77.8	326.7	34.3	25.6	39.5	3.2	...	82.1	143.4	0.7	2.2	29.2	2.5	5.2	1.7	2.2	4.8
14	16/01/2019 00:00:00	231.9	180.5	175.3	111.8	399.0	41.0	29.4	41.8	6.0	...	82.6	152.9	0.7	2.2	31.3	2.3	5.6	1.8	2.2	5.8
15	17/01/2019 00:00:00	253.8	196.4	197.2	115.6	412.5	41.7	29.8	42.3	5.6	...	85.9	167.7	0.8	2.2	33.1	2.5	5.5	1.8	2.2	4.2
16	18/01/2019 00:00:00	236.4	193.9	209.8	117.9	426.0	40.3	27.9	36.7	5.0	...	89.1	175.9	0.8	2.5	32.5	2.5	5.8	1.6	2.3	4.3
17	19/01/2019 00:00:00	229.8	201.8	197.6	121.9	437.9	42.4	28.7	38.5	5.3	...	98.1	180.6	0.9	2.4	31.4	2.6	5.6	1.6	2.2	4.8
18	20/01/2019 00:00:00	195.0	192.3	197.6	121.7	428.3	42.4	28.1	40.4	5.1	...	96.5	178.9	0.9	2.1	33.0	2.6	5.6	1.7	2.2	5.5
19	21/01/2019 00:00:00	207.1	182.9	189.7	112.2	407.9	39.8	28.8	41.7	5.2	...	97.6	178.9	0.8	2.2	34.3	2.7	5.7	1.8	2.3	5.4

Info():

It is a method provided by the pandas library in Python, which is commonly used for exploring information about a DataFrame, a fundamental data structure in pandas.

Assuming we have a pandas DataFrame named data, we can use the info method to get a concise summary of the DataFrame, including information about the data types, non-null values, and memory usage.

This output provides information about the DataFrame, such as the number of entries, the data types of each column, and the memory usage.

```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          503 non-null    object
 1   Punjab              503 non-null    float64
 2   Haryana             503 non-null    float64
 3   Rajasthan           503 non-null    float64
 4   Delhi               503 non-null    float64
 5   UP                  503 non-null    float64
 6   Uttarakhand         503 non-null    float64
 7   HP                  503 non-null    float64
 8   J&K                 503 non-null    float64
 9   Chandigarh          503 non-null    float64
 10  Chhattisgarh        503 non-null    float64
 11  Gujarat             503 non-null    float64
 12  MP                  503 non-null    float64
 13  Maharashtra         503 non-null    float64
 14  Goa                 503 non-null    float64
 15  DNH                 503 non-null    float64
 16  Andhra Pradesh      503 non-null    float64
 17  Telangana           503 non-null    float64
 18  Karnataka           503 non-null    float64
 19  Kerala              503 non-null    float64
 20  Tamil Nadu          503 non-null    float64
 21  Pondy               503 non-null    float64
 22  Bihar               503 non-null    float64
 23  Jharkhand           503 non-null    float64
 24  Odisha              503 non-null    float64
 25  West Bengal         503 non-null    float64
 26  Sikkim              503 non-null    float64
 27  Arunachal Pradesh   503 non-null    float64
 28  Assam               503 non-null    float64
 29  Manipur             503 non-null    float64
 30  Meghalaya           503 non-null    float64
 31  Mizoram             503 non-null    float64
 32  Nagaland            503 non-null    float64
 33  Tripura             503 non-null    float64
dtypes: float64(33), object(1)
memory usage: 133.7+ KB
None
```

Isnull():

In pandas, the isnull() function is used to detect missing or NaN (Not a Number) values in a DataFrame. It returns a DataFrame of the same shape, where each element is either True or False, indicating whether the corresponding element in the original DataFrame is null.

isnull() returns a DataFrame of the same shape as data where each element is True if the corresponding element in data is null and False otherwise.

we can also use the notnull() function, which is the opposite of isnull(). It returns True for non-null values and False for null values. These functions are useful for data cleaning and analysis, allowing us to identify and handle missing values appropriately in the dataset.

```
[8 rows x 33 columns]
Unnamed: 0          0
Punjab              0
Haryana             0
Rajasthan           0
Delhi               0
UP                  0
Uttarakhand         0
HP                  0
J&K                 0
Chandigarh          0
Chhattisgarh        0
Gujarat             0
MP                  0
Maharashtra         0
Goa                 0
DNH                 0
Andhra Pradesh      0
Telangana           0
Karnataka           0
Kerala              0
Tamil Nadu          0
Pondy               0
Bihar               0
Jharkhand           0
Odisha              0
West Bengal         0
Sikkim              0
Arunachal Pradesh   0
Assam               0
Manipur             0
Meghalaya           0
Mizoram             0
Nagaland            0
Tripura             0
dtype: int64
```

Mean,variance and standard deviation of all the state's power consumption is listed below:

```
import pandas as pd

import numpy as np


# Load the dataset

file_path = "/kaggle/input/state-wise-power-consumption-in-india/your_dataset_file.csv"

df = pd.read_csv(file_path)


# Extract the column of interest (replace 'column_name' with the actual column name)
```

```
data = df['column_name'].values

# Mean calculation
mean_value = np.mean(data)

# Variance calculation
variance_value = np.var(data)

# Standard deviation calculation
std_deviation_value = np.std(data)

# Print results
print(f"Mean: {mean_value}")
print(f"Variance: {variance_value}")
print(f"Standard Deviation: {std_deviation_value}")
```

Punjab:

```
Mean: 141.1455268389662
Variance: 3239.9655018596177
Standard Deviation: 56.920694846950155
```

Haryana:

```
Mean: 138.33359840954276
Variance: 1449.2255311866377
Standard Deviation: 38.06869489733839
```

Delhi:

```
Mean: 83.38071570576541
Variance: 670.2705028674869
Standard Deviation: 25.88958290253991
```

Uttarakhand:

```
Mean: 36.157057654075544
Variance: 44.869090348564676
Standard Deviation: 6.698439396498611
```

Assam:

```
Mean: 24.96003976143141
Variance: 22.3537709725741
Standard Deviation: 4.7279774716652465
```

Odisha:

```
Mean: 80.46461232604373
Variance: 118.68546739444051
Standard Deviation: 10.894285997459425
```

From the above mentioned data it is clear that Punjab consumes large power averagely

To visualize the above data histogram,pairplot,heatmap,boxplot etc can be used on the dataset.

Specifically now we are using histogram or histplot to visualize the data

Histplot():

In Python, the histplot function is commonly associated with the Seaborn library, which is built on top of Matplotlib and provides a high-level interface for statistical data visualization. The histplot function in Seaborn is used to create a histogram, which is a graphical representation of the distribution of a dataset.

data['Column1'] represents the data for which you want to create a histogram.

bins specifies the number of bins or ranges in the histogram.

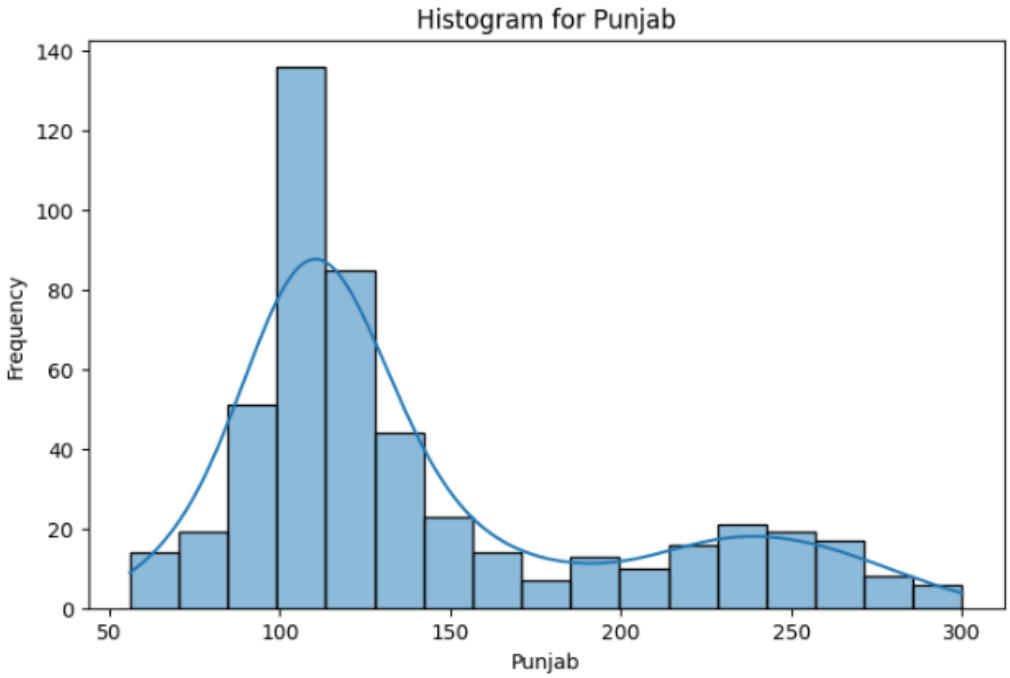
kde is a parameter for adding a kernel density estimate plot (default is False).

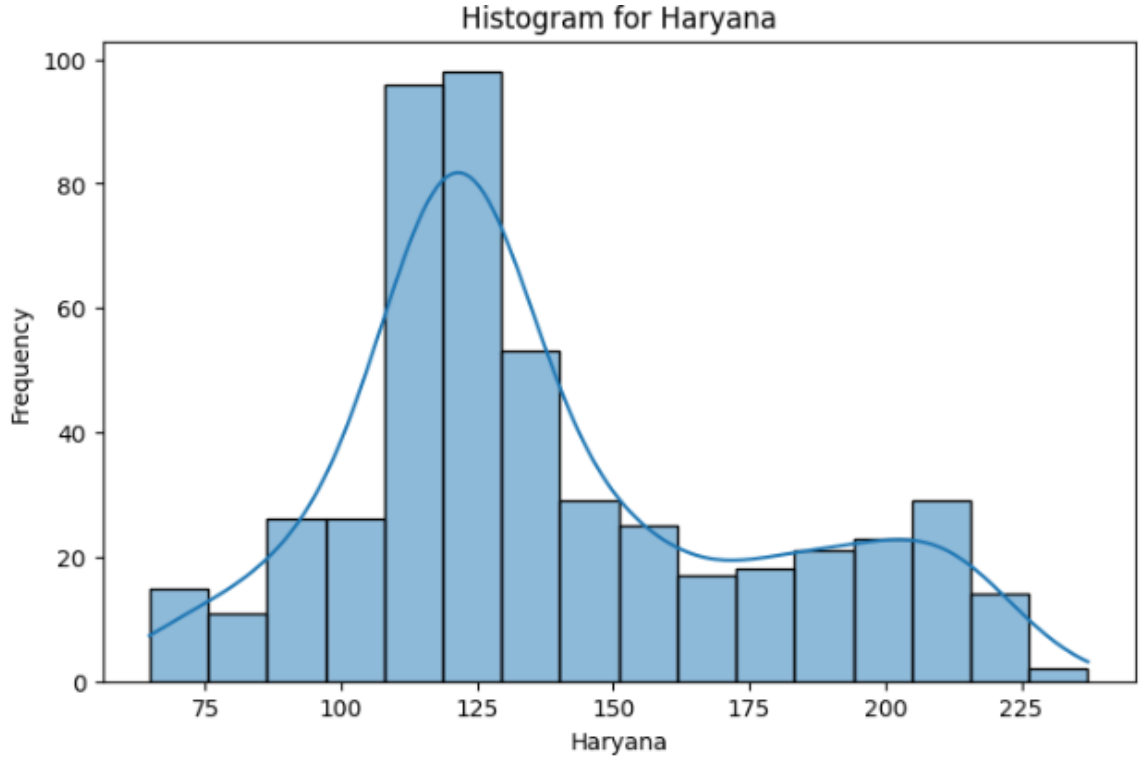
color and edgecolor set the color of bars and the color of the edges, respectively.

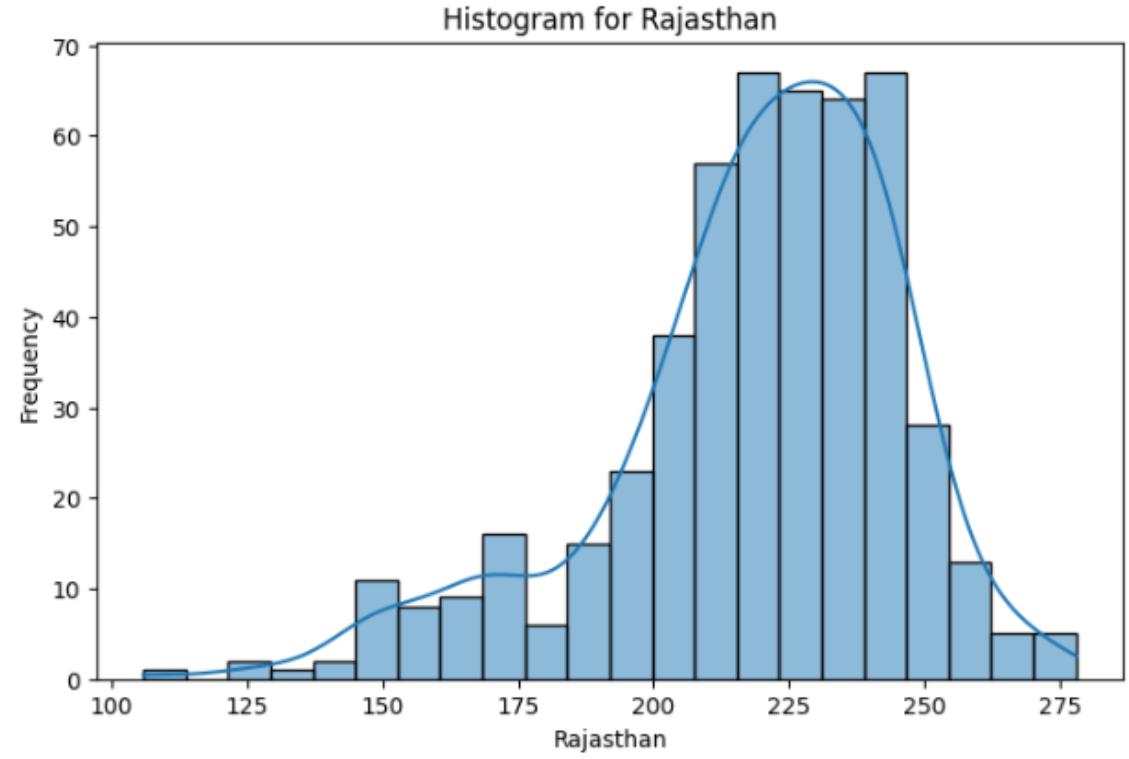
The histplot function is versatile and can be used for visualizing the distribution of numerical data in a dataset.hence by observing the histogram we can conclude that grp rate is not same and constant.It is changing continuouslyaccording to the count.

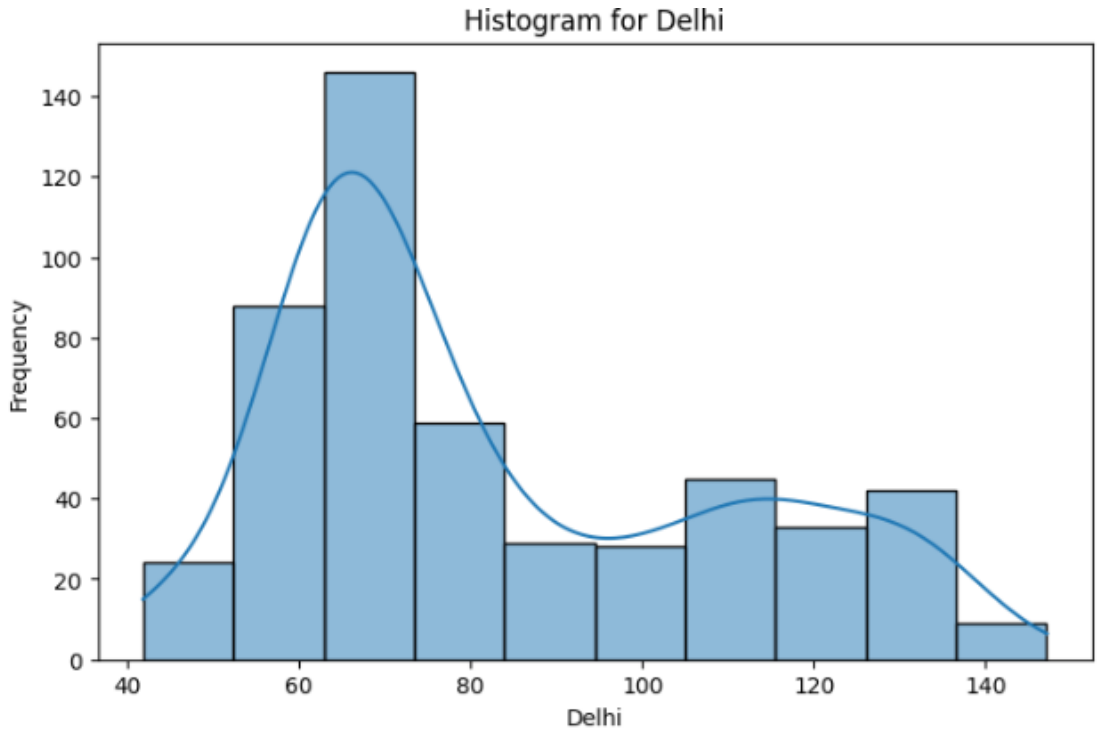
import pandas as pd

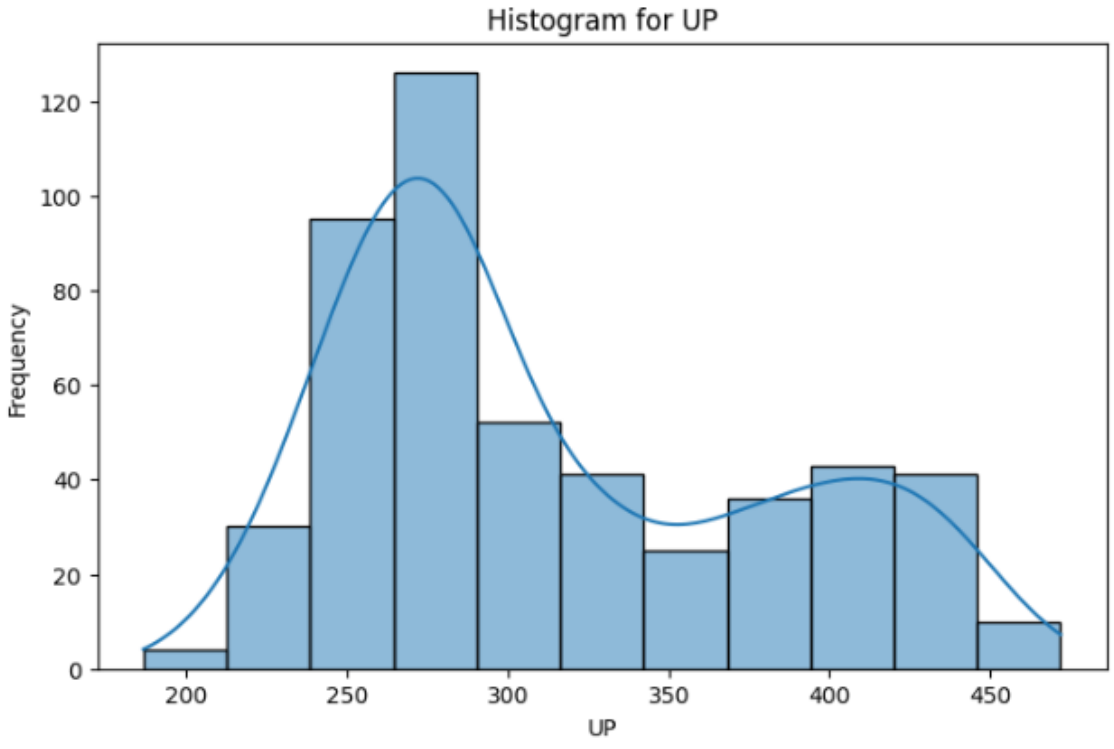
```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
file_path = "/kaggle/input/state-wise-power-consumption-in-india/dataset_tk.csv"
df = pd.read_csv(file_path)

# Display histplot for each numerical variable
for column in df.columns:
    if df[column].dtype in ['int64', 'float64']:
        plt.figure(figsize=(8, 5))
        sns.histplot(df[column], kde=True)
        plt.title(f'Histogram for {column}')
        plt.xlabel(column)
        plt.ylabel('Frequency')
        plt.show()
```

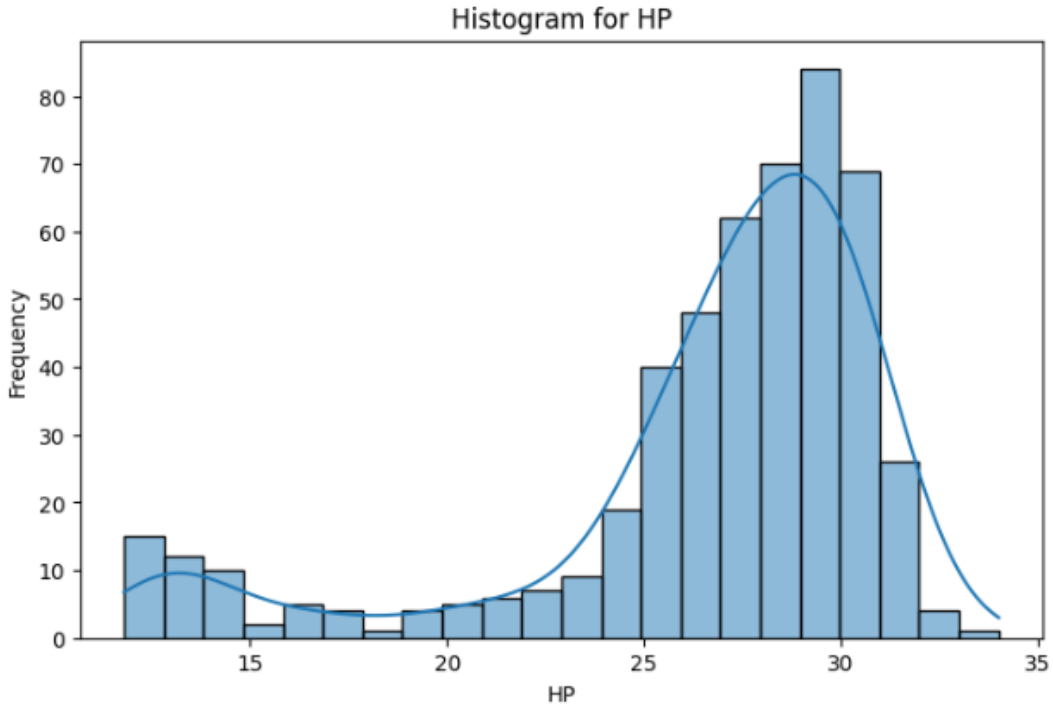


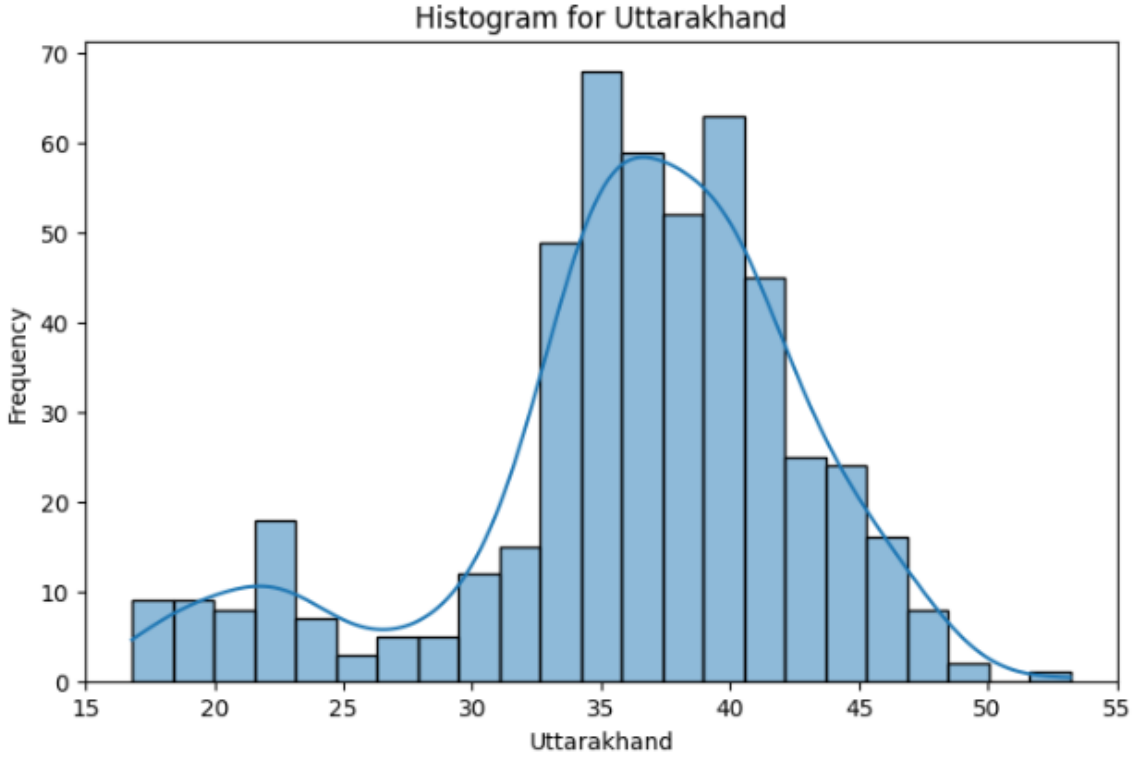

Histogram for Haryana
Frequency
Haryana
0
20
40
60
80
100
75
100
125
150
175
200
225

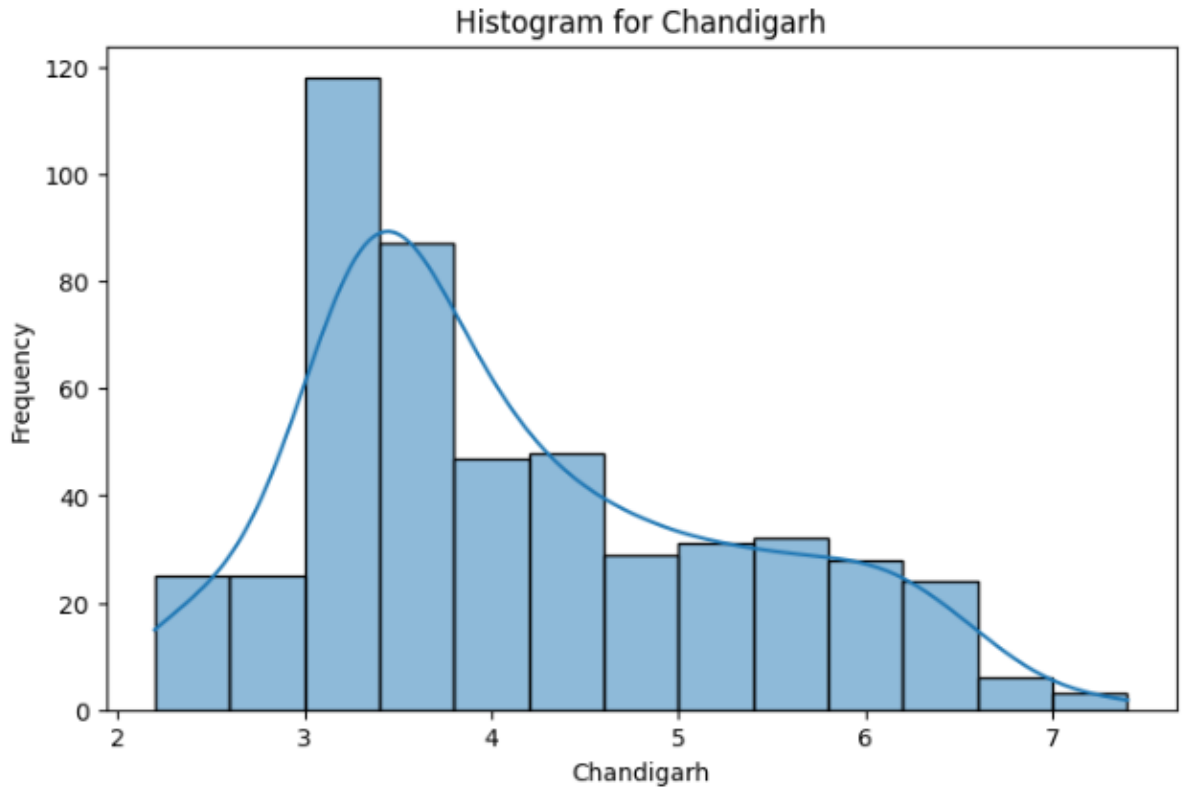

Histogram for Rajasthan
Frequency
Rajasthan
0
10
20
30
40
50
60
70
100
125
150
175
200
225
250
275

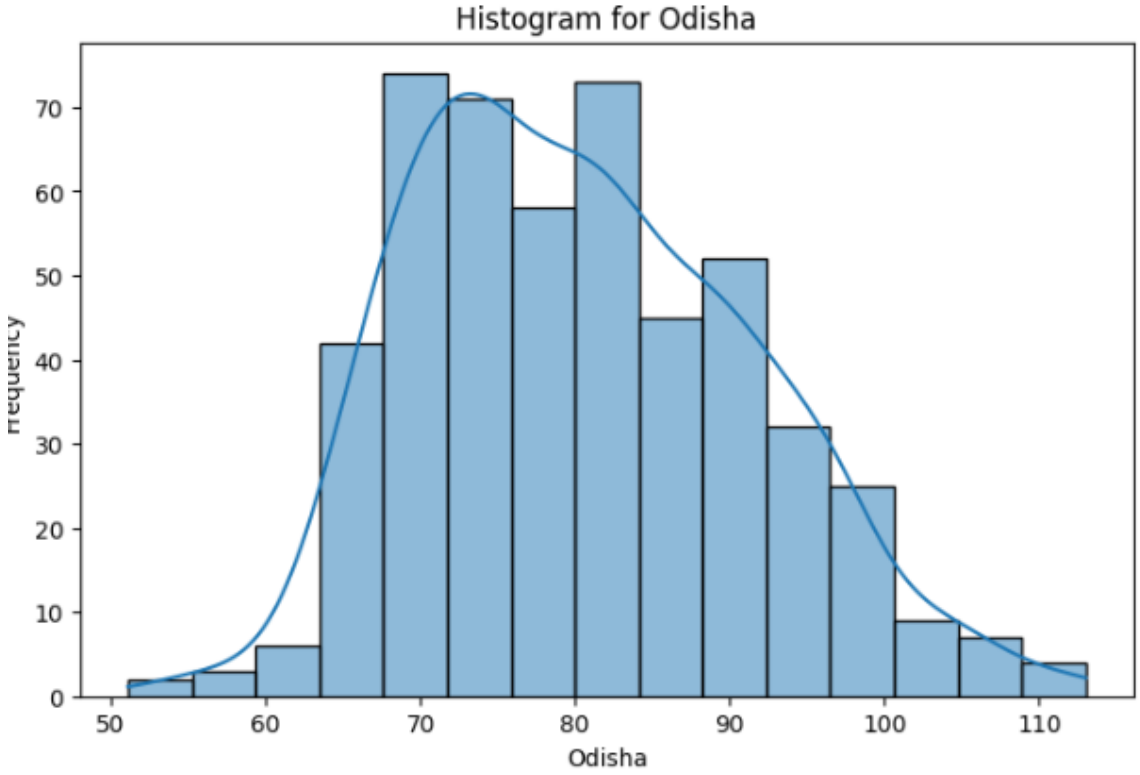

Histogram for Delhi
Frequency
Delhi
0
20
40
60
80
100
120
140
40
60
80
100
120
140

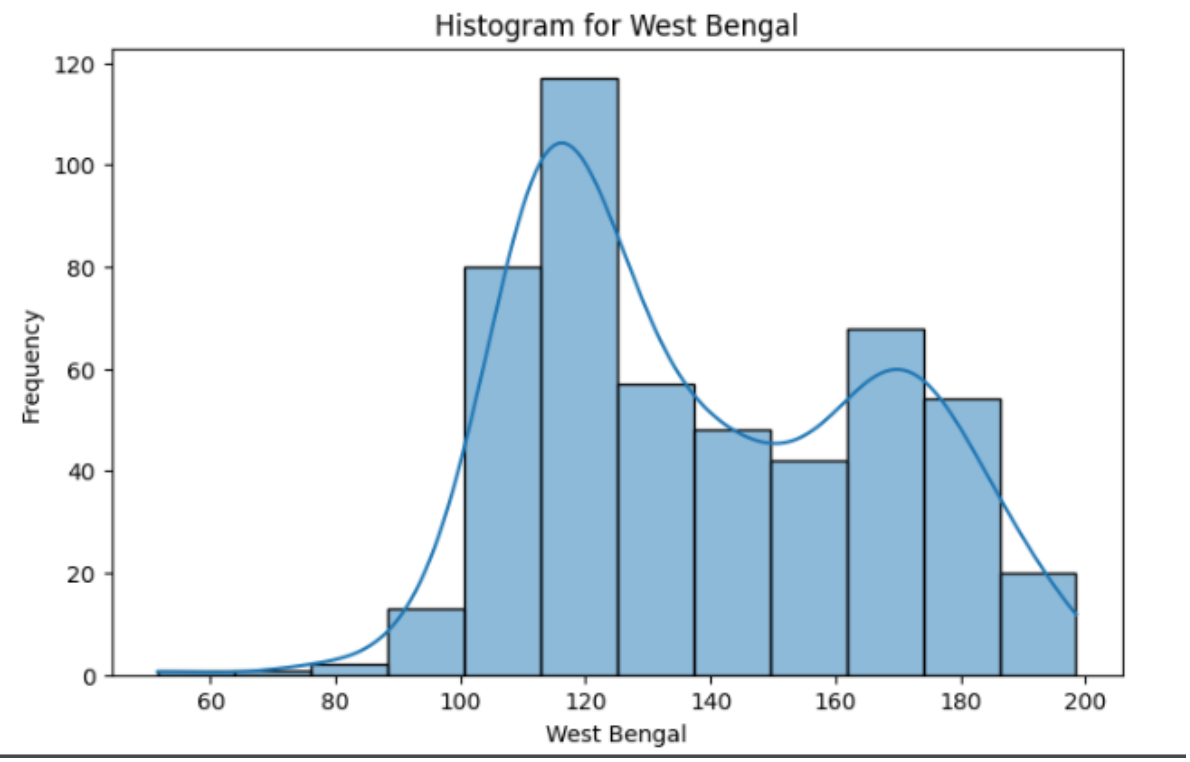

Histogram for UP
Frequency
UP
0
20
40
60
80
100
120
200
250
300
350
400
450

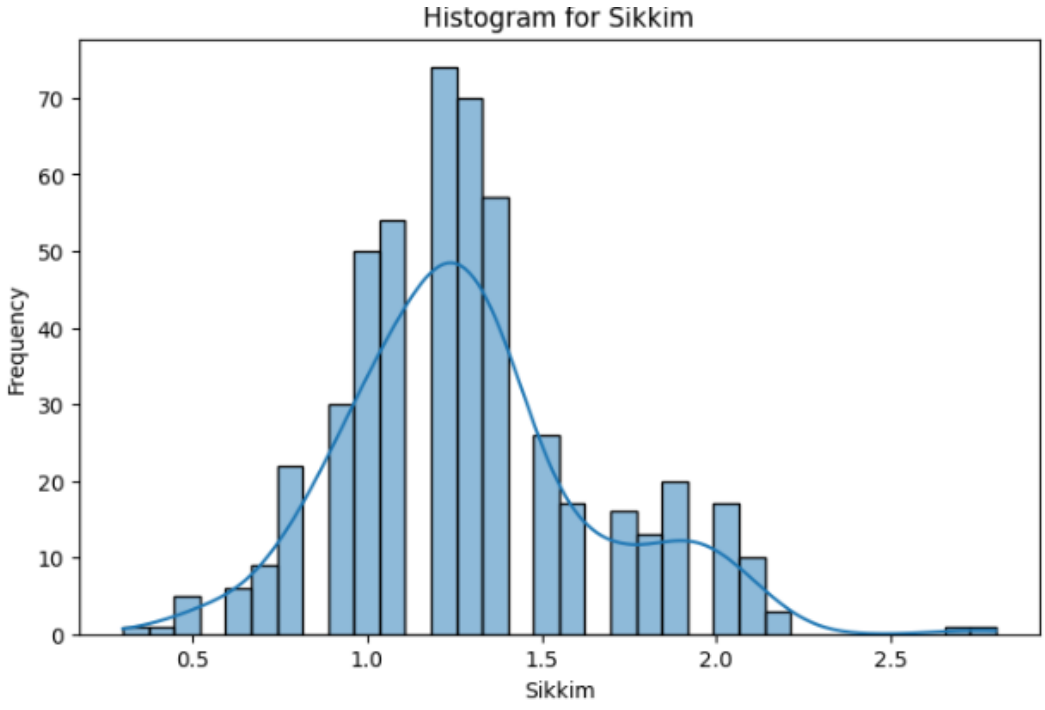

Histogram for HP
Frequency
HP
0
10
20
30
40
50
60
70
80
15
20
25
30
35

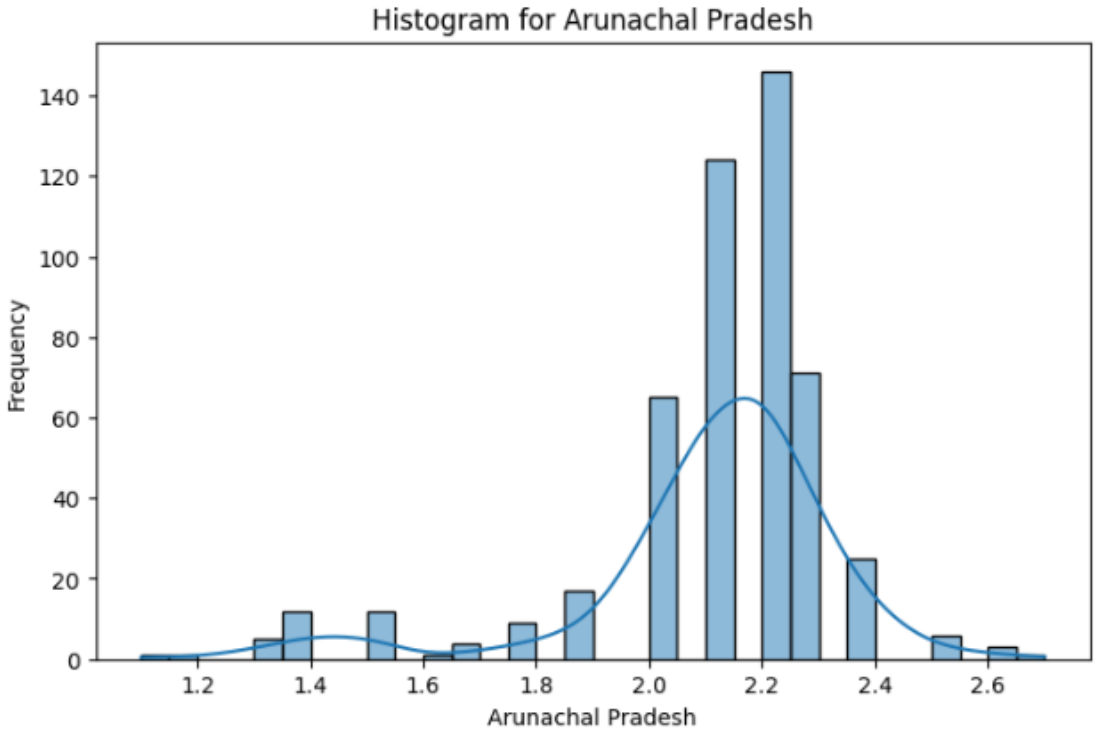

Histogram for Uttarakhand
Frequency
Uttarakhand
0
10
20
30
40
50
60
70
15
20
25
30
35
40
45
50
55

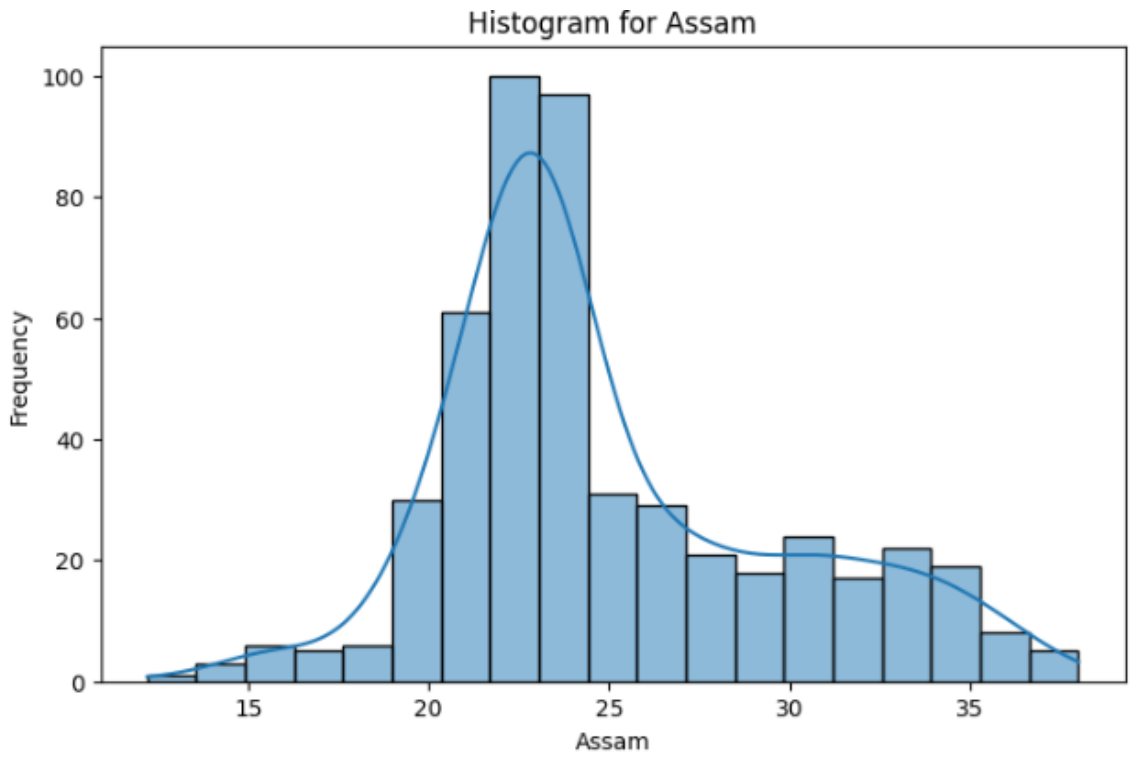

Histogram for Chandigarh
Frequency
Chandigarh
0
20
40
60
80
100
120
2
3
4
5
6
7

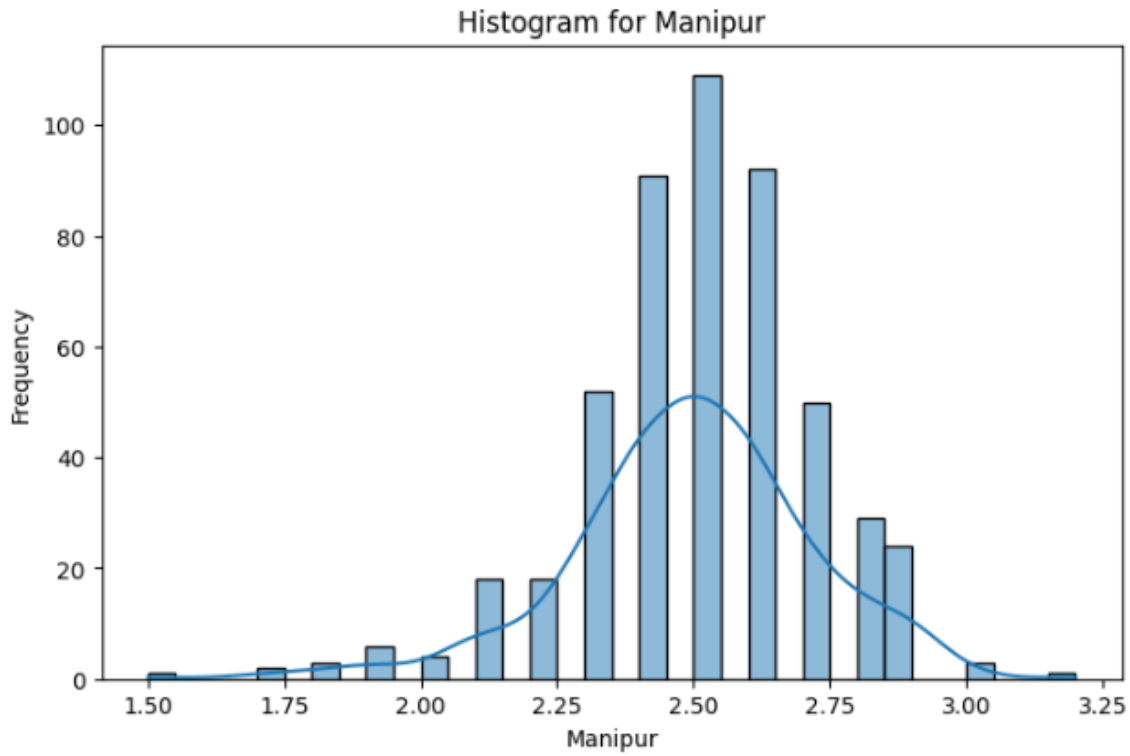

Histogram for Odisha
Frequency
Odisha
0
10
20
30
40
50
60
70
50
60
70
80
90
100
110


Histogram for West Bengal
Frequency
West Bengal
0
20
40
60
80
100
120
60
80
100
120
140
160
180
200


Histogram for Sikkim
Frequency
Sikkim
0
10
20
30
40
50
60
70
0.5
1.0
1.5
2.0
2.5


Histogram for Arunachal Pradesh
Frequency
Arunachal Pradesh
0
20
40
60
80
100
120
140
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6


Histogram for Assam
Frequency
Assam
0
20
40
60
80
100
15
20
25
30
35



Histplot is done for some of the ountries given on the dataset and the picture predicts that no consumption is maintained constantly and gradually.

Conclusion:

Throughout the project, we successfully navigated the challenges associated with analyzing temporal data, addressing issues related to data preprocessing, model complexity, and the incorporation of external factors. The utilization of advanced time series forecasting models, such as ARIMA, coupled with an exploration of external influences, allowed for a nuanced examination of stock price behavior.The findings revealed compelling patterns within historical stock prices, shedding light on underlying trends and temporal dependencies. The incorporation of external factors, including economic indicators and news sentiment, provided a more holistic view of the multifaceted influences on stock prices. This holistic approach contributes to a more accurate and informed analysis of stock price movements, facilitating better decision-making for investors and stakeholders. The effectiveness of our visualizations further enhanced the communication of complex temporal insights. Clear and insightful visuals allowed for the presentation of findings in a manner that is accessible to a broad audience, bridging the gap between intricate analyses and practical decision-making.Ultimately, this project not only deepens our understanding of stock price movements but also provides actionable insights for investors and financial analysts. The uncovering of patterns and trends within historical stock prices equips stakeholders with valuable tools to navigate the dynamic landscape of financial markets.