NAAN MUDHALVAN-IBM DATA ANALYTICS WITH COGNOS

PROJECT PHASE 4: DEVELOPMENT PART 2

PROJECT TITLE:

COMPREHENSIVE ANALYSIS OF COVID-19 VACCINATION DATA:

Enhancing deployment strategies for optimal public health impact

TEAM MEMBERS:

1. Dinesh K
   - Email: dineshreddykonda@gmail.com
2. Ezhil Oviya C
   - Email: oviyagarcia141@gmail.com
3. Geetika TK
   - Email: geetikatk@gmail.com
4. Gogilarasan S
   - Email: gogilarasan@gmail.com
5. Nijantha Nathan M
   - Email: nijanthanathan72@gmail.com

INTRODUCTION

In this phase, we will continue building the project by performing the following:

- Exploratory data analysis (EDA)
- Statistical analysis
- Visualization

The above processes are performed on the already loaded and pre-processed dataset. During this stage, we will carry on with the project by conducting various analysis on the prepared dataset and finally visualization techniques are used to represent the same for better and clearer understandings.

This step marks the next milestone of our project in understanding the data we have.

DATA COLLECTION

Collecting data for COVID-19 vaccine analysis is a vital aspect of understanding the effectiveness and impact of vaccination campaigns in order to perform proper analysis.

This process involves gathering a wide range of information related to vaccination efforts, including country-wise total vaccinations available, people vaccinated, adverse events, distribution logistics, and more. The source of the data for this project is,

https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress

DATA PRE-PROCESSING:

Once reliable data has been collected, it is now time to clean and prepare the data for analysis. This process is coined ad Data Pre-processing.

The above processes have already been performed and documented in the previous phase. Now,

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) is an essential initial step in data analysis. It is the method of studying and exploring data set to

recognize their traits, discover patterns, locate outliers, and identify relationships between variables.

EDA is essential for getting a clear picture of the data which is useful in subsequent decision-making and can be performed using various statistical and graphical techniques. It involves multiple iterations and proves especially beneficial in prepping data for machine learning or statistical modeling. It is performed in the project as follows,

Initially, we take a look at the different types of data we have in our dataset.

```
df_vaccination.dtypes # take a look of the data types that we dealing with (precisely the date column)
```

The output is,

```
country                              object
iso_code                             object
date                                 object
total_vaccinations                  float64
people_vaccinated                   float64
people_fully_vaccinated             float64
daily_vaccinations_raw              float64
daily_vaccinations                  float64
total_vaccinations_per_hundred      float64
people_vaccinated_per_hundred       float64
people_fully_vaccinated_per_hundred float64
daily_vaccinations_per_million      float64
vaccines                             object
source_name                          object
source_website                       object
dtype: object
```

Note that the “date” field is of object datatype and so for better analysis, it is converted to datetime format by,

```
In [4]: df_vaccination['date'] = pd.to_datetime(df_vaccination['date'], format="%Y-%m-%d")
        # to_datetime is a pandas method which helps to convert datetime string into pandas datetime object

In [5]: df_vaccination.dtypes # check our new data types after converting date(column) into datetime64[ns] by using pd.to_datetime()

Out[5]: country                                object
        iso_code                               object
        date                           datetime64[ns]
        total_vaccinations                    float64
        people_vaccinated                     float64
        people_fully_vaccinated               float64
        daily_vaccinations_raw                float64
        daily_vaccinations                    float64
        total_vaccinations_per_hundred        float64
        people_vaccinated_per_hundred         float64
        people_fully_vaccinated_per_hundred   float64
        daily_vaccinations_per_million        float64
        vaccines                               object
        source_name                            object
        source_website                         object
        dtype: object
```

Now it can be seen that the datatype has been changed which makes it easier to work with it.

After which, various other fields are being examined to make sure we have the perfect set of data to analyze.

```
df_vaccination.isnull().sum()

#There are no empty rows for country, iso_code or date columns.
```

```
country                                0
iso_code                               0
date                                   0
total_vaccinations                 42905
people_vaccinated                  45218
people_fully_vaccinated            47710
daily_vaccinations_raw             51150
daily_vaccinations                   299
total_vaccinations_per_hundred     42905
people_vaccinated_per_hundred      45218
people_fully_vaccinated_per_hundred 47710
daily_vaccinations_per_million       299
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

```
# General Overview of the calculations in data

df_vaccination.describe()
```

	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccina
count	43,607.00	41,294.00	38,802.00	35,362.00	86,213.00	43,607.00	
mean	45,929,644.64	17,705,077.79	14,138,299.85	270,599.58	131,305.49	80.19	
std	224,600,360.18	70,787,311.50	57,139,201.72	1,212,426.60	768,238.77	67.91	
min	0.00	0.00	1.00	0.00	0.00	0.00	
25%	526,410.00	349,464.25	243,962.25	4,668.00	900.00	16.05	
50%	3,590,096.00	2,187,310.50	1,722,140.50	25,309.00	7,343.00	67.52	
75%	17,012,303.50	9,152,519.75	7,559,869.50	123,492.50	44,098.00	132.74	
max	3,263,129,000.00	1,275,541,000.00	1,240,777,000.00	24,741,000.00	22,424,286.00	345.37	

Followed by,

It is not always necessary that all the fields/attributes in the collected dataset is/are useful for our analysis.

Therefore, the fields “source_name”,”source_website” and “vaccine_columns” are not required and hence are dropped for more efficient analysis.

```
#drop the source_name,source_website and vaccine columns

df_vaccine_country = df_vaccination.drop(['source_name','source_website','vaccines'],axis=1)
df_vaccine_country.head()
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per
0	Afghanistan	AFG	2021-02-22	0.00	0.00	NaN	NaN	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	1,367.00	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	1,367.00	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN	1,367.00	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN	1,367.00	

All the Nan values are then replaced by 0 to make calculations easier. From the screenshot below, it can be seen that the sum of all null values in every column is 0.

```
# convert Date column to date type and fill na values with 0 for calculation

df_vaccine_country["date"] = pd.to_datetime(df_vaccine_country["date"], format = '%Y-%m-%d')

df_vaccine_country = df_vaccine_country.replace([np.inf, -np.inf], np.nan)
df_vaccine_country = df_vaccine_country.fillna(0)
df_vaccine_country.isnull().sum()
```

```
country                              0
iso_code                             0
date                                 0
total_vaccinations                   0
people_vaccinated                    0
people_fully_vaccinated              0
daily_vaccinations_raw               0
daily_vaccinations                   0
total_vaccinations_per_hundred       0
people_vaccinated_per_hundred        0
people_fully_vaccinated_per_hundred  0
daily_vaccinations_per_million       0
dtype: int64
```

Once the dataset is prepared and ready for analysis, statistical analysis is performed on it.

STATISTICAL ANALYSIS

In statistical analysis, the total, average, maximum and minimum of different vaccinations status by country is calculated.

```
#########STATISTICAL ANALYSIS##########
#Function to find total, average, maximum  and minimum of different vaccinations status by country

def vaccination_country(col_name,func_name):

    '''
    Function that requires vaccination column name, and sum/mean/max/min function name as string arguments.
    '''
    if func_name == 'sum':

        return  (df_vaccine_country[['country',col_name]].groupby(by='country')
                                .sum()
                                .sort_values(by=col_name,ascending= False)
                                .reset_index()
                            )

    elif func_name == 'mean':

        return  (df_vaccine_country[['country',col_name]].groupby(by='country')
                                .mean()
                                .sort_values(by=col_name,ascending= False)
                                .reset_index()
                               )
    elif func_name == 'max':

        return  (df_vaccine_country[['country',col_name]].groupby(by='country')
                                .max()
                                .sort_values(by=col_name,ascending= False)
                                .reset_index()
                               )
    elif func_name == 'min':

        return  (df_vaccine_country[['country',col_name]].groupby(by='country')
                                .min()
                                .sort_values(by=col_name,ascending= False)
                                .reset_index()
                               )
```

The code snippet of function for finding country with maximum and minimum daily vaccinations is,

```
#Function for Country with maximum and minimum daily vaccinations
def daily_vaccination_country(col_name,func_name):

    '''
    A function that requires daily_vaccination column and max/min function name as string arguments.
    '''

    daily_vaccination = (df_vaccine_country
                                .pivot_table(index='country',columns='date',values=col_name)
                                   )
    if func_name == 'max':

        daily_vaccination['Highest Daily Vaccination'] = daily_vaccination.max(axis=1)
        daily_vaccination['Date - Highest Daily Vaccination'] = daily_vaccination.idxmax(axis=1)
        daily_vaccination.sort_values(by='Highest Daily Vaccination',ascending=False,inplace=True)
        daily_vaccination.rename_axis('',axis=1,inplace=True)

        return daily_vaccination[['Highest Daily Vaccination','Date - Highest Daily Vaccination']].reset_index()

    elif func_name == 'min':

        daily_vaccination.replace(0.00,np.nan,inplace=True)
        daily_vaccination['Lowest Daily Vaccination'] = daily_vaccination.min(axis=1)
        daily_vaccination['Date - Lowest Daily Vaccination'] = daily_vaccination.idxmin(axis=1)
        daily_vaccination.sort_values(by='Lowest Daily Vaccination',ascending=False,inplace=True)
        daily_vaccination.rename_axis('',axis=1,inplace=True)

        return daily_vaccination[['Lowest Daily Vaccination','Date - Lowest Daily Vaccination']].reset_index()
```

Finally, calculating the highest and lowest daily vaccination and the respective dates.

```
#Calculating highest and lowest daily vaccination and the respective dates.
highest_daily_vaccination = daily_vaccination_country('daily_vaccinations','max')
lowest_daily_vaccination = daily_vaccination_country('daily_vaccinations','min')
```

Once all necessary aspects are calculated, it now time for visualization i.e., representing the analyzed records graphically for better understanding of complex data patterns and relations.

VISUALIZATION

Data visualization is the use of graphical elements such as charts, graphs, and maps to represent data and information visually. The use of visualization tools provides an accessible way to see and understand trends, outliers, and patterns in data.

There are various techniques in data visualization. Few of them are described below,

- ***Histograms***: Plot the frequency distribution of numerical variables to identify patterns and distributions.
- ***Box Plots***: Display the distribution, central tendency, and outliers in numerical data.
- ***Scatter Plots***: Visualize relationships between two numerical variables to identify correlations or patterns.
- ***Bar Charts***: Used for categorical data to show the frequency of different categories.
- ***Heatmaps***: Display the correlation between variables using color gradients.
- ***Pair Plots***: When dealing with multiple numerical variables, pair plots help visualize relationships between them.

```
##########VISUALIZATION##########
# Calculating different vaccinations for visualizations
max_total_vaccinations = vaccination_country('total_vaccinations','max')

sum_people_vaccinated = vaccination_country('people_vaccinated','sum')
sum_people_fully_vaccinated = vaccination_country('people_fully_vaccinated','sum')

avg_total_vaccinations = vaccination_country('total_vaccinations_per_hundred','mean')
avg_people_vaccinated = vaccination_country('people_vaccinated_per_hundred','mean')
avg_people_fully_vaccinated = vaccination_country('people_fully_vaccinated_per_hundred','mean')
avg_daily_vaccinations = vaccination_country('daily_vaccinations_per_million','mean')
```

First, all required parameters are calculated using the previously created functions.

```
#Set sns theme and default figsize for all the sns visualizations.

sns.set_theme(style='whitegrid')
sns.set(rc={'figure.figsize' : (12,5)})

fig, axes  = plt.subplots(2,1)

sns.barplot(x='country',y='total_vaccinations',data=max_total_vaccinations.head(),ax=axes[0])
axes[0].set(xlabel = '', ylabel = 'Total Vaccinations', title ='Top 5 Countries in terms of total vaccinations!')

sns.barplot(x='country',y='total_vaccinations',data=max_total_vaccinations.tail(),ax=axes[1])
axes[1].set(xlabel = '', ylabel = 'Total Vaccinations', title ='Bottom 5 Countries in terms of total vaccinations!')


fig.tight_layout()
plt.show()
```

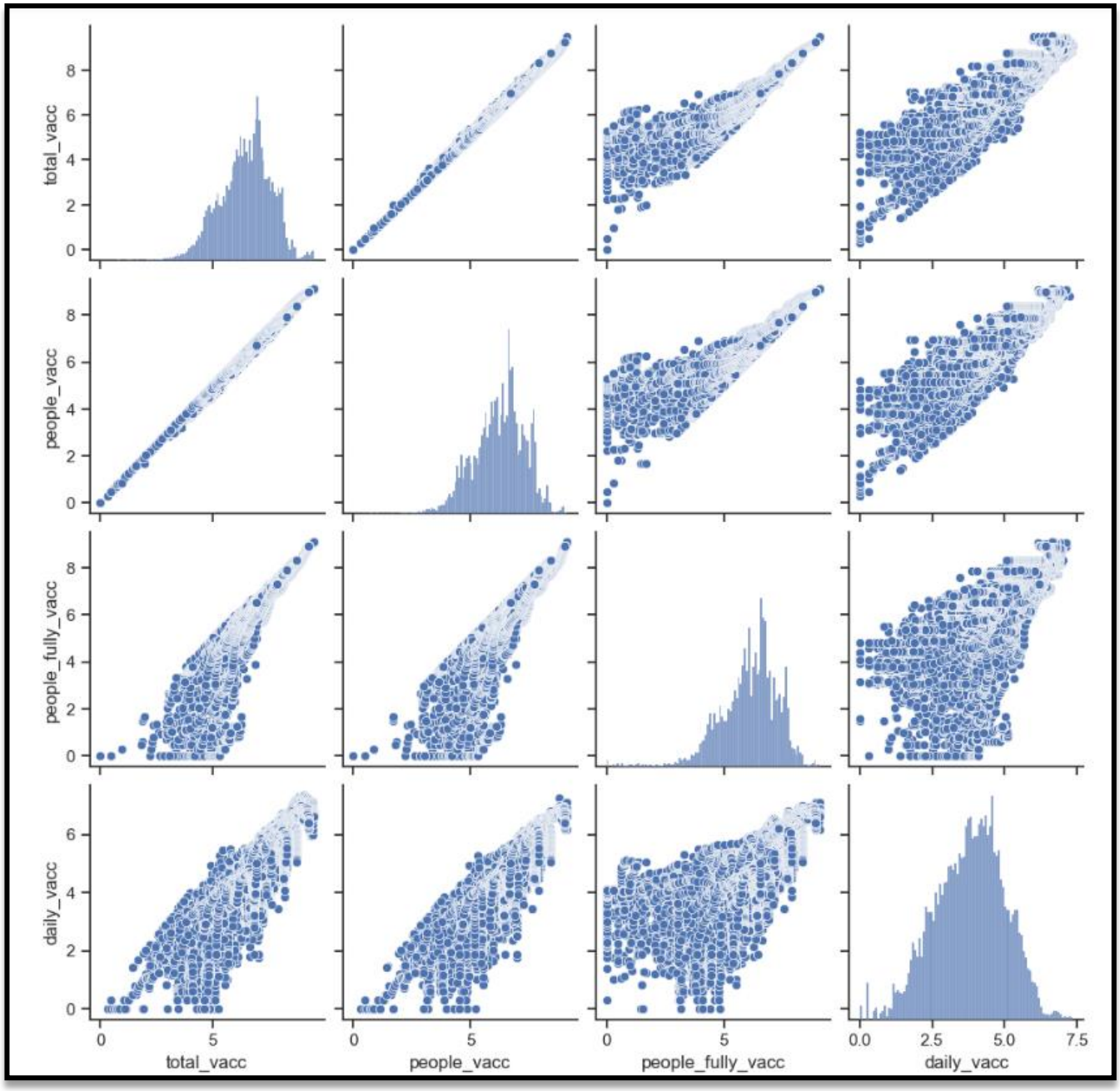
Then, a bar graph is used to represent the Top 5 and Bottom 5 countries in terms of total vaccinations.

```
#Plotting scatterplot matrix using Seaborn
#create dataframe with important features.
df_vaccination['total_vacc'] = np.log10(df_vaccination['total_vaccinations'])
df_vaccination['people_vacc'] = np.log10(df_vaccination['people_vaccinated'])
df_vaccination['people_fully_vacc'] = np.log10(df_vaccination['people_fully_vaccinated'])
df_vaccination['daily_vacc'] = np.log10(df_vaccination['daily_vaccinations'])

#drop the original nontransformed columns
df_vaccination = df_vaccination.drop(columns = ['total_vaccinations','people_vaccinated','people_fully_vaccinated', 'daily_vaccir


covid_features = df_vaccination[['date', 'total_vacc', 'people_vacc' , 'people_fully_vacc' , 'daily_vacc']]
sns.set_theme(style="ticks")
sns.pairplot(covid_features)
```

Here scatter plot is used for which the output is,



CONCLUSION:

At the end of this phase, the collected and prepared data has been gone through Exploratory data analysis (EDA) and Statistical analysis. And finally, visualization tools have been used to graphically represent the analyzed data which helps in deeper insights on the same.