unemployment-in-india

May 5, 2024

1 Unemployment Analysis in India during covid pandamic

- Unemployment is measured by the unemployment rate which is the number of people who are unemployed as a percentage of the total labour force.
- During the covid-19 period there was a sharp increase in the unemployment rate.
- The aim is to analyze the unemployment rate using python.

The dataset contains information about unemployment across different states in India during the COVID-19 pandemic. It includes:

States: Various regions within India where unemployment rates were measured. Date: Specific recording dates of unemployment rates. Measuring Frequency: How often measurements were collected (monthly). Estimated Unemployment Rate (%): The percentage of unemployed individuals in each state. Estimated Employed Individuals: The number of people currently engaged in employment. Estimated Labour Participation Rate (%): The percentage of the working-age population actively participating in the job market. This dataset is valuable for understanding how unemployment rates fluctuated throughout the pandemic across different parts of India. It provides insights into the impacts of the pandemic on employment numbers and labor force participation.

The goal of this analysis is to examine the widespread effects of the COVID-19 pandemic on India's employment landscape. By studying this dataset, we aim to gain insights into how unemployment rates, employment figures, and labor participation rates were affected during this challenging period. This analysis will shed light on the socio-economic consequences of the pandemic on India's workforce and labor market dynamics.

[1]:
```
#import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import calendar
import plotly.graph_objects as go

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Load the csv file into a pandas dataframe

```
[2]: df = pd.read_csv("Unemployment_in_india.csv")
     df.head()
```

```
[2]:            Region         Date  Frequency   Estimated Unemployment Rate (%)  \
     0  Andhra Pradesh   31-01-2020          M                              5.48
     1  Andhra Pradesh   29-02-2020          M                              5.83
     2  Andhra Pradesh   31-03-2020          M                              5.79
     3  Andhra Pradesh   30-04-2020          M                             20.51
     4  Andhra Pradesh   31-05-2020          M                             17.43

         Estimated Employed   Estimated Labour Participation Rate (%) Region.1  \
     0             16635535                                     41.02    South
     1             16545652                                     40.90    South
     2             15881197                                     39.18    South
     3             11336911                                     33.10    South
     4             12988845                                     36.46    South

        longitude  latitude
     0    15.9129     79.74
     1    15.9129     79.74
     2    15.9129     79.74
     3    15.9129     79.74
     4    15.9129     79.74
```

```
[3]: df.tail()
```

```
[3]:          Region         Date  Frequency   Estimated Unemployment Rate (%)  \
     262  West Bengal   30-06-2020          M                              7.29
     263  West Bengal   31-07-2020          M                              6.83
     264  West Bengal   31-08-2020          M                             14.87
     265  West Bengal   30-09-2020          M                              9.35
     266  West Bengal   31-10-2020          M                              9.98

           Estimated Employed   Estimated Labour Participation Rate (%) Region.1  \
     262             30726310                                     40.39     East
     263             35372506                                     46.17     East
     264             33298644                                     47.48     East
     265             35707239                                     47.73     East
     266             33962549                                     45.63     East

          longitude  latitude
     262    22.9868    87.855
     263    22.9868    87.855
     264    22.9868    87.855
     265    22.9868    87.855
     266    22.9868    87.855
```

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 267 entries, 0 to 266
Data columns (total 9 columns):
 #   Column                                    Non-Null Count  Dtype  
---  ------                                    --------------  -----  
 0   Region                                    267 non-null    object 
 1    Date                                     267 non-null    object 
 2    Frequency                                267 non-null    object 
 3    Estimated Unemployment Rate (%)          267 non-null    float64
 4    Estimated Employed                       267 non-null    int64  
 5    Estimated Labour Participation Rate (%)  267 non-null    float64
 6   Region.1                                  267 non-null    object 
 7   longitude                                 267 non-null    float64
 8   latitude                                  267 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 18.9+ KB
```

1.0.1 Renaming the attributes

1. Region = state
2. Date = date
3. Frequency = frequency
4. Estimated Unemployment Rate (%) = estimated unemployment rate
5. Estimated Employed = estimated employment
6. Estimated Labour Participation Rate (%) = estimated labour participation rate
7. Region.1 = region
8. longitude = longitude
9. latitude = latitude

Updating column names:

[5]:

```
df.columns = ['state','date','frequency','estimated unemployment␣
 ↪rate','estimated employed','estimated labour participation␣
 ↪rate','region','longitude','latitude']
df.head()
```

[5]:

```
            state         date frequency  estimated unemployment rate  \
0  Andhra Pradesh   31-01-2020         M                         5.48   
1  Andhra Pradesh   29-02-2020         M                         5.83   
2  Andhra Pradesh   31-03-2020         M                         5.79   
3  Andhra Pradesh   30-04-2020         M                        20.51   
4  Andhra Pradesh   31-05-2020         M                        17.43   

   estimated employed  estimated labour participation rate region  longitude  \
0            16635535                                41.02  South    15.9129   
1            16545652                                40.90  South    15.9129   
```

```
2            15881197                                       39.18  South    15.9129
3            11336911                                       33.10  South    15.9129
4            12988845                                       36.46  South    15.9129

   latitude
0     79.74
1     79.74
2     79.74
3     79.74
4     79.74
```

Revealing basic information of the dataset

```
[6]: df.shape
```

```
[6]: (267, 9)
```

```
[7]: df.columns
```

```
[7]: Index(['state', 'date', 'frequency', 'estimated unemployment rate',
            'estimated employed', 'estimated labour participation rate', 'region',
            'longitude', 'latitude'],
           dtype='object')
```

```
[8]: df.describe()
```

```
[8]:        estimated unemployment rate  estimated employed  \
     count                   267.000000        2.670000e+02
     mean                     12.236929        1.396211e+07
     std                      10.803283        1.336632e+07
     min                       0.500000        1.175420e+05
     25%                       4.845000        2.838930e+06
     50%                       9.650000        9.732417e+06
     75%                      16.755000        2.187869e+07
     max                      75.850000        5.943376e+07

            estimated labour participation rate   longitude    latitude
     count                           267.000000  267.000000  267.000000
     mean                             41.681573   22.826048   80.532425
     std                               7.845419    6.270731    5.831738
     min                              16.770000   10.850500   71.192400
     25%                              37.265000   18.112400   76.085600
     50%                              40.390000   23.610200   79.019300
     75%                              44.055000   27.278400   85.279900
     max                              69.690000   33.778200   92.937600
```

```
[9]: df.isnull().sum()
```

```
[9]: state                                  0
     date                                   0
     frequency                              0
     estimated unemployment rate            0
     estimated employed                     0
     estimated labour participation rate    0
     region                                 0
     longitude                              0
     latitude                               0
     dtype: int64
```

```
[10]: df.duplicated().any()
```

```
[10]: False
```

```
[11]: df.state.value_counts()
```

```
[11]: state
      Andhra Pradesh      10
      Assam               10
      Uttarakhand         10
      Uttar Pradesh       10
      Tripura             10
      Telangana           10
      Tamil Nadu          10
      Rajasthan           10
      Punjab              10
      Puducherry          10
      Odisha              10
      Meghalaya           10
      Maharashtra         10
      Madhya Pradesh      10
      Kerala              10
      Karnataka           10
      Jharkhand           10
      Himachal Pradesh    10
      Haryana             10
      Gujarat             10
      Goa                 10
      Delhi               10
      Chhattisgarh        10
      Bihar               10
      West Bengal         10
      Jammu & Kashmir      9
      Sikkim               8
      Name: count, dtype: int64
```

1.0.2 Changing the datatype of ‘date’ from object to datetime

[12]:

```
df['date'] = pd.to_datetime(df['date'],dayfirst = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 267 entries, 0 to 266
Data columns (total 9 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   state                             267 non-null    object
 1   date                              267 non-null    datetime64[ns]
 2   frequency                         267 non-null    object
 3   estimated unemployment rate       267 non-null    float64
 4   estimated employed                267 non-null    int64
 5   estimated labour participation rate  267 non-null    float64
 6   region                            267 non-null    object
 7   longitude                         267 non-null    float64
 8   latitude                          267 non-null    float64
dtypes: datetime64[ns](1), float64(4), int64(1), object(3)
memory usage: 18.9+ KB
```

1.0.3 Extracting month from date attribute

[13]:

```
df['month_int'] = df['date'].dt.month
df.head()
```

[13]:

```
            state       date frequency  estimated unemployment rate  \
0  Andhra Pradesh 2020-01-31         M                         5.48
1  Andhra Pradesh 2020-02-29         M                         5.83
2  Andhra Pradesh 2020-03-31         M                         5.79
3  Andhra Pradesh 2020-04-30         M                        20.51
4  Andhra Pradesh 2020-05-31         M                        17.43

   estimated employed  estimated labour participation rate region  longitude  \
0            16635535                                41.02  South    15.9129
1            16545652                                40.90  South    15.9129
2            15881197                                39.18  South    15.9129
3            11336911                                33.10  South    15.9129
4            12988845                                36.46  South    15.9129

   latitude  month_int
0     79.74          1
1     79.74          2
2     79.74          3
3     79.74          4
4     79.74          5
```

The months are in integer datetype. We need to convert the months into words for better analysis,

[14]:
```
df['month'] = df['month_int'].apply(lambda x: calendar.month_abbr[x])
df.head()
```

[14]:
```
             state       date frequency  estimated unemployment rate  \
0  Andhra Pradesh 2020-01-31         M                         5.48
1  Andhra Pradesh 2020-02-29         M                         5.83
2  Andhra Pradesh 2020-03-31         M                         5.79
3  Andhra Pradesh 2020-04-30         M                        20.51
4  Andhra Pradesh 2020-05-31         M                        17.43

   estimated employed  estimated labour participation rate region  longitude  \
0            16635535                                41.02  South    15.9129
1            16545652                                40.90  South    15.9129
2            15881197                                39.18  South    15.9129
3            11336911                                33.10  South    15.9129
4            12988845                                36.46  South    15.9129

   latitude  month_int month
0     79.74          1   Jan
1     79.74          2   Feb
2     79.74          3   Mar
3     79.74          4   Apr
4     79.74          5   May
```

Numeric data grouped by months

[15]:
```
data = df.groupby(['month'])[['estimated unemployment rate','estimated␣
 ↪employed','estimated labour participation rate']].mean()
data=pd.DataFrame(data).reset_index()
```

Bar plot of unemployment rate and labour participation rate

[16]:
```
month = data.month
unemployment_rate = data['estimated unemployment rate']
labour_participation_rate = data['estimated labour participation rate']

fig = go.Figure()

fig.add_trace(go.Bar(x = month,y = unemployment_rate,name = 'Unemployment␣
 ↪Rate'))
fig.add_trace(go.Bar(x = month,y = labour_participation_rate,name = 'Labour␣
 ↪Participation Rate'))

fig.update_layout(title = 'Unemployment Rate and Labour Participation',
                     xaxis = {'categoryorder':'array','categoryarray':
 ↪['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct']}      )
```

```
fig.show()
```

Bar plot of estimated employed citizen in every month

[17]:
```
import plotly.express as px
```

[18]:
```
fig = px.bar(data,x='month',y='estimated employed',color='month',
            category_orders ={'month':
 ↪['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct']},
            title='Estimated employed people from Jan 2020 to Oct 2020')
fig.show()
```

1.1 State wise Analysis

[19]:
```
state =  df.groupby(['state'])[['estimated unemployment rate','estimated␣
 ↪employed','estimated labour participation rate']].mean()
state = pd.DataFrame(state).reset_index()
```

[20]:
```
# Box plot

fig = px.box(data_frame=df,x='state',y='estimated unemployment␣
 ↪rate',color='state',title='Unemployment rate')
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```

[21]:
```
# average unemployment rate bar plot

fig = px.bar(state,x='state',y='estimated unemployment␣
 ↪rate',color='state',title='Average unemployment rate (State)')
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```

Hariyana and Tripura were having the highest average amount of Unemployment rate

Meghalaya was having the lowest average amount of Unemployment rate

[22]:
```
# Bar plot Unemployment Rate (monthly)

fig = px.bar(df,x='state',y='estimated unemployment␣
 ↪rate',animation_frame='month',color='state',
            title='Unemployment rate from Jan 2020 to Oct 2020(State)')

fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```

[23]:
```
# Filter data before and during lockdown
before_lockdown = df[df['date'] < '2020-03-25']
```

```
during_lockdown = df[df['date'] >= '2020-03-25']

# Average Unemployment Rate before and during lockdown
avg_unemployment_before = before_lockdown['estimated unemployment rate'].mean()
avg_unemployment_during = during_lockdown['estimated unemployment rate'].mean()

print(f"Average Unemployment Rate before lockdown: {avg_unemployment_before:.
 ↪2f}%")
print(f"Average Unemployment Rate during lockdown: {avg_unemployment_during:.
 ↪2f}%")

# Percentage change in Unemployment Rate
percentage_change = ((avg_unemployment_during - avg_unemployment_before) /␣
 ↪avg_unemployment_before) * 100
print(f"Percentage Change in Unemployment Rate: {percentage_change:.2f}%")
```

```
Average Unemployment Rate before lockdown: 9.23%
Average Unemployment Rate during lockdown: 12.96%
Percentage Change in Unemployment Rate: 40.43%
```

Monthly unemployment rate

[24]:
```
fig=px.scatter_geo(df,'longitude','latitude',color='state',
                  hover_name='state',size='estimated unemployment rate',
                  animation_frame='month',scope='asia',title='Impact of␣
 ↪lockdown on employment in India')

fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] =2000
fig.
 ↪update_geos(lataxis_range=[5,40],lonaxis_range=[65,100],oceancolor='lightblue',
               showocean=True)

fig.show()
```

1.2 Regional Analysis

[25]:
```
df.region.unique()
```

[25]:
```
array(['South', 'Northeast', 'East', 'West', 'North'], dtype=object)
```

[26]:
```
# numeric data grouped by region

region = df.groupby(['region'])[['estimated unemployment rate','estimated␣
 ↪employed','estimated labour participation rate']].mean()
region = pd.DataFrame(region).reset_index()
```

[27]:

```
import plotly.express as px

# Specify dimensions and color parameter for the scatter matrix plot
dimensions = ['estimated unemployment rate', 'estimated employed', 'estimated␣
 ↪labour participation rate']
color_column = 'region'

# Create scatter matrix plot with Plotly Express
fig = px.scatter_matrix(
    df,
    dimensions=dimensions,
    color=color_column,
    title='Scatter Matrix Plot Colored by Region'
)

# Display the plot
fig.show()
```

[28]:

```
import dash
import dash_core_components as dcc
import dash_html_components as html

app = dash.Dash(__name__)

# Define layout
app.layout = html.Div([
    dcc.Graph(
        id='unemployment-trend',
        figure={
            'data': [
                {'x': df['date'], 'y': df['estimated unemployment rate'],␣
 ↪'type': 'line', 'name': 'Unemployment Rate'}
            ],
            'layout': {
                'title': 'Unemployment Rate Over Time'
            }
        }
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

```
<IPython.lib.display.IFrame at 0x1686911ac90>
```

[29]:

```
# Average Unemployment Rate
```

```
fig = px.bar(region,x='region',y='estimated unemployment␣
 ↪rate',color='region',title='Average unemployment rate(region)')
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```

[30]:
```
fig = px.bar(df,x='region',y='estimated unemployment␣
 ↪rate',animation_frame='month',color='state',
            title='Unemployment rate from Jan 2020 to Oct 2020')

fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] =2000

fig.show()
```

[31]:
```
unemployment =df.groupby(['region','state'])['estimated unemployment rate'].
 ↪mean().reset_index()
unemployment.head()
```

[31]:
```
  region        state  estimated unemployment rate
0   East        Bihar                       19.471
1   East    Jharkhand                       19.539
2   East       Odisha                        6.462
3   East  West Bengal                       10.192
4  North        Delhi                       18.414
```

[32]:
```
fig = px.sunburst(unemployment,path=['region','state'],values='estimated␣
 ↪unemployment rate',
                 title ='Unemployment rate in state and region',height=600)
fig.show()
```

1.3 Unemployment rate before and after Lockdown

[33]:
```
# data representation before and after lockdown

before_lockdown = df[(df['month_int']>=1) &(df['month_int'] <4)]
after_lockdown = df[(df['month_int'] >=4) & (df['month_int'] <=6)]
```

[34]:
```
af_lockdown = after_lockdown.groupby('state')['estimated unemployment rate'].
 ↪mean().reset_index()

lockdown = before_lockdown.groupby('state')['estimated unemployment rate'].
 ↪mean().reset_index()
lockdown['unemployment rate before lockdown'] = af_lockdown['estimated␣
 ↪unemployment rate']
```

```
lockdown.columns = ['state','unemployment rate before lockdown','unemployment␣
 ↪rate after lockdown']
lockdown.head()
```

```
[34]:             state  unemployment rate before lockdown  \
      0  Andhra Pradesh                           5.700000
      1           Assam                           4.613333
      2           Bihar                          12.110000
      3    Chhattisgarh                           8.523333
      4           Delhi                          18.036667

         unemployment rate after lockdown
      0                         13.750000
      1                          7.070000
      2                         36.806667
      3                          9.380000
      4                         25.713333
```

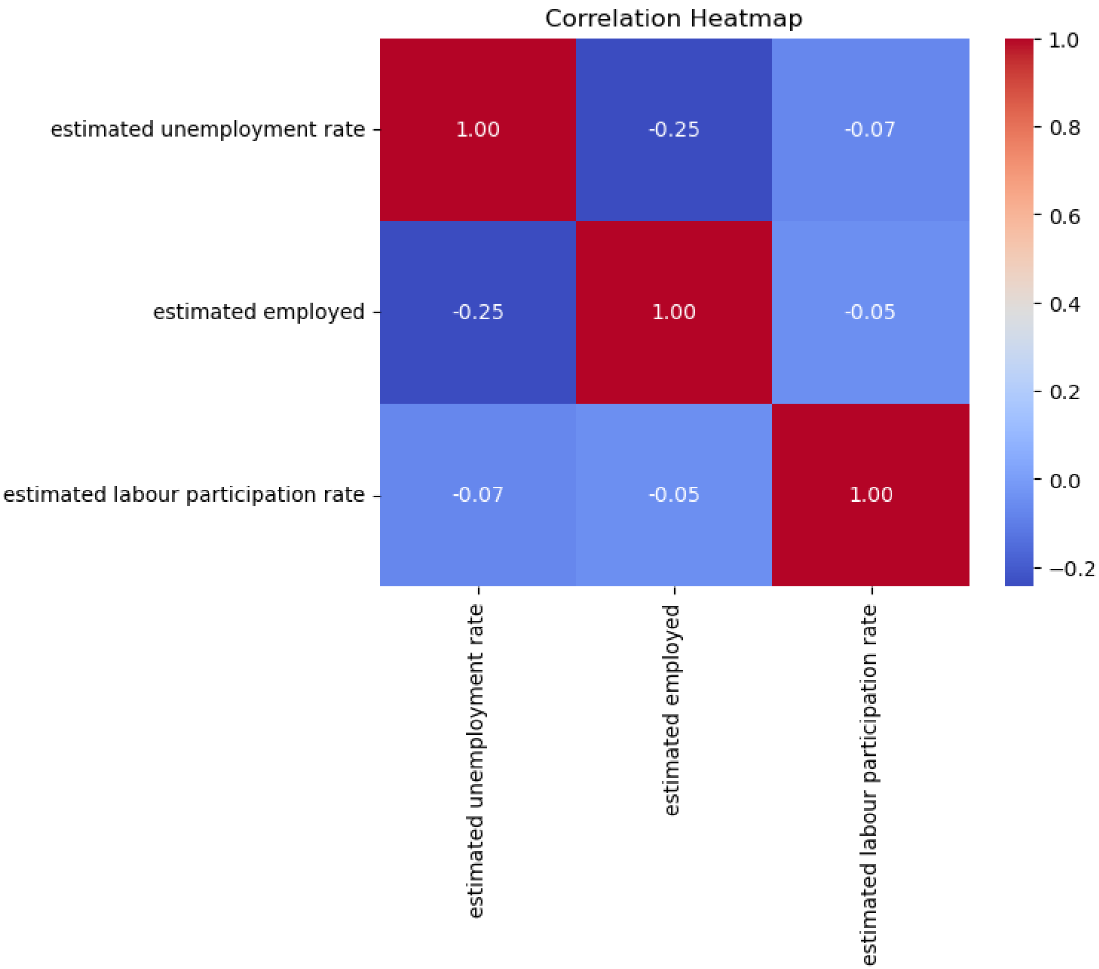
```
[35]: # unenployment rate change after lockdown

      lockdown['rate change in unemployment'] =round(lockdown['unemployment rate␣
       ↪before lockdown']-lockdown['unemployment rate before lockdown']
                                                 /lockdown['unemployment rate␣
       ↪after lockdown'],2)
```

```
[36]: fig = px.bar(lockdown,x='state',y='rate change in unemployment',color='rate␣
       ↪change in unemployment',
                   title='Percentage change in Unemployment rate in each state after␣
       ↪lockdown',template='ggplot2')
      fig.update_layout(xaxis={'categoryorder':'total ascending'})
      fig.show()
```

```
[37]: # Calculate correlation matrix
      correlation_matrix = df[['estimated unemployment rate', 'estimated employed',␣
       ↪'estimated labour participation rate']].corr()

      # Plot correlation heatmap
      sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
      plt.title('Correlation Heatmap')
      plt.show()
```



[38]:
```python
from statsmodels.tsa.arima.model import ARIMA

# Fit ARIMA model
model = ARIMA(df['estimated unemployment rate'], order=(1, 1, 1))
model_fit = model.fit()

# Forecast future unemployment rates
forecast = model_fit.forecast(steps=12)
print(forecast)
```

```
267    11.084691
268    11.640383
269    11.919913
270    12.060524
271    12.131256
272    12.166836
273    12.184734
```

```
274    12.193737
275    12.198266
276    12.200544
277    12.201690
278    12.202267
Name: predicted_mean, dtype: float64
```

[ ]: