**Group No. 1**

**Group Members: Parth Phalke (5019147)**

**Brian Mendes (5019138)**

**Catherine Sarah Sunil (5019108)**

## Working Code

## Designing the basic GUI:

```
from __future__ import print_function
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import folium
from streamlit_folium import folium_static
import pandas_datareader as pdr
from datetime import datetime
from plotly import graph_objs as go


def main():
    st.title("Covid-19 Analysis")
    menu = ["Public Health", "Climate", "Economy"]
    choice = st.sidebar.selectbox("Parameter", menu)

    for i in range(1, 18):
        cols = st.sidebar.beta_columns(3)
        if i == 15:
            cols[1].image("./github-512.png", width=100)

        elif i == 16:
            link = 'https://github.com/Parth-ops'
            cols[0].image("./parth.png", width=70)
            cols[1].markdown(link, unsafe_allow_html=True)

        elif i == 17:
            link = 'https://github.com/Brianrmendes'
            cols[0].image("./brian.png", width=70)
            cols[1].markdown(link, unsafe_allow_html=True)
```

```
        else:
            cols[0].text("")
            cols[1].text("")
            cols[2].text("")
```

## Public Health Screen:

```
if choice == "Public Health":
    st.subheader("Public Health")
    # import packages

    # # Reading Datasets

    # In[4]:

    # Datasets

    death_df = pd.read_csv(
        "https://raw.githubusercontent.com/CSSEGISandData/COVID-
19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_glob
al.csv")
    confirmed_df = pd.read_csv(
        "https://raw.githubusercontent.com/CSSEGISandData/COVID-
19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_
global.csv")
    recovered_df = pd.read_csv(
        "https://raw.githubusercontent.com/CSSEGISandData/COVID-
19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_g
lobal.csv")
    country_df = pd.read_csv(
        "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-
data/data/cases_country.csv")

    # In[5]:

    death_df.head()

    # In[6]:

    confirmed_df.head()
```

```
# In[7]:

recovered_df.head()

# In[8]:

country_df.head()

# # Data cleaning

# In[9]:

# data cleaning -renaming

country_df.columns = map(str.lower, country_df.columns)
recovered_df.columns = map(str.lower, recovered_df.columns)
death_df.columns = map(str.lower, death_df.columns)
confirmed_df.columns = map(str.lower, confirmed_df.columns)

# In[10]:

confirmed_df = confirmed_df.rename(columns={'province/state': 'state', 'country/region':
'country'})
recovered_df = recovered_df.rename(columns={'province/state': 'state', 'country/region':
'country'})
death_df = death_df.rename(columns={'province/state': 'state', 'country/region': 'country'})
country_df = country_df.rename(columns={'country_region': 'country'})

# In[11]:

confirmed_df.drop(["state"], axis=1, inplace=True)
recovered_df.drop(["state"], axis=1, inplace=True)
death_df.drop(["state"], axis=1, inplace=True)

sorted_country_df = country_df.sort_values('confirmed', ascending=False).head(5)

# In[12]:

# # Highlighting the useful columns for plotting

# In[13]:

def highlight_col(x):
    r = 'background-color: red'
    p = 'background-color: purple'
    g = 'background-color: grey'
```

```
        temp_df = pd.DataFrame('', index=x.index, columns=x.columns)
        temp_df.iloc[:, 4] = p
        temp_df.iloc[:, 5] = r
        temp_df.iloc[:, 6] = g
        return temp_df

    st.markdown("Top 5 countries affected by Covid-19:")
    sorted_country_df.style.apply(highlight_col, axis=None)

    # # Plotting the data in bubble  graph

    # In[14]:

    st.set_option('deprecation.showPyplotGlobalUse', False)

    st.write(px.scatter(sorted_country_df.head(10), x='country', y='confirmed',
size='confirmed', color='country',
                        hover_name="country", size_max=60))

    # # Line Graph for confirmed cases Vs confirmed deaths

    # In[15]:

    def plot_cases_for_country(country):
        labels = ['confirmed', 'deaths']
        colors = ['blue', 'red']
        mode_size = [6, 8]
        line_size = [4, 5]

        df_list = [confirmed_df, death_df]

        fig = go.Figure()

        for i, df in enumerate(df_list):
            if country == 'World' or country == 'world':
                x_data = np.array(list(df.iloc[:, 5:].columns))
                y_data = np.sum(np.asarray(df.iloc[:, 5:]), axis=0)

            else:
                x_data = np.array(list(df.iloc[:, 5:].columns))
                y_data = np.sum(np.asarray(df[df['country'] == country].iloc[:, 5:]), axis=0)

            st.write(fig.add_trace(go.Scatter(x=x_data, y=y_data, mode='lines+markers',
                                      name=labels[i],
                                      line=dict(color=colors[i], width=line_size[i]),
                                      connectgaps=True,
```

```
                                text="Total " + str(labels[i]) + ": " + str(y_data[-1])
                                )))

# plot_cases_for_country('India')

st.markdown("Confirmed cases and deaths for a country.")
cnt = st.text_input("Enter the country name", "")
plot_cases_for_country(cnt)

# # Bar graph for top 5 worst affected countries (Confirmed Cases)

# In[16]:

st.write(px.bar(
    sorted_country_df.head(10),
    x="country",
    y="confirmed",
    title="Top 5 worst affected countries (Confirmed Cases)",  # the axis names
    color_discrete_sequence=["green"],
    height=500,
    width=800
))

# # Bar graph top 5 worst affected countries (Confirmed Cases)

# In[17]:

st.write(px.bar(
    sorted_country_df.head(10),
    x="country",
    y="deaths",
    title="Top 5 worst affected countries (Death Cases)",  # the axis names
    color_discrete_sequence=["Brown"],
    height=500,
    width=800
))

# In[18]:

confirmed_df = confirmed_df.dropna(subset=['long'])
confirmed_df = confirmed_df.dropna(subset=['lat'])

# # Plotting the cases on a World map

# In[19]:
```

```
    world_map = folium.Map(location=[11, 0], tiles="cartodbpositron", zoom_start=2,
max_zoom=6, min_zoom=2)

    for i in range(len(confirmed_df)):
        folium.Circle(
            location=[confirmed_df.iloc[i]['lat'], confirmed_df.iloc[i]['long']],
            fill=True,
            radius=(int((np.log(confirmed_df.iloc[i, -1] + 1.00001))) + 0.2) * 50000,
            fill_color='indigo',
            color='red',
            tooltip="<div style='margin: 0; background-color: black; color: white;'>" +
                    "<h4 style='text-align:center;font-weight: bold'>" +
confirmed_df.iloc[i]['country'] + "</h4>"
                                                                    "<hr
style='margin:10px;color: white;'>" +
                    "<ul style='color: white;;list-style-type:circle;align-item:left;padding-
left:20px;padding-right:20px'>" +
                    "<li>Confirmed: " + str(confirmed_df.iloc[i, -1]) + "</li>" +
                    "<li>Deaths:   " + str(death_df.iloc[i, -1]) + "</li>" +
                    "</ul></div>",
        ).add_to(world_map)
    st.subheader("Plotting the cases on a World map")
    folium_static(world_map)
```

## Climate Screen:

```
elif choice == "Climate":

    st.subheader("Climate")
    st.subheader("Air Quality parameters visualization across 6 big cities of India.")
    st.markdown("Note: Missing lines indicate missing data for that period.")
    # data- www.kaggle.com
    df = pd.read_csv("E:/Python_projects/Mini-Project/Main-Project/Files/AQI_city_day.csv")
    df["Date"] = pd.to_datetime(df["Date"])
    df = df.set_index("Date")
    df = df.dropna(how="all")

    ahmed = df[df["City"] == "Ahmedabad"]
    delhi = df[df["City"] == "Delhi"]
    mum = df[df["City"] == "Mumbai"]
    chen = df[df["City"] == "Chennai"]
    hyd = df[df["City"] == "Hyderabad"]
```

```
kol = df[df["City"] == "Kolkata"]

# In[5]:

ahmed = ahmed.resample("YS").mean()
mum = mum.resample("YS").mean()
delhi = delhi.resample("YS").mean()
chen = chen.resample("YS").mean()
hyd = hyd.resample("YS").mean()
kol = kol.resample("YS").mean()

mum.dropna(axis=0, thresh=11, inplace=True)

# ## AQI graph:
#
# Lesser the AQI of the city better is the air quality.
#
# Incomplete lines represent lack of data for that particular period.
#

# In[6]:

fig, ax = plt.subplots(figsize=(30, 15))

if st.checkbox("display_AQI_parameters"):
    select_cols = st.selectbox("Select the parameter", df.columns[1:-1])
    ax.plot(ahmed.index, ahmed[select_cols], linewidth=3)
    ax.plot(delhi.index, delhi[select_cols], linewidth=3)
    ax.plot(chen.index, chen[select_cols], linewidth=3)
    ax.plot(hyd.index, hyd[select_cols], linewidth=3)
    ax.plot(kol.index, kol[select_cols], linewidth=3)
    ax.plot(mum.index, mum[select_cols], linewidth=3)
    ax.tick_params(axis='x', labelsize=20)
    ax.tick_params(axis='y', labelsize=20)
    ax.set_xlabel("Year", fontsize=20)
    ax.set_ylabel("Air Quality Index(AQI)", fontsize=20)
    ax.legend(["Ahmedabad", "Delhi", "Chennai", "Hyderabad", "Kolkata", "Mumbai"],
loc="upper right",
              fontsize=20)
    st.pyplot(fig)

st.markdown("We can conclude that there is a decline in every Air Quality parameter
indicating  overall "
            "decrease in air pollution during the pandemic")
# ## Temperature Comparison between the years 2019-2020(Mumbai).
```

```
    # In[7]:

    # data- www.wunderground.com

    temp = pd.read_excel("E:/Python_projects/Mini-Project/Main-Project/Files/Temp.xlsx",
sheet_name='Sheet1')
    temp["Date"] = pd.to_datetime(temp["Date"])
    temp = temp.set_index(temp["Date"])
    twenty_19 = temp[(temp.index >= pd.datetime(2019, 1, 1)) & (temp.index <=
pd.datetime(2019, 12, 31))]
    twenty_20 = temp[(temp.index >= pd.datetime(2020, 1, 1)) & (temp.index <=
pd.datetime(2020, 12, 31))]

    # In[8]:

    st.subheader("Temperature Comparison between the years 2019-2020(Mumbai)")

    fig, ax = plt.subplots(figsize=(30, 15))

    index = np.arange(0, 12, 1)
    bar_width = 0.35

    ax.set_ylabel("Temperature in degree", fontsize=20)
    ax.set_xlabel("Date")
    ax.bar(index, twenty_19["Avg"], bar_width, color="darksalmon")

    ax.set_xlabel("Date", fontsize=20)
    ax.bar(index + bar_width, twenty_20["Avg"], bar_width, color="navy")

    ax.set_xticks(index + bar_width / 2)
    ax.set_xticklabels(["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct",
"Nov", "Dec"],
                       fontsize=20)
    ax.legend(("2019", "2020"), loc="upper right", fontsize=15)
    ax.set_ylim([20, 32])
    ax.tick_params(axis='y', labelsize=15)
    fig.autofmt_xdate()
    st.pyplot(fig)

    st.markdown("Looking at the above graph we can conclude that there was no significant
effect on the "
                "temperature of Mumbai due to the pandemic.")
```

## Economy Screen:

```
elif choice == "Economy":
   st.subheader("Economy")

   st.subheader("Visualizing a comparison of sector performance of the years 2019 vs 2020 and 2017 vs 2018")
   st.write("\n")

   @st.cache
   def first_plot():
      # datatset from kaggle
      df = pd.read_csv('E:/Python_projects/Mini-Project/Main-Project/Files/NIFTY50_all.csv')
      df["Date"] = pd.to_datetime(df["Date"])
      df = df.set_index("Date")
      sectors = {"MARUTI": "Automobile", "TATAMOTORS": "Automobile", "HEROMOTOCO": "Automobile",
                 "BAJAJ-AUTO": "Automobile", "EICHERMOT": "Automobile", "M&M": "Automobile",
                 "GRASIM": "Cement", "SHREECEM": "Cement", "ULTRACEMCO": "Cement", "ITC": "Cigarettes",
                 "HINDUNILVR": "Consumer Goods", "BRITANNIA": "Consumer Goods", "NESTLEIND": "Consumer Goods",
                 "TITAN": "Consumer Goods", "ASIANPAINT": "Consumer Goods",
                 "ONGC": "Energy", "NTPC": "Energy", "POWERGRID": "Energy", "BPCL": "Energy", "IOC": "Energy",
                 "RELIANCE": "Energy", "GAIL": "Energy",
                 "LT": "Engineering", "UPL": "Fertilizer",
                 "AXISBANK": "Financial Services", "BAJAJFINSV": "Financial Services",
                 "BAJFINANCE": "Financial Services", "HDFC": "Financial Services",
                 "HDFCBANK": "Financial Services",
                 "ICICIBANK": "Financial Services", "INDUSINDBK": "Financial Services",
                 "KOTAKBANK": "Financial Services", "SBIN": "Financial Services",
                 "HCLTECH": "Information Technology", "INFY": "Information Technology",
                 "TCS": "Information Technology", "TECHM": "Information Technology",
                 "WIPRO": "Information Technology",
                 "ZEEL": "Media & Entertainment",
                 "HINDALCO": "Metals & Mining", "VEDL": "Metals & Mining", "JSWSTEEL": "Metals & Mining",
                 "TATASTEEL": "Metals & Mining", "COALINDIA": "Metals & Mining",
                 "CIPLA": "Pharma", "DRREDDY": "Pharma", "SUNPHARMA": "Pharma",
                 "ADANIPORTS": "Shipping", "MUNDRAPORT": "Shipping",
                 "BHARTIARTL": "Telecom"
```

```
        }

    df["SECTORS"] = df["Symbol"].map(sectors)

    # In[4]:

    df_2017 = df[(df.index >= pd.datetime(2017, 1, 1)) & (df.index <= pd.datetime(2017,
12, 31))]

    cmp_17 = df_2017.groupby(["Symbol"])
    companies_17 = cmp_17.resample('MS').mean()

    # In[5]:

    df_2018 = df[(df.index >= pd.datetime(2018, 1, 1)) & (df.index <= pd.datetime(2018,
12, 31))]

    cmp_18 = df_2018.groupby(["Symbol"])
    companies_18 = cmp_18.resample('MS').mean()

    # In[6]:

    df_2019 = df[(df.index >= pd.datetime(2019, 1, 1)) & (df.index <= pd.datetime(2019,
12, 31))]

    cmp_19 = df_2019.groupby(["Symbol"])
    # companies_19

    # In[7]:

    df_2020 = df[(df.index >= pd.datetime(2020, 1, 1)) & (df.index <= pd.datetime(2020,
12, 31))]
    cmp_20 = df_2020.groupby(["Symbol"])
    # companies_20

    # In[8]:

    sec_17 = df_2017.groupby(["SECTORS"])
    ind_17 = sec_17.resample('Y').mean()
    ind_17 = ind_17.reset_index("Date")

    # In[9]:

    sec_18 = df_2018.groupby(["SECTORS"])
    ind_18 = sec_18.resample('Y').mean()
    ind_18 = ind_18.reset_index("Date")
```

```
        # In[10]:

        sec_19 = df_2019.groupby(["SECTORS"])
        ind_19 = sec_19.resample('Y').mean()
        ind_19 = ind_19.reset_index("Date")

        # In[11]:

        sec_20 = df_2020.groupby(["SECTORS"])
        ind_20 = sec_20.resample('Y').mean()
        ind_20 = ind_20.reset_index("Date")

        companies_19 = cmp_19.resample('Y').mean()
        companies_19 = companies_19.reset_index("Date")

        companies_20 = cmp_20.resample('Y').mean()
        companies_20 = companies_20.reset_index("Date")

        nifty_index = pdr.get_data_yahoo('^NSEI', datetime(2007, 1, 1), datetime(2021, 5, 1),
interval='m')

        bse_sensex = pdr.get_data_yahoo('^BSESN', datetime(2007, 9, 30), datetime(2021, 5,
1), interval='m')

        return companies_19, companies_20, ind_17, ind_18, ind_19, ind_20, nifty_index,
bse_sensex

        # ## Visualizing a comparison of sector performance of the years 2019 vs 2020 and
2017 vs 2018

        # In[12]:

    #  CACHED DATA STORED INTO THE RESPECTIVE VARIABLES
    companies_19, companies_20, ind_17, ind_18, ind_19, ind_20, nifty_index, bse_sensex =
first_plot()

    fig1 = plt.figure(figsize=(25, 15))
    ax1 = fig1.add_subplot(211)
    ax2 = fig1.add_subplot(212, sharey=ax1)

    index = np.arange(0, 14, 1)
    bar_width = 0.35

    ax1.set_yticklabels(["100K", "200K", "300K", "400K", "500K", "600K", "700K"],
fontsize=15)
```

```
    ax1.bar(index, ind_19["Turnover"], bar_width)
    ax1.set_xlabel("Sectors", fontsize=15)
    ax1.bar(index + bar_width, ind_20["Turnover"], bar_width)

    ax1.set_xticks(index + bar_width / 2)
    ax1.set_xticklabels(ind_19.index, fontsize=12)
    ax1.legend(("2019", "2020"), loc="upper right")
    fig1.autofmt_xdate()

    # for years 2017-18
    ax2.set_yticklabels(["100K", "200K", "300K", "400K", "500K", "600K", "700K"],
fontsize=15)
    ax2.set_ylabel("Turnover in Cr. --->", fontsize=15)
    ax2.set_xlabel("Sectors")
    ax2.bar(index, ind_17["Turnover"], bar_width)

    ax2.set_xlabel("Sectors", fontsize=15)
    ax2.bar(index + bar_width, ind_18["Turnover"], bar_width)

    ax2.set_xticks(index + bar_width / 2)
    ax2.set_xticklabels(ind_17.index, fontsize=12)
    ax2.legend(("2017", "2018"), loc="upper right")
    fig1.autofmt_xdate()
    st.pyplot()

    st.markdown("On comparing the two plots we can conclude that the following sectors were
profitable during the "
                "pandemic: ")
    st.markdown("Cigarettes")
    st.markdown("Energy")
    st.markdown("Financial Services")
    st.markdown("Information Technology")
    st.markdown("Pharma")
    st.markdown("Telecom.")

    # In[ ]:

    # From the above figure we can notice a considerable growth of the following sectors:
    #
    # 1. Financial Services
    # 2. Pharma
    # 3. Telecom

    # ## Visualizing Company wise performance for the years 2019 and 2020

    # In[13]:
```

```
    # In[14]:
    st.subheader("Visualizing Company wise performance for the years 2019 and 2020")
    fig2 = plt.figure(figsize=(30, 20))
    ax1 = fig2.add_subplot(211)

    ax1.set_ylabel("Turnover in Trillion Rs.--->", fontsize=25)
    ax1.set_xlabel("Companies", fontsize=25)
    ax1.set_xticklabels(companies_19.index, rotation=90, fontsize=20)
    ax1.set_yticklabels(["0", "20", "40", "60", "80", "100", "120"], fontsize=20)
    ax1.set_title("Average Annual Turnover of NIFTY-50 Companies for the year 2019",
fontsize=30)
    ax1.tick_params()

    ax2 = fig2.add_subplot(212)
    ax2.set_ylabel("Turnover in Trillion Rs.--->", fontsize=25)
    ax2.set_xlabel("Companies", fontsize=25)

    ax2.set_xticklabels(companies_20.index, rotation=90, fontsize=20)
    ax2.set_yticklabels(["0", "20", "40", "60", "80", "100", "120"], fontsize=20)
    ax2.set_title("Average Annual Turnover of NIFTY-50 Companies for the year 2020",
fontsize=30)

    fig2.tight_layout()
    ax1.bar(companies_19.index, companies_19["Turnover"], color="lightgreen")
    ax2.bar(companies_20.index, companies_20["Turnover"], color="skyblue")
    st.pyplot()

    st.markdown("Looking at the above graph, it is clear that most of the companies suffered
losses during the "
                "panddemic")
    st.markdown("The following companies performed well in 2020 compared to 2019:")
    st.markdown("BAJAJ Finance")
    st.markdown("RELIANCE")

    st.subheader("NIFTY and BSE Index over the years")

    # ##  Historical data scraping and visualization

    # In[15]:

    # data source- https://in.finance.yahoo.com/

    # In[16]:

    fig3 = plt.figure(figsize=(30, 15))
```

```
    ax1 = fig3.add_subplot(211)

    ax1.plot(nifty_index.index, nifty_index["Adj Close"], linewidth=4, color="teal")
    ax1.set_ylabel("Adjusted Close Index--->", fontsize=22)
    ax1.set_xlabel("Year", fontsize=22)
    ax1.set_title("NIFTY Index over the years", fontsize=20)
    ax1.annotate("* Global Financial Crisis", xy=(nifty_index.index[8], nifty_index["Adj
Close"][8]),
                 xycoords='data', xytext=(nifty_index.index[5], 6000), fontsize=20)
    ax1.annotate("* Covid-19 Pandemic", xy=(nifty_index.index[150], nifty_index["Adj
Close"][150]),
                 xycoords='data',
                 xytext=(nifty_index.index[148], 8000), fontsize=20)
    ax1.tick_params(axis='x', labelsize=20)
    ax1.tick_params(axis='y', labelsize=25)

    ax2 = fig3.add_subplot(212)
    ax2.plot(bse_sensex.index, bse_sensex["Adj Close"], linewidth=4, color="indigo")
    ax2.set_ylabel("Adjusted Close Index--->", fontsize=22)
    ax2.set_xlabel("Year", fontsize=22)
    ax2.set_title("BSE Index Over the years", fontsize=20)
    ax2.annotate("* Global Financial Crisis", xy=(bse_sensex.index[8], bse_sensex["Adj
Close"][8]),
                 xycoords='data',
                 xytext=(bse_sensex.index[5], 22000), fontsize=20)
    ax2.annotate("* Covid-19 Pandemic", xy=(bse_sensex.index[150], bse_sensex["Adj
Close"][150]),
                 xycoords='data',
                 xytext=(bse_sensex.index[148], 26000), fontsize=20)
    ax2.tick_params(axis='x', labelsize=20)
    ax2.tick_params(axis='y', labelsize=20)

    fig3.autofmt_xdate()
    fig3.tight_layout()
    st.pyplot()

    st.markdown("The sharp dips during the 2020 period is due to the global Covid-19
pandemic.")

    # for i in range(140, 155):
    # print(nifty_index.index[i] , ":" , nifty_index["Adj Close"][i])

    # ## Analysis of unemployment rates before and during the lockdown.

    # In[17]:
```

```
    # data source- https://data.worldbank.org/
    # DATA CLEANING
    st.subheader("Analysis of unemployment rates before and during the lockdown")

    # year_wise = pd.read_csv("E:/Python_projects/Mini-Project/Main-
Project/Files/india_unemployment_rate.csv")
    # year_wise["date"] = pd.to_datetime(year_wise["date"])
    # year_wise = year_wise.set_index("date")
    # year_wise = year_wise.dropna(axis=1, how="all")
    # year_wise = year_wise.fillna(value=0)
    # year_wise.columns = ["Unemployment Rate", "Annual Change"]
    # year_wise

    @st.cache
    def ex():
        xls = pd.ExcelFile('E:/Python_projects/Mini-Project/Main-
Project/Files/World_Unemployment.xls')
        world_data = pd.read_excel(xls, 'Data')
        for i in range(1960, 1992):
            world_data = world_data.drop(str(i), axis=1)
        world_data = world_data.drop(world_data.index[0])
        world_data = world_data.T
        world_data = world_data.drop(["Country Code", "Indicator Name", "Indicator Code"],
axis=0)
        world_data.rename(index={'Country Name': None}, inplace=True)

        world_data.index = pd.to_datetime(world_data.index)
        world_data.index = world_data.index.fillna("Date")
        world_data.columns = world_data.iloc[0]
        world_data = world_data[1:]
        world_data = world_data.dropna(axis=1, how="all")
        # world_data["World"]
        return world_data

    # In[18]:
    world_data = ex()
    fig, ax = plt.subplots(figsize=(30, 15))
    choice1 = st.text_input("Enter the first country")
    choice2 = st.text_input("Enter the second country")
    if choice1 and choice2 != "":
        ax.plot(world_data.index, world_data[choice1], linewidth=4)
        ax.plot(world_data.index, world_data[choice2], linewidth=4)
        ax.legend((choice1, choice2), loc="upper left", fontsize=20)
        ax.tick_params(axis='x', labelsize=20)
        ax.set_ylabel("Unemployment rate in %", fontsize=25)
        ax.set_xlabel("Year", fontsize=25)
```

```
        ax.set_title("Unemployment rate of {} and {}".format(choice1, choice2), fontsize=25)
        ax.tick_params(axis='y', labelsize=20)
        st.pyplot()

    st.markdown("From the above visualization we observe that there's been an increase in the unemployment rates "
                "across most of the countries.")


if __name__ == '__main__':
    main()
```

## Stock Prediction Notebook:

```
#!/usr/bin/env python
# coding: utf-8

# # Importing pac

# In[81]:


### Data Collection
import matplotlib.pyplot as plt
import pandas as pd
import pandas_datareader as pdr
import numpy as np
key="f7fee84c9ad6db319bb4e6cd55a797e5ab66561f"


# # Getting Tata Motors stock prices from Tiingo

# In[82]:


df = pdr.get_data_tiingo('TTM', api_key=key)


# In[83]:


df.to_csv('C:/Users/asus/TTM.csv')
```

# In[84]:

df=pd.read_csv('C:/Users/asus/TTM.csv')

# In[85]:

df.head()

# In[86]:

df.tail()

# In[87]:

df1=df.reset_index()['close']

# In[88]:

df1.shape

# # Plotting the current stock market graph

# In[89]:

plt.plot(df1)

# In[90]:

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))

# In[91]:

df1

# # Spliting the dataset into train and test

# In[92]:

```
training_size=int(len(df1)*0.65)
test_size=len(df1)-training_size
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

# In[93]:

training_size,test_size

# # Converting an array of values into a dataset matrix

# In[94]:

```
def create_dataset(dataset, time_step=1):
    dataX, dataY = [] , []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

# In[95]:

```
#reshape into X=t,t+1,t+2,t+3 and Y=t+4
import numpy
time_step = 100
X_train , y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)
```

# In[96]:

```
print(X_train.shape), print(y_train.shape)
```

# In[97]:

```
print(X_test.shape), print(y_test.shape)
```

# # Reshaping input to be samples, time steps,features for LSTM

# In[98]:

```
X_train = X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)
```

# # Creating the STACKED LSTM model

# In[99]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

# In[100]:

```
model = Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

# In[101]:

model.summary()

# # Training the model for 100 epochs

# In[102]:

model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=100,batch_size=64,verbose=1)

# In[116]:

import tensorflow as tf

# In[117]:

tf.__version__

# In[152]:

#check performance metrics
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)

# In[153]:

#Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)

# # Calculate RMSE performance metrics

# In[120]:

```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))


# In[121]:


# Test Data RMSE
math.sqrt(mean_squared_error(y_test,test_predict))


# # Shift train predictions for plotting

# In[122]:


look_back=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(df1))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()


# In[123]:


len(test_data)


# In[137]:


x_input=test_data[341:].reshape(1,-1)
x_input.shape


# In[138]:
```

```
x_input=test_data[341:].reshape(1,-1)


# In[139]:


x_input.shape


# In[140]:


temp_input=list(x_input)
temp_input=temp_input[0].tolist()


# # Prediction for next 10 days

# In[141]:


from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
```

```
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)


# In[143]:


day_new=np.arange(1,101)
day_pred=np.arange(101,131)


# In[144]:


len(df1)


# In[145]:


df3 = df1.tolist()
df3.extend(lst_output)


# # Prediction for next 10 days on a graph

# In[150]:


plt.plot(day_new,scaler.inverse_transform(df1[1158:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))

# # Plotting the final predicted graph for Tata motors

# In[151]:

df3 = df1.tolist()
df3.extend(lst_output)
plt.plot(df3[1200:])
```

## Running the project: