Data Science

Time Series Forecasting of Covid-19 cases + EDA

Lucas Steinmetz

RIS

Table of Contents

Intro

The COVID-19 pandemic has had a profound effect on the planet, leaving an influence on people's everyday lives all across the world. The virus has significantly disrupted social, economic, and political institutions and caused widespread disease and death. In this situation, it is crucial to comprehend the pandemic's propagation and effects. We will be concentrating on the COVID-19 instances in the United States, Italy, and the United Kingdom in this paper. We will investigate the trends and patterns in the COVID-19 cases in these three nations using exploratory data analysis (EDA) and time series forecasting. Our investigation will provide light on the dynamics of the pandemic in many areas and be helpful in determining the course of the pandemic and formulating sensible countermeasures.

Description of Data

The COVID-19 dataset from Our World in Data (OWID) is a thorough and often updated dataset that offers a wealth of information about the COVID-19 pandemic. It is one of the most frequently used sources of COVID-19 data by academics, decision-makers, and the general public. For most nations and territories throughout the globe, the dataset contains data on COVID-19 cases, fatalities, hospitalizations, testing, vaccinations, and other associated variables. The data is derived from the World Health Organization (WHO), national health ministries, and other dependable sources. The OWID COVID-19 dataset, which contains both historical and recent data, is updated every day. It's presented in a uniform format makes it simple to compare data through time and between various nations. The OWID team offers a number of tools and visualizations in addition to the data itself so that users may study the data and recognize patterns and trends in the epidemic. These tools, which allow users to study the data in numerous ways and at varying degrees of detail, include interactive charts, maps, and dashboards.

Importing Packages

```
import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from sklearn.metrics import mean_squared_log_error
from keras.layers import SimpleRNN
import statistics
from tensorflow.keras.optimizers import Adam
```

1. **import pandas as pd**: This imports the sophisticated Python data analysis module pandas. It offers data cleaning, preparation, analysis, and visualization tools and data structures for effectively storing and processing massive, structured datasets.
2. **import matplotlib.pyplot as plt**: This imports the main Python model for creating images that display data. It can create all types of graphs.
3. **import math**: The Python math package contains trigonometric, logarithmic, and exponential functions.
4. **import numpy as np**: Python's popular numpy library is imported here. Multidimensional arrays, linear algebra, random number generation, and more are supported.
5. **from keras.models import Sequential**: This imports Sequential from keras.models. Keras, a Python-based neural networks API, runs on TensorFlow, CNTK, or Theano. Sequential layers are used to develop neural network models.
6. **from keras.layers import Dense, LSTM, Dropout, SimpleRNN**: These import Keras' layers for neural network modeling. Dense layers generate fully connected layers, whereas LSTM, SimpleRNN layers build recurrent neural network layers. Regularization reduces overfitting with dropout.
7. **from sklearn.metrics import mean_squared_log_error**: Scikit-learn, a prominent Python machine learning package, imports the mean squared log error function. This statistic assesses time series forecasting model accuracy.
8. **import statistics**: The Python statistics library offers statistical operations on datasets.

9. **from tensorflow.keras.optimizers import Adam**: Keras imports Adam. Adam optimizes neural network models. It extends stochastic gradient descent with adaptable learning rates.

Importing Data Sets

```
data=pd.read_csv(r"C:\Users\ljste\OneDrive - The Hill School\Coding Projects (python)\data sets\owid-covid-data.csv")
```

This line of code reads a CSV file named "owid-covid-data.csv" located at the specified file path "C:\Users\ljste\OneDrive - The Hill School\Coding Projects (python)\data sets" into a Pandas DataFrame called data.

Examining the Data

The data.head() function is a built-in function in pandas that is used to display the first few rows of a DataFrame. By default, it displays the first 5 rows of the DataFrame.

```
data.head()
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	NaN	NaN	NaN	...	NaN
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	NaN	NaN	NaN	...	NaN
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	NaN	NaN	NaN	...	NaN
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	NaN	NaN	NaN	...	NaN
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	NaN	NaN	NaN	...	NaN

5 rows × 67 columns

As we can see, there are 67 different stats being tracked by our dataset. These include, vaccinations, cases, gdp, and general population health.

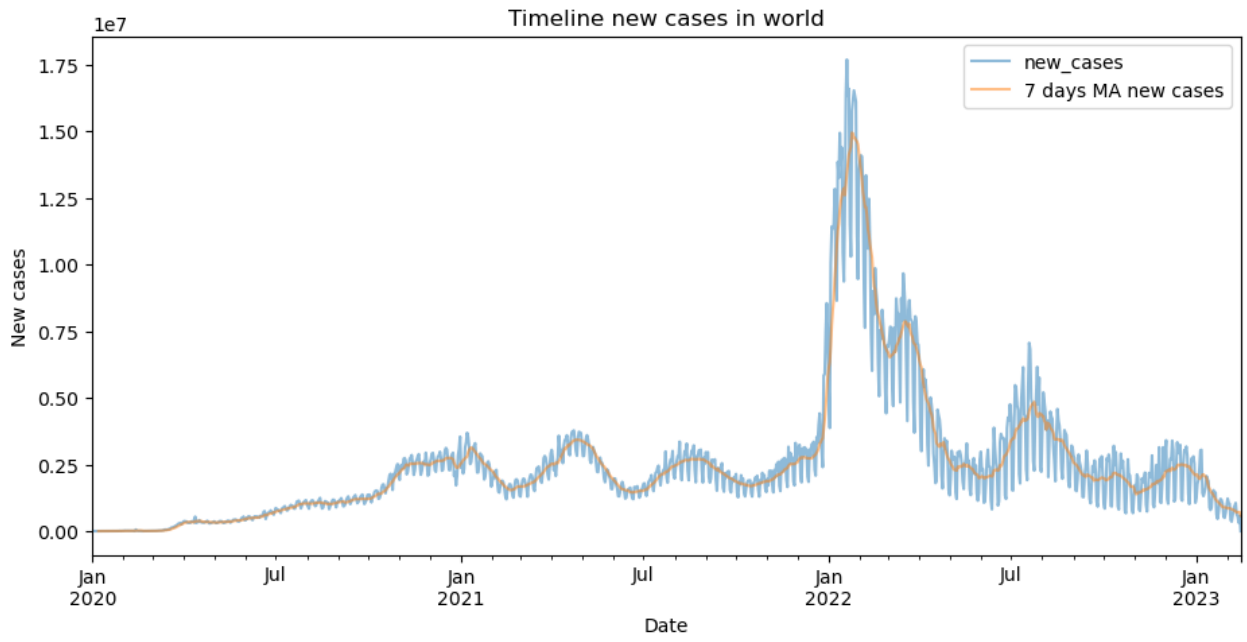
Exploratory Data Analysis

Global Scale

```
data2 = data.copy()
data2.date = pd.to_datetime(data2['date'])
data2 = data2.groupby('date').sum()
data2['7 days MA new cases'] = 0
data2['7 days MA new cases'] = data2['new_cases'].rolling(7).mean() #Moving average of new cases with window=10
data2['7 days MA new deaths'] = 0
data2['7 days MA new deaths'] = data2['new_deaths'].rolling(7).mean()  #Moving average of new deaths with window=10

data2[['new_cases', '7 days MA new cases']].plot(figsize = (11, 5), alpha = 0.5)
plt.title('Timeline new cases in world')
plt.xlabel('Date')
plt.ylabel('New cases')
```

```
Text(0, 0.5, 'New cases')
```

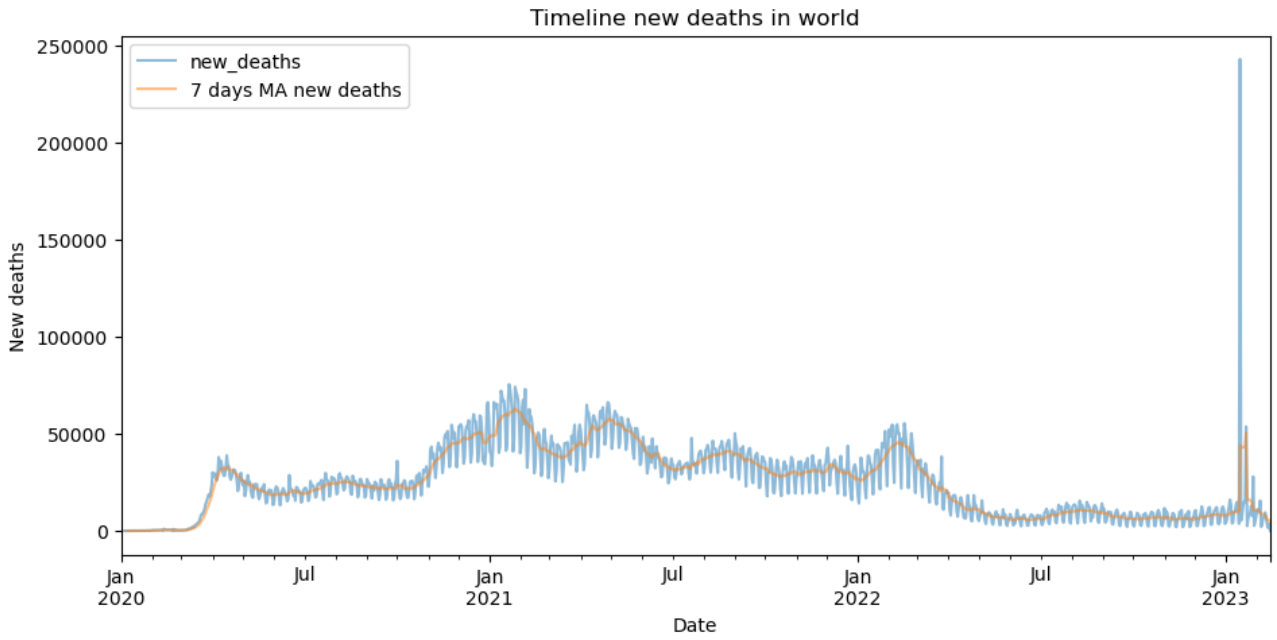


The code calculates and presents the new cases' 7-day moving average, which aids in identifying the data's underlying trend. The algorithm is able to record the general trend direction and smooth out daily volatility by using a rolling window of seven days. The trend in the 7-day moving average of new cases can provide valuable insights into the trajectory of the event. A rising trend could signify that current measures to curb the spread of the event are ineffective, or that new factors are driving its expansion. On the other hand, a declining trend may suggest that measures to address the underlying issues or to limit the event's transmission are working effectively. When the trend in the 7-day moving average of new

cases remains consistent over time, it may indicate that the event has reached a plateau. This could suggest that the current efforts to stop its spread are maintaining the status quo, or that other factors are counteracting the spread. It is important to continually monitor and analyze the trend in the 7-day moving average of new cases to inform decisions about appropriate responses and interventions. We may gauge the data's volatility and the extent to which daily changes affect the trend by comparing the 7-day moving average of new cases to the original data (i.e., the daily new cases). A large difference between the two might be a sign that the trend is unstable or that the data is noisy.

```
data2[['new_deaths', '7 days MA new deaths']].plot(figsize = (11, 5), alpha = 0.5)
plt.title('Timeline new deaths in world')
plt.xlabel('Date')
plt.ylabel('New deaths')
```

```
Text(0, 0.5, 'New deaths')
```



This code displays a very similar looking graph as seen prior, except now, the focus is on deaths. As we can see, the peak of deaths was in early 2021. On the contrary, the peak of cases was early 2022. Perhaps this is due to better treatment becoming more accessible across various parts of the globe. Now, we will take a look into more country-specific graphs and data.

United States of America

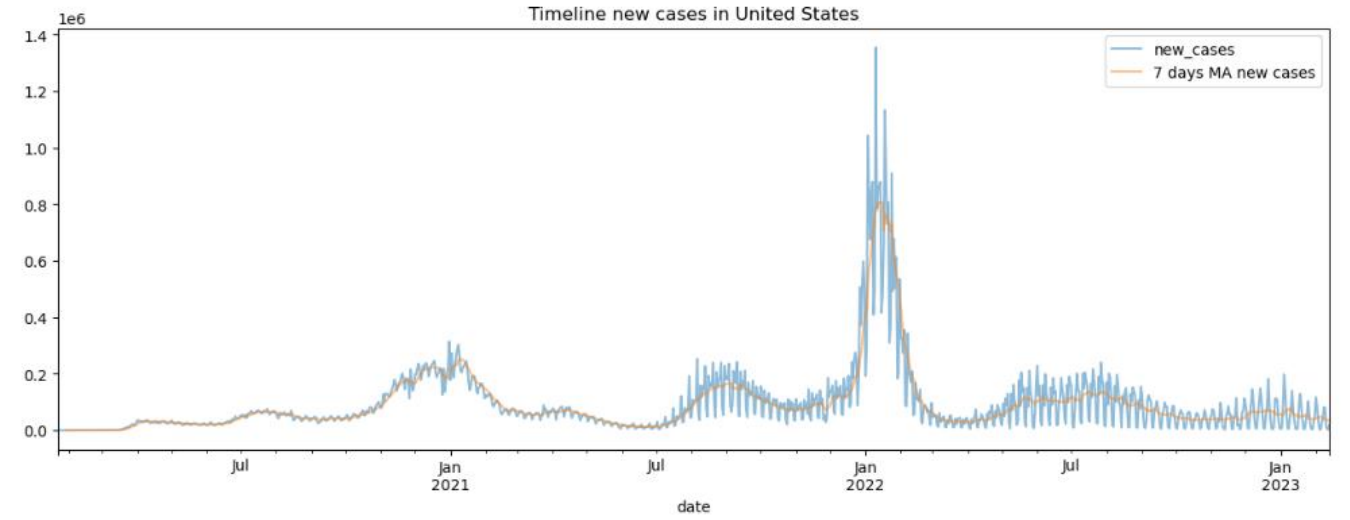
```python
def create_and_plot_df(df, country):
    df=df[df['location']==country].copy()
    df=df[['date','total_cases','new_cases',
         'total_deaths','new_deaths',
         'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred',
         'new_deaths_per_million', 'new_cases_per_million']].copy()
    df.date = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)
    df['7 days MA new cases'] = 0
    df['7 days MA new cases'] = df['new_cases'].rolling(7).mean()
    df['7 days MA new deaths'] = 0
    df['7 days MA new deaths'] = df['new_deaths'].rolling(7).mean()
    df['7 days MA new cases per million'] = 0
    df['7 days MA new cases per million'] = df['new_cases_per_million'].rolling(7).mean()
    df['7 days MA new deaths per million'] = 0
    df['7 days MA new deaths per million'] = df['new_deaths_per_million'].rolling(7).mean()
    df[['new_cases', '7 days MA new cases']].plot(figsize = (15, 5), alpha = 0.5)
    plt.title(f'Timeline new cases in {country}')
    df[['new_deaths', '7 days MA new deaths']].plot(figsize = (15, 5), alpha = 0.5)
    plt.title(f'Timeline new deaths in {country}')
    df[['people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred']].plot(figsize = (15, 5), alpha = 0.5)
    plt.ylim([0,100])
    plt.fill_between(df.index, df.people_vaccinated_per_hundred)
    plt.fill_between(df.index, df.people_fully_vaccinated_per_hundred)
    plt.title(f'Timeline percentage of people vaccinated in {country}')
    return df
```

This program creates a function called create and plot df that accepts a pandas DataFrame and a string that denotes a country's name as parameters. The following operations are carried out by the function:

- Selects the rows in the DataFrame that match the specified nation and the relevant fields (date, total cases, new cases, total deaths, new deaths, people vaccinated per hundred, people fully vaccinated per hundred, new deaths per million, new cases per million).
- The date column is changed to datetime format.
- Changes a few numbers in the DataFrame for the UK to fix various data flaws (e.g., incorrect reporting of new cases on certain dates).
- Establishes the date column as the DataFrame's index.
- Creates new columns in the DataFrame and adds them after calculating the 7-day moving average for the new cases, new deaths, new cases per million, and new deaths per million columns.
- Plots the new cases and new deaths and new deaths and new people vaccinated per hundred and people fully vaccinated per hundred columns for the specified nation.

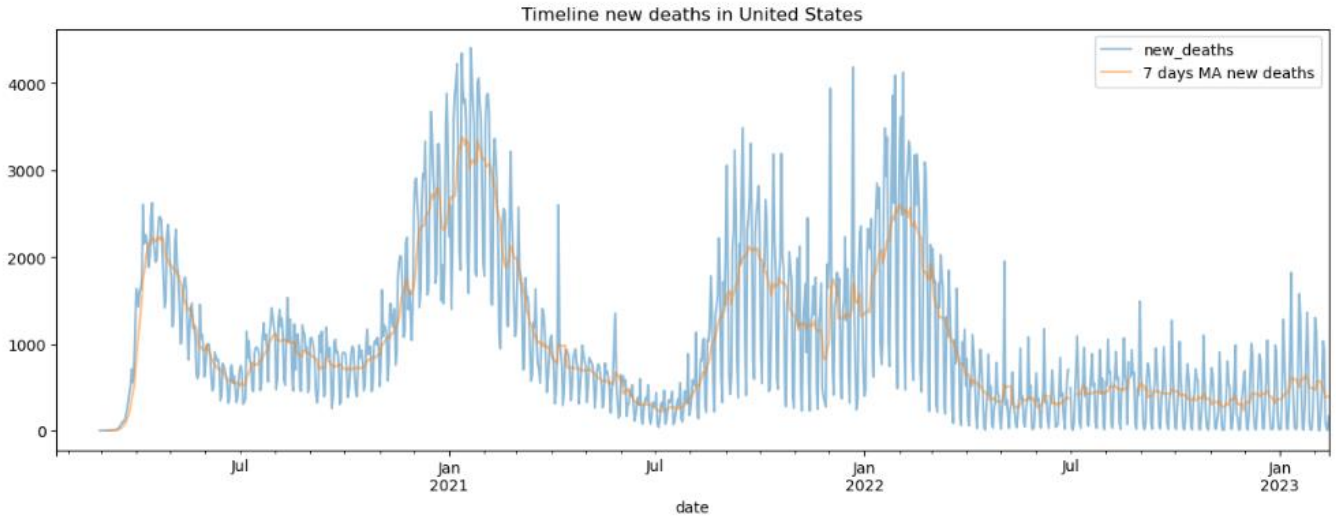
The function then returns the updated DataFrame. This is the baseline function for all of the countries I examined. I created a function instead of doing it individually to save time and space. Functions allow us to modularize our code. By writing a function for a particular action, we can encapsulate that functionality within a single block of code. This makes it easier to read and maintain, as we don't need to repeat the same code multiple times throughout the program.

```
df_us=create_and_plot_df(data, 'United States')
```

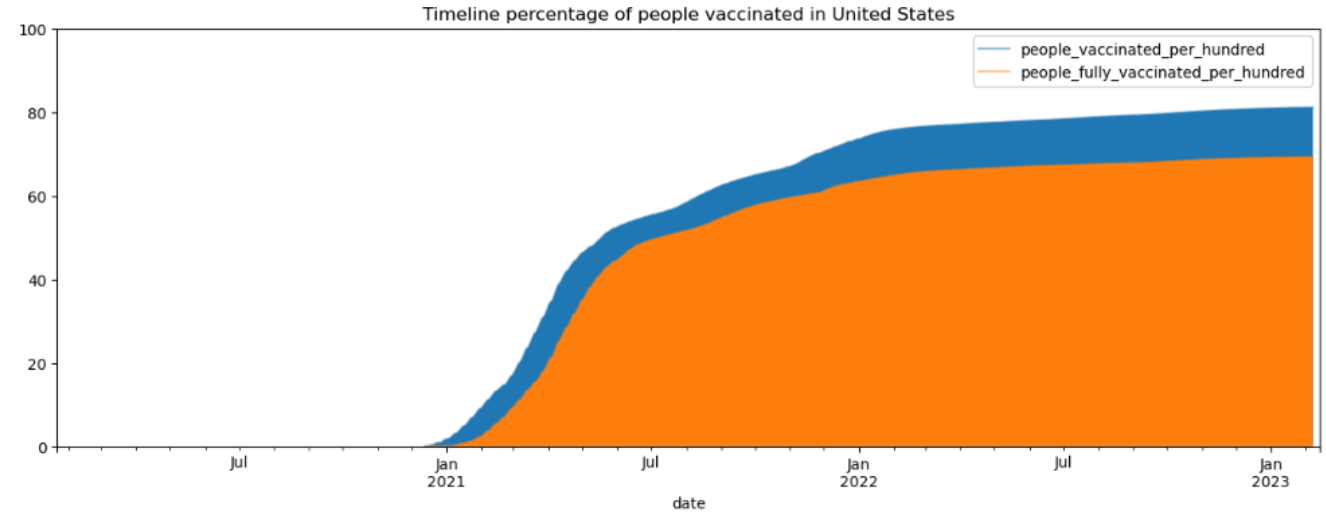


This first graph that we will be looking at displays the timeline of new cases per million in the United States between the start of the pandemic and the current day. The blue line is the raw number that each day has. While the orange line is the moving average, which is the main stat that most sources use. Based on the data, it can be observed that the United States experienced the initial wave of COVID-19 cases in early 2020. This was followed by a second wave in the summer of 2020, which can be seen as a spike in the graph. There was then a decline in cases before a third wave occurred in the fall and winter of 2020-2021. The third wave is visible on the graph as a sharp increase in cases. The most recent wave was at the start of 2022, which was the biggest one, yet least spoken about on news outlets and newspapers. Since then, there has been a decline in the number of COVID-19 cases in the United States, which is likely due to the rollout of vaccines and other preventive measures such as social distancing and mask-wearing. However, recent spikes in cases can be observed in some parts of the United States, which can

be attributed to the highly transmissible Delta and Omicron variant of the virus. It can be noted that the trend of COVID-19 cases in the United States is heavily influenced by several factors, including vaccination rates, adherence to preventive measures, and the emergence of new variants of the virus.



Similar to the rest of the world, the United States has the highest death peak at the start of 2021, and the case peak at the start of 2022.

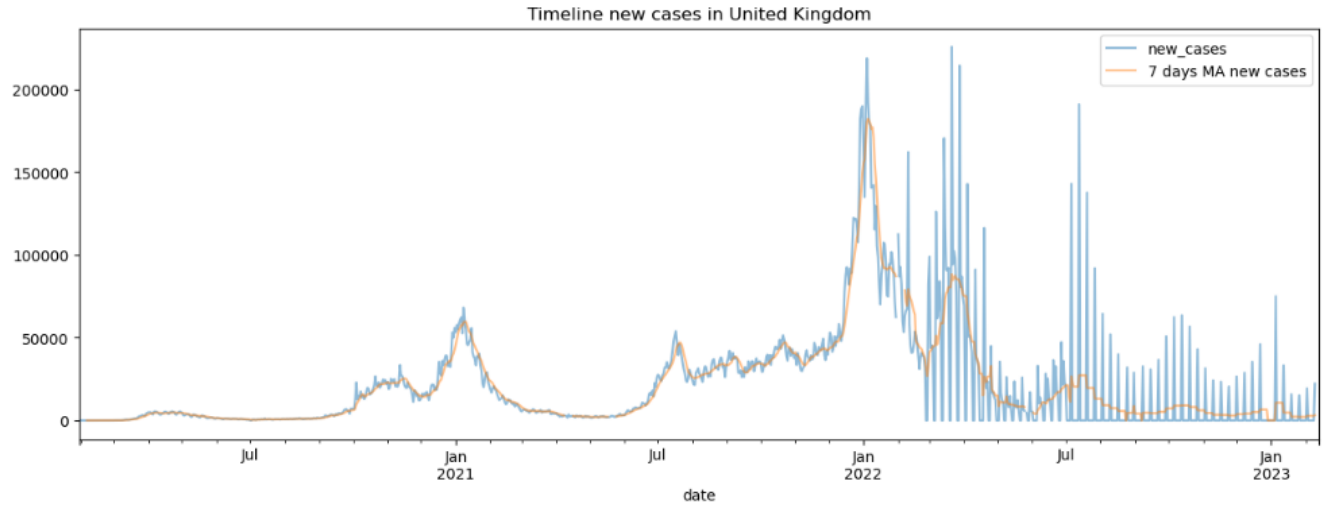


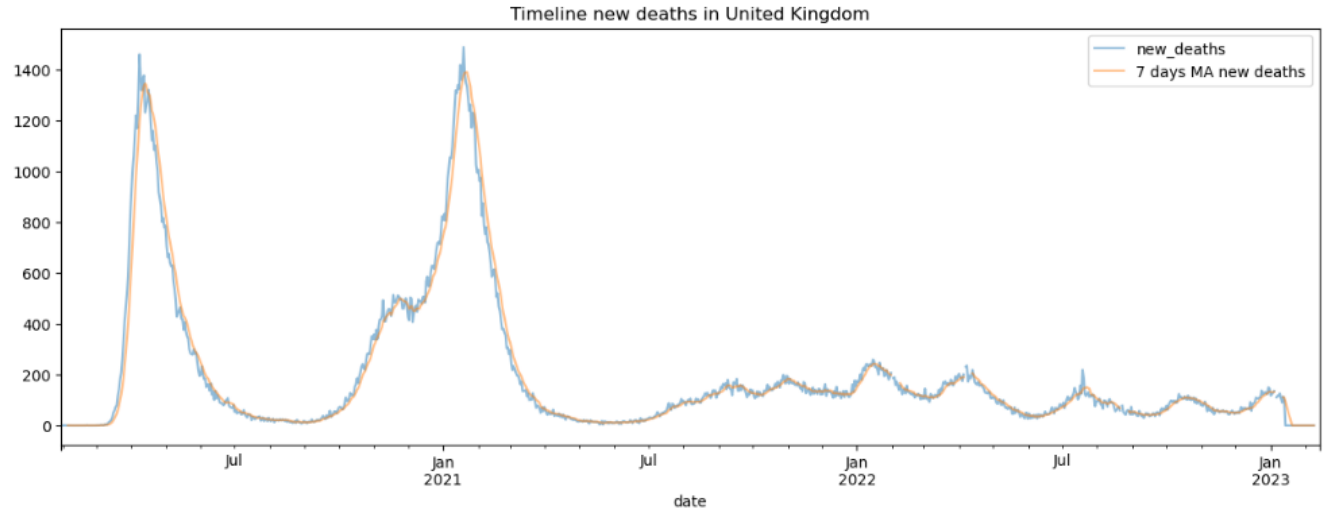
This graph shows us the comparison between full vaccinated people to partly vaccinated people. It also gives us the percentage of the population for each. The USA has a fairly decent fully vaccinated to vaccinated ratio. As vaccination percentage increased, we can clearly see the deaths decrease.

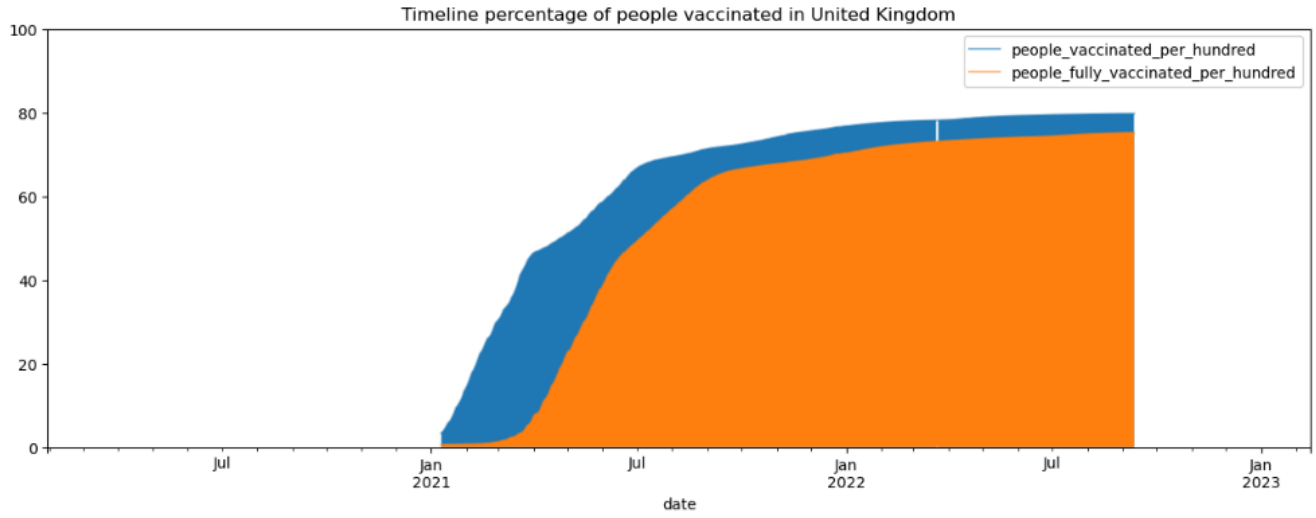
United Kingdom

Although my main focus of time series forecasting is the United States, it is still important to test my code on more than 1 country to ensure accuracy in my code.

```
1 df_uk=create_and_plot_df(data, 'United Kingdom')
```



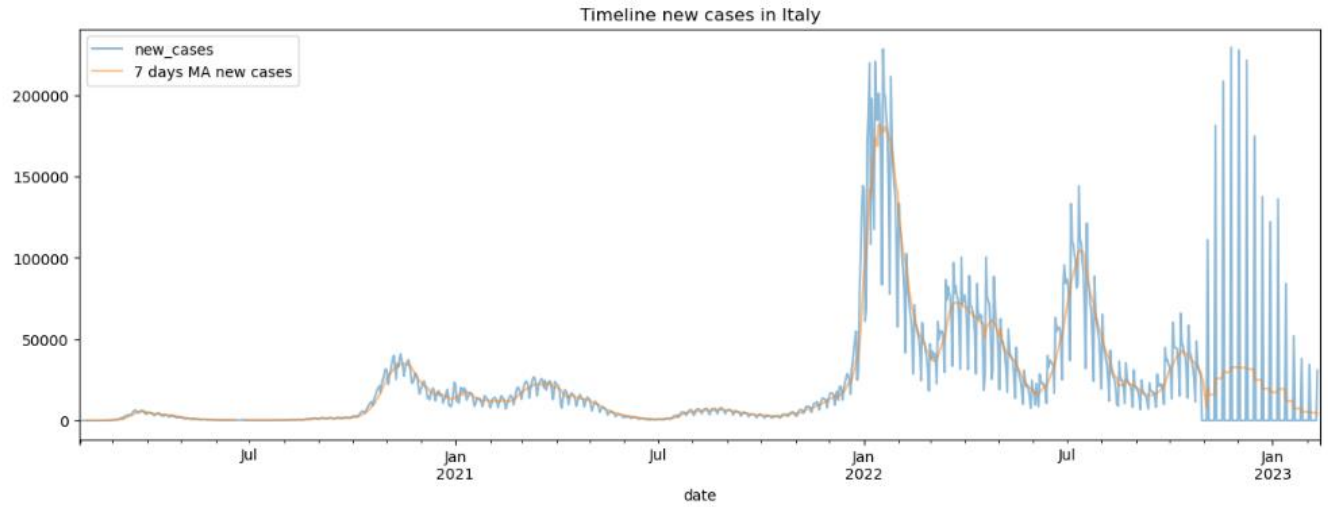


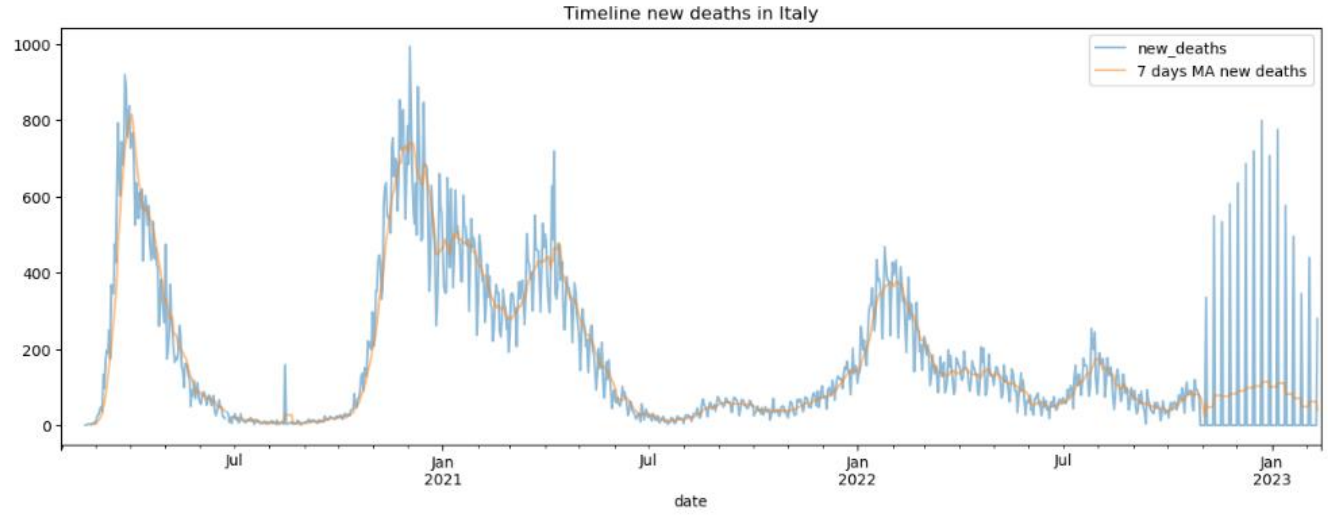


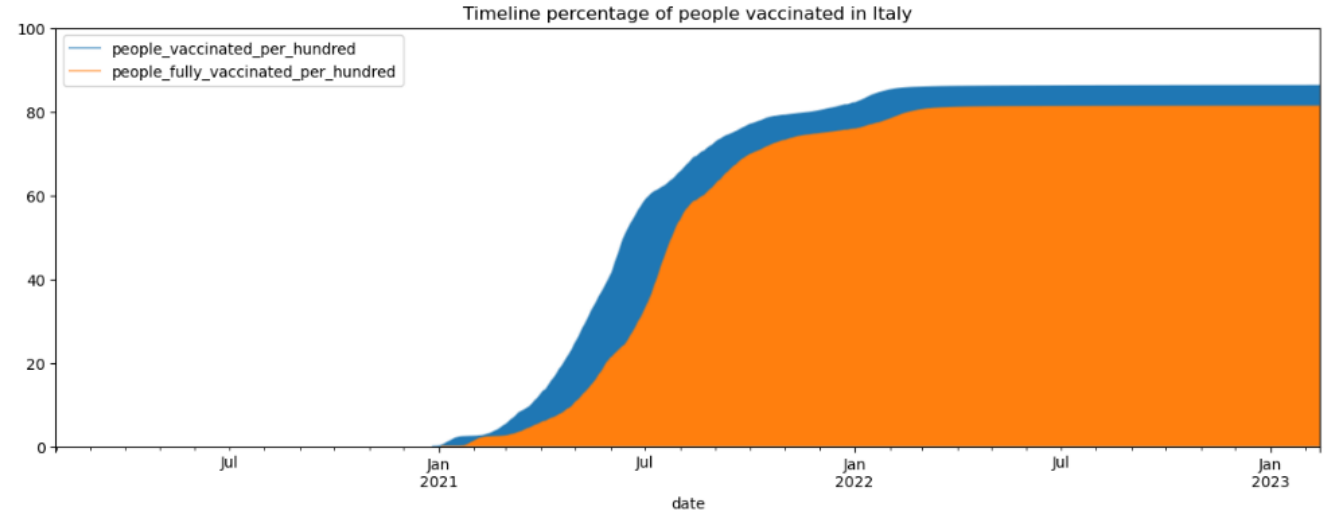
The number of people in the United Kingdom who have gotten all of the recommended immunizations has been steadily going up. This could be because more people are taking responsibility for their health. And since 2021, there has been a trend toward fewer people dying. This is different from the situation in the United States, where the death rate has continued to rise even after this time period has passed. It's likely that the number of incidents reported in the UK followed patterns similar to those seen in other places. This is most likely the case because there is a link between the number of cases and how well the population as a whole is immunized. This is an interesting idea that needs to be looked into more. We need to learn more about this topic.

Italy

```
1 df_italy=create_and_plot_df(data, 'Italy')
```



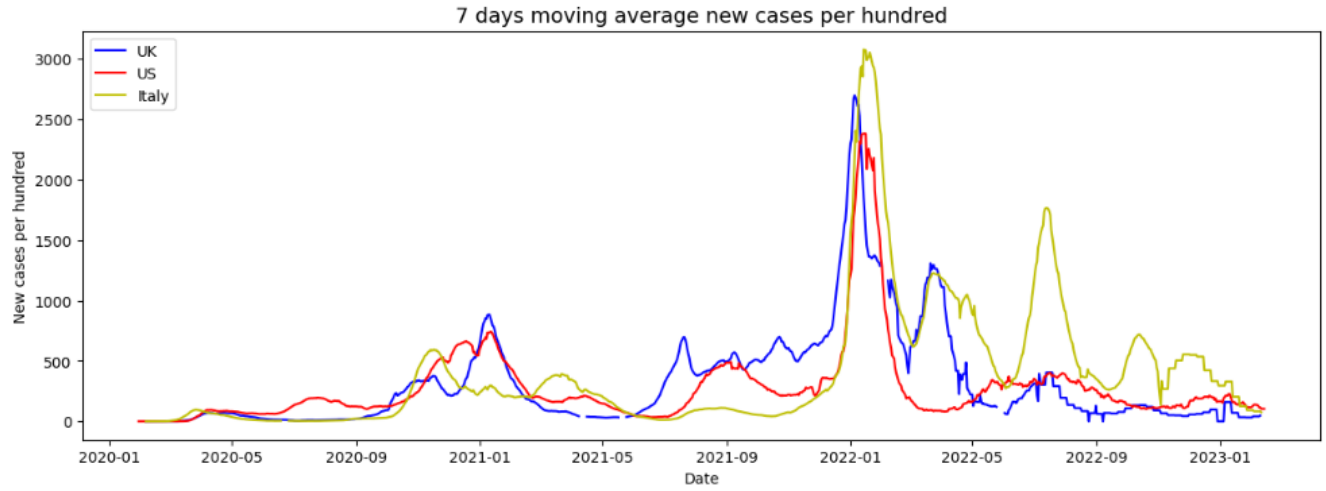




Combined

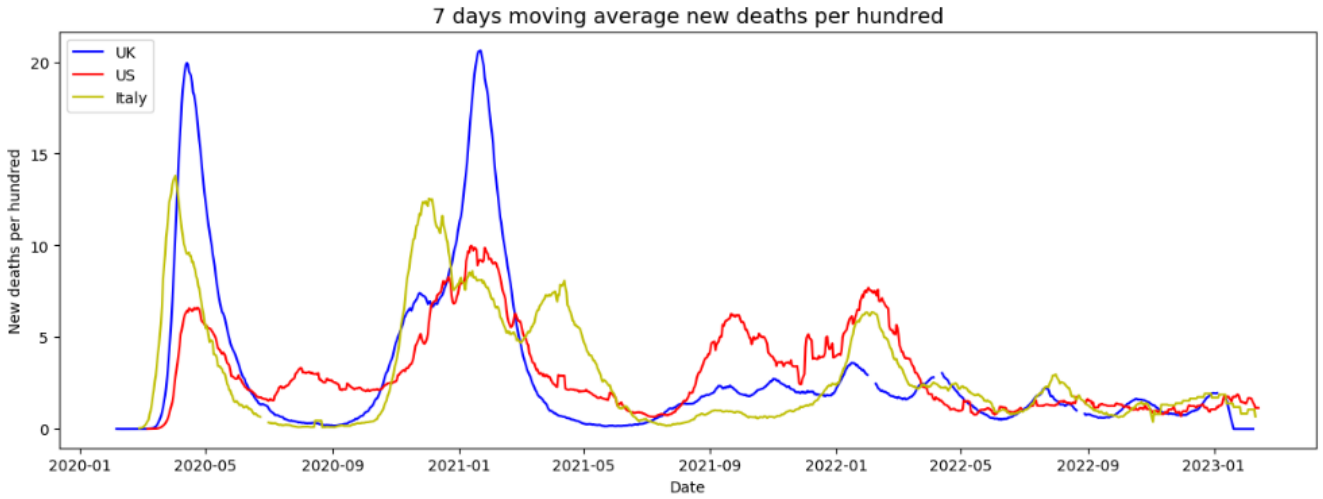
```
df3_countries=pd.DataFrame()
df3_countries['New cases per million uk'] = df_uk['7 days MA new cases per million']
df3_countries['New cases per million us'] = df_us['7 days MA new cases per million']
df3_countries['New cases per million italy'] = df_italy['7 days MA new cases per million']
df3_countries['New deaths per million uk'] = df_uk['7 days MA new deaths per million']
df3_countries['New deaths per million us'] = df_us['7 days MA new deaths per million']
df3_countries['New deaths per million italy'] = df_italy['7 days MA new deaths per million']
df3_countries['date'] = df_uk.index
df3_countries.set_index('date', inplace=True)
```

```
plt.figure(figsize=(15, 5))
plt.plot(df3_countries.index, df3_countries['New cases per million uk'], "-b", label="UK")
plt.plot(df3_countries.index, df3_countries['New cases per million us'], "-r", label="US")
plt.plot(df3_countries.index, df3_countries['New cases per million italy'], "-y", label="Italy")
plt.legend(loc="upper left")
plt.title("7 days moving average new cases per hundred",fontsize=14)
plt.ylabel('New cases per hundred')
plt.xlabel('Date')
plt.show()
```



This code creates the 7-day moving average of new cases per million and new deaths per million for three countries: the United Kingdom, the United States, and Italy. The date column is the dataframe's index. The code then makes a line plot of the 7-day moving average new cases per million for the three countries using the matplotlib tool. The x-axis of the figure shows the date, while the y-axis shows the number of new cases per million. Each country is colored uniquely and has its own tale. Finally, the plot is shown using the plt.show() method.

```
plt.figure(figsize=(15, 5))
plt.plot(df3_countries.index, df3_countries['New deaths per million uk'], "-b", label="UK")
plt.plot(df3_countries.index, df3_countries['New deaths per million us'], "-r", label="US")
plt.plot(df3_countries.index, df3_countries['New deaths per million italy'], "-y", label="Italy")
plt.legend(loc="upper left")
plt.title("7 days moving average new deaths per hundred",fontsize=14)
plt.ylabel('New deaths per hundred')
plt.xlabel('Date')
plt.show()
```



This graph provides a visual representation of the number of deaths in the USA, Italy, and the UK over time, using the same method as previously described. By comparing the three graphs, we can observe some interesting trends. One key observation is that the UK initially had the highest death peaks of the three countries, with a sharp increase in fatalities in the early stages of the pandemic. However, after implementing a successful immunization campaign, as shown by the vaccination chart, the number of deaths in the UK has decreased significantly and has remained relatively low compared to the other countries. In contrast, the United States experienced a much slower decline in the number of COVID-19 deaths, taking significantly longer to reach zero death tolls. This could be due to a number of factors, including differences in the speed and effectiveness of vaccination campaigns, variations in public health policies, and differences in population demographics. Similarly, Italy experienced a significant surge in COVID-19 deaths early in the pandemic, followed by a gradual decline. Although Italy's death toll remained relatively high compared to the other two countries, it did eventually decrease over time.

Time Series Forecasting

Validation and Model Building

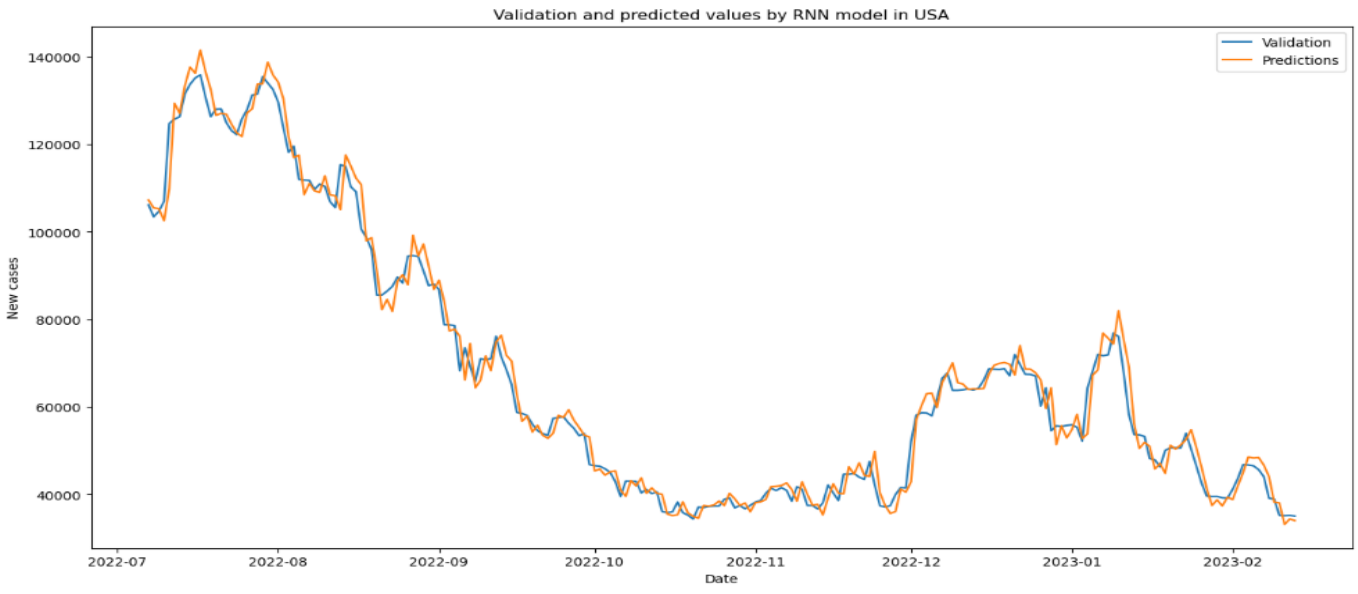
This code creates the function build model, which constructs and trains a basic RNN model using the Keras and Tensorflow libraries. Based on the 7-day moving average of new cases in a particular nation, the model is trained to estimate the number of new COVID-19 cases. The function accepts two input parameters: country data, a pandas DataFrame containing COVID-19 data for a specified country, and name country, a string holding the name of the nation for which the model is being generated.

```
def build_model(country_data, name_country):
  new_df=country_data['7 days MA new cases']
  if name_country == 'USA':
    new_df=new_df[7:].copy()
  else:
    new_df=new_df[6:].copy()
  window=10
  train_len = math.ceil(len(new_df)*0.8)
  train_data = new_df[0:train_len]
  X_train=[]
  Y_train=[]
  for i in range(window, len(train_data)):
      X_train.append(train_data[i-window:i])
      Y_train.append(train_data[i])
  X_train, Y_train= np.array(X_train), np.array(Y_train)
  X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
  test_data = new_df[train_len-window:]
  X_val=[]
  Y_val=[]
  for i in range(window, len(test_data)):
      X_val.append(test_data[i-window:i])
      Y_val.append(test_data[i])
  X_val, Y_val = np.array(X_val), np.array(Y_val)
  X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1],1))
  r1=[]
  r2=[]
  model=Sequential()
  model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
  model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
  model.add(Dense(100))
  model.add(Dense(25))
  model.add(Dense(1))
  opt1=Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
  model.compile(loss='mean_squared_error', optimizer=opt1)
  model.fit(X_train, Y_train, epochs=100, batch_size=10, verbose=0)
  lstm_train_pred = model.predict(X_train)
  lstm_valid_pred = model.predict(X_val)
  lstm_train_pred[lstm_train_pred<0] = 0
  lstm_valid_pred[lstm_valid_pred<0] = 0
  valid = pd.DataFrame(new_df[train_len:])
  valid['Predictions'] = model.predict(X_val)
  plt.figure(figsize=(16,8))
  plt.title(f'Validation and predicted values by RNN model in {name_country}')
  plt.xlabel('Date')
  plt.ylabel('New cases')
  plt.plot(valid[['7 days MA new cases','Predictions']])
  plt.legend(['Validation','Predictions'])
  plt.show()
  return new_df.copy(), model, valid
```

The function first calculates the 7-day moving average of new instances for the selected nation before dividing the data into a training set and a validation set. The training set contains 80% of the data, whereas the validation set contains 20%. The data is preprocessed by reshaping it into the RNN model's necessary input format, which is (batch size, timesteps, input dim). The RNN model is built using Keras's Sequential class and comprises of two SimpleRNN layers with 50 neurons each, followed by two dense layers with 100 and 25 neurons, respectively. The Adam optimizer and the mean squared error loss function are used to build the model. With a batch size of 10, the model is trained for 100 epochs. The model is used to generate predictions on the validation set after training. The predicted and actual values are presented on a graph to visually compare the model's performance. Finally, the function returns a duplicate of the original data, as well as the trained model and validation data. I experimented with ARIMA algorithm and found that the Keras and Tensorflow libraries are more accurate for this example. In addition, deep learning models like RNNs can capture complex patterns in time series data that may not be easily captured by more traditional statistical models like ARIMA. RNNs can also handle non-linear relationships and dependencies between time series data points, making them potentially more powerful for time series prediction tasks.

```
1 df_us_cut, model_us, valid_df_us = build_model(df_us, 'USA')
```

```
28/28 [==============================] - 0s 1ms/step
7/7 [==============================] - 0s 1ms/step
7/7 [==============================] - 0s 1ms/step
```



It appears that the predicted values generated by the Recurrent Neural Network (RNN) model are very similar to the actual cases of COVID-19. This suggests that the RNN model has been successful in capturing the patterns and trends in the data, and is able to make accurate predictions based on that information. The accuracy of the RNN model in predicting past COVID-19 cases provides some evidence that it may also be successful in predicting future cases. By training the model on historical data and evaluating its performance on a test set, we can see that the model is able to capture patterns and trends in the data, and generate accurate predictions.

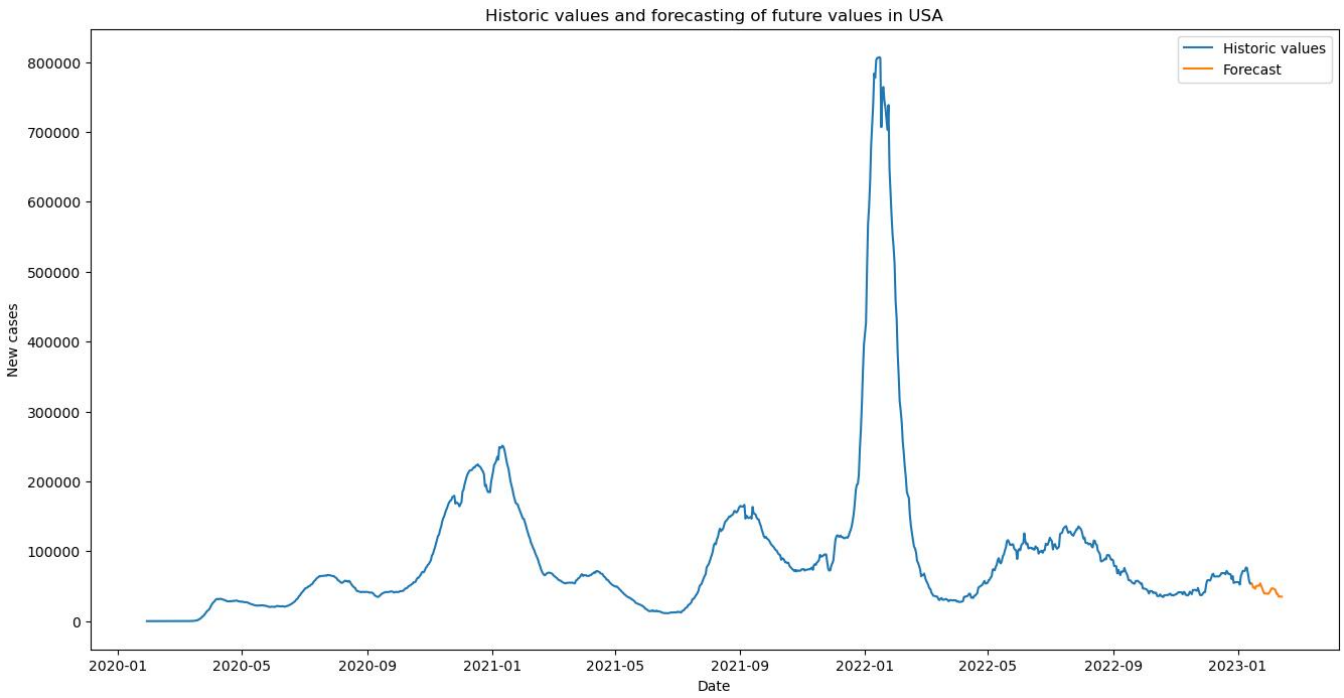
Forecasting and Predicting

```
def future_values(model, future_df, days, name_country):
  for i in range(1,days+1,1):
    last_10_days=future_df[-10:].values
    X_test=[]
    X_test.append(last_10_days)
    X_test=np.array(X_test)
    X_test=np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1))
    pred_new_cases=model.predict(X_test)
    idx = pd.date_range(future_df.index[-1], periods=1, freq='D')[1:]
    future_df=future_df.append(pd.Series(pred_new_cases[0][0], index=idx)).copy()

  plt.figure(figsize=(16,8))
  plt.title(f'Historic values and forecasting of future values in {name_country}')
  plt.xlabel('Date')
  plt.ylabel('New cases')
  plt.plot(future_df[:-days])
  plt.plot(future_df[-days:])
  plt.legend(['Historic values','Forecast'])
  plt.show()

  return future_df
```

Using the function above, this function builds off of that. Based on the data from the last 10 days from the input "future df," it builds a loop to forecast the number of new cases for the next "days" days using the RNN model. Next, the anticipated values are added to the "future df," and a plot is made to show the historical values and the anticipated values for the next "days" days. The modified "future df" is the function's return value. The result can be seen below.



The blue line represents the Covid cases that we already know about. The orange line is what the prediction is. As we can see, the trend is headed downwards. As we go farther into the future, the prediction loses accuracy. Because when we project farther into the future, unanticipated occurrences and changes in the underlying data become more likely, making it increasingly difficult to make precise forecasts. As we go deeper into the future for COVID-19 forecasting, a variety of things may have an impact on the model's accuracy. For instance, changes in governmental regulations, societal norms, the accessibility and effectiveness of vaccines, and the appearance of novel variations may all have a substantial influence on how the pandemic develops. Beyond a certain point, it becomes difficult to provide accurate projections since these components are often unexpected and tough to model. Furthermore, the model also gets more sensitive to minor changes in the input data as we project farther into the future. The prediction may become less accurate if there are even little variations in the input data that have a major influence on the predicted values.

Conclusion

What we learned

Using deep learning methods and exploratory data analysis, we conducted a study of the COVID-19 pandemic, and the findings supplied us with up-to-date information on the disease's development in a number of nations located in a range of different areas throughout the world. We were able to forecast how the pandemic would evolve in the next years by studying historical data

To begin, through our real life experiences, we know that the COVID-19 pandemic has had a huge impact not only on people's health, but also on their economy and the way they live their lives throughout the world. It is critical that fast and firm action be made to bring the virus under control in order to cope with the alarmingly high number of cases and deaths that have been reported in different parts of the world. According to our data, the virus spread swiftly, and the number of countries reporting new instances of the disease has been gradually growing over the previous several years. This is a worldwide trend.

We built models that can predict how the epidemic would evolve in the next years. We accomplished this by using deep learning technologies. We moved on to the next step. We discovered that forecasting models were capable of correctly forecasting the number of cases across multiple countries, particularly in the short term. On the other side, as we moved farther into the future, we saw that the projections were getting less accurate. This was because we were looking farther forward. It didn't matter how far into the

future we looked; this was always the case. We ascribe this to the virus's high dynamic nature, and there are several elements that may influence its spread, such as changes in government regulations, new strains of the virus, and variances in human behavior. This is also due to the fact that there are several strains of the virus. Despite this, the models were able to provide important insights into how the pandemic may evolve in the future, and we believe they might be critical tools for policymakers and public health experts to utilize.

According to our study results, some countries have been more effective than others in limiting the slope of the growing number of new cases; this result suggests that certain remedies have been more successful than others. Through the information that Covid-19 forecasting and data analysis gives us, we may be able to assist policymakers and public health professionals identify optimal practices and methods for reducing the virus's spread.

Predicting COVID-19 is crucial for several reasons. First, it provides policymakers, healthcare professionals, and the public with vital information about the disease's potential impacts. Predicting the pandemic's progress allows decision-makers to plan for hospital beds, ventilators, and PPE. They may also organize and allocate resources to minimize shortages and respond quickly to demand surges. It also aids early detection and pandemic response policy development. As the epidemic continues to evolve and new strains emerge, forecasting is as crucial as ever.