GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, Tiruttani

1103-GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT TITLE

Covid-19 Vaccines Analysis

Rahul Sakthevel SP
3rd yr, 5th sem
Reg no.:110321104701
Falkonzrahul125@gmail.com
Ph: 8124107010

Feature engineering :

Data Understanding:

First, load the dataset and understand its structure and contents. Explore the columns, data types, and any missing values.

Data Cleaning:

Handle missing data: Decide how to deal with missing values, either by imputation (e.g., using mean, median, or mode) or removing rows with missing values if appropriate.

Feature Extraction:

Extract useful information from existing features. For example, you can extract the day of the week, month, or year from date columns.Calculate new features that might be relevant, such as the daily vaccination rate, percentage of the population vaccinated, or the number of days since the start of vaccination.

Feature Transformation:

Normalize or standardize numeric features if necessary.Encode categorical features using techniques like one-hot encoding or label encoding.

Feature Selection:

Choose the most relevant features to include in your model. You can use techniques like correlation analysis, feature importance from tree-based models, or dimensionality reduction methods.

Feature Scaling:

Scale your features to ensure they have similar scales, especially if you're using algorithms sensitive to feature scales, such as K-Means or Support Vector Machines.

Feature Engineering for Time Series (if applicable):

If your dataset includes time series data, consider lag features, rolling statistics, and seasonality components like day of the week or month.

Feature Visualization:

Create visualizations to understand the relationships between features and the target variable, as this can guide further feature engineering.

Model Building:

After feature engineering, you can build and train machine learning models. Be sure to split your data into training and testing sets to evaluate model performance.

Iterate:

Evaluate the model's performance and iteratively refine your feature engineering based on insights gained from model results. Remember that feature engineering is an iterative process, and it's crucial to experiment with different features and transformations to improve your model's accuracy and gain valuable insights from your data.

Training

Data Preparation:

Download the dataset from Kaggle or the source you mentioned.Preprocess the data, which may involve handling missing values, encoding categorical variables, and scaling features if needed.Split the data into training and testing sets.

Select a Machine Learning Model:

Decide on the type of model you want to use. This could be a regression model for predicting vaccination coverage or a classification model for other tasks like predicting vaccination success.

Feature Engineering:

If necessary, create new features or transform existing ones that may improve your model's performance.

Model Training:

Train your selected machine learning model on the training data. You can use libraries like scikit-learn for this purpose.Tune hyperparameters to optimize model performance. This may involve techniques like cross-validation and grid search.

Model Evaluation:

Evaluate the model's performance on the testing dataset using appropriate metrics. For regression tasks, common metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). For classification tasks, you can use accuracy, precision, recall, F1-score, etc.

Visualization and Interpretation:

Visualize the results and create informative plots or charts to help you understand the model's predictions and the dataset.

Model Deployment (Optional):

If you plan to use the model in a real-world application, you can deploy it using frameworks like Flask or Django for web applications.

Documentation:

Keep thorough documentation of your code, data preprocessing steps, model architecture, and evaluation results.

Here are a few Python libraries that can be helpful for this process:

scikit-learn: A powerful library for machine learning in Python.

Pandas: For data manipulation and analysis.

Matplotlib and Seaborn: For data visualization.

Jupyter Notebook: An interactive development environment for data science and machine learning.Be sure to review the dataset's documentation and perform necessary data cleaning and exploration steps specific to the dataset you're working with. Model selection and evaluation will also depend on the specific goals of your analysis, such as predicting vaccination progress or understanding factors that affect it.

Step 1: Data Exploration and Preprocessing :

Load the dataset: Download the dataset from Kaggle and load it into your analysis environment (e.g.,Python with pandas).

Explore the data: Understand the structure of the dataset, including the columns and their meanings.

Check for missing data: Identify and handle missing values appropriately.

Data cleaning: Clean and preprocess the data as needed. This may involve data type conversion, renaming columns, and handling outliers.

Step 2: Descriptive Analysis :

Summarize the data: Provide basic statistics on the dataset, such as mean, median, standard deviation, and percentiles for relevant columns.Visualizations: Create visualizations to display the progress of COVID-19 vaccinations over time and across countries. Use tools like line charts, bar charts, and heatmaps to represent the data effectively.Identify trends: Analyze trends in vaccination rates, vaccination types, and coverage by country and date.

Step 3: Comparative Analysis :

Compare vaccination progress between countries: Identify countries that have made significant progress in vaccination and those that lag behind.

Vaccine types: Analyze the usage of different vaccine types and their effectiveness.

Calculate vaccination rates: Calculate the number of vaccine doses administered per 100 people for each country.

Group countries: Group countries by region or income level and compare their vaccination progress.

Step 4: Time Series Analysis :

Time series decomposition: Decompose the time series data to understand the underlying trends, seasonality, and irregular components.

Forecasting: Use time series forecasting techniques to predict future vaccination rates.

Step 5: Geospatial Analysis :

Plot vaccination progress on a map: Utilize geospatial visualization tools to create maps that show vaccination coverage by country.

Analyze geographical patterns: Identify regions where vaccination progress is higher or lower and explore potential reasons for these differences.

Step 6: Advanced Analysis :

Correlation analysis: Investigate correlations between vaccination progress and various factors like GDP, healthcare infrastructure, and COVID-19 cases.

Regression analysis: Build regression models to understand the factors that influence vaccination progress.

Step 7: Conclusion and Insights :

Summarize key findings: Provide a concise summary of your analysis, highlighting significant trends and insights.

Recommendations: Suggest recommendations or actions based on your analysis.

Limitations: Discuss any limitations in the data or analysis.

Step 8: Document Creation and Sharing :

Create a report or Jupyter Notebook that documents your analysis, findings, and insights.

Include code, visualizations, and explanations.

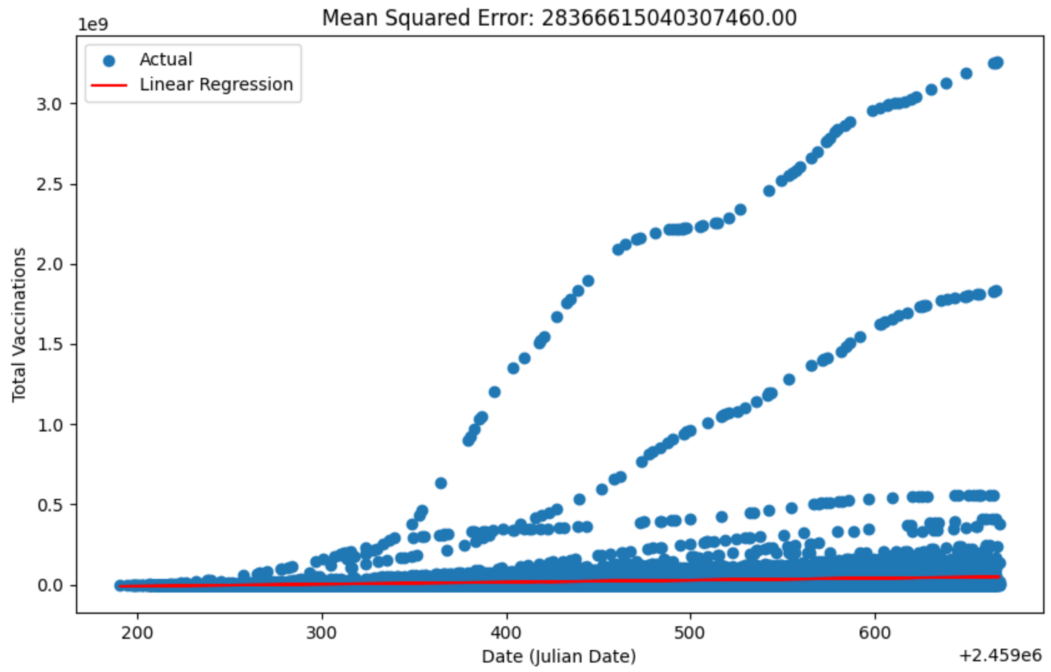
Share the document with relevant stakeholders or for assessment.

<u>Coding :</u>

```
# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt


# Load the dataset

    data_url = "https://raw.githubusercontent.com/datasets/gpreda/covid-19-world-
    vaccination-progress/master/country_vaccinations.csv"

    df = pd.read_csv(data_url)


# Data preprocessing

# For simplicity, we'll focus on a few columns

    df = df[['date', 'total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated']]

    df['date'] = pd.to_datetime(df['date'])
```

```
        df.set_index('date', inplace=True)

# Fill missing values with 0
        df.fillna(0, inplace=True)

# Split the data into features (X) and target (y)
        X = df.index.to_julian_date().values.reshape(-1, 1)
        y = df['total_vaccinations'].values

# Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model training
        model = LinearRegression()
        model.fit(X_train, y_train)
# Model evaluation
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)

# Plot the data and regression line
        plt.figure(figsize=(10, 6))
        plt.scatter(X_test, y_test, label='Actual')
        plt.plot(X_test, y_pred, color='red', label='Linear Regression')
        plt.xlabel('Date (Julian Date)')
        plt.ylabel('Total Vaccinations')
        plt.title(f'Mean Squared Error: {mse:.2f}')
        plt.legend()
        plt.show()
```

Output :


Mean Squared Error: 28366615040307460.00
1e9
Actual
Linear Regression
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Vaccinations
200
300
400
500
600
Date (Julian Date)
+2.459e6