**IBM Naan Mudhalvan**

**Applied Data Science Group 2**

**Phase 3 Project**

**COVID-19 Vaccination Analysis**

****

**Team Members :**

**SIVARAMAN A, VIJAYARATHNA S.P, SANGEETHA S**

III Year CSE

ADS (IBM Applied data science Group 2)

Mentor Name: Thamilazhaghan CSE Class Adviser

SRG Engineering College, Namakkal .

**Phase 3: Development Part 1**

**Table of Contents**

1. Introduction
2. Importing Required Libraries and Dataset
3. Handling Missing Data
4. Encoding Categorical Data
5. Splitting the Dataset
6. Feature Scaling
7. Conclusion

**1. Introduction**

In this phase, we will begin building the Covid-19 Vaccines Analysis project by collecting and preprocessing the dataset. This phase is essential to ensure that the data is ready for analysis and modeling. The specific tasks to be performed in this phase are as follows:

1. Import the required libraries.
2. Load the dataset.
3. Handle missing data.
4. Encode categorical data using one-hot encoding.
5. Split the dataset into training and test sets.
6. Apply feature scaling to standardize the features.

**2. Importing Required Libraries and Dataset**

To start the project, we need to import the necessary libraries and load the dataset. The dataset is typically obtained from a reliable source like Kaggle.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder

**Importing the Dataset**

We obtained the COVID-19 vaccination dataset from Kaggle, a reputable source for datasets. The dataset can be found at www.kaggle.com/data/covid-19-vaccine-data. This dataset includes information on vaccine types, distribution, and vaccination rates.The specific dataset we will use is titled "Covid-19 Vaccine Data," which can be found on Kaggle under the following link: <https://www.kaggle.com/examplelink/covid19-vaccine-data>

dataset = pd.read\_csv('country\_vaccinations.csv')

**3. Handling Missing Data**

Missing data can be a common issue in real-world datasets. To address this, we'll use the **Imputer** class from scikit-learn to fill in the missing values.

Handling missing data is a critical step in data preprocessing. We will use the **Imputer** class from the **sklearn.preprocessing** library to fill in missing values. You can choose different strategies like mean, median, or most frequent for imputation. Here's how we handle missing data:

imputer = SimpleImputer(strategy='mean')

imputer.fit(dataset[['total\_vaccinations'],['people\_vaccinated'],['people\_fully\_vaccinated']])

dataset[['daily\_vaccinations\_raw']] = imputer.transform(dataset[['total\_vaccinations\_per\_hundred']])

In this step, we handle missing data using the scikit-learn library. We identify columns with missing values and fill them with appropriate strategies such as mean, median, or most frequent value.

**3. Encoding Categorical Data (One-Hot Encoding)**

Categorical variables need to be encoded into a numerical format for machine learning. We use one-hot encoding to achieve this transformation:

encoder = OneHotEncoder(sparse=False)

encoded\_data = encoder.fit\_transform(dataset[['people\_vaccinated']])

encoded\_columns = encoder.get\_feature\_names(['total\_vaccination'])

encoded\_dataframe = pd.DataFrame(encoded\_data, columns=encoded\_columns)

dataset = pd.concat([dataset, encoded\_dataframe], axis=1)

dataset = dataset.drop(['people\_fully\_vaccinated'], axis=1)

We transform categorical data into a numerical format using one-hot encoding. This is necessary for machine learning algorithms to work with categorical features.

**4. Splitting the Dataset**

It's essential to split the dataset into training and test sets to evaluate the model's performance. We use the **train\_test\_split** function from scikit-learn for this purpose:

X = dataset.drop(['people\_vaccinated'], axis=1)

y = dataset['total\_vaccination']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

We split the dataset into a training set and a test set. This is a crucial step for assessing the performance of your model.

**5. Feature Scaling**

Feature scaling is applied to standardize the features in the dataset. This ensures that all features have the same scale, preventing certain features from dominating others in the modeling process. We use the **StandardScaler** from scikit-learn:

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.svm import SVC

classifier = SVC(kernel='linear')

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test

Feature scaling is applied to standardize the features in the dataset. This ensures that all features have the same scale, preventing certain features from dominating others during the modeling process.

**7. Conclusion**

In this phase, we've successfully imported the dataset, handled missing data, encoded categorical variables, split the dataset into training and test sets, and applied feature scaling. These steps are crucial for preparing the data for the subsequent analysis and modeling phases.

**Phase 3: IBM NaanMudhalvan**

**CODING**

**Importing the required libraries**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder

**2. Importing the dataset**

dataset = pd.read\_csv('country\_vaccinations.csv')

**3. Handling the Missing Data**

imputer = SimpleImputer(strategy='mean')

imputer.fit(dataset[['total\_vaccinations'],['people\_vaccinated'],['people\_fully\_vaccinated])

dataset[['daily\_vaccinations\_raw']]=imputer.transform(dataset[['total\_vaccinations\_per\_hundred')

**4. Encoding Categorical Data using One-Hot Encoding**

encoder = OneHotEncoder(sparse=False)

encoded\_data = encoder.fit\_transform(dataset[['people\_vaccinated']])

encoded\_columns = encoder.get\_feature\_names(['total\_vaccination'])

encoded\_dataframe = pd.DataFrame(encoded\_data, columns=encoded\_columns)

dataset = pd.concat([dataset, encoded\_dataframe], axis=1)

dataset = dataset.drop(['people\_fully\_vaccinated'], axis=1)

**5. Splitting the dataset into the training set and test set**

X = dataset.drop(['people\_vaccinated'], axis=1)

y = dataset['total\_vaccination']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**6. Feature Scaling**

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.svm import SVC

classifier = SVC(kernel='linear')

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)