**COVID Vaccines Analysis**

**Phase 2 Submission Document**

**Project:** COVID Vaccines Analysis



**Introduction:**

* The "COVID Vaccines Analysis" project is a comprehensive exploration of critical aspects related to COVID-19 vaccines. This project delves into the data surrounding vaccine distribution, administration, and the occurrence of adverse effects, aiming to provide insights that contribute to informed decision-making and public health strategies.
* Our analysis encompasses a wide range of methodologies, from data mining and machine learning techniques to statistical modelling and trend analysis. By leveraging advanced data analytics, we seek to uncover hidden patterns, identify areas of concern, and make meaningful predictions regarding the trajectory of the pandemic and the success of vaccination efforts.

**Content for Project Phase 2:**

* Exploring advanced machine learning techniques like clustering and time series forecasting can be instrumental in uncovering hidden patterns in vaccine distribution and adverse effects data. Here's a more detailed explanation of how you can utilize these techniques.

**Advanced Machine Learning Techniques:**

**Clustering Analysis:**

**Objective**: Clustering analysis groups similar data points together to reveal patterns and relationships within the data. In the context of your project on vaccine distribution and adverse effects, clustering can help identify geographical, temporal, or demographic clusters with similar characteristics.

**Implementation**:

* **Feature Selection**: Choose relevant features from your dataset, such as location (e.g., countries, states), time (e.g., months, weeks), vaccine types, and adverse effects.
* **Data Preprocessing**: Clean and preprocess your data to handle missing values, outliers, and normalize data if necessary.
* **Algorithm Selection**: Utilize clustering algorithms like K-Means, Hierarchical Clustering, or DBSCAN, depending on the nature of your data and the desired cluster shape.
* **Clustering Interpretation**: Analyze and interpret the clusters. For instance, you might discover clusters of regions with similar vaccination rates, clusters of time periods with similar adverse effects, or groups of people with similar vaccine response patterns.
* **Visualization**: Create visualizations, such as heatmaps or geographic maps, to illustrate the clusters and make it easier to communicate your findings.

**Time Series Forecasting:**

**Objective**: Time series forecasting involves predicting future values based on historical data. In the context of your project, time series forecasting can help in predicting future vaccine distribution trends and identifying potential outbreaks of adverse effects.

**Implementation**:

* **Data Preparation**: Organize your data into time series format with timestamps and relevant variables (e.g., vaccine distribution counts or adverse effect occurrences).
* **Train-Test Split**: Split your time series data into training and testing sets to evaluate the accuracy of your forecasting model.
* **Model Selection**: Choose an appropriate time series forecasting model like ARIMA (AutoRegressive Integrated Moving Average), Exponential Smoothing, or Long Short-Term Memory (LSTM) networks for deep learning.
* **Hyperparameter Tuning**: Fine-tune the model hyperparameters to optimize performance.
* **Forecasting and Evaluation**: Train the model on the training data and use it to make predictions for the test set. Evaluate the model's accuracy using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE).
* **Visualization**: Visualize the predicted values alongside the actual data to assess the model's performance and identify trends or anomalies.

**Program**

IN[1]: import numpy as np

IN[2]: import pandas as pd

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

IN[3]: df = pd.read\_csv("/kaggle/input/covid-world-vaccination-progress/country\_vac cinations\_by\_manufacturer.csv")

IN[4]: df.head(10)

OUT[4]:

|  |  |  |  |
| --- | --- | --- | --- |
| location | date | vaccine | total\_vaccinations |
| 0 | Argentina | 2020-12-29 | Moderna | 2 |
| 1 | Argentina | 2020-12-29 | Oxford/AstraZeneca | 3 |
| 2 | Argentina | 2020-12-29 | Sinopharm/Beijing | 1 |
| 3 | Argentina | 2020-12-29 | Sputnik V | 20481 |
| 4 | Argentina | 2020-12-30 | Moderna | 2 |
| 5 | Argentina | 2020-12-30 | Oxford/AstraZeneca | 3 |
| 6 | Argentina | 2020-12-30 | Sinopharm/Beijing | 1 |
| 7 | Argentina | 2020-12-30 | Sputnik V | 40583 |
| 8 | Argentina | 2020-12-31 | Moderna | 2 |
| 9 | Argentina | 2020-12-31 | Oxford/AstraZeneca | 3 |

IN[5]: df["location"].nunique()

OUT[5]: 43

IN[6]: df.isnull().sum()

OUT[6]:

location 0

date 0

vaccine 0

total\_vaccinations 0

dtype: int64

IN[7]: df.dtypes

OUT[7]:

location object

date object

vaccine object

total\_vaccinations int64

dtype: object

IN[8]: df['date'] = pd.to\_datetime(df['date'])

IN[9]: data=pd.DataFrame(columns=['Country', 'Vaccine', 'Total\_vaccine'])

for country **in** df["location"].unique():

for vaccine **in** df["vaccine"].unique():

filtered\_data = df[(df['location'] == country) & (df['vaccine'] == vaccine)]

total\_count = filtered\_data['total\_vaccinations'].max()

data = pd.concat([data, pd.DataFrame({'Country': [country], 'Vaccine': [vaccine], 'Total\_vaccine': [total\_count]})], ignore\_index=True)

IN[10]: data.head(10)

OUT[10]:

|  |  |  |
| --- | --- | --- |
| Country | Vaccine | Total\_vaccine |
| 0 | Argentina | Moderna | 6507561 |
| 1 | Argentina | Oxford/AstraZeneca | 25977231 |
| 2 | Argentina | Sinopharm/Beijing | 28322602 |
| 3 | Argentina | Sputnik V | 20405678 |
| 4 | Argentina | CanSino | 610540 |
| 5 | Argentina | Pfizer/BioNTech | 14681054 |
| 6 | Argentina | Johnson&Johnson | NaN |
| 7 | Argentina | Novavax | NaN |
| 8 | Argentina | Sinovac | NaN |
| 9 | Argentina | Covaxin | NaN |

IN[11]:data.dropna(axis=0,inplace=True)

In [12]:

IN[12]:data.head(20)

OUT[12]:

|  |  |  |
| --- | --- | --- |
| Country | Vaccine | Total\_vaccine |
| 0 | Argentina | Moderna | 6507561 |
| 1 | Argentina | Oxford/AstraZeneca | 25977231 |
| 2 | Argentina | Sinopharm/Beijing | 28322602 |
| 3 | Argentina | Sputnik V | 20405678 |
| 4 | Argentina | CanSino | 610540 |
| 5 | Argentina | Pfizer/BioNTech | 14681054 |
| 10 | Austria | Moderna | 1585063 |
| 11 | Austria | Oxford/AstraZeneca | 1588222 |
| 15 | Austria | Pfizer/BioNTech | 14584985 |
| 16 | Austria | Johnson&Johnson | 363548 |
| 17 | Austria | Novavax | 3682 |
| 20 | Belgium | Moderna | 4267394 |
| 21 | Belgium | Oxford/AstraZeneca | 2846716 |
| 25 | Belgium | Pfizer/BioNTech | 17451842 |
| 26 | Belgium | Johnson&Johnson | 425639 |
| 27 | Belgium | Novavax | 36 |
| 30 | Bulgaria | Moderna | 491663 |
| 31 | Bulgaria | Oxford/AstraZeneca | 478541 |
| 35 | Bulgaria | Pfizer/BioNTech | 2852218 |
| 36 | Bulgaria | Johnson&Johnson | 511702 |

IN[13]: data\_2=pd.DataFrame(columns=['Country', 'Vaccine'])

data["Total\_vaccine"] = pd.to\_numeric(data["Total\_vaccine"], errors="coerce")

for country **in** data["Country"].unique():

new\_data = data[data["Country"] == country]

max\_vaccine = new\_data.loc[new\_data["Total\_vaccine"].idxmax(), "Vaccine"]

data\_2 = pd.concat([data\_2, pd.DataFrame({'Country': [country], 'Vaccine': [max\_vaccine]})], ignore\_index=True)

IN[14]: data\_2.head()

OUT[14]:

Out[14]:

|  |  |  |
| --- | --- | --- |
|  | Country | Vaccine |
| 0 | Argentina | Sinopharm/Beijing |
| 1 | Austria | Pfizer/BioNTech |
| 2 | Belgium | Pfizer/BioNTech |
| 3 | Bulgaria | Pfizer/BioNTech |
| 4 | Chile | Sinovac |

IN[15]:data\_2["Vaccine"].value\_counts().plot(kind="bar",

color=["Red","Gray","Gray","Gray"])

OUT[15]:

<Axes: >



IN[16]:number\_of\_days = (df["date"].max() -df["date"].min() ).days

In [17]:

dtfrm=data[data["Vaccine"]=="Pfizer/BioNTech"]

dtfrm = dtfrm.drop(dtfrm[dtfrm['Country'] == 'European Union'].index)

In [18]:

dtfrm.head(10)

dtfrm.head(10)

Out[18]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Country | Vaccine | Total\_vaccine |
| 5 | Argentina | Pfizer/BioNTech | 14681054 |
| 15 | Austria | Pfizer/BioNTech | 14584985 |
| 25 | Belgium | Pfizer/BioNTech | 17451842 |
| 35 | Bulgaria | Pfizer/BioNTech | 2852218 |
| 45 | Chile | Pfizer/BioNTech | 7910264 |
| 55 | Croatia | Pfizer/BioNTech | 3921503 |
| 65 | Cyprus | Pfizer/BioNTech | 1188656 |
| 75 | Czechia | Pfizer/BioNTech | 14604323 |
| 85 | Denmark | Pfizer/BioNTech | 10259219 |
| 95 | Ecuador | Pfizer/BioNTech | 8552679 |

In [19]:

dtfrm["average\_vaccination\_count"] = dtfrm["Total\_vaccine"] / number\_of\_days

dtfrm["average\_vaccination\_count"] =dtfrm["average\_vaccination\_count"].astype(int)

In [20]:

dtfrm.head(15)

IN[21]:

color=["Lightblue","Purple","Green","Orange","darkgoldenrod","tan","Gray","Blue","Pink","Lightgreen"]

dtfrm["average\_vaccination\_count"].sort\_values(ascending=False).head(10).plot(kind="bar",color=color)

Out[22]:

<Axes: xlabel='Country'>



**Number of countries where vaccines are used**

In [23]:

number\_of\_vaccines = data.groupby('Vaccine')['Country'].nunique()

In [24]:

number\_of\_vaccines.sort\_values(ascending=False).plot(kind="bar",color="r")

Out[24]:

<Axes: xlabel='Vaccine'>



IN[25]:

fig = px.choropleth(data\_frame=dtfrm,

locations=dtfrm.index,

locationmode='country names',

color='Total\_vaccine',

color\_continuous\_scale='YlOrRd',

title='Ülkelerde Yapılan Biontech Aşıları')

fig.update\_layout(title\_x=0.5)

IN[26]:

m = folium.Map(location=[0, 0], zoom\_start=2)

Choropleth(

geo\_data='https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json',

name='choropleth',

data=dtfrm,

columns=[dtfrm.index, 'Total\_vaccine'],

key\_on='feature.properties.name',

fill\_color='YlOrRd',

fill\_opacity=0.7,

line\_opacity=0.2,

legend\_name='Aşı Sayısı',

).add\_to(m)

Out[26]:

<folium.features.Choropleth at 0x7d414f2b7430>