COVID-19 VACCINES ANALYSIS

Data loading and preprocessing are critical steps in the field of data analytics. These processes are essential for preparing raw data for analysis, ensuring its quality, and making it suitable for various data-driven tasks such as statistical analysis, machine learning and data visualization.

COVID VACCINES ANALYSIS :-

DATA LOADING:

Data loading is a critical step in any data analysis project, including COVID VACCINES analysis. In a covid vaccination analysis project, you typically collect vaccination data from various sources. Here's a step-by-step guide on the data loading process:

1. Identify Data Sources:

• Determine where your data is coming from. Common sources include government agencies, research institutions, or private sensor networks.

2. Access Data Sources:

• Find out how to access the data from your chosen sources. This may involve visiting websites, making API requests, downloading data files, or setting up data subscriptions.

3. Data Format:

• Understand the format of the data. Vaccination data can come in various formats such as CSV, Excel, JSON, or database records. Ensure you are familiar with the format to be able to read it correctly.

4. Data Retrieval:

• Retrieve the data using the appropriate methods based on your data source. Some common approaches include: • Use Python libraries like `requests` for web scraping and API calls. • Use Pandas to read data from CSV or Excel files. • Use database queries (e.g., SQL) for data stored in databases.

DATA PREPROCESSING:

Data preprocessing is a crucial step in an covid vaccine analysis project as it ensures that your data is clean, consistent, and suitable for further analysis. Here is a step by-step guide on how to preprocess air quality data:

1. Data Cleaning:

• Handle missing data: Identify and deal with missing values, which are common in real-world datasets. You can choose to impute missing values (e.g., using mean, median, or machine learning techniques) or remove records with missing data, depending on the context.

• Detect and handle outliers: Identify and address outliers in your data. Outliers can significantly impact analysis results. You may choose to remove them or apply robust statistical methods.

2. Data Transformation:

• Convert data types: Ensure that your data types are appropriate. For example, convert date and time columns to datetime objects if necessary.

• Encoding categorical variables: If your data contains categorical variables, encode them into numerical values. This can be done using one-hot encoding or label encoding.

• Aggregating data: Depending on the granularity of your data, you might need to aggregate data to a specific time scale for analysis.

• Feature scaling: Normalize or standardize numerical features to ensure they are on the same scale, especially if you plan to use machine learning algorithms.

3. Data Visualization:

• Use data visualization libraries like Matplotlib, Seaborn, or geographic mapping tools to explore your preprocessed data and gain insights.

PYTHON SCRIPT:

```
import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split


# Load COVID vaccine data (replace 'covid_vaccine_data.csv' with your data file)

data_file = "country_vaccinations.csv"

covid_vaccine_data = pd.read_csv(data_file)


# Remove duplicates (if applicable)

covid_vaccine_data.drop_duplicates(inplace=True)


# Fill missing values with the mean of each column

covid_vaccine_data.fillna(covid_vaccine_data.mean(), inplace=True)


# Convert the 'timestamp' column to datetime (if applicable)
```

```
covid_vaccine_data['timestamp'] = pd.to_datetime(covid_vaccine_data['timestamp'])


# Compute daily averages (adjust columns accordingly)
daily_averages = covid_vaccine_data.groupby(covid_vaccine_data['timestamp'].dt.date).mean()


# Scale the data using Min-Max scaling (adjust columns accordingly)
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(covid_vaccine_data[['feature1', 'feature2']])


# Split the data into training and testing sets
X = covid_vaccine_data.drop(columns=['target_column'])  # Adjust 'target_column' to your target variable
y = covid_vaccine_data['target_column']


 # Adjust 'target_column' to your target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Save daily averages to a CSV file (if needed)
daily_averages.to_csv("preprocessed_country_vaccinations.csv", index=False)
```