Load files

In [12]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

datewise = pd.read_csv('datewise.csv')
datewise = pd.DataFrame(datewise)
```

In [13]:

```
datewise.head()
```

Out[13]:

	ObservationDate	Confirmed	Recovered	Deaths	Days Since	WeekOfYear	Mortalit Rat
0	2020-01-22	555.0	28.0	17.0	0 days 00:00:00.000000000	4	3.06306
1	2020-01-23	653.0	30.0	18.0	1 days 00:00:00.000000000	4	2.75650
2	2020-01-24	941.0	36.0	26.0	2 days 00:00:00.000000000	4	2.76301
3	2020-01-25	1438.0	39.0	42.0	3 days 00:00:00.000000000	4	2.92072
4	2020-01-26	2118.0	52.0	56.0	4 days 00:00:00.000000000	4	2.64400

In [14]:

```
datewise.index = datewise['ObservationDate']
datewise.index = pd.to_datetime(datewise.index)
datewise.drop(columns='ObservationDate')
```

Out[14]:

ObservationDate	Confirmed	Recovered	Deaths	Days Since	WeekOfYear	Mortality Rate
2020-01-22	555.0	28.0	17.0	0 days 00:00:00.000000000	4	3.063063
2020-01-23	653.0	30.0	18.0	1 days 00:00:00.000000000	4	2.756508
2020-01-24	941.0	36.0	26.0	2 days 00:00:00.000000000	4	2.763018
2020-01-25	1438.0	39.0	42.0	3 days 00:00:00.000000000	4	2.920723
2020-01-26	2118.0	52.0	56.0	4 days 00:00:00.000000000	4	2.644004
...	...	...	...	...	...	..
2020-06-25	9609829.0	4838921.0	489312.0	155 days 00:00:00.000000000	26	5.091787
2020-06-26	9801572.0	4945557.0	494181.0	156 days 00:00:00.000000000	26	5.041855
2020-06-27	9979535.0	5051864.0	498710.0	157 days 00:00:00.000000000	26	4.997327
2020-06-28	10145791.0	5140899.0	501893.0	158 days 00:00:00.000000000	26	4.946810
2020-06-29	10302151.0	5235813.0	505505.0	159 days 00:00:00.000000000	27	4.906791

160 rows × 9 columns

In [ ]:

In [ ]:

Prediction using Machine Learning Models

Linear Regression Model for Confirm Cases Prediction

In [15]:

```
datewise['Days Since'] = datewise.index - datewise.index[0]
datewise['Days Since'] = datewise['Days Since'].dt.days
```

In [82]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.95):]
model_scores=[]
```

In [83]:

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
```

In [84]:

```
import numpy as np

lin_reg.fit(np.array(train_ml['Days Since']).reshape(-1, 1),
            np.array(train_ml['Confirmed']).reshape(-1,1))
```

Out[84]:

```
LinearRegression(normalize=True)
```

In [85]:

```
prediction_valid_linreg = lin_reg.predict(np.array(valid_ml['Days Since']).reshape(-1,1))
```

In [86]:

```
from sklearn.metrics import mean_squared_error

model_scores.append(np.sqrt(mean_squared_error(valid_ml['Confirmed'],
                                               prediction_valid_linreg)))
print('Root Mean Square Error for Linear Regression: ',
      np.sqrt( mean_squared_error(valid_ml['Confirmed'], prediction_valid_linreg) ) )
```

```
Root Mean Square Error for Linear Regression:  2598361.511152013
```

In [87]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(11, 6))
prediction_linreg = lin_reg.predict(np.array(datewise['Days Since']).reshape(-1, 1))

linreg_output = []
for i in range(prediction_linreg.shape[0]):
    linreg_output.append(prediction_linreg[i][0])
```

```
<Figure size 792x432 with 0 Axes>
```

In [88]:

```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=linreg_output,
                         mode='lines', name='Linear Regression Best Fit Line',
              line = dict(color='black', dash='dot')))

fig.update_layout(title='Confirmed Cases Linear Regression Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                   legend=dict(x=0, y=1))

fig.show()
```

The Linear Regression Model is absolutely falling apart. As it is clearly visible that the trend of Confirmed Cases in absolutely not Linear.

In [89]:

```
from datetime import timedelta

new_date = []
new_prediction_lr = []

for i in range(1, 100):
    new_date.append(datewise.index[-1]+timedelta(days=i))
    new_prediction_lr.append(lin_reg.predict(np.array(datewise['Days Since'].max() + i).reshape(
-1,1))[0][0])
```

In [ ]:

```
```

Polynomial Regression for Prediction of Confirmed Cases

In [26]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.9):]
```

In [90]:

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 8)
```

In [91]:

```
train_poly=poly.fit_transform(np.array(train_ml["Days Since"]).reshape(-1,1))
valid_poly=poly.fit_transform(np.array(valid_ml["Days Since"]).reshape(-1,1))
y=train_ml["Confirmed"]
```

In [92]:

```
linreg=LinearRegression(normalize=True)
linreg.fit(train_poly,y)
```

Out[92]:

```
LinearRegression(normalize=True)
```

In [93]:

```
prediction_poly=linreg.predict(valid_poly)
rmse_poly=np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_poly))
model_scores.append(rmse_poly)
print("Root Mean Squared Error for Polynomial Regression: ",rmse_poly)
```

```
Root Mean Squared Error for Polynomial Regression:  339437.01978744625
```

In [94]:

```
comp_data=poly.fit_transform(np.array(datewise["Days Since"]).reshape(-1,1))
plt.figure(figsize=(11,6))
predictions_poly=linreg.predict(comp_data)

fig=go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                         mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=datewise.index, y=predictions_poly,
                         mode='lines',name="Polynomial Regression Best Fit",
                         line=dict(color='black', dash='dot')))
fig.update_layout(title="Confirmed Cases Polynomial Regression Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",
                  legend=dict(x=0,y=1,traceorder="normal"))
fig.show()
```

<Figure size 792x432 with 0 Axes>

In [95]:

```
new_prediction_poly = []

for i in range(1, 100):
    new_date_poly = poly.fit_transform(np.array(datewise['Days Since'].max()+i).reshape(-1,1))
    new_prediction_poly.append(linreg.predict(new_date_poly)[0])
```

In [96]:

```
pd.set_option('display.float_format', lambda x: '%.6f' % x)

model_predictions = pd.DataFrame(zip(new_date, new_prediction_lr, new_prediction_poly),
                                 columns=['Dates', 'Linear Regression Prediction', 'Polynomial Re
gression Prediction'])
model_predictions.head()
```

Out[96]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction
0	2020-06-30	7380259.723910	9798236.241235
1	2020-07-01	7438229.365146	9847282.240619
2	2020-07-02	7496199.006382	9881310.459843
3	2020-07-03	7554168.647618	9898732.751112
4	2020-07-04	7612138.288853	9897865.109540

In [ ]:

```

```

Support Vector Machine ModelRegressor for Prediction of Confirmed Cases

Takes too long times ,so skip it

In [22]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.9):]
```

In [23]:

```
from sklearn.svm import SVR

# Initializing SVR Model
svm = SVR(C=1, degree=5, kernel='poly', epsilon=0.01)
```

In [24]:

```
# Fitting model on the training data
svm.fit(np.array(train_ml['Days Since']).reshape(-1, 1), np.array(train_ml['Confirmed']).reshape
(-1,1))
```

Out[24]:

```
SVR(C=1, degree=5, epsilon=0.01, kernel='poly')
```

In [25]:

```
prediction_valid_svm = svm.predict(np.array(valid_ml['Days Since']).reshape(-1,1))
```

In [26]:

```
model_scores.append(np.sqrt(mean_square_error(valid_ml['Confirmed'], prediction_valid_svm)))
print('Root Mean Square Error for Support Vector Machine: ', np.sqrt(mean_squared_error(
    valid_ml['Confirmed'], prediction_valid_svm)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-26-78f770b00c7c> in <module>
----> 1 model_scores.append(np.sqrt(mean_square_error(valid_ml['Confirmed'], p
rediction_valid_svm)))
      2 print('Root Mean Square Error for Support Vector Machine: ', np.sqrt(mean_
squared_error(
      3     valid_ml['Confirmed'], prediction_valid_svm)))

NameError: name 'mean_square_error' is not defined
```

In [27]:

```
plt.figure(figsize=(11, 6))
prediction_svm = svm.predict(np.array(datewise['Days Since']).reshape(-1,1))

fig = go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=prediction_svm,
                         mode='lines', name='Support Vector Machine Best fit Kernel',
                         line=dict(color='black', dash='dot')))
fig.update_layout(title='Confirmed Cases Support Vector Machine Regressor Prediction',
                  xais_title='Date', yaxis_title='Confirmed Cases',
                   legend = dict(x=0,y=1, 'normal'))

fig.show()
```

```
  File "<ipython-input-27-5a59e741244f>", line 12
    legend = dict(x=0,y=1, 'normal'))
                           ^
SyntaxError: positional argument follows keyword argument
```

In [28]:

```
new_date = []
new_prediction_svm = []

for i in range(1, 100):
    new_date.append(datewise.index[-1]+timedelta(days=i))
    new_prediction_svm.append(svm.predict(np.array(datewise['Days Since'].max() + i).reshape(-1,
1))[0])
```

In [29]:

```
pd.set_option('display.float_format', lambda x: '%.6f' % x)

model_predictions = pd.DataFrame(zip(new_date_poly, new_prediction_lr, new_prediction_poly),
                                 columns=['Dates', 'Linear Regression Prediction', 'Polynomial Re
gression Prediction', 'SVM Prediction'])
model_predictions.head()
```

```
---------------------------------------------------------------------------
---
AssertionError                            Traceback (most recent call last)
~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
list_to_arrays(data, columns, coerce_float, dtype)
    499         result = _convert_object_array(
--> 500             content, columns, dtype=dtype, coerce_float=coerce_float
    501         )

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
convert_object_array(content, columns, coerce_float, dtype)
    582                 "{col:d} columns passed, passed data had "
--> 583                 "{con} columns".format(col=len(columns), con=len(co
ntent))
    584             )

AssertionError: 4 columns passed, passed data had 3 columns

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
<ipython-input-29-7d81091c768c> in <module>
      2
      3 model_predictions = pd.DataFrame(zip(new_date_poly, new_prediction_lr, new
_prediction_poly),
----> 4                                    columns=['Dates', 'Linear Regres
sion Prediction', 'Polynomial Regression Prediction', 'SVM Prediction'])
      5 model_predictions.head()

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩frame.py in __init__(self, dat
a, index, columns, dtype, copy)
    448                     if is_named_tuple(data[0]) and columns is None:
    449                         columns = data[0]._fields
--> 450                     arrays, columns = to_arrays(data, columns, dtype=
dtype)
    451                     columns = ensure_index(columns)
    452

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in t
o_arrays(data, columns, coerce_float, dtype)
    462         return [], []  # columns if columns is not None else []
    463     if isinstance(data[0], (list, tuple)):
--> 464         return _list_to_arrays(data, columns, coerce_float=coerce_floa
t, dtype=dtype)
    465     elif isinstance(data[0], abc.Mapping):
    466         return _list_of_dict_to_arrays(

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
list_to_arrays(data, columns, coerce_float, dtype)
    501         )
    502     except AssertionError as e:
--> 503         raise ValueError(e) from e
    504     return result
    505

ValueError: 4 columns passed, passed data had 3 columns
```

Predictions of Linear Regression are nowhere close to actual values.

In [ ]:

```
```

Time Series Forecasting

Holt's Linear Model

In [34]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid = datewise.iloc[int(datewise.shape[0]*0.9):]
y_pred = valid.copy()
```

In [97]:

```
from statsmodels.tsa.api import Holt, SimpleExpSmoothing, ExponentialSmoothing


holt = Holt(np.asarray(model_train['Confirmed'])).fit(
    smoothing_level=0.2, smoothing_slope=1.8, optimized=False)
```

In [98]:

```
y_pred['Holt'] = holt.forecast(len(valid))
model_scores.append(np.sqrt(mean_squared_error(y_pred['Confirmed'], y_pred['Holt'])))
print("Root Mean Square Error Holt's Linear Model: ", np.sqrt(mean_squared_error(y_pred['Confirm
ed'], y_pred['Holt'])))
```

```
Root Mean Square Error Holt's Linear Model:  182429.61166897498
```

In [99]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred['Holt'],
                         mode='lines+markers', name='Prediction of Confirmed Cases'))
fig.update_layout(title="Confirmed Cases Holt's Linear Model Prediction",
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                   legend = dict(x=0,y=1,traceorder='normal'))

fig.show()
```

In [100]:

```
from datetime import timedelta

holt_new_date = []
holt_new_prediction = []

for i in range(1, 100):
    holt_new_date.append(datewise.index[-1]+timedelta(days=i))
    holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holt's Linear Model Prediction"] = holt_new_prediction
model_predictions.head()
```

Out[100]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085

In [ ]:

```

```

Holt's Winter Model for Daily Time Series

In [39]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid = datewise.iloc[int(datewise.shape[0]*0.9):]
y_pred = valid.copy()
```

In [101]:

```
import warnings
warnings.filterwarnings('ignore')
es = ExponentialSmoothing(np.asarray(model_train['Confirmed']),
                          seasonal_periods = 11, trend='mul', seasonal='add').fit()
```

In [102]:

```
y_pred["Holt's Winter Model"] = es.forecast(len(valid))
model_scores.append(np.sqrt(mean_squared_error(y_pred['Confirmed'],
                                               y_pred["Holt's Winter Model"])))
print("Root Mean Square Error for Holt's Winter Model: ",
     np.sqrt(mean_squared_error(y_pred['Confirmed'], y_pred["Holt's Winter Model"])))
```

```
Root Mean Square Error for Holt's Winter Model:  52727.7197348411
```

In [103]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt's Winter Model"],
                         mode='lines+markers', name='Prediction of Confirmed Cases'))
fig.update_layout(title="Confirmed Cases Holt's Winter Model Prediction",
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend= dict(x=0,y=1, traceorder='normal'))

fig.show()
```

In [104]:

```
holt_winter_new_prediction = []

for i in range(1, 100):
    holt_winter_new_prediction.append(es.forecast((len(valid)+i))[-1])

model_predictions["Holt's Winter Model Prediction"] = holt_winter_new_prediction
model_predictions.head()
```

Out[104]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612

In [ ]:

```

```

AR Model (using AUTO ARIMA)

In [44]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid = datewise.iloc[int(datewise.shape[0]*0.9):]
y_pred = valid.copy()
```

In [105]:

```
from tqdm.notebook import tqdm   # 진행표시바
import time

for i in tqdm(range(100), desc="ing"):
    time.sleep(0.1)
```

In [106]:

```
from sys import executable
print(executable)
```

C:₩Users₩kheed₩Anaconda3₩python.exe

In [107]:

```
!cd
```

C:₩Users₩kheed₩Desktop₩PNU₩Senior₩학부연구생₩covid19

In [99]:

```
!pip install pmdarima
# !를 사용함으로써 커맨드창에서 입력하는 것과 동일하게 처리가 가능하다.
```

```
Requirement already satisfied: pmdarima in c:₩users₩kheed₩appdata₩roaming₩python₩p
ython37₩site-packages (1.7.0)
Requirement already satisfied: statsmodels>=0.11 in c:₩users₩kheed₩anaconda3₩lib₩s
ite-packages (from pmdarima) (0.11.1)
Requirement already satisfied: joblib>=0.11 in c:₩users₩kheed₩anaconda3₩lib₩site-p
ackages (from pmdarima) (0.13.2)
Requirement already satisfied: urllib3 in c:₩users₩kheed₩anaconda3₩lib₩site-packag
es (from pmdarima) (1.24.2)
Requirement already satisfied: Cython<0.29.18,>=0.29 in c:₩users₩kheed₩anaconda3₩l
ib₩site-packages (from pmdarima) (0.29.13)
Requirement already satisfied: scipy>=1.3.2 in c:₩users₩kheed₩anaconda3₩lib₩site-p
ackages (from pmdarima) (1.4.1)
Requirement already satisfied: pandas>=0.19 in c:₩users₩kheed₩anaconda3₩lib₩site-p
ackages (from pmdarima) (0.25.1)
Requirement already satisfied: scikit-learn>=0.22 in c:₩users₩kheed₩appdata₩roamin
g₩python₩python37₩site-packages (from pmdarima) (0.23.2)
Requirement already satisfied: numpy>=1.17.3 in c:₩users₩kheed₩anaconda3₩lib₩site-
packages (from pmdarima) (1.19.1)
Requirement already satisfied: patsy>=0.5 in c:₩users₩kheed₩anaconda3₩lib₩site-pac
kages (from statsmodels>=0.11->pmdarima) (0.5.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:₩users₩kheed₩anaconda3
₩lib₩site-packages (from pandas>=0.19->pmdarima) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:₩users₩kheed₩anaconda3₩lib₩site-p
ackages (from pandas>=0.19->pmdarima) (2019.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:₩users₩kheed₩appdata₩roam
ing₩python₩python37₩site-packages (from scikit-learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: six in c:₩users₩kheed₩anaconda3₩lib₩site-packages
(from patsy>=0.5->statsmodels>=0.11->pmdarima) (1.12.0)
```

In [100]:

```
!pip list
```

Package	Version
-tatsmodels	0.10.1
-umpy	1.16.5
absl-py	0.9.0
alabaster	0.7.12
anaconda-client	1.7.2
anaconda-navigator	1.9.7
anaconda-project	0.8.3
appdirs	1.4.4
asn1crypto	1.0.1
astroid	2.3.1
astropy	3.2.1
astunparse	1.6.3
atomicwrites	1.3.0
attrs	19.2.0
Babel	2.7.0
backcall	0.1.0
backports.functools-lru-cache	1.5
backports.os	0.1.1
backports.shutil-get-terminal-size	1.0.0
backports.tempfile	1.0
backports.weakref	1.0.post1
beautifulsoup4	4.8.0
bitarray	1.0.1
bkcharts	0.2
bleach	3.1.0
bokeh	1.3.4
boto	2.49.0
Bottleneck	1.2.1
cachetools	4.1.0
certifi	2019.9.11
cffi	1.12.3
chardet	3.0.4
Click	7.0
cloudpickle	1.2.2
clyent	1.2.2
colorama	0.4.1
comtypes	1.1.7
conda	4.7.12
conda-build	3.18.9
conda-package-handling	1.6.0
conda-verify	3.4.2
contextlib2	0.6.0
cryptography	2.7
cycler	0.10.0
Cython	0.29.13
cytoolz	0.10.0
dask	2.5.2
decorator	4.4.0
defusedxml	0.6.0
distlib	0.3.1
distributed	2.5.2
docutils	0.15.2
entrypoints	0.3
et-xmlfile	1.0.1
fastcache	1.1.0
filelock	3.0.12
Flask	1.1.1
fsspec	0.5.2
future	0.17.1

gast	0.3.3
gevent	1.4.0
glob2	0.7
google-auth	1.16.1
google-auth-oauthlib	0.4.1
google-pasta	0.2.0
greenlet	0.4.15
grpcio	1.29.0
h5py	2.10.0
HeapDict	1.0.1
html5lib	1.0.1
hupper	1.10.2
idna	2.8
imageio	2.6.0
imagesize	1.1.0
importlib-metadata	0.23
ipykernel	5.1.2
ipython	7.8.0
ipython-genutils	0.2.0
ipywidgets	7.5.1
isort	4.3.21
itsdangerous	1.1.0
jdcal	1.4.1
jedi	0.15.1
Jinja2	2.10.3
joblib	0.13.2
json5	0.8.5
jsonschema	3.0.2
jupyter	1.0.0
jupyter-client	5.3.3
jupyter-console	6.0.0
jupyter-core	4.5.0
jupyterlab	1.1.4
jupyterlab-server	1.0.6
Keras-Preprocessing	1.1.2
keyring	18.0.0
kiwisolver	1.1.0
lazy-object-proxy	1.4.2
libarchive-c	2.8
llvmlite	0.29.0
locket	0.2.0
lxml	4.4.1
Markdown	3.2.2
MarkupSafe	1.1.1
matplotlib	3.1.1
mccabe	0.6.1
menuinst	1.4.16
mistune	0.8.4
mkl-fft	1.0.14
mkl-random	1.1.0
mkl-service	2.3.0
mock	3.0.5
more-itertools	7.2.0
mpmath	1.1.0
msgpack	0.6.1
multipledispatch	0.6.0
navigator-updater	0.2.1
nbconvert	5.6.0
nbformat	4.4.0
networkx	2.3
nltk	3.4.5

nose	1.3.7
notebook	6.0.1
numba	0.45.1
numexpr	2.7.0
numpy	1.19.1
numpydoc	0.9.1
oauthlib	3.1.0
olefile	0.46
openpyxl	3.0.0
opt-einsum	3.2.1
packaging	19.2
pandas	0.25.1
pandocfilters	1.4.2
parso	0.5.1
partd	1.0.0
PasteDeploy	2.1.0
path.py	12.0.1
pathlib2	2.3.5
patsy	0.5.1
pep8	1.7.1
pickleshare	0.7.5
Pillow	6.2.0
pip	20.2.2
pkginfo	1.5.0.1
plaster	1.0
plaster-pastedeploy	0.7
plotly	4.8.1
pluggy	0.13.0
ply	3.11
pmdarima	1.7.0
prometheus-client	0.7.1
prompt-toolkit	2.0.10
protobuf	3.12.2
psutil	5.6.3
py	1.8.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycodestyle	2.5.0
pycosat	0.6.3
pycparser	2.19
pycrypto	2.6.1
pycurl	7.43.0.3
pyflakes	2.1.1
Pygments	2.4.2
pylint	2.4.2
PyMySQL	0.9.3
pyodbc	4.0.27
pyOpenSSL	19.0.0
pyparsing	2.4.2
pyramid	1.10.4
pyreadline	2.1
pyrsistent	0.15.4
PySocks	1.7.1
pytest	5.2.1
pytest-arraydiff	0.3
pytest-astropy	0.5.0
pytest-doctestplus	0.4.0
pytest-openfiles	0.4.0
pytest-remotedata	0.3.2
python-dateutil	2.8.0
pytz	2019.3

PyWavelets	1.0.3
pywin32	223
pywinpty	0.5.5
PyYAML	5.1.2
pyzmq	18.1.0
QtAwesome	0.6.0
qtconsole	4.5.5
QtPy	1.9.0
requests	2.22.0
requests-oauthlib	1.3.0
retrying	1.3.3
rope	0.14.0
rsa	4.0
ruamel-yaml	0.15.46
scikit-image	0.15.0
scikit-learn	0.23.2
scipy	1.4.1
seaborn	0.9.0
Send2Trash	1.5.0
setuptools	41.4.0
simplegeneric	0.8.1
singledispatch	3.4.0.3
six	1.12.0
snowballstemmer	2.0.0
sortedcollections	1.1.2
sortedcontainers	2.1.0
soupsieve	1.9.3
Sphinx	2.2.0
sphinxcontrib-applehelp	1.0.1
sphinxcontrib-devhelp	1.0.1
sphinxcontrib-htmlhelp	1.0.2
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.2
sphinxcontrib-serializinghtml	1.1.3
sphinxcontrib-websupport	1.1.2
spyder	3.3.6
spyder-kernels	0.5.2
SQLAlchemy	1.3.9
statsmodels	0.11.1
sympy	1.4
tables	3.5.2
tblib	1.4.0
tensorboard	2.2.2
tensorboard-plugin-wit	1.6.0.post3
tensorflow	2.2.0
tensorflow-estimator	2.2.0
termcolor	1.1.0
terminado	0.8.2
testpath	0.4.2
threadpoolctl	2.1.0
toolz	0.10.0
tornado	6.0.3
tqdm	4.36.1
traitlets	4.3.3
translationstring	1.4
unicodecsv	0.14.1
urllib3	1.24.2
venusian	3.0.0
virtualenv	20.0.25
wcwidth	0.1.7
webencodings	0.5.1

WebOb	1.8.6
Werkzeug	0.16.0
wheel	0.33.6
widgetsnbextension	3.5.1
win-inet-pton	1.1.0
win-unicode-console	0.5
wincertstore	0.2
wrapt	1.11.2
xlrd	1.2.0
XlsxWriter	1.2.1
xlwings	0.15.10
xlwt	1.3.0
zict	1.0.0
zipp	0.6.0
zope.deprecation	4.4.0
zope.interface	5.1.0

In [101]:

```
!conda list
```

```
# packages in environment at C:₩Users₩kheed₩Anaconda3:
#
# Name                    Version                   Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0                     py37_0
absl-py                   0.9.0                     pypi_0    pypi
alabaster                 0.7.12                    py37_0
anaconda                  2019.10                   py37_0
anaconda-client           1.7.2                     py37_0
anaconda-navigator        1.9.7                     py37_0
anaconda-project          0.8.3                       py_0
appdirs                   1.4.4                     pypi_0    pypi
asn1crypto                1.0.1                     py37_0
astroid                   2.3.1                     py37_0
astropy                   3.2.1            py37he774522_0
astunparse                1.6.3                     pypi_0    pypi
atomicwrites              1.3.0                     py37_1
attrs                     19.2.0                      py_0
babel                     2.7.0                       py_0
backcall                  0.1.0                     py37_0
backports                 1.0                         py_2
backports.functools_lru_cache 1.5                       py_2
backports.os              0.1.1                     py37_0
backports.shutil_get_terminal_size 1.0.0                    py37_2
backports.tempfile        1.0                         py_1
backports.weakref         1.0.post1                   py_1
beautifulsoup4            4.8.0                     py37_0
bitarray                  1.0.1            py37he774522_0
bkcharts                  0.2                       py37_0
blas                      1.0                         mkl
bleach                    3.1.0                     py37_0
blosc                     1.16.3               h7bd577a_0
bokeh                     1.3.4                     py37_0
boto                      2.49.0                    py37_0
bottleneck                1.2.1            py37h452e1ab_1
bzip2                     1.0.8                he774522_0
ca-certificates           2019.8.28                    0
cachetools                4.1.0                     pypi_0    pypi
certifi                   2019.9.11                 py37_0
cffi                      1.12.3           py37h7a1dbc1_0
chardet                   3.0.4                 py37_1003
click                     7.0                       py37_0
cloudpickle               1.2.2                       py_0
clyent                    1.2.2                     py37_1
colorama                  0.4.1                     py37_0
comtypes                  1.1.7                     py37_0
conda                     4.7.12                    py37_0
conda-build               3.18.9                    py37_3
conda-env                 2.6.0                        1
conda-package-handling    1.6.0            py37h62dcd97_0
conda-verify              3.4.2                       py_1
console_shortcut          0.1.1                        3
contextlib2               0.6.0                     py_0
cryptography              2.7              py37h7a1dbc1_0
curl                      7.65.3               h2a8f88b_0
cycler                    0.10.0                    py37_0
cython                    0.29.13          py37ha925a31_0
cytoolz                   0.10.0           py37he774522_0
dask                      2.5.2                       py_0
dask-core                 2.5.2                       py_0
decorator                 4.4.0                     py37_1
defusedxml                0.6.0                       py_0
```

distlib	0.3.1	pypi_0	pypi
distributed	2.5.2	py_0	
docutils	0.15.2	py37_0	
entrypoints	0.3	py37_0	
et_xmlfile	1.0.1	py37_0	
fastcache	1.1.0	py37he774522_0	
filelock	3.0.12	py_0	
flask	1.1.1	py_0	
freetype	2.9.1	ha9979f8_1	
fsspec	0.5.2	py_0	
future	0.17.1	py37_0	
gast	0.3.3	pypi_0	pypi
get_terminal_size	1.0.0	h38e98db_0	
gevent	1.4.0	py37he774522_0	
glob2	0.7	py_0	
google-auth	1.16.1	pypi_0	pypi
google-auth-oauthlib	0.4.1	pypi_0	pypi
google-pasta	0.2.0	pypi_0	pypi
greenlet	0.4.15	py37hfa6e2cd_0	
grpcio	1.29.0	pypi_0	pypi
h5py	2.10.0	pypi_0	pypi
hdf5	1.10.4	h7ebc959_0	
heapdict	1.0.1	py_0	
html5lib	1.0.1	py37_0	
hupper	1.10.2	pypi_0	pypi
icc_rt	2019.0.0	h0cc432a_1	
icu	58.2	ha66f8fd_1	
idna	2.8	py37_0	
imageio	2.6.0	py37_0	
imagesize	1.1.0	py37_0	
importlib_metadata	0.23	py37_0	
intel-openmp	2019.4	245	
ipykernel	5.1.2	py37h39e3cac_0	
ipython	7.8.0	py37h39e3cac_0	
ipython_genutils	0.2.0	py37_0	
ipywidgets	7.5.1	py_0	
isort	4.3.21	py37_0	
itsdangerous	1.1.0	py37_0	
jdcal	1.4.1	py_0	
jedi	0.15.1	py37_0	
jinja2	2.10.3	py_0	
joblib	0.13.2	py37_0	
jpeg	9b	hb83a4c4_2	
json5	0.8.5	py_0	
jsonschema	3.0.2	py37_0	
jupyter	1.0.0	py37_7	
jupyter_client	5.3.3	py37_1	
jupyter_console	6.0.0	py37_0	
jupyter_core	4.5.0	py_0	
jupyterlab	1.1.4	pyhf63ae98_0	
jupyterlab_server	1.0.6	py_0	
keras-preprocessing	1.1.2	pypi_0	pypi
keyring	18.0.0	py37_0	
kiwisolver	1.1.0	py37ha925a31_0	
krb5	1.16.1	hc04afaa_7	
lazy-object-proxy	1.4.2	py37he774522_0	
libarchive	3.3.3	h0643e63_5	
libcurl	7.65.3	h2a8f88b_0	
libiconv	1.15	h1df5818_7	
liblief	0.9.0	ha925a31_2	
libpng	1.6.37	h2a8f88b_0	

libsodium	1.0.16	h9d3ae62_0	
libssh2	1.8.2	h7a1dbc1_0	
libtiff	4.0.10	hb898794_2	
libxml2	2.9.9	h464c3ec_0	
libxslt	1.1.33	h579f668_0	
llvmlite	0.29.0	py37ha925a31_0	
locket	0.2.0	py37_1	
lxml	4.4.1	py37h1350720_0	
lz4-c	1.8.1.2	h2fa13f4_0	
lzo	2.10	h6df0209_2	
m2w64-gcc-libgfortran	5.3.0	6	
m2w64-gcc-libs	5.3.0	7	
m2w64-gcc-libs-core	5.3.0	7	
m2w64-gmp	6.1.0	2	
m2w64-libwinpthread-git	5.0.0.4634.697f757	2	
markdown	3.2.2	pypi_0	pypi
markupsafe	1.1.1	py37he774522_0	
matplotlib	3.1.1	py37hc8f65d3_0	
mccabe	0.6.1	py37_1	
menuinst	1.4.16	py37he774522_0	
mistune	0.8.4	py37he774522_0	
mkl	2019.4	245	
mkl-service	2.3.0	py37hb782905_0	
mkl_fft	1.0.14	py37h14836fe_0	
mkl_random	1.1.0	py37h675688f_0	
mock	3.0.5	py37_0	
more-itertools	7.2.0	py37_0	
mpmath	1.1.0	py37_0	
msgpack-python	0.6.1	py37h74a9793_1	
msys2-conda-epoch	20160418	1	
multipledispatch	0.6.0	py37_0	
navigator-updater	0.2.1	py37_0	
nbconvert	5.6.0	py37_1	
nbformat	4.4.0	py37_0	
networkx	2.3	py_0	
nltk	3.4.5	py37_0	
nose	1.3.7	py37_2	
notebook	6.0.1	py37_0	
numba	0.45.1	py37hf9181ef_0	
numexpr	2.7.0	py37hdce8814_0	
numpy	1.19.1	pypi_0	pypi
numpydoc	0.9.1	py_0	
oauthlib	3.1.0	pypi_0	pypi
olefile	0.46	py37_0	
openpyxl	3.0.0	py_0	
openssl	1.1.1d	he774522_2	
opt-einsum	3.2.1	pypi_0	pypi
packaging	19.2	py_0	
pandas	0.25.1	py37ha925a31_0	
pandoc	2.2.3.2	0	
pandocfilters	1.4.2	py37_1	
parso	0.5.1	py_0	
partd	1.0.0	py_0	
pastedeploy	2.1.0	pypi_0	pypi
path.py	12.0.1	py_0	
pathlib2	2.3.5	py37_0	
patsy	0.5.1	py37_0	
pep8	1.7.1	py37_0	
pickleshare	0.7.5	py37_0	
pillow	6.2.0	py37hdc69c19_0	
pip	20.2.2	pypi_0	pypi

pkginfo	1.5.0.1	py37_0	
plaster	1.0	pypi_0	pypi
plaster-pastedeploy	0.7	pypi_0	pypi
plotly	4.8.1	pypi_0	pypi
pluggy	0.13.0	py37_0	
ply	3.11	py37_0	
powershell_shortcut	0.0.1	2	
prometheus_client	0.7.1	py_0	
prompt_toolkit	2.0.10	py_0	
protobuf	3.12.2	pypi_0	pypi
psutil	5.6.3	py37he774522_0	
py	1.8.0	py37_0	
py-lief	0.9.0	py37ha925a31_2	
pyasn1	0.4.8	pypi_0	pypi
pyasn1-modules	0.2.8	pypi_0	pypi
pycodestyle	2.5.0	py37_0	
pycosat	0.6.3	py37hfa6e2cd_0	
pycparser	2.19	py37_0	
pycrypto	2.6.1	py37hfa6e2cd_9	
pycurl	7.43.0.3	py37h7a1dbc1_0	
pyflakes	2.1.1	py37_0	
pygments	2.4.2	py_0	
pylint	2.4.2	py37_0	
pymysql	0.9.3	pypi_0	pypi
pyodbc	4.0.27	py37ha925a31_0	
pyopenssl	19.0.0	py37_0	
pyparsing	2.4.2	py_0	
pyqt	5.9.2	py37h6538335_2	
pyramid	1.10.4	pypi_0	pypi
pyreadline	2.1	py37_1	
pyrsistent	0.15.4	py37he774522_0	
pysocks	1.7.1	py37_0	
pytables	3.5.2	py37h1da0976_1	
pytest	5.2.1	py37_0	
pytest-arraydiff	0.3	py37h39e3cac_0	
pytest-astropy	0.5.0	py37_0	
pytest-doctestplus	0.4.0	py_0	
pytest-openfiles	0.4.0	py_0	
pytest-remotedata	0.3.2	py37_0	
python	3.7.4	h5263a28_0	
python-dateutil	2.8.0	py37_0	
python-libarchive-c	2.8	py37_13	
pytz	2019.3	py_0	
pywavelets	1.0.3	py37h8c2d366_1	
pywin32	223	py37hfa6e2cd_1	
pywinpty	0.5.5	py37_1000	
pyyaml	5.1.2	py37he774522_0	
pyzmq	18.1.0	py37ha925a31_0	
qt	5.9.7	vc14h73c81de_0	
qtawesome	0.6.0	py_0	
qtconsole	4.5.5	py_0	
qtpy	1.9.0	py_0	
requests	2.22.0	py37_0	
requests-oauthlib	1.3.0	pypi_0	pypi
retrying	1.3.3	pypi_0	pypi
rope	0.14.0	py_0	
rsa	4.0	pypi_0	pypi
ruamel_yaml	0.15.46	py37hfa6e2cd_0	
scikit-image	0.15.0	py37ha925a31_0	
scikit-learn	0.21.3	py37h6288b17_0	
scipy	1.4.1	pypi_0	pypi

```
seaborn                   0.9.0              py37_0
send2trash                1.5.0              py37_0
setuptools                41.4.0             py37_0
simplegeneric             0.8.1              py37_2
singledispatch            3.4.0.3            py37_0
sip                       4.19.8          py37h6538335_0
six                       1.12.0             py37_0
snappy                    1.1.7             h777316e_3
snowballstemmer           2.0.0                py_0
sortedcollections         1.1.2              py37_0
sortedcontainers          2.1.0              py37_0
soupsieve                 1.9.3              py37_0
sphinx                    2.2.0                py_0
sphinxcontrib             1.0                py37_1
sphinxcontrib-applehelp   1.0.1                py_0
sphinxcontrib-devhelp     1.0.1                py_0
sphinxcontrib-htmlhelp    1.0.2                py_0
sphinxcontrib-jsmath      1.0.1                py_0
sphinxcontrib-qthelp      1.0.2                py_0
sphinxcontrib-serializinghtml 1.1.3                    py_0
sphinxcontrib-websupport  1.1.2                py_0
spyder                    3.3.6              py37_0
spyder-kernels            0.5.2              py37_0
sqlalchemy                1.3.9           py37he774522_0
sqlite                    3.30.0             he774522_0
statsmodels               0.10.1             pypi_0    pypi
sympy                     1.4                py37_0
tbb                       2019.4             h74a9793_0
tblib                     1.4.0                py_0
tensorboard               2.2.2              pypi_0    pypi
tensorboard-plugin-wit    1.6.0.post3        pypi_0    pypi
tensorflow                2.2.0              pypi_0    pypi
tensorflow-estimator      2.2.0              pypi_0    pypi
termcolor                 1.1.0              pypi_0    pypi
terminado                 0.8.2              py37_0
testpath                  0.4.2              py37_0
tk                        8.6.8             hfa6e2cd_0
toolz                     0.10.0               py_0
tornado                   6.0.3           py37he774522_0
tqdm                      4.36.1               py_0
traitlets                 4.3.3              py37_0
translationstring         1.4                pypi_0    pypi
unicodecsv                0.14.1             py37_0
urllib3                   1.24.2             py37_0
vc                        14.1              h0510ff6_4
venusian                  3.0.0              pypi_0    pypi
virtualenv                20.0.25            pypi_0    pypi
vs2015_runtime            14.16.27012       hf0eaf9b_0
wcwidth                   0.1.7              py37_0
webencodings              0.5.1              py37_1
webob                     1.8.6              pypi_0    pypi
werkzeug                  0.16.0               py_0
wheel                     0.33.6             py37_0
widgetsnbextension        3.5.1              py37_0
win_inet_pton             1.1.0              py37_0
win_unicode_console       0.5                py37_0
wincertstore              0.2                py37_0
winpty                    0.4.3                 4
wrapt                     1.11.2          py37he774522_0
xlrd                      1.2.0              py37_0
xlsxwriter                1.2.1                py_0
```

```
xlwings                   0.15.10             py37_0
xlwt                      1.3.0               py37_0
xz                        5.2.4            h2fa13f4_4
yaml                      0.1.7            hc54c509_2
zeromq                    4.3.1            h33f27b4_3
zict                      1.0.0                 py_0
zipp                      0.6.0                 py_0
zlib                      1.2.11           h62dcd97_3
zope-deprecation          4.4.0               pypi_0    pypi
zope-interface            5.1.0               pypi_0    pypi
zstd                      1.3.7            h508b16e_0
```

In [108]:

```
from pmdarima import *   # pyramid-arima 는 예전에 쓰던 모듈명
# from pmdarima import model_selection
```

In [109]:

```
model_ar = auto_arima(model_train['Confirmed'], trace=True, error_action='ignore',
                      start_p = 0, start_q=0, max_p=5, max_q=0,
                      suppress_warnings=True, stepwise=False, seasonal=False)
model_ar.fit(model_train['Confirmed'])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=3004.672, Time=0.01 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=2995.018, Time=0.02 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=2995.651, Time=0.03 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=2990.037, Time=0.06 sec
 ARIMA(4,2,0)(0,0,0)[0] intercept   : AIC=2985.449, Time=0.06 sec
 ARIMA(5,2,0)(0,0,0)[0] intercept   : AIC=2970.848, Time=0.10 sec
Total fit time: 0.314 seconds
```

Out[109]:

```
ARIMA(order=(5, 2, 0), scoring_args={}, suppress_warnings=True)
```

In [110]:

```
prediction_ar = model_ar.predict(len(valid))
y_pred["AR Model Prediction"] = prediction_ar
```

In [111]:

```
model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],
                                               y_pred["AR Model Prediction"])))
print("Root Mean Square Error for AR Model: ", np.sqrt(mean_squared_error(y_pred["Confirmed"],
                                                                          y_pred["AR Model Predic
tion"])))
```

```
Root Mean Square Error for AR Model:  199901.21425497934
```

In [112]:

```
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                    mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                    mode='lines+markers',name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["AR Model Prediction"],
                    mode='lines+markers',name="Prediction of Confirmed Cases",))
fig.update_layout(title="Confirmed Cases AR Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder
="normal"))
fig.show()
```

In [113]:

```
AR_model_new_prediction = []
for i in range(1, 100):
    AR_model_new_prediction.append(model_ar.predict(len(valid)+i)[-1])

model_predictions["AR Model Prediction"] = AR_model_new_prediction
model_predictions.head()
```

Out[113]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictio
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.56527
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.95027
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.18790
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.98348
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.27881

In [ ]:

```

```

In [ ]:

```

```

MA Model (using AUTO ARIMA)

In [54]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid = datewise.iloc[int(datewise.shape[0]*0.9):]
y_pred = valid.copy()
```

In [114]:

```
model_ma = auto_arima(model_train["Confirmed"], trace=True, error_action='ignore',
                      start_p=0, start_q=0, max_p=0, max_q=5,
                      suppress_warnings=True, stepwise=False, seasonal=False)
model_ma.fit(model_train["Confirmed"])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=3004.672, Time=0.01 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=2990.744, Time=0.04 sec
 ARIMA(0,2,2)(0,0,0)[0] intercept   : AIC=2995.515, Time=0.04 sec
 ARIMA(0,2,3)(0,0,0)[0] intercept   : AIC=3002.327, Time=0.16 sec
 ARIMA(0,2,4)(0,0,0)[0] intercept   : AIC=2992.933, Time=0.16 sec
 ARIMA(0,2,5)(0,0,0)[0] intercept   : AIC=2983.116, Time=0.12 sec
Total fit time: 0.559 seconds
```

Out[114]:

```
ARIMA(order=(0, 2, 5), scoring_args={}, suppress_warnings=True)
```

In [115]:

```
prediction_ma = model_ar.predict(len(valid))
y_pred["MA Model Prediction"] = prediction_ma
```

In [116]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["MA Model Prediction"],
                          mode='lines+markers', name="Prediction for Confirmed Cases"))

fig.update_layout(title='Confirmed Cases MA Model Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend=dict(x=0, y=1, traceorder='normal'))

fig.show()
```

In [117]:

```
MA_model_new_prediction = []

for i in range(1, 100):
    MA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])

model_predictions["MA Model Prediction"] = MA_model_new_prediction
model_predictions
```

Out[117]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Pre
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.
...	...	...	...	...	...	
94	2020-10-02	12829406.000087	-737621895.380941	22998915.129886	47802323.311642	27748386.
95	2020-10-03	12887375.641323	-768602963.472418	23136278.010528	48532856.405936	27980091.
96	2020-10-04	12945345.282559	-800575250.229044	23273640.891170	49274231.846084	28212718.
97	2020-10-05	13003314.923795	-833560667.977087	23411003.771812	50026595.166598	28446267.
98	2020-10-06	13061284.565031	-867581334.739752	23548366.652455	50790049.273035	28680737.

99 rows × 7 columns

In [ ]:

```
```

ARIMA Model (using AUTOARIMA)

In [120]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid = datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred = valid.copy()
```

In [121]:

```
model_arima = auto_arima(model_train['Confirmed'], trace=True, error_action='ignore',
                         start_p=1, start_q=1, max_p=3, max_q=3,
                         suppress_warnings=True, stepwise=False, seasonal=False)
model_arima.fit(model_train['Confirmed'])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=3238.491, Time=0.01 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=3200.833, Time=0.03 sec
 ARIMA(0,2,2)(0,0,0)[0] intercept   : AIC=3207.106, Time=0.05 sec
 ARIMA(0,2,3)(0,0,0)[0] intercept   : AIC=3212.557, Time=0.10 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=3199.257, Time=0.02 sec
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=3201.281, Time=0.06 sec
 ARIMA(1,2,2)(0,0,0)[0] intercept   : AIC=3184.943, Time=0.29 sec
 ARIMA(1,2,3)(0,0,0)[0] intercept   : AIC=3180.276, Time=0.36 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=3201.266, Time=0.03 sec
 ARIMA(2,2,1)(0,0,0)[0] intercept   : AIC=3203.272, Time=0.12 sec
 ARIMA(2,2,2)(0,0,0)[0] intercept   : AIC=3190.009, Time=0.41 sec
 ARIMA(2,2,3)(0,0,0)[0] intercept   : AIC=3181.137, Time=0.40 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=3194.263, Time=0.05 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=3188.798, Time=0.10 sec
 ARIMA(3,2,2)(0,0,0)[0] intercept   : AIC=inf, Time=nan sec
Total fit time: 2.635 seconds
```

Out[121]:

```
ARIMA(order=(1, 2, 3), scoring_args={}, suppress_warnings=True)
```

In [122]:

```
prediction_arima = model_arima.predict(len(valid))
y_pred["ARIMA Model Prediction"] = prediction_arima
```

In [123]:

```
model_scores.append(np.sqrt(mean_squared_error(valid['Confirmed'], prediction_arima)))
print("Root Mean Squares Error for ARIMA Model: ", np.sqrt(mean_squared_error(valid['Confirmed'
], prediction_arima)))
```

```
Root Mean Squares Error for ARIMA Model:  68691.38400632553
```

In [124]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["ARIMA Model Prediction"],
                          mode='lines+markers', name="Prediction for Confirmed Cases"))

fig.update_layout(title='Confirmed Cases ARIMA Model Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend=dict(x=0, y=1, traceorder='normal'))

fig.show()
```

In [125]:

```
ARIMA_model_new_prediction = []

for i in range(1, 100):
    ARIMA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])

model_predictions["ARIMA Model Prediction"] = ARIMA_model_new_prediction
model_predictions.head()
```

Out[125]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictio
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.56527
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.95027
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.18790
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.98348
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.27881

In [ ]:

```
```

SARIMA Model (using AUTO ARIMA)

In [126]:

```
model_sarima = auto_arima(model_train["Confirmed"], trace=True, error_action='ignore',
                          start_p=0, start_q=0, max_p=2, max_q=2, m=7,
                          suppress_warnings=True, stepwise=True, seasonal=True)
model_sarima.fit(model_train["Confirmed"])
```

```
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(1,0,1)[7]             : AIC=3234.221, Time=0.10 sec
 ARIMA(0,2,0)(0,0,0)[7]             : AIC=3237.815, Time=0.01 sec
 ARIMA(1,2,0)(1,0,0)[7]             : AIC=3187.681, Time=0.08 sec
 ARIMA(0,2,1)(0,0,1)[7]             : AIC=3199.411, Time=0.06 sec
 ARIMA(1,2,0)(0,0,0)[7]             : AIC=3200.554, Time=0.02 sec
 ARIMA(1,2,0)(2,0,0)[7]             : AIC=3184.908, Time=0.21 sec
 ARIMA(1,2,0)(2,0,1)[7]             : AIC=3189.470, Time=0.31 sec
 ARIMA(1,2,0)(1,0,1)[7]             : AIC=3176.520, Time=0.17 sec
 ARIMA(1,2,0)(0,0,1)[7]             : AIC=3192.735, Time=0.06 sec
 ARIMA(1,2,0)(1,0,2)[7]             : AIC=3178.504, Time=0.37 sec
 ARIMA(1,2,0)(0,0,2)[7]             : AIC=3190.957, Time=0.12 sec
 ARIMA(1,2,0)(2,0,2)[7]             : AIC=inf, Time=0.41 sec
 ARIMA(2,2,0)(1,0,1)[7]             : AIC=3176.978, Time=0.27 sec
 ARIMA(1,2,1)(1,0,1)[7]             : AIC=3176.185, Time=0.21 sec
 ARIMA(1,2,1)(0,0,1)[7]             : AIC=3194.392, Time=0.10 sec
 ARIMA(1,2,1)(1,0,0)[7]             : AIC=3185.692, Time=0.10 sec
 ARIMA(1,2,1)(2,0,1)[7]             : AIC=3184.884, Time=0.32 sec
 ARIMA(1,2,1)(1,0,2)[7]             : AIC=3178.131, Time=0.38 sec
 ARIMA(1,2,1)(0,0,0)[7]             : AIC=3202.055, Time=0.07 sec
 ARIMA(1,2,1)(0,0,2)[7]             : AIC=3188.845, Time=0.27 sec
 ARIMA(1,2,1)(2,0,0)[7]             : AIC=3181.420, Time=0.21 sec
 ARIMA(1,2,1)(2,0,2)[7]             : AIC=3179.300, Time=0.46 sec
 ARIMA(0,2,1)(1,0,1)[7]             : AIC=3181.556, Time=0.37 sec
 ARIMA(2,2,1)(1,0,1)[7]             : AIC=3175.874, Time=0.35 sec
 ARIMA(2,2,1)(0,0,1)[7]             : AIC=3192.915, Time=0.14 sec
 ARIMA(2,2,1)(1,0,0)[7]             : AIC=3185.495, Time=0.17 sec
 ARIMA(2,2,1)(2,0,1)[7]             : AIC=3185.212, Time=0.50 sec
 ARIMA(2,2,1)(1,0,2)[7]             : AIC=3177.818, Time=1.00 sec
 ARIMA(2,2,1)(0,0,0)[7]             : AIC=3201.573, Time=0.09 sec
 ARIMA(2,2,1)(0,0,2)[7]             : AIC=3188.620, Time=0.26 sec
 ARIMA(2,2,1)(2,0,0)[7]             : AIC=3181.573, Time=0.34 sec
 ARIMA(2,2,1)(2,0,2)[7]             : AIC=3179.033, Time=0.65 sec
 ARIMA(2,2,2)(1,0,1)[7]             : AIC=3166.525, Time=0.79 sec
 ARIMA(2,2,2)(0,0,1)[7]             : AIC=3180.290, Time=0.76 sec
 ARIMA(2,2,2)(1,0,0)[7]             : AIC=3174.759, Time=0.45 sec
 ARIMA(2,2,2)(2,0,1)[7]             : AIC=3175.636, Time=1.29 sec
 ARIMA(2,2,2)(1,0,2)[7]             : AIC=3168.344, Time=1.20 sec
 ARIMA(2,2,2)(0,0,0)[7]             : AIC=3187.060, Time=0.24 sec
 ARIMA(2,2,2)(0,0,2)[7]             : AIC=3177.625, Time=1.64 sec
 ARIMA(2,2,2)(2,0,0)[7]             : AIC=3171.853, Time=1.11 sec
 ARIMA(2,2,2)(2,0,2)[7]             : AIC=inf, Time=1.56 sec
 ARIMA(1,2,2)(1,0,1)[7]             : AIC=3169.733, Time=0.70 sec
 ARIMA(2,2,2)(1,0,1)[7] intercept   : AIC=3180.834, Time=0.68 sec

Best model:  ARIMA(2,2,2)(1,0,1)[7]
Total fit time: 18.720 seconds
```

Out[126]:

```
ARIMA(order=(2, 2, 2), scoring_args={}, seasonal_order=(1, 0, 1, 7),
      suppress_warnings=True, with_intercept=False)
```

In [127]:

```
prediction_sarima = model_sarima.predict(len(valid))
y_pred["SARIMA Model Prediction"] = prediction_sarima
```

In [134]:

```
model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"], y_pred["SARIMA Model Predict
ion"])))
print("Root Mean Square Error for SARIMA Model: ", np.sqrt(mean_squared_error(y_pred["Confirmed"
], y_pred["SARIMA Model Prediction"])))
```

```
Root Mean Square Error for SARIMA Model:  70212.49513155753
```

In [129]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["SARIMA Model Prediction"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.update_layout(title="Confirmed Cases SARIMA Model Prediction",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [130]:

```
SARIMA_model_new_prediction = []

for i in range(1, 100):
    SARIMA_model_new_prediction.append(model_sarima.predict(len(valid)+i)[-1])

model_predictions["SARIMA Model Prediction"] = SARIMA_model_new_prediction
model_predictions.head()
```

Out[130]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictio
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.56527
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.95027
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.18790
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.98348
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.27881

In [ ]:

In [ ]:

In [ ]:

Summarization of Forecasts using different Models

Case1 : Results Using 5% Validation Set

In [131]:

```
model_predictions_5 = model_predictions
model_predictions_5
```

Out[131]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Pre
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.
...	...	...	...	...	...	
94	2020-10-02	12829406.000087	-737621895.380941	22998915.129886	47802323.311642	27748386.
95	2020-10-03	12887375.641323	-768602963.472418	23136278.010528	48532856.405936	27980091.
96	2020-10-04	12945345.282559	-800575250.229044	23273640.891170	49274231.846084	28212718.
97	2020-10-05	13003314.923795	-833560667.977087	23411003.771812	50026595.166598	28446267.
98	2020-10-06	13061284.565031	-867581334.739752	23548366.652455	50790049.273035	28680737.

99 rows × 9 columns

In [135]:

```
print(list(zip(model_names, model_scores)))
```

```
[('Linear Regression', 2598361.511152013), ('Polynomial Regression', 339437.019787
44625), ("Holt's Linear", 182429.61166897498), ("Holt's Winter Model", 52727.71973
48411), ('Auto Regressive Model (AR)', 199901.21425497934), ('Moving Average Model
(MA)', 68691.38400632553), ('ARIMA Model', 70212.49513155753), ('SARIMA Model', 70
212.49513155753)]
```

In [136]:

```
model_names=["Linear Regression", "Polynomial Regression","Holt's Linear","Holt's Winter Model",
             "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Mode
l"]
model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model Name","Root Mean Square
d Error"]).sort_values(["Root Mean Squared Error"])
model_summary
```

Out[136]:

	Model Name	Root Mean Squared Error
3	Holt's Winter Model	52727.719735
5	Moving Average Model (MA)	68691.384006
6	ARIMA Model	70212.495132
7	SARIMA Model	70212.495132
2	Holt's Linear	182429.611669
4	Auto Regressive Model (AR)	199901.214255
1	Polynomial Regression	339437.019787
0	Linear Regression	2598361.511152

In [ ]:

```

```

In [ ]:

```

```

In [128]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                         mode='lines+markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Holt's Winter Model
 Prediction"],
                         mode='lines+markers', name="Holt's Winter Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["SARIMA Model Predict
ion"],
                         mode='lines+markers', name="SARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Holt's Linear Model
 Prediction"],
                         mode='lines+markers', name="Holt's Linear Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["AR Model Prediction"
],
                         mode='lines+markers', name="AR Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["MA Model Prediction"
],
                         mode='lines+markers', name="MA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["ARIMA Model Predicti
on"],
                         mode='lines+markers', name="ARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Linear Regression Pr
ediction"],
                         mode='lines+markers', name="Linear Regression Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Polynomial Regressio
n Prediction"],
                         mode='lines+markers', name="Polynomial Regression Model Prediction"))

fig.update_layout(title="Compare Prediction by the best model with 5% Validation Set",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=-1, traceorder="normal"))

fig.show()
```

In [ ]:

```
```

Case2 : Results Using 10% Validation Set

In [79]:

```
model_predictions_10 = model_predictions
model_predictions_10
```

Out[79]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	A Pr
0	2020-06-30	6899263.797007	5553059.654346	10086804.349516	10573460.804313	10083394
1	2020-07-01	6952906.075865	4970776.973441	10224167.230159	10761314.342326	10229086
2	2020-07-02	7006548.354724	4300888.528809	10361530.110801	10953361.048453	10375078
3	2020-07-03	7060190.633582	3535363.573804	10498892.991443	11147808.718753	10522468
4	2020-07-04	7113832.912440	2665639.382201	10636255.872085	11345481.962612	10669943
...	...	...	...	...	...	
94	2020-10-02	11941638.009679	-5339812100.397134	22998915.129886	47802323.311642	27748386
95	2020-10-03	11995280.288537	-5589944764.441430	23136278.010528	48532856.405936	27980091
96	2020-10-04	12048922.567395	-5850037808.690240	23273640.891170	49274231.846084	28212718
97	2020-10-05	12102564.846254	-6120422335.913735	23411003.771812	50026595.166598	28446267
98	2020-10-06	12156207.125112	-6401438387.148759	23548366.652455	50790049.273035	28680737

99 rows × 9 columns

In [80]:

```
model_scores
```

Out[80]:

```
[2665388.3159043086,
 1983232.8167772032,
 182429.61166897498,
 52727.7197348411,
 199901.21425497934,
 144600.732100827,
 235299.4332788909,
 235299.4332788909]
```

In [81]:

```
model_names=["Linear Regression", "Polynomial Regression","Holt's Linear","Holt's Winter Model",
             "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Mode
l"]
model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model Name","Root Mean Square
d Error"]).sort_values(["Root Mean Squared Error"])
model_summary
```

Out[81]:

	Model Name	Root Mean Squared Error
3	Holt's Winter Model	52727.719735
5	Moving Average Model (MA)	144600.732101
2	Holt's Linear	182429.611669
4	Auto Regressive Model (AR)	199901.214255
6	ARIMA Model	235299.433279
7	SARIMA Model	235299.433279
1	Polynomial Regression	1983232.816777
0	Linear Regression	2665388.315904

In [72]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                         mode='lines+markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Holt's Winter Model
 Prediction"],
                         mode='lines+markers', name="Holt's Winter Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["SARIMA Model Predict
ion"],
                         mode='lines+markers', name="SARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Holt's Linear Model
 Prediction"],
                         mode='lines+markers', name="Holt's Linear Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["AR Model Prediction"
],
                         mode='lines+markers', name="AR Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["MA Model Prediction"
],
                         mode='lines+markers', name="MA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["ARIMA Model Predicti
on"],
                         mode='lines+markers', name="ARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Linear Regression Pr
ediction"],
                         mode='lines+markers', name="Linear Regression Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Polynomial Regressio
n Prediction"],
                         mode='lines+markers', name="Polynomial Regression Model Prediction"))

fig.update_layout(title="Compare Prediction by the best model with 10% Validation Set",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=-1, traceorder="normal"))

fig.show()
```

In [ ]:

In [ ]:

The Best Model is "Holt's Winter Model"

In [137]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions_5["Dates"], y=model_predictions_5["Holt's Winter Mo
del Prediction"],
                         mode='markers', name="Holt's Winter Model Prediction with 5% Valid Set"
))
fig.add_trace(go.Scatter(x=model_predictions_10["Dates"], y=model_predictions_10["Holt's Winter
 Model Prediction"],
                         mode='markers', name="Holt's Winter Model Prediction with 10% Valid Set"
))

fig.update_layout(title="Compare The Best Model by Validation Set",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [ ]:

```

```

In [130]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=(datewise["Confirmed"]-datewise["Confirmed"].shift
().fillna(0)),
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions_5["Dates"], y=(model_predictions_5["Holt's Winter M
odel Prediction"]-model_predictions_5["Holt's Winter Model Prediction"].shift()),
                         mode='markers', name="Holt's Winter Model Prediction with 5% Valid Set"
))
fig.add_trace(go.Scatter(x=model_predictions_10["Dates"], y=(model_predictions_10["Holt's Winter
Model Prediction"]-model_predictions_10["Holt's Winter Model Prediction"].shift()),
                         mode='markers', name="Holt's Winter Model Prediction with 10% Valid Set"
))

fig.update_layout(title="Amount of Change by a day",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [131]:

```
datewise
```

Out[131]:

ObservationDate	ObservationDate	Confirmed	Recovered	Deaths	Days Since
2020-01-22	2020-01-22	555.000000	28.000000	17.000000	0
2020-01-23	2020-01-23	653.000000	30.000000	18.000000	1
2020-01-24	2020-01-24	941.000000	36.000000	26.000000	2
2020-01-25	2020-01-25	1438.000000	39.000000	42.000000	3
2020-01-26	2020-01-26	2118.000000	52.000000	56.000000	4
...	...	...	...	...	...
2020-06-25	2020-06-25	9609829.000000	4838921.000000	489312.000000	155
2020-06-26	2020-06-26	9801572.000000	4945557.000000	494181.000000	156
2020-06-27	2020-06-27	9979535.000000	5051864.000000	498710.000000	157
2020-06-28	2020-06-28	10145791.000000	5140899.000000	501893.000000	158
2020-06-29	2020-06-29	10302151.000000	5235813.000000	505505.000000	159

160 rows × 10 columns

In [132]:

```
datewise.shift()
```

Out[132]:

	ObservationDate	Confirmed	Recovered	Deaths	Da Sin
ObservationDate					
2020-01-22	NaN	nan	nan	nan	n
2020-01-23	2020-01-22	555.000000	28.000000	17.000000	0.0000
2020-01-24	2020-01-23	653.000000	30.000000	18.000000	1.0000
2020-01-25	2020-01-24	941.000000	36.000000	26.000000	2.0000
2020-01-26	2020-01-25	1438.000000	39.000000	42.000000	3.0000
...	...	...	...	...	
2020-06-25	2020-06-24	9430516.000000	4746118.000000	482753.000000	154.0000
2020-06-26	2020-06-25	9609829.000000	4838921.000000	489312.000000	155.0000
2020-06-27	2020-06-26	9801572.000000	4945557.000000	494181.000000	156.0000
2020-06-28	2020-06-27	9979535.000000	5051864.000000	498710.000000	157.0000
2020-06-29	2020-06-28	10145791.000000	5140899.000000	501893.000000	158.0000

160 rows × 10 columns

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Export file

In [134]:

```
model_predictions
```

Out[134]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Pre
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.
...	...	...	...	...	...	
94	2020-10-02	12829406.000087	-737621895.380941	22998915.129886	47802323.311642	27748386.
95	2020-10-03	12887375.641323	-768602963.472418	23136278.010528	48532856.405936	27980091.
96	2020-10-04	12945345.282559	-800575250.229044	23273640.891170	49274231.846084	28212718.
97	2020-10-05	13003314.923795	-833560667.977087	23411003.771812	50026595.166598	28446267.
98	2020-10-06	13061284.565031	-867581334.739752	23548366.652455	50790049.273035	28680737.

99 rows × 9 columns

In [135]:

```
model_predictions.to_csv('./model_predictions.csv', sep=',', na_rep='NaN')
```

In [ ]:

```

```