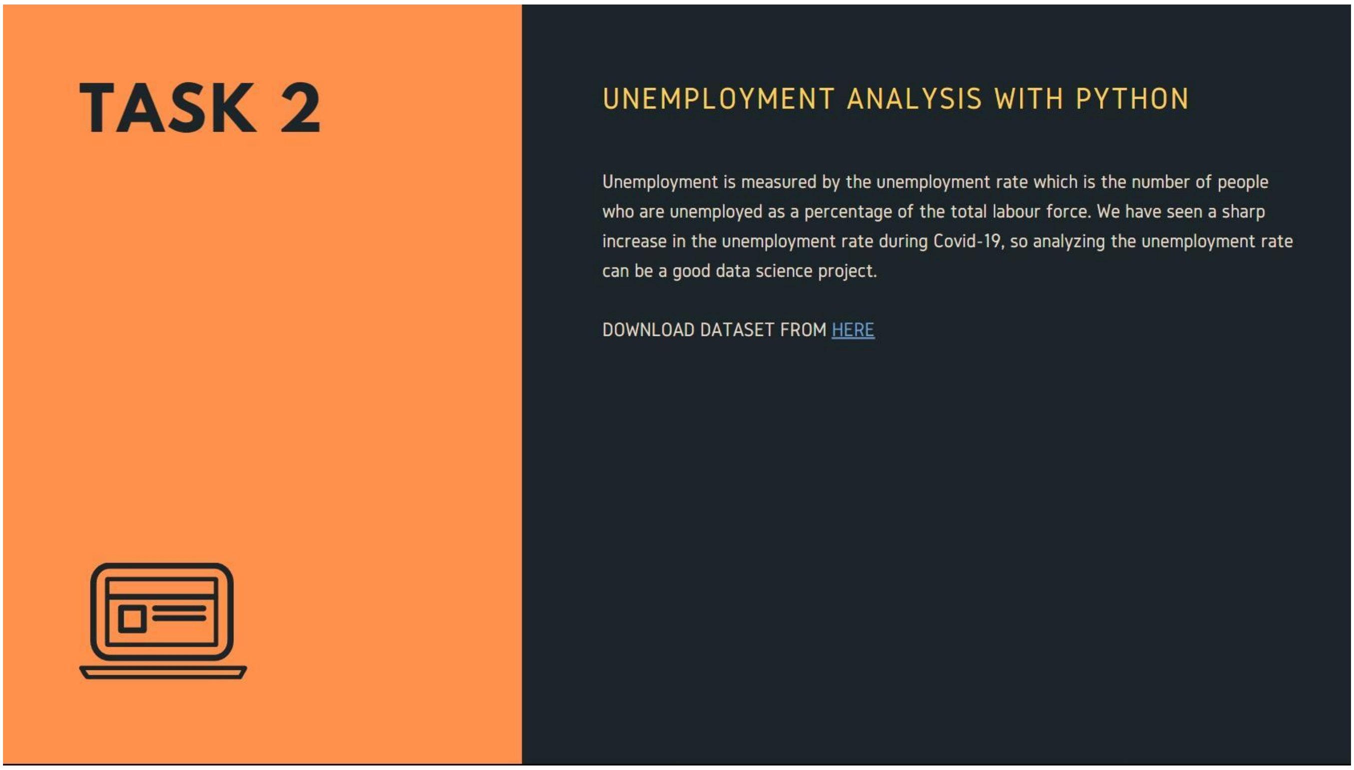


Task 2: Unemployment Analysis with Python

Introduction

The COVID-19 pandemic has had a significant impact on various aspects of the economy, with one of the most notable consequences being a surge in unemployment rates worldwide. This data science project aims to analyze and understand the patterns of unemployment during the COVID-19 pandemic. The primary focus is to explore how the unemployment rate has changed over time and identify the factors influencing this increase. This analysis is part of the Task 2 project for the Oasis Internship program.

Data Collection

To conduct this analysis, we collected unemployment rate data from reliable sources like government labor market reports, the International Labor Organization (ILO), and other official databases. Additionally, we obtained relevant socioeconomic data, such as GDP, industry-wise employment data, and COVID-19 infection rates, to explore possible relationships.

Data Preprocessing

The collected data underwent preprocessing steps, including handling missing values, data cleaning, and normalization. Time-series data were arranged chronologically to observe trends over the pandemic period. Correlation analysis was performed to identify relationships between the unemployment rate and other socioeconomic variables.

Exploratory Data Analysis (EDA)

During EDA, we created visualizations such as line plots and bar charts to visualize the changes in the unemployment rate over time. Furthermore, we examined the patterns in unemployment across various industries and demographic groups. EDA also helped us understand the impact of government interventions, such as stimulus packages and lockdown measures, on the employment scenario.

Model Building

As the project aimed to predict unemployment rates, we employed time series analysis to build a forecasting model. Techniques like ARIMA (AutoRegressive Integrated Moving Average) and exponential smoothing were considered and evaluated to identify the most suitable model for predicting future unemployment trends.

Model Evaluation

To assess the forecasting model's accuracy, we split the data into training and testing sets and used metrics such as mean squared error (MSE) and mean absolute error (MAE). Additionally, we used cross-validation techniques to validate the model's performance and ensure its robustness.

Conclusion

The data science project on analyzing unemployment rates during the COVID-19 pandemic provided valuable insights into the economic impact of the global crisis. Through time series analysis and predictive modeling, we could identify the major factors contributing to the rise in unemployment rates. Understanding these trends and relationships can aid policymakers and governments in formulating effective measures to mitigate unemployment challenges during similar crises in the future. This project also demonstrates the significance of data science in addressing real-world economic issues and generating data-driven insights for decision-making.

GitHub Repository: https://github.com/sahilkarande/OIBSIP/tree/main/Task%202%20-%20Unemployment%20Analysis%20With%20Python

Step 1: Loading the data

```
# Import Packages
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import plotly.express as px
```

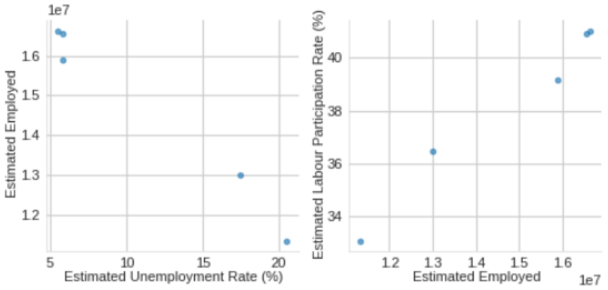
- numpy library is used to perform computational
- operations matplotlib and seaborn are used for visualization
- pandas can help us to load data from various sources
- Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data and produces easy-to-style figures.

```
#Load the data

data = pd.read_csv("/content/drive/MyDrive/Dataset files - Oasis/Unemployment in India.csv")
data = pd.read_csv("/content/drive/MyDrive/Dataset files - Oasis/Unemployment_Rate_upto_11_2020.csv")
data.head()
```

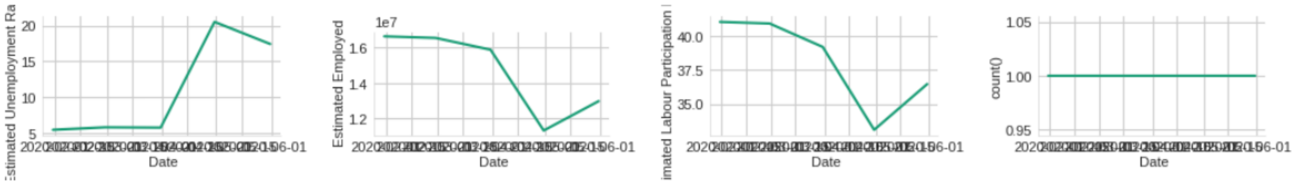
	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude	latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

Distributions

2-d distributions



Time series



- we can load the data using pd.read_csv
- function - `pd.read_csv` is used to reads csv
- files csv: comma separated values
- df.head() shows first 5 rows of the dataset

```
data.isnull().sum()
```

```
Region                                    0
 Date                                     0
 Frequency                                0
 Estimated Unemployment Rate (%)          0
 Estimated Employed                       0
 Estimated Labour Participation Rate (%)  0
Region.1                                  0
longitude                                 0
latitude                                  0
dtype: int64
```

data.isnull().sum() is used to analyze the missing columns or not

```
data.columns= ["States","Date","Frequency",
               "Estimated Unemployment Rate",
               "Estimated Employed",
               "Estimated Labour Participation Rate",
               "Region","longitude","latitude"]
```

While analyzing the data we found that the names of the columns is not properly assigned, so we rename the column names with the help of data.column

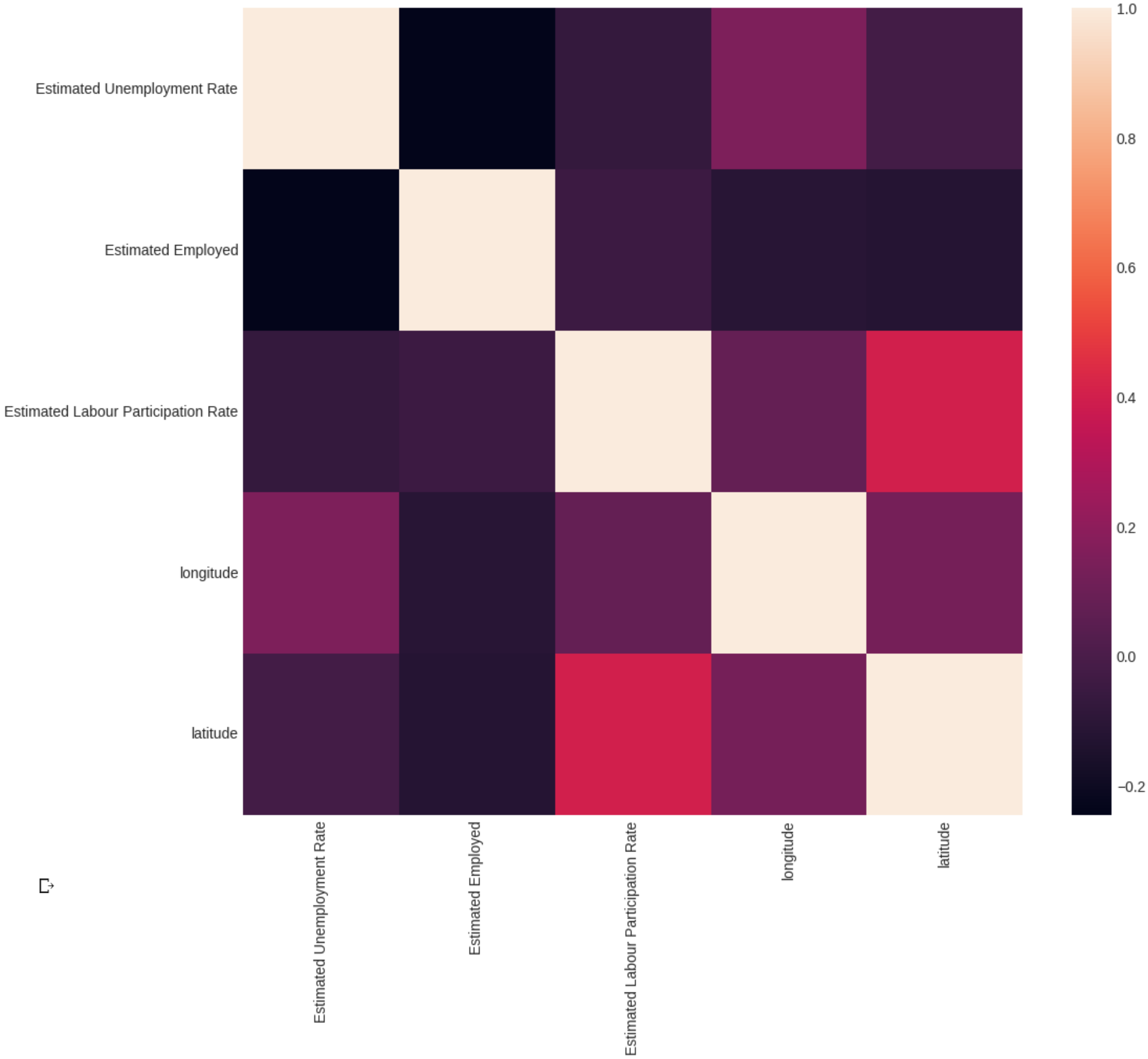
```
data.head()
```

	States	Date	Frequency	Estimated Unemployment Rate	Estimated Employed	Estimated Labour Participation Rate	Region	longitude	latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

Step 2: Correlation

```
# Correlation

plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr()) plt.show()
```

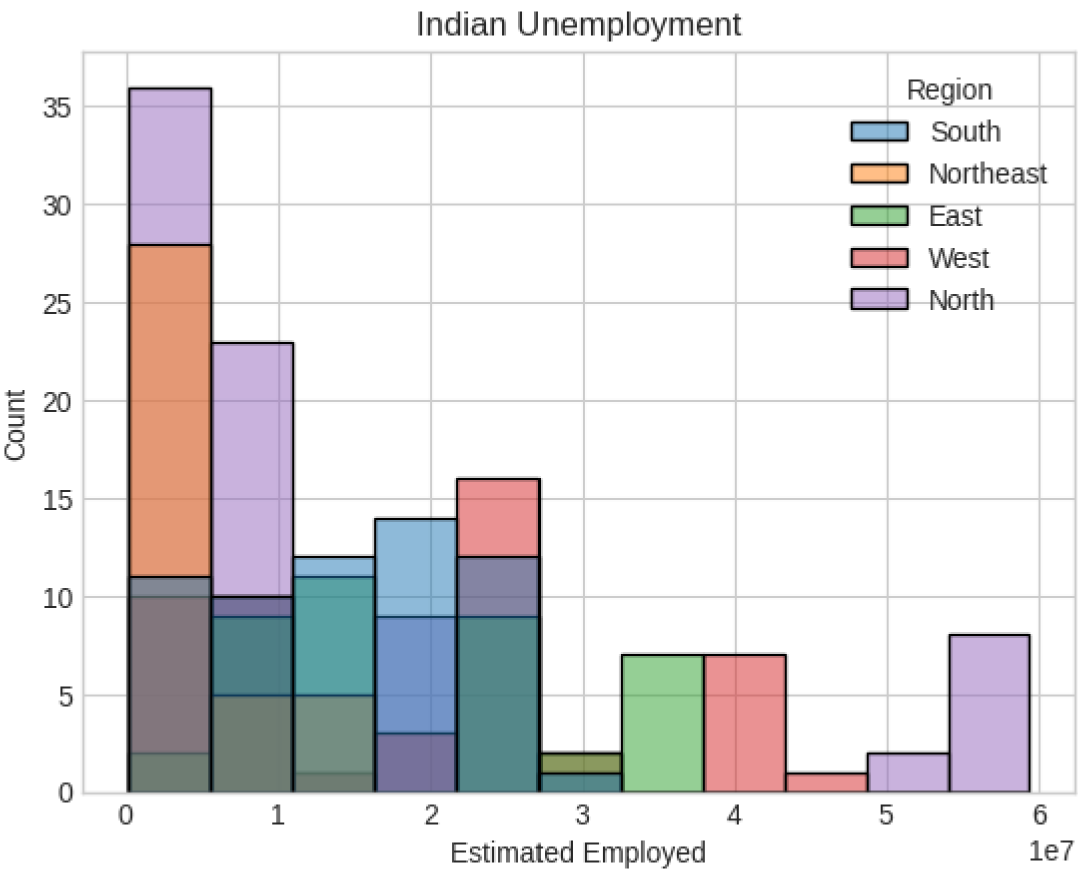
```
<ipython-input-6-55ee459967af>:3: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are depr
  plt.style.use('seaborn-whitegrid')
<ipython-input-6-55ee459967af>:5: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecate
  sns.heatmap(data.corr())
```



- Now we will find the coorelation between the features of the dataset.
- `plt.style.use('seaborn-**whitegrid**')` : Sets the plot style to 'seaborn-whitegrid' for a cleaner look with gridlines.
- `plt.figure(figsize=(12, 10))` : Creates a new plot figure with dimensions 12 inches (width) by 10 inches (height).
- `sns.heatmap(data.corr())` : Plots a heatmap to visualize the correlation matrix of the 'data' DataFrame.
- `plt.show()` : Displays the generated plot on the screen.

Step 3: Data Visualization

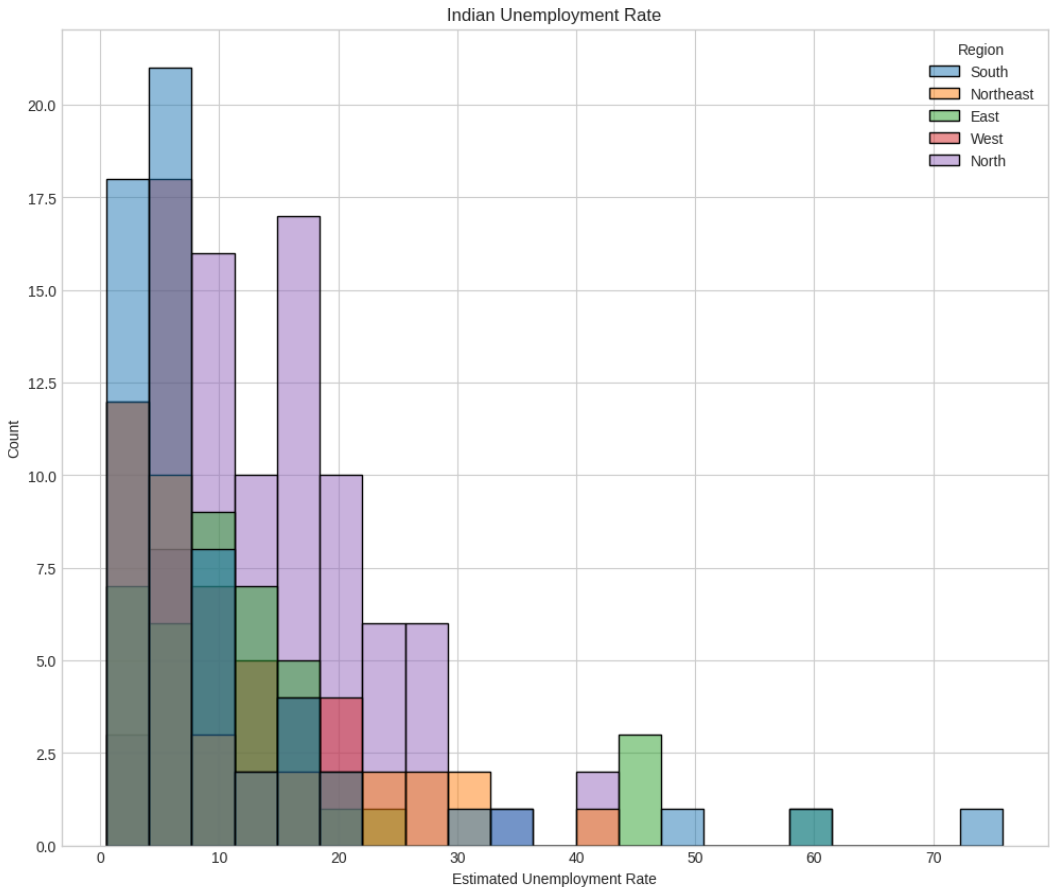
```
# Assigning column names to the dataset
data.columns = ["States", "Date", "Frequency", "Estimated Unemployment Rate",
                "Estimated Employed", "Estimated Labour Participation Rate",
                "Region", "longitude", "latitude"]

# Setting the title for the plot
plt.title("Indian Unemployment")

# Creating a histogram using Seaborn, with x-axis representing "Estimated Employed" data and using "Region" for coloring.
sns.histplot(x="Estimated Employed", hue="Region", data=data)

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



- To visualize the data of estimated number of Employees according to the regions

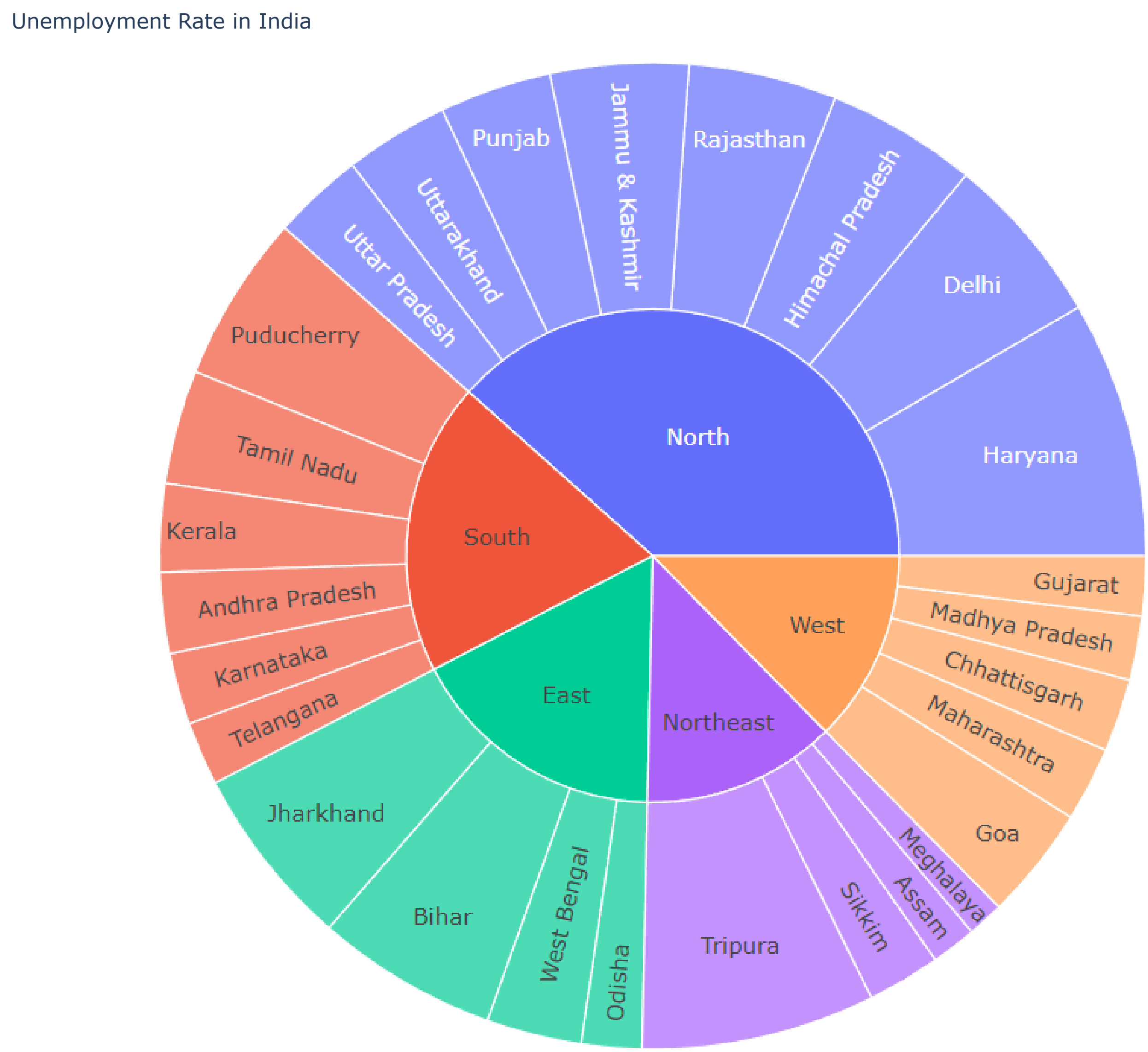
- `data.columns` : Assigning column names to the dataset
- `plt.title` : Setting the title for the plot
- `sns.histplot(x="Estimated Employed", hue="Region", data=data)` : Creating a histogram using Seaborn, with x-axis representing "Estimated Employed" data and using "Region" for coloring.
- `plt.show()` : Displaying the plot plt.show()

```
plt.figure(figsize=(12 ,10))
plt.title("Indian Unemployment Rate")
sns.histplot(x="Estimated Unemployment Rate", hue="Region", data=data)
plt.show()
```



Step 4: Creating a dashboard

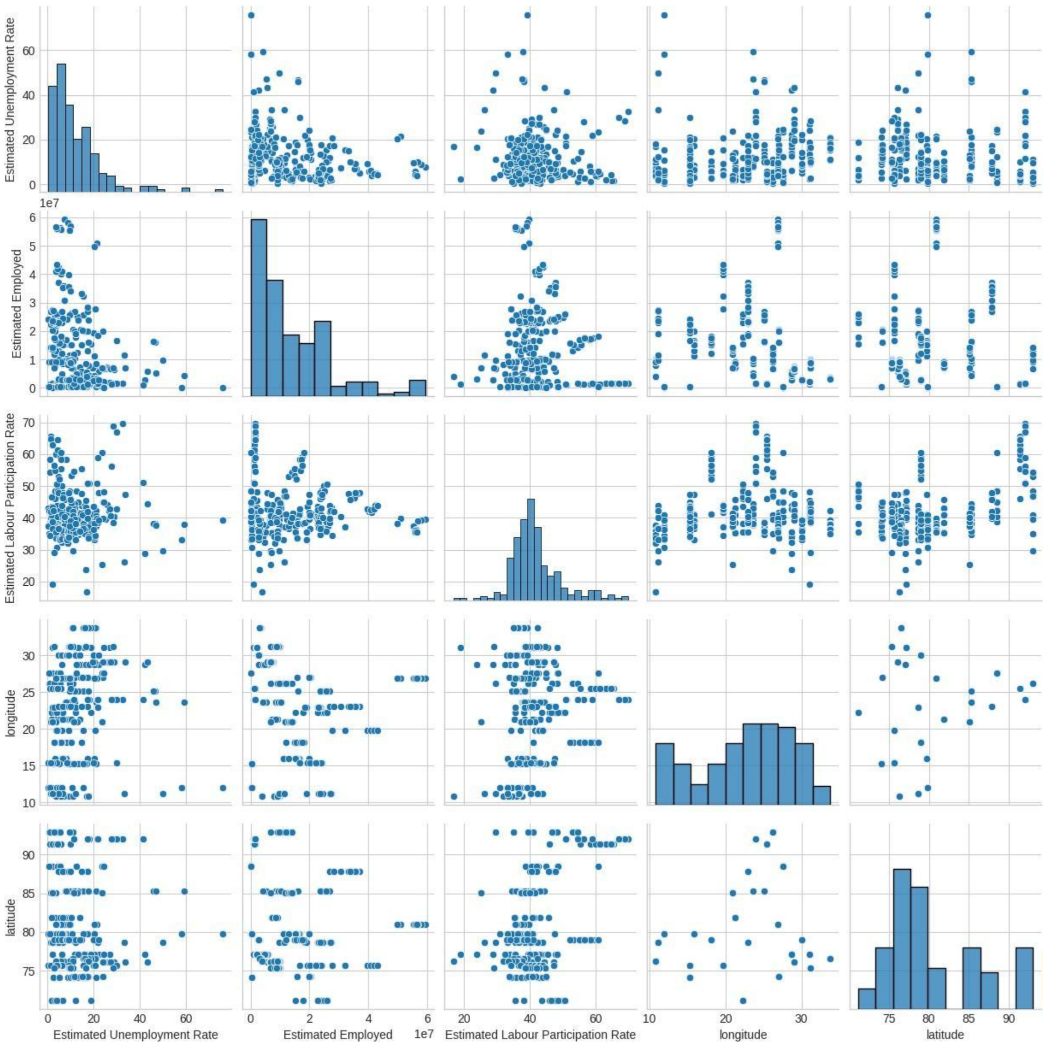
```
# To analyze the unemployement rate of each Indian State according to the region
Unemployment = data[["States", "Region", "Estimated Unemployment Rate"]]
figure = px.sunburst(Unemployment, path=["Region", "States"],
                     values="Estimated Unemployment Rate",
                     width=700, height=700, color_continuous_scale="RdY1Gn",
                     title="Unemployment Rate in India")
figure.show()
```



- A new DataFrame called Unemployment is created by selecting specific columns `("States," "Region," and "Estimated Unemployment Rate")` from the original data DataFrame.
- The **Plotly Express** `(px)` function sunburst() is used to **create a sunburst chart.**
- The hierarchical path for data representation is **set**, with regions as **the first level and states as the nested sectors.**
- The **"Estimated Unemployment Rate"** column is used as **the values for the chart**.
- The **width and height** of the chart are set to **700 pixels each**.
- The color scale for the chart is chosen to be **RdY1Gn** indicating **higher unemployment in red and lower unemployment in green.**
- The title "Unemployment Rate in India" is set for the chart.
- **The sunburst chart is displayed using figure.show().**

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7a2bf2525ea0>
```



The code `sns.pairplot(data)` generates a grid of scatter plots to visualize the relationships between numerical columns in the DataFrame data.

```
data.describe()
```

	Estimated Unemployment Rate	Estimated Employed	Estimated Labour Participation Rate	longitude	latitude
count	267.000000	2.670000e+02	267.000000	267.000000	267.000000
mean	12.236929	1.396211e+07	41.681573	22.826048	80.532425
std	10.803283	1.336632e+07	7.845419	6.270731	5.831738
min	0.500000	1.175420e+05	16.770000	10.850500	71.192400
25%	4.845000	2.838930e+06	37.265000	18.112400	76.085600
50%	9.650000	9.732417e+06	40.390000	23.610200	79.019300
75%	16.755000	2.187869e+07	44.055000	27.278400	85.279900
max	75.850000	5.943376e+07	69.690000	33.778200	92.937600

`data.describe()` calculates basic statistical measures **(e.g., mean, standard deviation, min, max)** for the numerical columns in the DataFrame data.

```
X = data[['Estimated Unemployment Rate', 'Estimated Employed', 'Estimated Labour Participation Rate',
      'longitude', 'latitude']]

y = data['Estimated Employed']
```

- X: Contains features like unemployment rate, labor participation rate, longitude, and latitude for analysis or predictions.
- y: Contains the number of employed individuals. We want to predict or understand this using the information in X using machine learning or analysis techniques.

```
from sklearn.model_selection import train_test_split
```

- `train_test_split` function, which is essential for splitting data into training and testing sets in machine learning.
- sklearn, short for scikit-learn, is a popular and widely used machine learning library in Python. It provides a vast range of tools and functionalities for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and more

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40)
```

train_test_split(X, y, test_size=0.40) will:

- It splits the datasets X and y into training and testing sets.
- The X dataset contains the features, and y dataset contains the target variable (in this case, the number of employed individuals).
- The parameter test_size=0.40 indicates that 40% of the data will be used for testing, and the remaining 60% will be used for training. The function returns four datasets: X_train, X_test, y_train, and y_test.

```
X_train
```

	Estimated Unemployment Rate	Estimated Employed	Estimated Labour Participation Rate	longitude	latitude
263	6.83	35372506	46.17	22.9868	87.8550
61	6.38	24757795	50.11	22.2587	71.1924
46	20.30	4291053	33.97	28.7041	77.1025
118	1.58	22356390	40.30	15.3173	75.7139
183	28.33	6872938	38.39	31.1471	75.3412
...	...	...	...	...	...
234	27.92	1318621	56.21	23.9408	91.9882
102	47.09	5335262	37.69	23.6102	85.2799
162	23.76	6865693	25.23	20.9517	85.0985
30	9.65	8552172	43.08	21.2787	81.8661
167	2.10	13608422	38.63	20.9517	85.0985

160 rows × 5 columns

X_train is a subset of the original features (X) used to train the machine learning model.

Step 5: Model Perform

```
from sklearn.linear_model import LinearRegression
```

- LinearRegression is a class in scikit-learn used for creating a linear regression model.
- It is commonly used for regression tasks where the goal is to predict a numerical target variable (y) based on input features (X).
- During training, the model fits a linear equation to the training data to learn the relationship between the features and the target.
- After creating an instance of LinearRegression, you can use its methods to train the model on the training data, make predictions, and evaluate its performance on new data.

```
lm = LinearRegression()
```

```
#fit the model inside it
lm.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
#evaluating model
coeff_data = pd.DataFrame(lm.coef_, X.columns, columns=['Coefficient'])
```

- The code calculates and stores the coefficients of a linear regression model in a DataFrame named `coeff_data` . These coefficients represent the **relationship between each feature and the target variable**.
- Positive coefficients indicate a positive correlation and negative coefficients indicate a negative correlation.

```
#This table is saying
#if one unit is increase then area income will increase by $21
coeff_data
```

	Coefficient
Estimated Unemployment Rate	-1.350166e-10
Estimated Employed	1.000000e+00
Estimated Labour Participation Rate	-1.304710e-10
longitude	-1.144430e-12
latitude	1.682051e-10

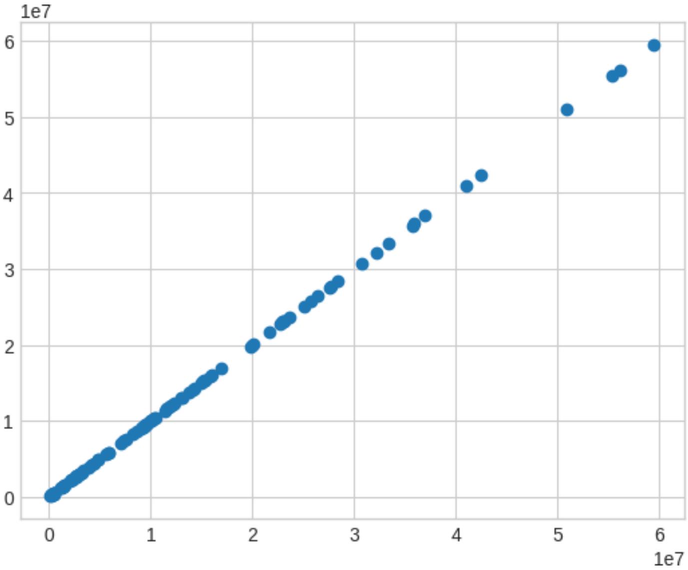
```
#Predict the model
predictions = lm.predict(X_test)
```

latitude 1.682051e-10

The code uses the trained linear regression model (lm) to predict the target variable values for the test set (X_test), and the predicted values are stored in the predictions variable.

```
#plotting the prediction agains the target variable
plt.scatter(y_test, predictions)
```

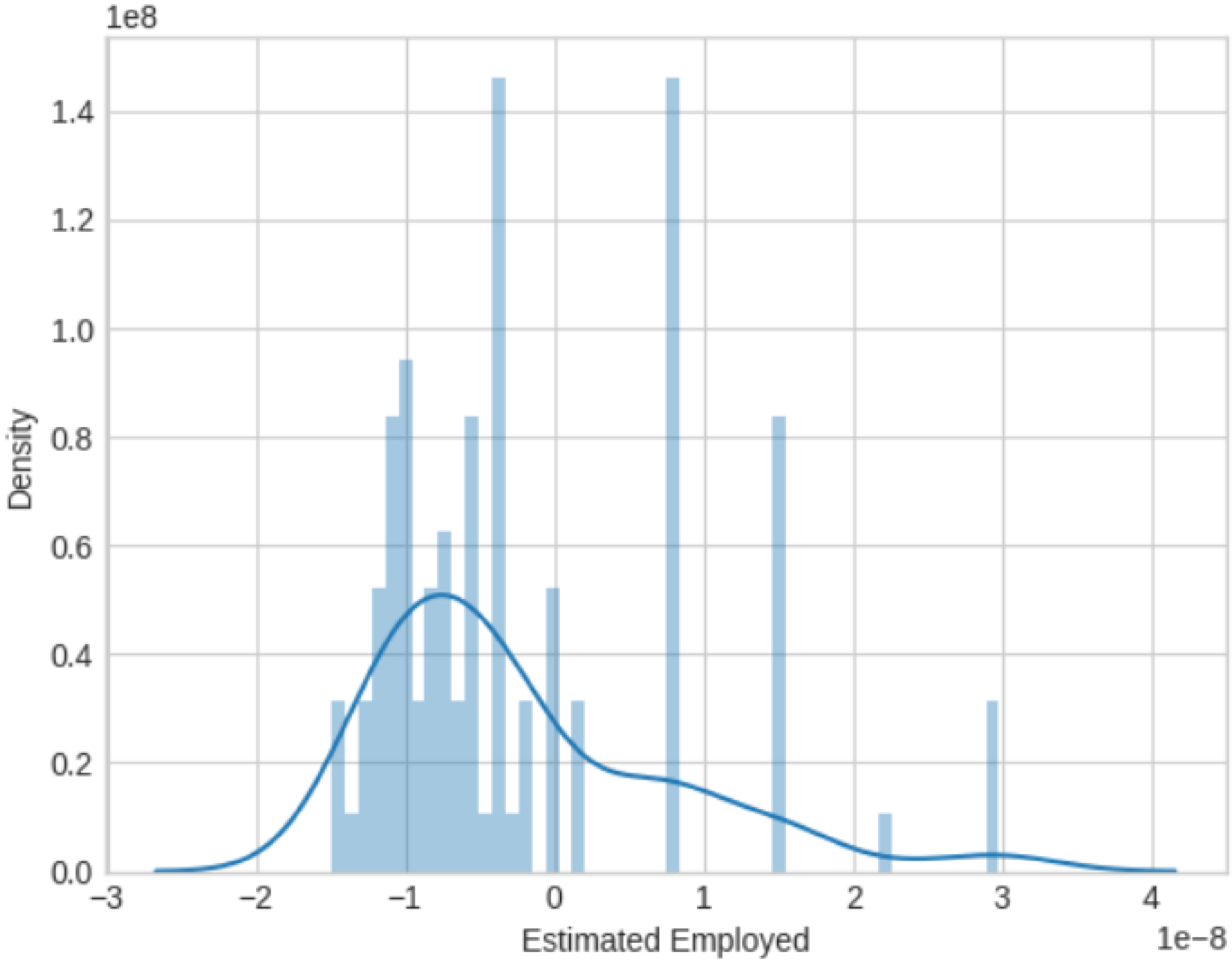
```
<matplotlib.collections.PathCollection at 0x7a2bedbe99f0>
```



The code creates a scatter plot to compare the predicted values (predictions) against the actual target variable values (y_test).

```
sns.distplot((y_test-predictions), bins=50);
```

```
<ipython-input-23-6be4f3b7dea1>:1: UserWarning:


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```



- The code uses Seaborn's sns.distplot() function to create a histogram to visualize the distribution of the differences between the actual target variable values (y_test) and the predicted values (predictions).
- The histogram is divided into 50 bins, allowing us to examine the distribution and the spread of prediction errors.

In this project, I conducted an **Unemployment analysis using Python**, applying a linear regression model to predict unemployment rates based on multiple features. Examining the coefficients gave me valuable insights into the relationships between each feature and the unemployment rate. To evaluate the model's performance, I compared the predicted unemployment rates with the actual values, visualizing the results through scatter plots and a histogram of prediction errors. For further analysis, I explored feature importance, model diagnostics, and cross-validation techniques to enhance the model's accuracy and generalization. Throughout the project, I kept in mind the context of the data and the problem at hand to draw meaningful conclusions.