### *Step 1: Importing Libraries(Using Darts for timeseries forecasting)*

In [1]:

```python
from darts import TimeSeries
```

### *Step 2: Importing covid cases data from covid19h (Source: John Hopkins) from 1st November and relevant data for analysis*

In [2]:

```python
# Importing data obtained from Data Acquisition Team (This data contains 40% o
data = pd.read_csv('data_sa_new.csv')
```

In [3]:

In [4]:

```python
from datetime import datetime
from covid19dh import covid19
x, src = covid19(countries, start = datetime(2021,11,1), end = "2022-03-04")
```

```
C:\Users\Aditya\anaconda3\lib\site-packages\covid19dh\_cite.py:85: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  src = src.append(sources[
C:\Users\Aditya\anaconda3\lib\site-packages\covid19dh\_cite.py:96: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  references = references.append(src)
C:\Users\Aditya\anaconda3\lib\site-packages\covid19dh\_cite.py:85: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  src = src.append(sources[
C:\Users\Aditya\anaconda3\lib\site-packages\covid19dh\_cite.py:96: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  references = references.append(src)
C:\Users\Aditya\anaconda3\lib\site-packages\covid19dh\_cite.py:85: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
```

### *Step3: Data Preparation for Analysis*

In [5]:

```
# Divinding data based on the country
import numpy as np
countries=list(data["Countries"].unique())
countries_data=[data.loc[data["Countries"] == i] for i in countries]
start, end = list(countries_data[0].day)[0],list(countries_data[0].day)[-1]
dt_ran = pd.date_range(start=start,end=end,freq="D")
t_index=pd.DatetimeIndex(dt_ran)
countries_data = [i.set_index("day").reindex(t_index).fillna(0).reset_index()
for i in countries_data:
    pop_d = [i for i in list(i['Population Density'].unique()) if i!=0]
    dest_cnt = [i for i in list(i['dest cou'].unique()) if i!=0]
    i['Population Density'] = i['Population Density'].replace(0,pop_d[0]).valu
    i['dest cou'] = i['dest cou'].replace(0,dest_cnt[0]).values
```

In [6]:

```
# Mapping covid cases to each country
import datetime
for cnt in countries_data:
    cases=[]
    population = []
    for index,i in list(cnt.iterrows()):
        start = i["index"]+datetime.timedelta(days=2)
        end = i["index"]+datetime.timedelta(days=1)
        y = x[x["iso_alpha_2"]==i["dest cou"]]
        date_range = y[(y["date"]>=end) & (y["date"]<=start)]
        confirmed = list(date_range["confirmed"])
        if(len(confirmed)>1):
            cases.append(confirmed[1]-confirmed[0])
        else:
            cases.append(cases[-1])
        population.append(list(y["population"])[0])
    cnt["cases"] = cases
```

```
C:\Users\Aditya\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:73:
FutureWarning: Comparison of Timestamp with datetime.date is deprecated in or
der to match the standard library behavior. In a future version these will be
considered non-comparable. Use 'ts == pd.Timestamp(date)' or 'ts.date() == da
te' instead.
  result = libops.scalar_compare(x.ravel(), y, op)
C:\Users\Aditya\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:73:
FutureWarning: Comparison of Timestamp with datetime.date is deprecated in or
der to match the standard library behavior. In a future version these will be
considered non-comparable. Use 'ts == pd.Timestamp(date)' or 'ts.date() == da
te' instead.
  result = libops.scalar_compare(x.ravel(), y, op)
C:\Users\Aditya\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:73:
FutureWarning: Comparison of Timestamp with datetime.date is deprecated in or
der to match the standard library behavior. In a future version these will be
considered non-comparable. Use 'ts == pd.Timestamp(date)' or 'ts.date() == da
te' instead.
  result = libops.scalar_compare(x.ravel(), y, op)
C:\Users\Aditya\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:73:
```

In [7]:

### *Step4: Conversion to time series*

In [8]:

```
series_main = [TimeSeries.from_dataframe(i, 'index', ['cases',"Seats"],fill_mi
```

### *Step5: Standardization of Data*

In [9]:

```
from darts.dataprocessing.transformers import Scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler_cases = [Scaler(scaler) for i in series_main]
scaler_seats = [Scaler(scaler) for i in series_main]
series_cases =  [y.fit_transform(i["cases"]) for y,i in zip(scaler_cases,serie
```

In [10]:

### *Step6: Applying Algorithms on data*

### *LSTM*

In [11]:

```
from darts.models import BlockRNNModel
# Model takes minimum 10 days and forecasts maximum 20 days of future data
model = BlockRNNModel(input_chunk_length =10,output_chunk_length=20,model="LST
```

```
2022-08-26 21:11:18 pytorch_lightning.utilities.rank_zero INFO: GPU availabl
e: False, used: False
2022-08-26 21:11:18 pytorch_lightning.utilities.rank_zero INFO: TPU availabl
e: False, using: 0 TPU cores
2022-08-26 21:11:18 pytorch_lightning.utilities.rank_zero INFO: IPU availabl
e: False, using: 0 IPUs
2022-08-26 21:11:18 pytorch_lightning.utilities.rank_zero INFO: HPU availabl
e: False, using: 0 HPUs
2022-08-26 21:11:18 pytorch_lightning.callbacks.model_summary INFO:
  | Name          | Type              | Params
---------------------------------------------------
0 | criterion     | MSELoss           | 0
1 | train_metrics | MetricCollection  | 0
2 | val_metrics   | MetricCollection  | 0
3 | rnn           | LSTM              | 2.9 K
4 | fc            | Sequential        | 1.0 K
---------------------------------------------------
3.9 K     Trainable params
0         Non-trainable params
3.9 K     Total params
0.032     Total estimated model params size (MB)
```

Epoch 99: 100% 34/34 [00:01<00:00, 24.90it/s, loss=0.0168, train_loss=0.0218]

```
2022-08-26 21:13:27 pytorch_lightning.utilities.rank_zero INFO: `Trainer.fit`
stopped: `max_epochs=100` reached.
```

Out[11]:

```
<darts.models.forecasting.block_rnn_model.BlockRNNModel at 0x1fc4cb9cc70>
```

#### *Step7: Inverse Standardization*

In [12]:

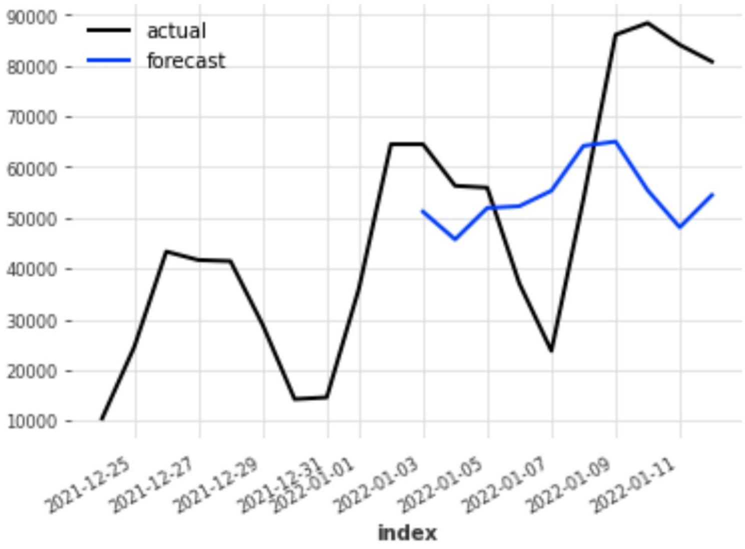
```
pred = model.predict(n=10,series=series[4][53:63])

inverse_series_cases = [scaler_cases[index].inverse_transform(i["cases"][:94])
inverse_series_seats = [scaler_seats[index].inverse_transform(i["Seats"][:94])
inverse_pred_cases = scaler_cases[4].inverse_transform(pred["cases"])
inverse_pred_seats = scaler_seats[4].inverse_transform(pred["Seats"])
```

Predicting DataLoader 0: 100% 1/1 [00:00<00:00, 20.24it/s]

In [13]:

```
# Plotting actual against forecast for a random country
import matplotlib.pyplot as plt
inverse_series_cases[4][53:73]["cases"].plot(label="actual")
```



### ***Step8: Calculating error percentage ((actual-forecast)*100/actual)***

In [15]:

```
from darts.metrics import mae
forecast_data = []
mape_nos = []
inverse_series_cases = [scaler_cases[index].inverse_transform(i["cases"][:94])
for index,i in enumerate(countries):
    pred = model.predict(n=11,series=series[index][53:73])
    inverse_pred_cases = scaler_cases[index].inverse_transform(pred["cases"])
    print(i)
    print("MAE for cases = {:.2f}".format(mae(inverse_series_cases[index],inve
    mae_no = mae(inverse_series_cases[index],inverse_pred_cases)
    mape_no = mae_no/(int(inverse_series_cases[index].pd_dataframe().max())+1)
    mape_nos.append(mape_no)
    print(inverse_series_cases[index].pd_dataframe().max())
    forecast_df = inverse_pred_cases.pd_dataframe().reset_index()
    forecast_df["Country"] = i
```

```
Predicting DataLoader 0: 100%                                1/1 [00:00<00:00, 55.60it/s]
```

```
United Arab Emirates (the)
MAE for cases = 1235.45
component
cases    3116.0
dtype: float64
```

```
Predicting DataLoader 0: 100%                                1/1 [00:00<00:00, 58.86it/s]
```

```
Belgium
MAE for cases = 21726.92
component
cases    76034.0
dtype: float64
```

```
Predicting DataLoader 0: 100%                                1/1 [00:00<00:00, 50.02it/s]
```

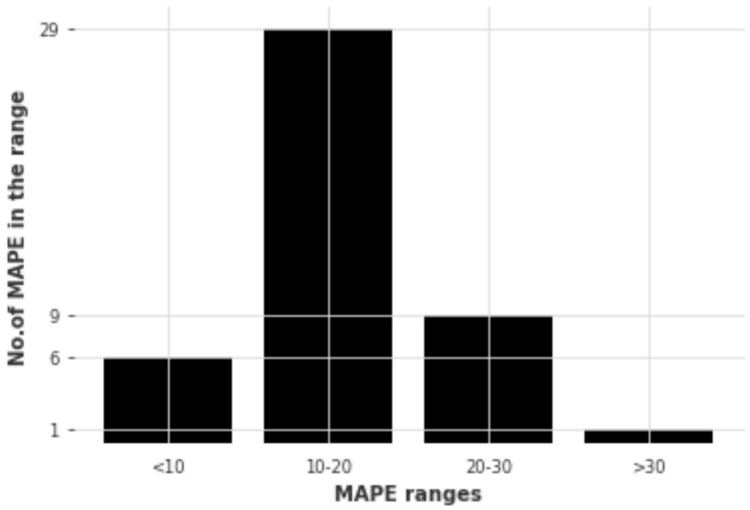
### ***Step9: Plotting the errors of country as bar plots***

In [16]:

```
first=len([i for i in mape_nos if i<10])
second=len([i for i in mape_nos if i>10 and i<30])
third = len([i for i in mape_nos if i>30 and i<50])
fourth = len([i for i in mape_nos if i>50])

weights = [1,2,3,4]
bars_list = [first,second,third,fourth]

import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
ax1 = plt.subplot(1,1,1)
ax1.set_xticks(x)
ax1.set_yticks(bars_list)
plt.bar(x,bars_list)
ax1.set_xticklabels(["<10","10-20","20-30",">30"])
ax1.set_yticklabels(bars_list)
plt.ylabel("No.of MAPE in the range")
plt.xlabel("MAPE ranges")
```



In [17]:

*Step10: Finding Incidence Rate (no. of. cases in the country/(population of the country * time frame of consideration))*

In [18]:

In [19]:

```
pop_density = []
for row in class_data.iterrows():
    pop_density.append(list(data[data["Countries"]==row[1]["Country"]]["Popula

class_data["Population"] = pop_density
```

In [20]:

In [21]:

```
class_data["Incidence"] = class_data["cases"]*1000000/class_data["Population"]
```

### *Step11: One hot encoding for Country names*

In [22]:
```
one_hot = pd.get_dummies(class_data["Country"])
data_sa = class_data.drop("Country", axis=1)
```

### *Step12: Normalization of data*

In [23]:
```
from sklearn import preprocessing

x = np.array(data_sa["Incidence"]).reshape(-1,1) #returns a numpy array
min_max_scaler = preprocessing.StandardScaler()
x_scaled = min_max_scaler.fit_transform(x)
x_scaled = [i[0] for i in x_scaled]
```

In [24]:
```
# deleting unused columns
del data_sa["index"]
del data_sa["Population"]
```

### *Step13: Clustering using K means clustering*

In [25]:
```
from dtaidistance import dtw,clustering
from dtaidistance.clustering import kmeans
model = kmeans.KMeans(k=3)
series_clu =  data_sa.to_numpy()
```

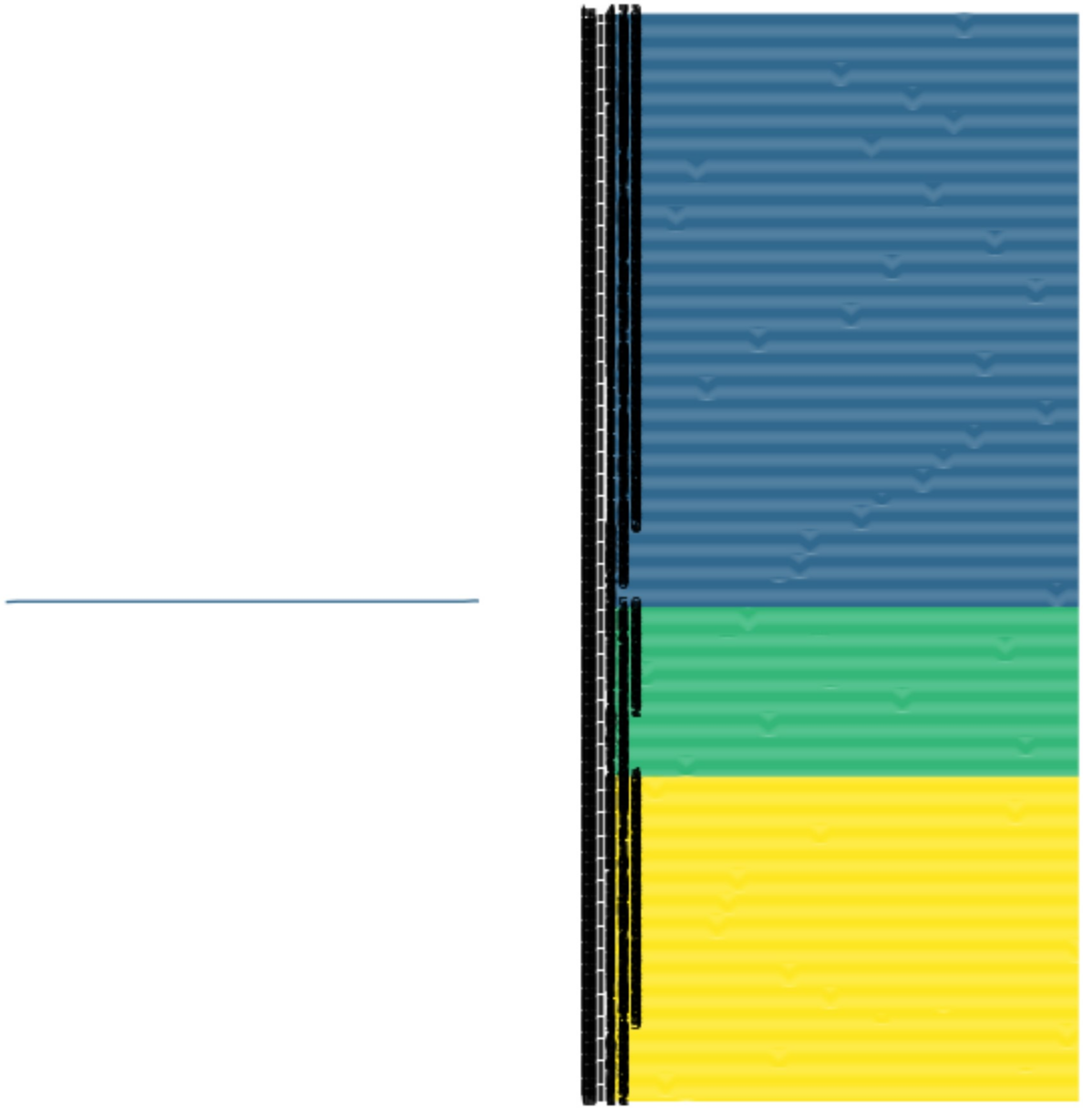
```
2022-08-26 21:15:21 be.kuleuven.dtai.distance WARNING: Warning: loading libra
ry to link with numpy returned an error
2022-08-26 21:15:21 be.kuleuven.dtai.distance WARNING: numpy.ndarray size cha
nged, may indicate binary incompatibility. Expected 96 from C header, got 88
from PyObject
 40%|████████████████████████
| 4/10 [01:08<01:42, 17.05s/it]
```

In [26]:

In [27]:
```
for i in cluster_idx.keys():
    for j in cluster_idx[i]:
```

In [28]:
```
# Plotting the clusters
import matplotlib.pyplot as plt
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10, 10))
show_ts_label = lambda idx: "ts-" + str(idx)
model.plot(axes=ax, show_ts_label=show_ts_label,
           show_tr_label=True, ts_label_margin=-10,
```

Out[28]:
```
(None, array([<AxesSubplot:>, <AxesSubplot:>], dtype=object))
```



In [29]:

In [30]:
```
class_data=class_data.groupby("Country").agg({"Clusters":pd.Series.mode,
In [31]: for i in range(3):
```

```
['Argentina', 'Austria', 'Bahrain', 'Croatia', 'Cyprus', 'Czech Republic (th
e)', 'Germany', 'Greece', 'Ireland', 'Italy', 'Sweden', 'United Kingdom of Gr
eat Britain and Northern Ireland (the)', 'United States of America (the)']
['Australia', 'Belgium', 'Denmark', 'France', 'Netherlands (the)', 'Spain', '
Switzerland']
['Algeria', 'Bangladesh', 'Brazil', 'Canada', 'Egypt', 'Hong Kong', 'India',
'Kazakhstan', 'Korea (the Republic of)', 'Lebanon', 'Malaysia', 'Malta', 'Mor
occo', 'Pakistan', 'Philippines (the)', 'Poland', 'Qatar', 'Romania', 'Russia
n Federation (the)', 'Saudi Arabia', 'Singapore', 'South Africa', 'Turkey', '
Ukraine', 'United Arab Emirates (the)']
```

### *Step14: Export the results into excel file for visualization*

```
In [32]:
```