Vaccination Period Prediction

Final Project of Data Science

Iman Fahru Syuhada (JCDS-1104)

Background

To help understand how many days needed to vaccinate of 80% total population with the available data

PROBLEMS

Inequity of Vaccines distribution

between developed countries

and non-developed countries

Days required to vaccinate

80% of population

IMPACTS

Non-developed countries were able to order the vaccines in lower quantity due to unavailable of vaccines. Sometimes, they have the leftovers from canceled transactions. Late start of vaccinations that led to more deaths.

GOALS

- **Find** the timeline of vaccination
- **Analyze** the vaccination per 100, 1 mio and Daily Vaccination
- **Build Machine Learning model** to predict duration of how much time needed to vaccinate 80% of total population in each country

METHODS

Data Cleansing

Redundant data removal

Outliers handling

Dashboard

Show data sample

Display graphs

Perform prediction

EDA

Extract insights

Prepare appropriate
data for ML Modeling

Machine Learning

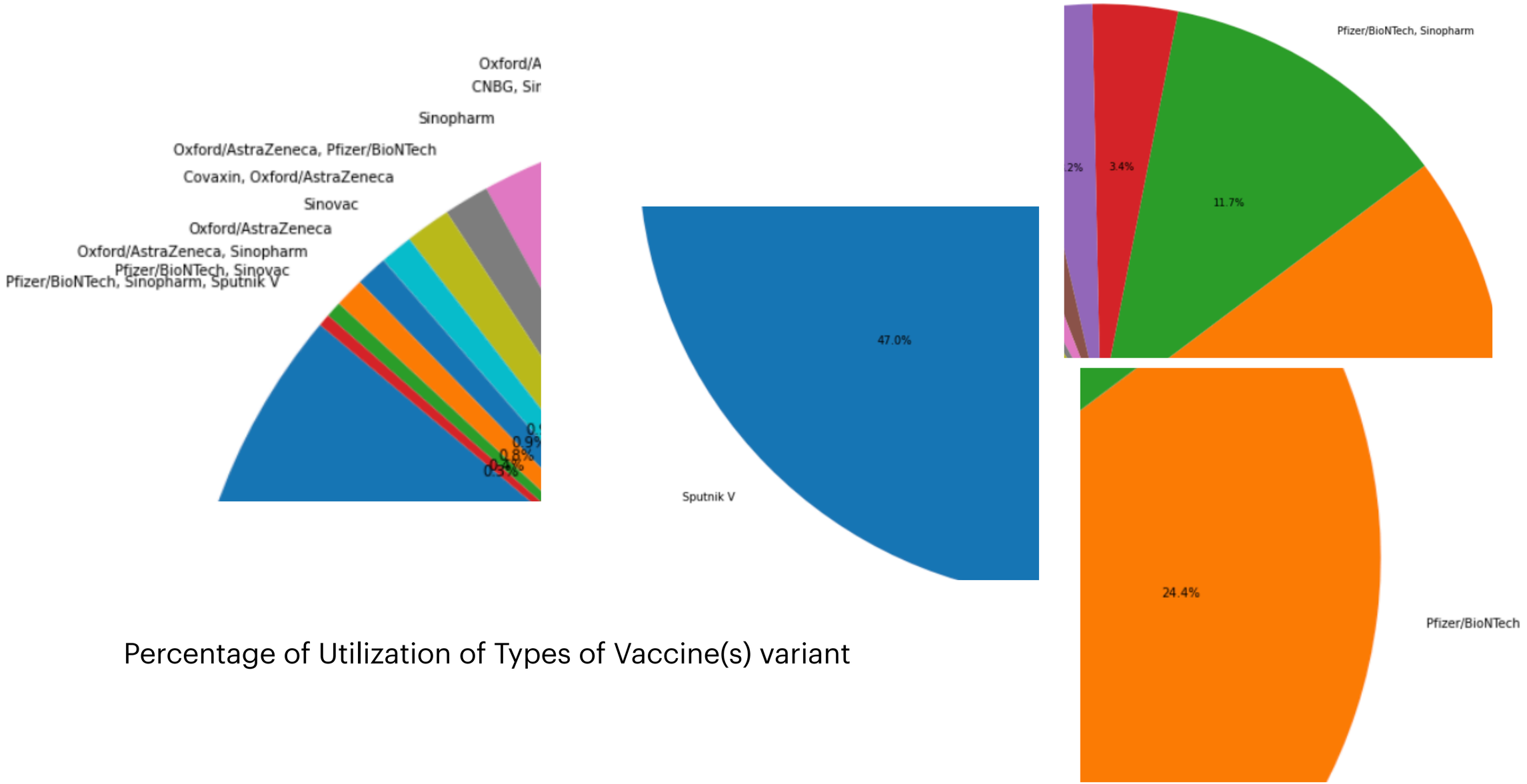
Auto Regression Integrated Moving Average(ARIMA)

TimeSeries

Literature Review

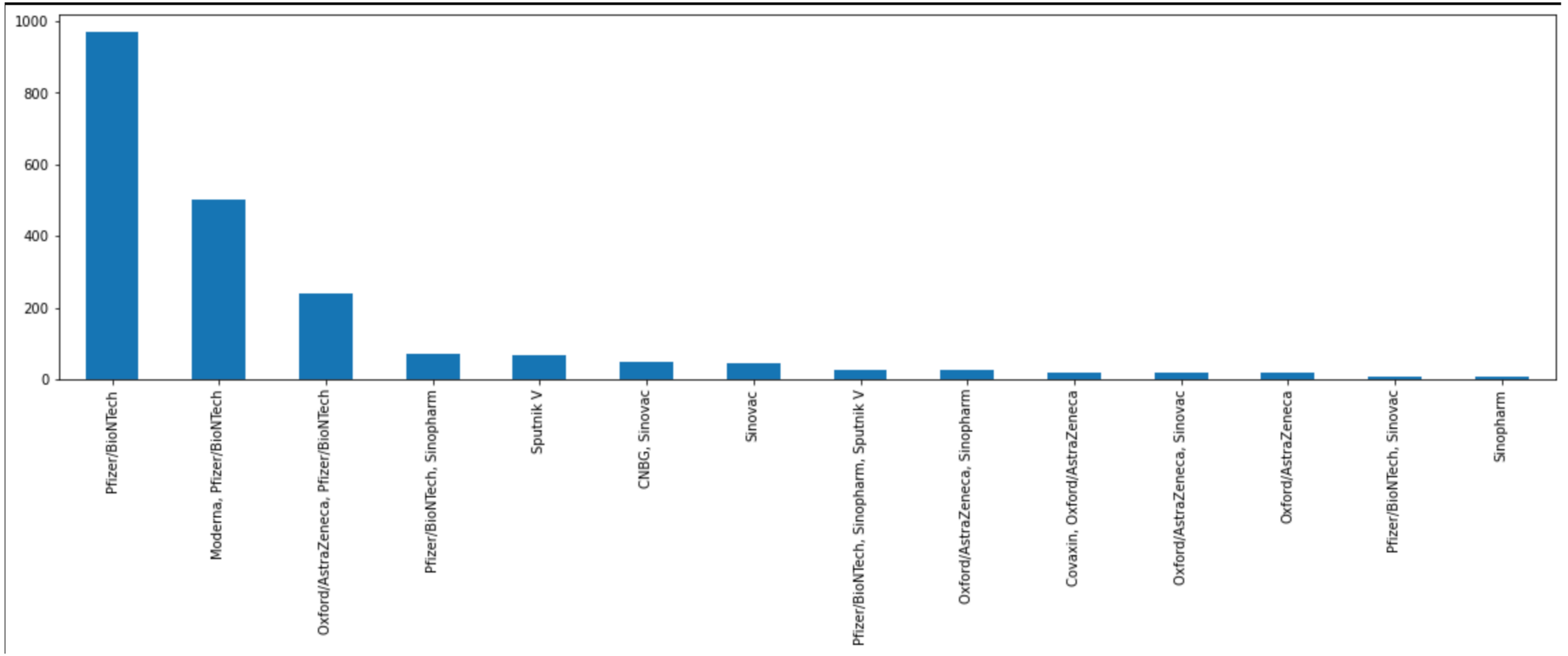
- Prediction in each country based on 80% of total population
- Prediction options: Average of Daily Vaccinations, Vaccinations per 100 and 1,000,000 people
- Utilisation of different types of vaccine(s) variant

Exploratory Data Analysis



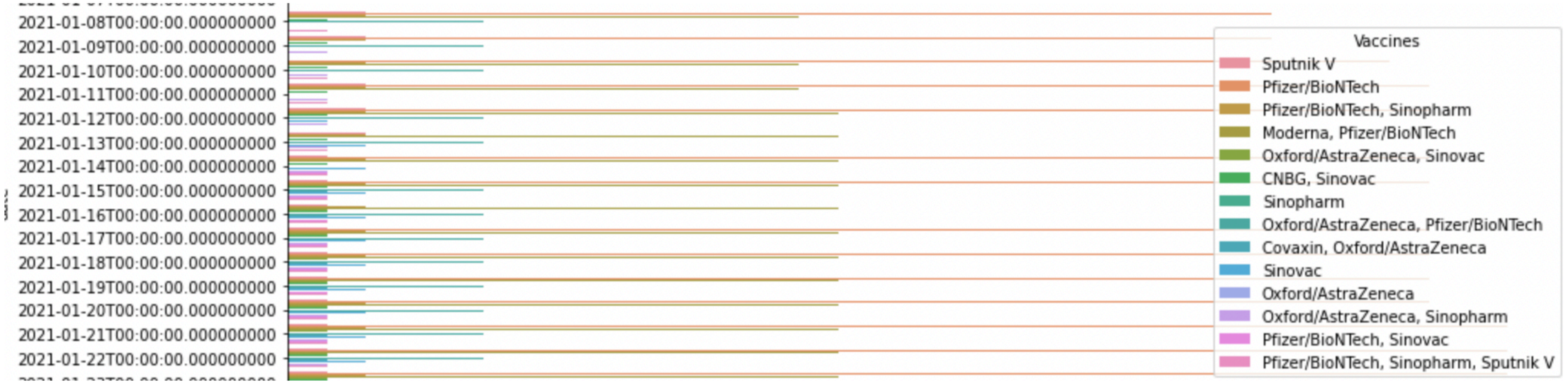
Percentage of Utilization of Types of Vaccine(s) variant

Data Cleansing



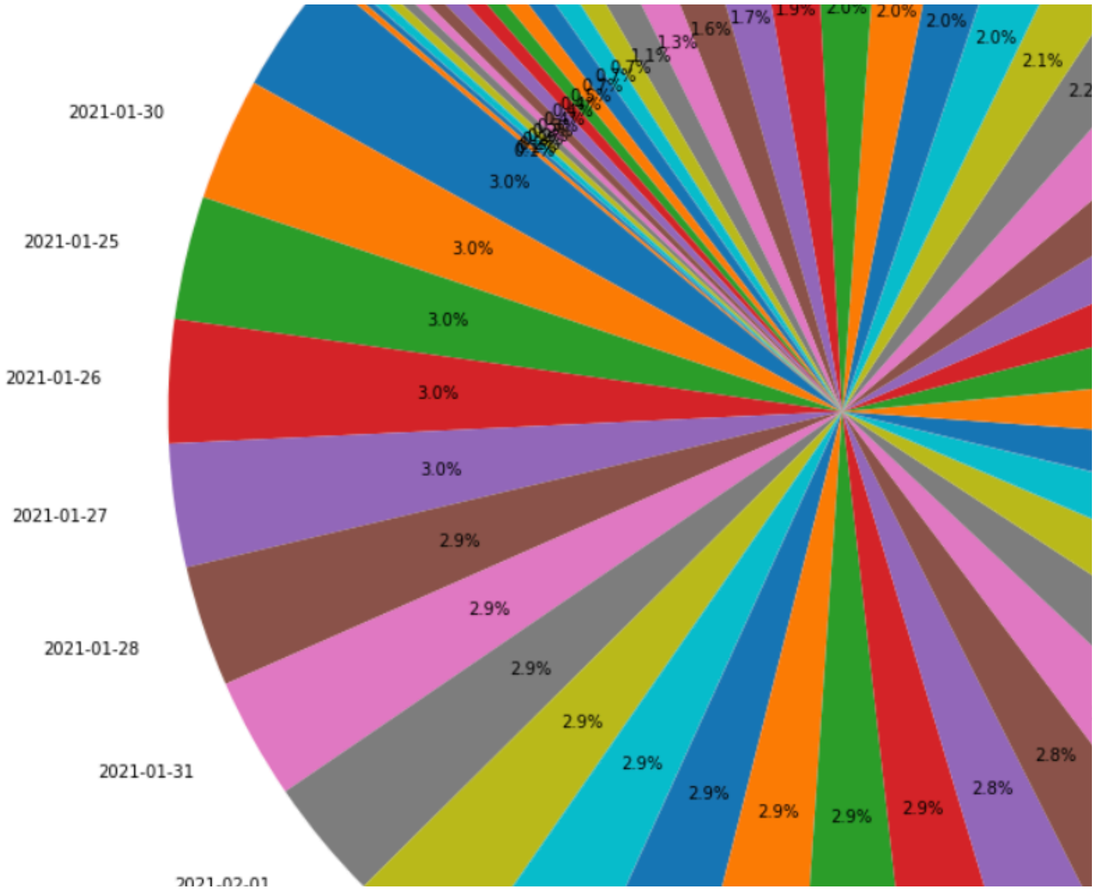
Utilisations of Vaccine(s) variant

DATA CLEANSING



Utilisation of Vaccine(s) variant by DateTime

EXPLORATORY DATA ANALYSIS



Highest percentage of Vaccination by DateTime

EXPLORATORY DATA ANALYSIS

:

	country	date	Total Vaccinations	Daily Vaccinations	People Vaccinated 100	Daily Vaccination 1 mio	Vaccines
0	Algeria	2021-01-29	0.000	0.000	0.000	0.000	Sputnik V
1	Algeria	2021-01-30	30.000	30.000	0.000	1.000	Sputnik V
2	Andorra	2021-01-25	576.000	0.000	0.750	0.000	Pfizer/BioNTech
3	Andorra	2021-01-26	0.000	66.000	0.000	854.000	Pfizer/BioNTech
4	Andorra	2021-01-27	0.000	66.000	0.000	854.000	Pfizer/BioNTech
...	...	...	...	...	...	...	...
2059	Wales	2021-01-28	362970.000	21463.000	11.490	6807.000	Oxford/AstraZeneca, Pfizer/BioNTech
2060	Wales	2021-01-29	378950.000	19705.000	12.000	6250.000	Oxford/AstraZeneca, Pfizer/BioNTech
2061	Wales	2021-01-30	404249.000	19885.000	12.800	6307.000	Oxford/AstraZeneca, Pfizer/BioNTech
2062	Wales	2021-01-31	417147.000	20824.000	13.200	6605.000	Oxford/AstraZeneca, Pfizer/BioNTech
2063	Wales	2021-02-01	440706.000	21508.000	13.940	6822.000	Oxford/AstraZeneca, Pfizer/BioNTech

2064 rows × 7 columns

Final data to be utilized

Machine Learning

[67]:
```
# to perform time series analysis (TSA), we need to get the whole month data
# some countries only have data for several days, so we plan to remove them from the dataset

# listing all countries
countries = df['country'].unique()

# counting the number of data in each country
num = []
for i in countries:
    a = len(df[df["country"]==i])
    num.append(a)

# creating dataframe that contain country and number of data
dataNum = pd.DataFrame(list(zip(countries, num)), columns = ["country", "total data"])
dataNum

# removing country with data less than 30 rows
sufCountry = dataNum[dataNum['total data']>=30]
print(f"There are {len(sufCountry)} countries that has sufficient data.")

# sort the data
sufCountry.sort_values(by=['total data', 'country'], ascending=[False, True]).head(10)

# Northern Ireland, Scotland, and Wales has the most data.
# We can use them to build  time series analysis model
```

```
There are 40 countries that has sufficient data.
```

[67]:

	country	total data
48	Northern Ireland	51
57	Scotland	51
71	Wales	51
11	China	48
33	Israel	46
70	United States	45
19	England	44
69	United Kingdom	44
4	Bahrain	42
10	Chile	41

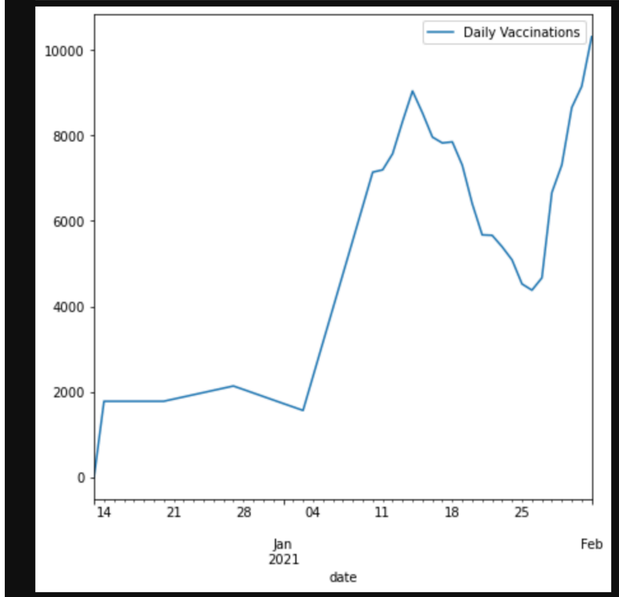
Preparing North Ireland (NIRL) Dataset for ARIMA

- Setting date as inde

]:
```
NIRL.set_index('date', inplace=True)
```

- Graphing value of daily vaccination over time

]:
```
NIRL = NIRL[['Daily Vaccinations']]
NIRL.plot(figsize=(7,7))
plt.show()
```



Filtration of countries with sufficient data (30 days)

Machine Learning

[71]:
```
# checking stationarity
# stationarity means that the statistical properties of a a time series (or rather the process generating it) do not change over time

# creating user-defined function
from statsmodels.tsa.stattools import adfuller
def ad_test(dataset):
    dftest = adfuller(dataset, autolag='AIC')
    print("ADF : ", dftest[0])
    print("P-Value : ", dftest[1])
    print("# Lags : ", dftest[2])
    print("# Observation : ", dftest[3])
    print("Critical Value : ")
    for key, val in dftest[4].items():
        print("\t", key, "= ", val)
# source: https://www.youtube.com/watch?v=8FCDpFhd1zk&ab_channel=NachiketaHebbar

ad_test(NIRL['Daily Vaccinations'])
# P-Value is larger than critical value (means the data is not stationary)
```

```
ADF :  -0.9703564408607157
P-Value :  0.7639220128663259
# Lags :  7
# Observation :  43
Critical Value :
        1% =  -3.5925042342183704
        5% =  -2.931549768951162
        10% =  -2.60406594375338
```

Checking stationarity

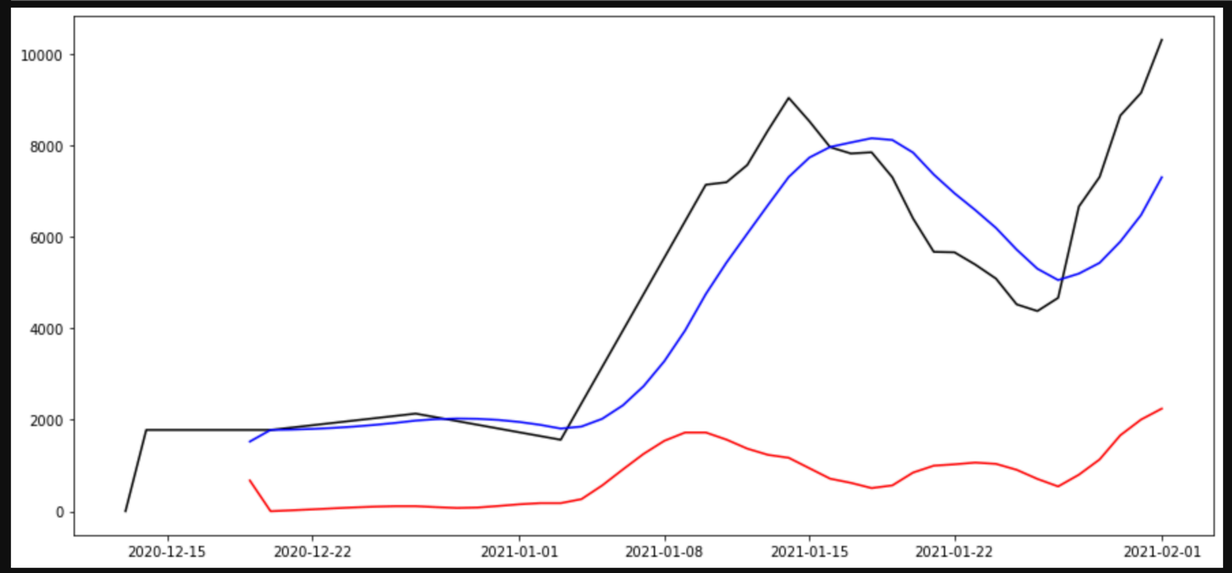
Stationarity means that the statistical properties of a a time series (or rather the process generating it) do not change over time

Machine Learning

[72]:
```
# calculating rolling mean and stdev
rollmean = NIRL.rolling(window=7).mean()
rollstd = NIRL.rolling(window=7).std()
# window is 7 to see the movement weekly

# plotting 'daily vaccination' value
plt.figure(figsize=(15,7))
plt.plot(NIRL['Daily Vaccinations'], color='black', label='original')
plt.plot(rollmean, color='blue', label='mean')
plt.plot(rollstd, color='red', label='stdev')
plt.show()

# rolling mean and stdev tend to increase over time
```



To check the Mean and StDev Trend of Daily Vaccination

Machine Learning

Fitting using ARIMA

```
3]: from pmdarima import auto_arima
    import warnings
    warnings.filterwarnings('ignore')
```

- Fitting the Data

```
4]: stepwiseFit = auto_arima(NIRL['Daily Vaccinations'], trace=True, suppress_warning=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=768.642, Time=0.13 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=783.951, Time=0.15 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=769.402, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=776.293, Time=0.03 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=787.712, Time=0.00 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=769.741, Time=0.07 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=767.919, Time=0.06 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=770.391, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=770.105, Time=0.05 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=769.373, Time=0.08 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=771.441, Time=0.06 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.12 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=768.008, Time=0.04 sec

Best model:  ARIMA(2,1,1)(0,0,0)[0] intercept
Total fit time: 0.867 seconds
```

```
[75]: stepwiseFit.summary()
```

[75]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	51
Model:	SARIMAX(2, 1, 1)	**Log Likelihood**	-378.960
Date:	Tue, 02 Mar 2021	**AIC**	767.919
Time:	10:12:06	**BIC**	777.479
Sample:	0	**HQIC**	771.560
	- 51		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	177.3837	166.625	1.065	0.287	-149.196	503.963
ar.L1	-0.2883	0.176	-1.643	0.100	-0.632	0.056
ar.L2	0.6573	0.122	5.370	0.000	0.417	0.897
ma.L1	0.8181	0.268	3.057	0.002	0.294	1.343
sigma2	2.168e+05	4.14e+04	5.242	0.000	1.36e+05	2.98e+05

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	18.85
Prob(Q):	0.95	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.61	**Skew:**	0.88
Prob(H) (two-sided):	0.34	**Kurtosis:**	5.43

Fitting ARIMA and Summary

Machine Learning

Training and Testing Data

[76]:
```
from statsmodels.tsa.arima_model import ARIMA, ARIMAResults
```

- Splitting the data into train set and test set

[77]:
```
train = NIRL.iloc[:-10]
test = NIRL.iloc[-10:]
```

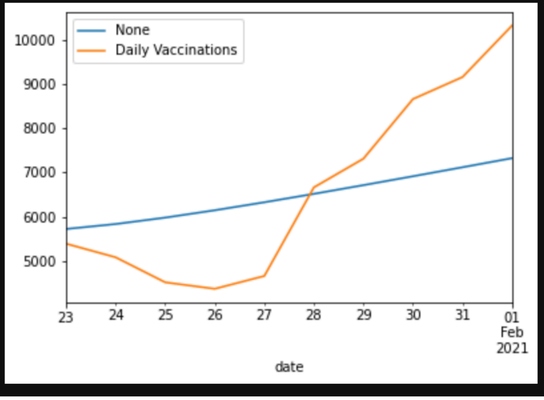
- Train set fitting

[78]:
```
model = ARIMA(train['Daily Vaccinations'], order=(1,1,0))
model = model.fit()
```

- Prediction making

[79]:
```
start = len(train)
end = len(train)+len(test)-1
pred = model.predict(start=start, end=end, typ='levels')
```

[80]:
```
# compare prediction to test set
pred.plot(legend='Prediction')
test['Daily Vaccinations'].plot(legend='Orig')
plt.show()
```



Training and Testing Data

Machine Learning

[81]:
```
# error analysis

#import libraries
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

# calculating MSE and RMSE
print('MSE = ', mean_squared_error(pred, test['Daily Vaccinations']))
print('RMSE = ', sqrt(mean_squared_error(pred, test['Daily Vaccinations'])))
print('R2 Score = ', r2_score(pred, test['Daily Vaccinations']))
```

```
MSE =  2517437.916782179
RMSE =  1586.643601059223
R2 Score =  -8.25570164608901
```

[82]:
```
# saving model
model = ARIMA(train['Daily Vaccinations'], order=(1,1,0))
model_fit = model.fit()
model_fit.save('arima.pkl')
```

Error Analysis, the ML is still weak, I need more time to optimize the model (RMSE numbers are still thousands)

Machine Learning

- Setting date as index

[115]:
```
df.set_index('date', inplace=True)
```

- User will input country, so we should make it as variable
- Assume we use Malta as input

[116]:
```
country = 'Italy'
```

- Creating dataframe containing data from a certain country

[117]:
```
data = df[df['country']==country]
data.head(5)
```

[117]:

date	country	Daily Vaccinations
2020-12-27	Italy	0.0
2020-12-28	Italy	948.0
2020-12-29	Italy	927.0
2020-12-30	Italy	2150.0
2020-12-31	Italy	7892.0

Machine Learning testing, country = Italy

Machine Learning

Checking The Performance

[119]:
```
import warnings
warnings.filterwarnings('ignore')
```

- Splitting train and test set

[120]:
```
train = data.iloc[:-10]
test = data.iloc[-10:]
```

- Training the data

[121]:
```
model = ARIMA(train['Daily Vaccinations'], order=(0,1,0))
model = model.fit()
```

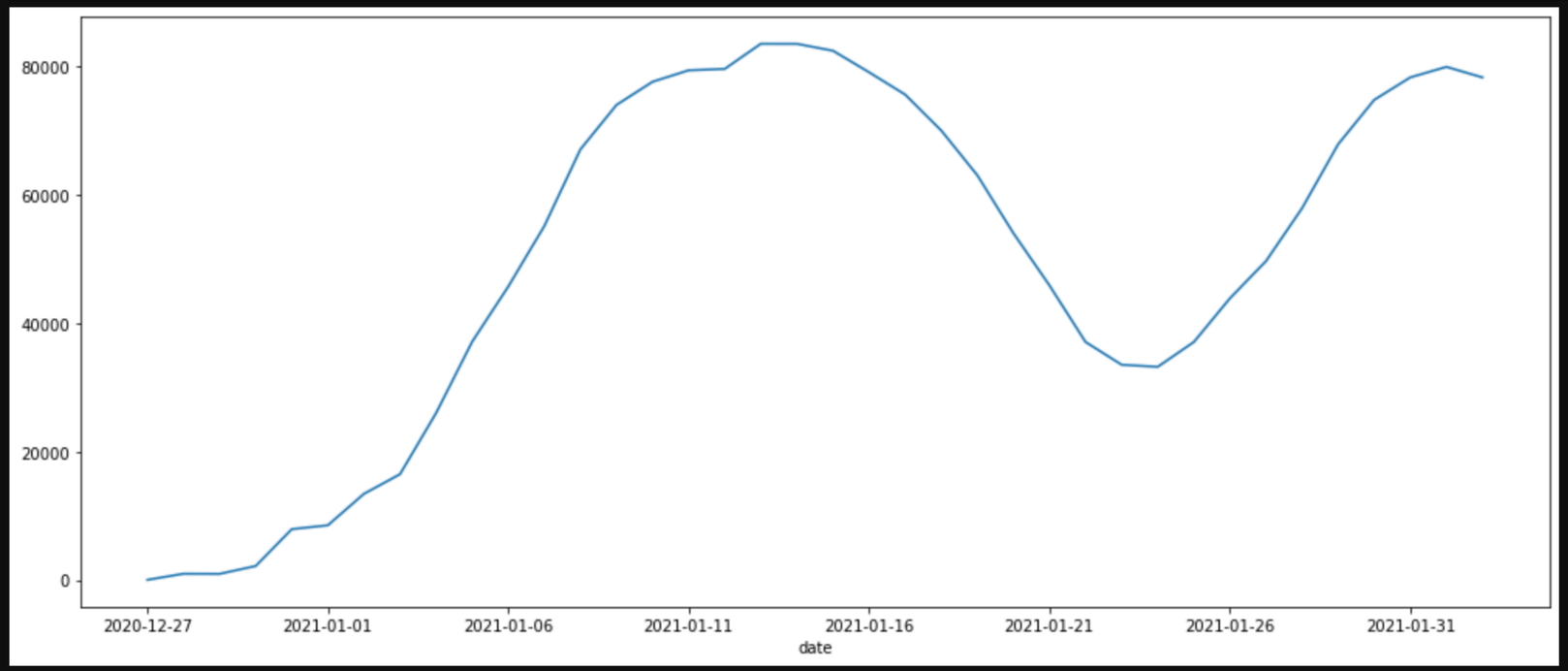
- Making Prediction

[122]:
```
start = len(train)
end = len(train)+len(test)-1
pred = model.predict(start=start, end=end, typ='levels')
```

Checking the performance

Machine Learning

```
[118]: # plotting the data
       data['Daily Vaccinations'].plot(figsize=(17,7))
       plt.show()
```



Daily Vaccination Progress

Machine Learning

Performing Machine Learning

- Etraxting data of country population

[126]:
```
population = pd.read_csv('02 - Country.csv')
citizenNum = population[population['Country (or dependency)']==country].iloc[0][1]
```

[127]:
```
# providing variables
history = list(data['Daily Vaccinations'])
vaccinated = data['Daily Vaccinations'].sum()
day = 0

# looping to find number of vaccinated citizen
while vaccinated < 80/100*citizenNum:
    model = ARIMA(history, order=(0,1,0))
    model = model.fit()
    start = len(history)
    pred = model.predict(start=start, end=start, typ='levels')
    vaccinated += int(pred[0])
    history.append(int(pred[0]))
    day += 1
```

- How many days we need to vaccinate at least 80% of population?

[128]:
```
day
```

[128]: 176

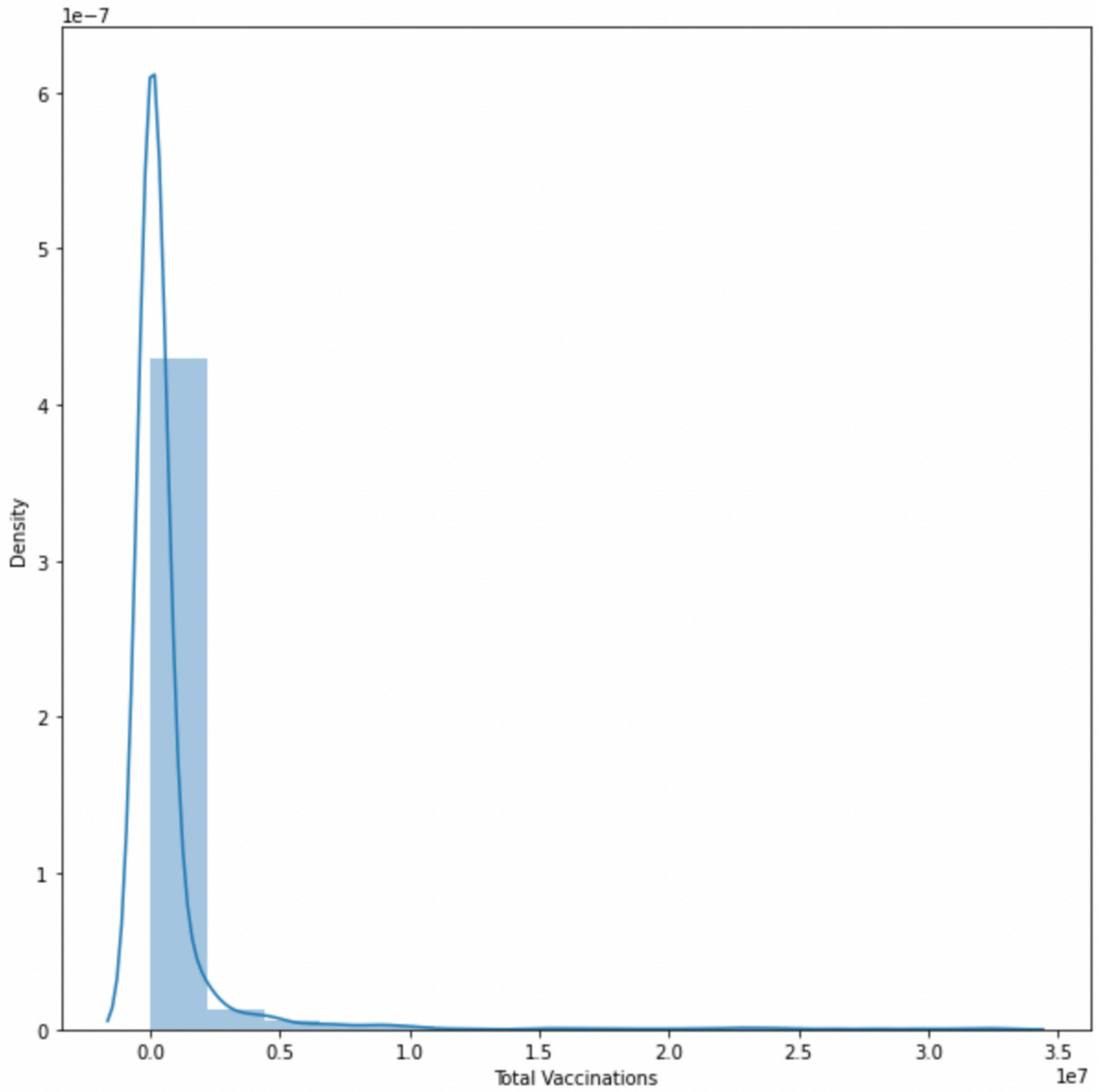
Final Machine Learning Input

CONCLUSION

- Only a few countries are able to be predicted thanks to it’s sufficient data.
- **From the statement above only**, we can assume that it means that other countries will suffer from the pandemic longer.
- Sputnik V made by Russia has the highest percentage due to it’s unfinished third clinical trial instead they advanced the utilization of the vaccine without proper data.

CONCLUSION

- Many non-developed countries are at the level of 0 for total vaccinations from average of 30 days due to it’s unavailability to purchase the vaccines



100%
RECOMMENDATION

The world needs to work together to put the pandemic to an end, none will survive if we don’t vaccinate all human across the continent.

If we don’t vaccinate everyone, new mutations such as South African and Brazilian variant of the virus will only accelerate the numbers of deaths globally.

Lastly, wear a mask and get tested! #itsaveslives