

Time Series Project

Forecasting the Total Daily Covid-19 Vaccinations Across the World

Team Members

1. Harsh Jaiswal (215280022)
2. Indresh Singh (215280011)
3. Anurag Yadav (215280003)
4. Neeraj Meena (215280014)

INTRODUCTION:

Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points intermittently or randomly. There are different types of models available for the analysis of time series data. We apply different types of models depending on the nature of the time series data set.

ABOUT THE DATASET:

The data is collected from OWID (Our World in Data) GitHub repository, which is updated on daily bases and we have collected it from Kaggle.

Link: <https://www.kaggle.com/rsrishav/covid-vaccination-dataset>

It contains the records of vaccination doses received by people from all the countries.

- location: name of the country (or region within a country).
- iso_code: ISO 3166-1 alpha-3 – three-letter country codes.
- **date**: date of the observation.
- total_vaccinations: total number of doses administered. This is counted as a single dose, and may not equal the total number of people vaccinated, depending on the specific dose regime (e.g. people receive multiple doses). If a person receives one dose of the vaccine, this metric goes up by 1. If they receive a second dose, it goes up by 1 again.
- total_vaccinations_per_hundred: total_vaccinations per 100 people in the total population of the country.

- **daily_vaccinations_raw**: daily change in the total number of doses administered. It is only calculated for consecutive days. This is a raw measure provided for data checks and transparency, but we strongly recommend that any analysis on daily vaccination rates be conducted using **daily_vaccinations** instead.
- **daily_vaccinations**: new doses administered per day (7-day smoothed). For countries that don't report data on a daily basis, we assume that doses changed equally on a daily basis over any periods in which no data was reported. This produces a complete series of daily figures, which is then averaged over a rolling 7-day window.
- **daily_vaccinations_per_million**: **daily_vaccinations** per 1,000,000 people in the total population of the country.
- **people_vaccinated**: total number of people who received at least one vaccine dose. If a person receives the first dose of a 2-dose vaccine, this metric goes up by 1. If they receive the second dose, the metric stays the same.
- **people_vaccinated_per_hundred**: **people_vaccinated** per 100 people in the total population of the country.
- **people_fully_vaccinated**: total number of people who received all doses prescribed by the vaccination protocol. If a person receives the first dose of a 2-dose vaccine, this metric stays the same. If they receive the second dose, the metric goes up by 1.
- **people_fully_vaccinated_per_hundred**: **people_fully_vaccinated** per 100 people in the total population of the country.

We have dataset of the COVID-19 vaccinated people on daily basis from February 22, 2021 to July 9, 2021. Thus, we have the dataset of dimension of 32411 rows and 12 columns.

OBJECTIVE:

The main objective of this project is to 'fit an appropriate time series model and to forecast the total number of COVID-19 vaccinated people across the world for the upcoming days.

Thus, we require a series of columns 'daily_vaccinations' and 'date' for our analysis.

IN THIS PROJECT WE INTEND TO:

- Cleaning out the data to obtain the series for our analysis.
- Perform Box-Cox method to stabilize the variance of the series.
- Test the stationarity of the series and make it stationary.
- Find out which model fits the data and estimate its parameters.
- Forecast the total daily covid-19 vaccinations across the world.

METHODOLOGY:

We use the Box-Jenkins methodology for the time series modelling and forecasting.

ARIMA:

An ARIMA model is a class of statistical model for analyzing and forecasting time series data. ARIMA is an acronym that stands for **AutoRegressive Integrated Moving Average**. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration.

- **AR: Autoregression.** A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I: Integrated.** The use of differencing of raw observations (i.e. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA: Moving Average.** A model that uses the dependency between an observation and residual errors from a moving average model applied to lagged observations.

A standard notation is used of $ARIMA(p,d,q)$ where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used. The parameters of the ARIMA model are defined as follows:

- **p :** The number of lag observations included in the model, also called the lag order.
- **d :** The number of times that the raw observations are differenced, also called the degree of differencing.
- **q :** The size of the moving average window, also called the order of moving average.

ARIMA(p,d,q) model is given by the equation below:

$$Y_t = c + \phi_1 Y_{t-1} + \dots + \phi_p Y_{t-p} + \theta_1 e_{t-1} + \dots + \theta_q e_{t-q} + e_t$$

Where:

- Y_t = the variable that will be explained in time t ;
- c = constant or intercept;
- ϕ = coefficient of each parameter p ;
- θ = coefficient of each parameter q ;
- e_t = Residuals or errors in time t .

BOX-JENKINS METHODOLOGY:

The approach starts with the assumption that the process that generated the time series can be approximated using an ARMA model if it is stationary or an ARIMA model if it is non-stationary. The 2016 5th edition of the textbook (Part Two, page 177) refers to the process as a stochastic model building and that it is an iterative approach that consists of the following 3 steps:

1. **Identification**
2. **Estimation**
3. **Diagnostic Checking**

It is an iterative process, so that as new information is gained during diagnostics, we can circle back to step 1 and incorporate that into new model classes.

1. IDENTIFICATION:

The identification step is further broken down into:

- Assess whether the time series is stationary, and if not, how many differences are required to make it stationary.
- Identify the parameters of an ARMA model for the data.

1.1 DIFFERENCING:

- **Unit Root Tests.** Use unit root statistical tests on the time series to determine whether or not it is stationary. Repeat after each round of differencing.
- **Avoid over differencing.** Differencing the time series more than is required can result in the addition of extra serial correlation and additional complexity.

1.2 CONFIGURING AR AND MA:

Two diagnostic plots can be used to help choose the p and q parameters of the ARMA or ARIMA. They are:

- **Autocorrelation Function (ACF).** The plot summarizes the correlation of an observation with lag values. The x-axis shows the lag and the y-axis shows the correlation coefficient between -1 and 1 for negative and positive correlation.

$$\rho_k = E[(Y_t - \mu)(Y_{t+k} - \mu)] / \sqrt{(E[(Y_t - \mu)^2]E[(Y_{t+k} - \mu)^2])}$$

- **Partial Autocorrelation Function (PACF).** The plot summarizes the correlations for an observation with lag values that is not accounted for by prior lagged observations.

Both plots are drawn as bar charts showing the 95% and 99% confidence intervals as horizontal lines. Bars that cross these confidence intervals are therefore more significant and worth noting.

- The model is AR if the ACF trails off after a lag and has a hard cut-off in the PACF after a lag. This lag is taken as the value for p .
- The model is MA if the PACF trails off after a lag and has a hard cut-off in the ACF after the lag. This lag value is taken as the value for q .
- The model is a mix of AR and MA if both the ACF and PACF trail off.

2. ESTIMATION:

We use the data to estimate the parameters coefficients of the model.

3. DIAGNOSTIC CHECKING:

The idea of diagnostic checking is to look for evidence that the model is not a good fit for the data e.g. residual errors diagnostics.

DATA CLEANING:

- First, we check the dimension of the dataset and we found that it has 32411 rows and 12 columns.
- We check for NaN values in the dataset.

```
location          0
iso_code          0
date              0
total_vaccinations 13317
people_vaccinated 14163
people_fully_vaccinated 17161
daily_vaccinations_raw 16266
daily_vaccinations 249
total_vaccinations_per_hundred 13317
people_vaccinated_per_hundred 14163
people_fully_vaccinated_per_hundred 17161
daily_vaccinations_per_million 249
```

- Now we replace all the NaN values by 0.
- We check for unique values in 'date' column.

```
2021-06-05    221
2021-06-02    221
2021-06-10    221
2021-06-03    221
2021-06-04    221
...
2020-12-07     6
2020-12-05     6
2020-12-04     6
2020-12-03     4
2020-12-02     4
Name: date, Length: 221
```

Which shows how many countries are having vaccination on a particular date.

- We drop the 'location' and 'iso_code' columns from dataset which is not necessary for us. So, dimension of the dataset reduces to 32411 rows and 10 columns.
- We make a list of numerical-value columns in dataset having length 9.

```
['total_vaccinations',
 'people_vaccinated',
 'people_fully_vaccinated',
 'daily_vaccinations_raw',
 'daily_vaccinations',
 'total_vaccinations_per_hundred',
 'people_vaccinated_per_hundred',
 'people_fully_vaccinated_per_hundred',
 'daily_vaccinations_per_million']
```

- Sort the dataframe by 'date' in ascending order.
- Group dataframe by 'date' and summing them on same date. After this, dimension of the dataframe reduces to 221 rows × 10 columns.
- Now we make our dataframe continuous by adding a random number using uniform (0,1).
- Take 'daily_vaccinations' and 'date' columns only in new dataframe for further analysis.

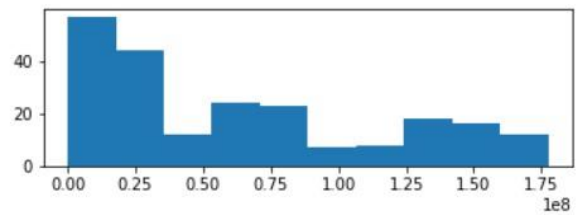
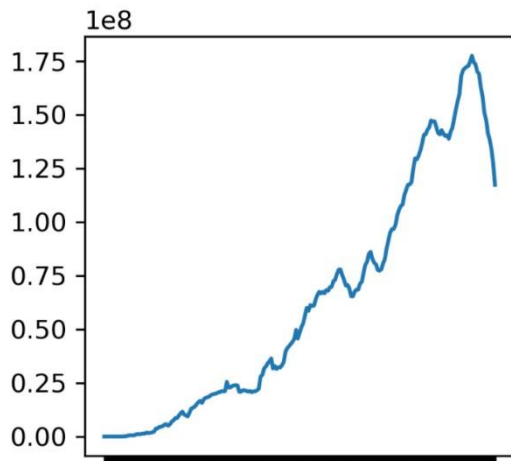
	daily_vaccinations	Date
0	4.985070e-01	2020-12-02
1	5.121923e-01	2020-12-03
2	3.733408e-01	2020-12-04
3	5.216472e-01	2020-12-05
4	5.475862e-01	2020-12-06
...
216	1.413055e+08	2021-07-06
217	1.385741e+08	2021-07-07
218	1.338450e+08	2021-07-08
219	1.271848e+08	2021-07-09
220	1.172732e+08	2021-07-10

221 rows × 2 columns

	daily_vaccinations
count	2.210000e+02
mean	6.232140e+07
std	5.455110e+07
min	1.650159e-01
25%	1.667417e+07
50%	4.970357e+07
75%	1.031357e+08
max	1.774500e+08

LINE PLOT AND HISTOGRAM OF 'daily_vaccinations':

Now, as we get our time series of 'daily_vaccinations' across the world on daily basis, we plot the line diagram to see the nature of the series. From the line plot, we can see that variance is not stable, hence we need to apply the 'Box-Cox' transformation on the series to stabilize the variance.



BOX-COX TRANSFORMATION FOR VARIANCE STABILIZATION OF THE SERIES:

A Box cox transformation is defined as a way to transform non-normal dependent variables in our data to a normal shape through which we can run a lot more tests than we could have. Box-cox Transformation only cares about computing the value of λ which varies from -5 to 5 . A value of λ is said to be best if it is able to approximate the non-normal curve to a normal curve. This function requires input to be positive.

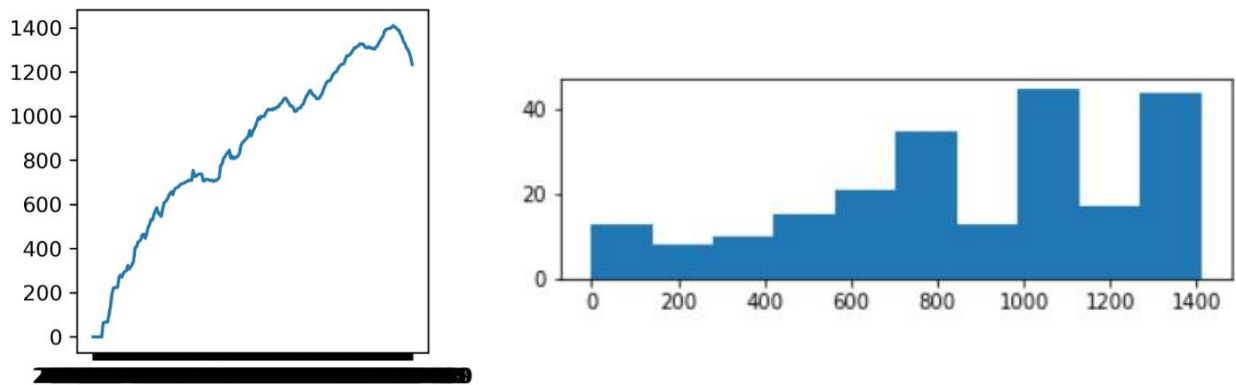
The transformation equation is as follows:

$$y(\lambda) = \begin{cases} (y^\lambda - 1) / \lambda & \text{if } \lambda \neq 0 \\ \log y & \text{if } \lambda = 0 \end{cases}$$

After applying the Box-Cox transformation, we get $\lambda = 0.322258$ and the series after the transformation updates as the following:

```

0          -0.623579
1          -0.601844
2          -0.844168
3          -0.587057
4          -0.547400
...
216       1309.856208
217       1301.623385
218       1287.105166
219       1266.057016
220       1233.303050
Name: daily_vaccinations, Length: 221, dtype: float64
```



Above plots represents the line plot and histogram of the updated series after applying the transformation.

DIVIDING THE SERIES INTO TRAINING AND TEST SETS:

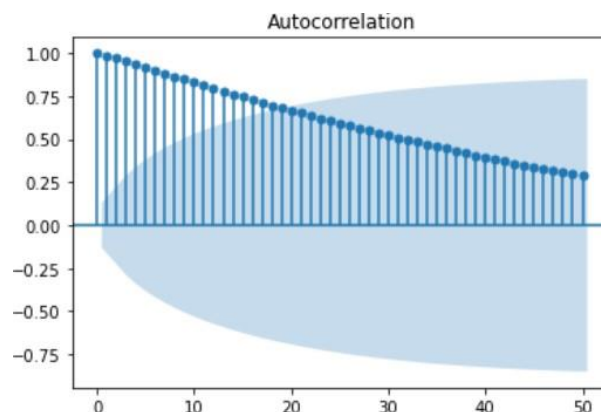
We divide our series into two parts - training and test set. Training and test sets are disjoint portions of the series which is used to fit the model while the test set is used to see how accurate our fitted model is. Here, train set contains first 154 observations (approx. 70% of total) and test set contains remaining last 67 observations (approx. 30% of total) of the series.

ACF PLOT OF THE SERIES WITH NO DIFFERENCING UPTO LAG 50:

ACF (auto-correlation function) is a function which gives us values of auto-correlation of any series with its lagged values. We plot these values along with the confidence band. In simple terms, it describes how well the present value of the series is related with its past values.

In time series analysis, the partial autocorrelation function (PACF) gives the partial correlation of a stationary time series with its own lagged values, regressed the values of the time series at all shorter lags.

We plot the ACF plot of our series and see that it goes to zero in linear fashion, suggesting that series is non-stationary. Now, we check for the stationarity of the series by different tests.



TESTING THE STATIONARITY OF THE SERIES

Time series is said to be stationary if they do not have trend or seasonal effects. Summary statistics calculated on the time series are consistent over time, like the mean or the variance of the observations. When the time series is not stationary then it is called non-stationary time series.

When a time series is stationary, it can be easier to model. Statistical modelling methods assume or require the time series to be stationary to be effective. We use ADF and KPSS test for this purpose.

- **ADF test:**

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

Null Hypothesis (H0): series has a unit root, that is series is not stationary.

Alternate Hypothesis (H1): series does not have a unit root, that is series is stationary.

We apply the ADF test on the series with no differencing and see that the p-value is 0.029776 which is less than 0.05 level of significance, hence we reject the null hypothesis and conclude that the series is stationary.

```
ADF Statistic: -3.058553
p-value: 0.029776
Critical Values:
  1%: -3.461
  5%: -2.875
 10%: -2.574
```

- **KPSS test:**

The KPSS test, short for, Kwiatkowski-Phillips-Schmidt-Shin (KPSS), is a type of Unit root test that tests for the stationarity of a given series around a deterministic trend.

Null Hypothesis (H0): series does not have a unit root, that is series is stationary.

Alternate Hypothesis (H1): series has a unit root, that is series is not stationary.

We apply the KPSS test on the series with no differencing and see that the p-value is 0.01 which is less than 0.05 level of significance, hence we reject the null hypothesis and conclude that the series is non-stationary.

```
(1.4238626879574532, 0.01, 15, {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

As we can see that, ADF and KPSS tests give different results, conclude that the series is non-stationary. For making the series stationary, we need differencing of it.

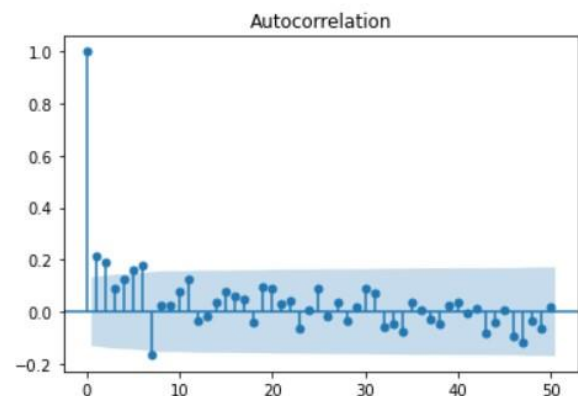
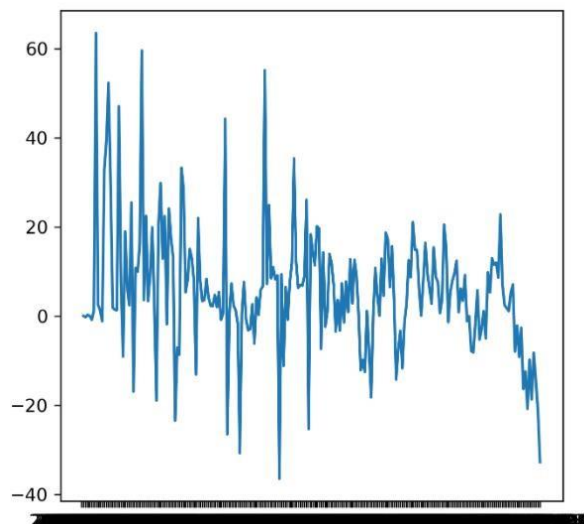
FIRST ORDER DIFFERENCING OF THE SERIES:

In the below table, the column 'daily_vaccinations_with_diff_1' shows the series with order of differencing 1.

	daily_vaccinations	Date	daily_vaccinations_with_diff_1
0	-0.623579	2020-12-02	NaN
1	-0.601844	2020-12-03	0.021735
2	-0.844168	2020-12-04	-0.242324
3	-0.587057	2020-12-05	0.257111
4	-0.547400	2020-12-06	0.039657
...
216	1309.856208	2021-07-06	-18.686686
217	1301.623385	2021-07-07	-8.232824
218	1287.105166	2021-07-08	-14.518219
219	1266.057016	2021-07-09	-21.048151
220	1233.303050	2021-07-10	-32.753966

221 rows × 3 columns

LINE PLOT OF THE TIME SERIES WITH ORDER OF DIFFERENCING 1 AND IT'S ACF PLOT:



By looking at the ACF plot, we can see that it cuts off the region and goes to zero quickly which is kind of sign of stationarity of the series. Now, we again apply the ADF and KPSS tests to check the stationarity of the series with order of differencing 1.

TESTING THE STATIONARITY OF THE SERIES WITH ORDER OF DIFFERENCING 1:

- **ADF test:**

Null Hypothesis (H0): series has a unit root, that is the series is not stationary.

Alternate Hypothesis (H1): series does not have a unit root, that is the series is stationary.

We apply the ADF test on the series with first order differencing and see that the pvalue is 0.225578 which is greater than 0.05 level of significance, hence we fail to reject the null hypothesis and conclude that the series is non-stationary.

```
ADF Statistic: -2.148205
p-value: 0.225578
Critical Values:
  1%: -3.462
  5%: -2.875
 10%: -2.574
```

- **KPSS test:**

Null Hypothesis (H0): series does not have a unit root, that is the series is stationary.

Alternate Hypothesis (H1): series has a unit root, that is the series is non-stationary.

We apply the KPSS test on the series with first order differencing and see that the pvalue is 0.01948 which is less than 0.05 level of significance, hence we reject the null hypothesis and conclude that the series is non-stationary.

```
(0.6346958167554975, 0.019482198476772954, 15, {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

As both the tests results in non-stationarity of the series with order of differencing 1, we need to difference it again and check for the stationarity.

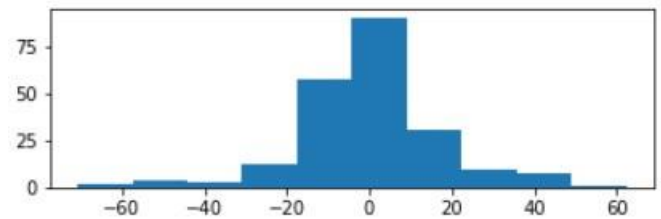
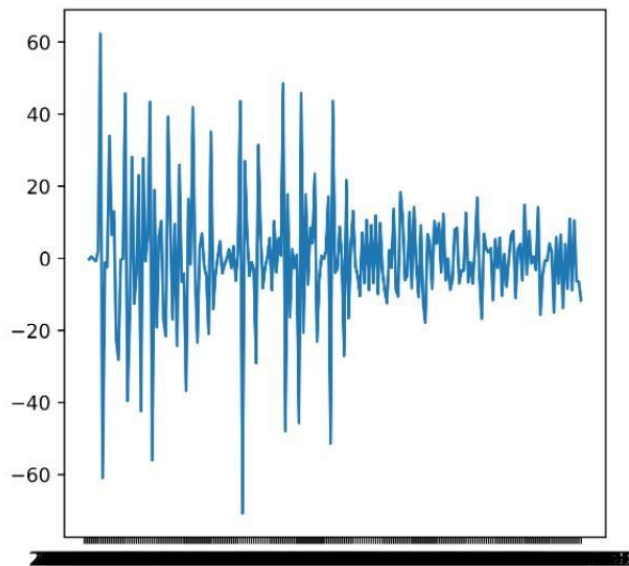
SECOND ORDER DIFFERENCING OF THE SERIES:

As series is not stationary with order of differencing 1, We need to perform 2nd order differencing on the series which is given below in the column 'daily_vaccinations_with_diff_2'.

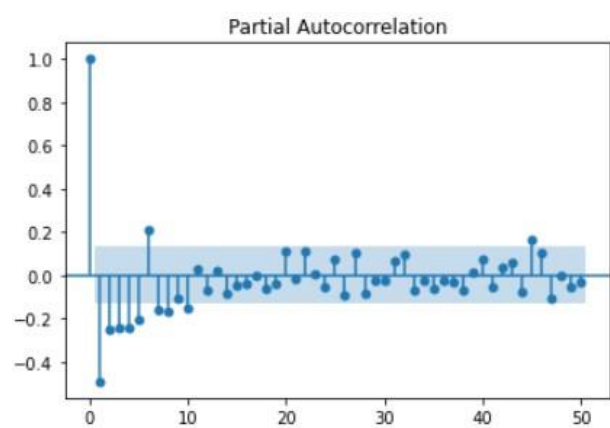
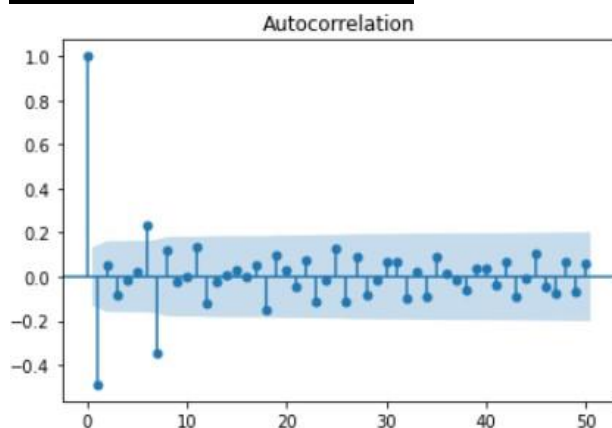
	daily_vaccinations	Date	daily_vaccinations_with_diff_1	daily_vaccinations_with_diff_2
0	-0.623579	2020-12-02	NaN	NaN
1	-0.601844	2020-12-03	0.021735	NaN
2	-0.844168	2020-12-04	-0.242324	-0.264059
3	-0.587057	2020-12-05	0.257111	0.499435
4	-0.547400	2020-12-06	0.039657	-0.217454
...
216	1309.856208	2021-07-06	-18.686686	-8.863791
217	1301.623385	2021-07-07	-8.232824	10.453863
218	1287.105166	2021-07-08	-14.518219	-6.285395
219	1266.057016	2021-07-09	-21.048151	-6.529932
220	1233.303050	2021-07-10	-32.753966	-11.705815

221 rows × 4 columns

LINE PLOT AND HISTOGRAM OF THE SERIES WITH 2ND ORDER OF DIFFERENCING:



ACF AND PACF PLOTS OF THE SERIES WITH 2ND ORDER OF DIFFERENCING:



TESTING THE STATIONARITY OF THE SERIES WITH 2ND ORDER DIFFERENCING:

- ADF test:

Null Hypothesis (H0): series has a unit root, that is the series is not stationary.

Alternate Hypothesis (H1): series does not have a unit root, that is the series is stationary.

We apply the ADF test on the series with 2nd order of differencing and see that the pvalue is 0.00 which is less than 0.05 level of significance, hence we reject the null hypothesis and conclude that the series is now stationary.

```

ADF Statistic: -7.989885
p-value: 0.000000
Critical Values:
1%: -3.462
5%: -2.875
10%: -2.574

```

- KPSS test:

Null Hypothesis (H0): series does not have a unit root, that is the series is stationary.

Alternate Hypothesis (H1): series have a unit root, that is the series is non-stationary.

We apply the KPSS test on the series with 2nd order differencing and see that the pvalue is 0.1 which is greater than 0.05 level of significance, hence we fail to reject the null hypothesis and conclude that the series is stationary.

(0.12533126967433905, 0.1, 15, {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})

With the second order of differencing our series has become stationary. We can proceed for the modelling stage.

ESTIMATING AR AND MA PARAMETERS:

Further, as we got our series is stationary with 2nd order of differencing, we find out the appropriate order of (p, q). We'll fit the ARIMA model of order (p, d=2, q) and calculate the value of Akaike Information Criteria and consider the corresponding value of p and q for which AIC is minimum.

d=2				
p\q	1	2	3	4
1	1269.365	1271.681	1272.102	1274.099
2	1270.982	1272.103	1265.547	1267.540
3	1272.543	1274.097	1267.541	1269.504
4	1274.510	1276.085	1269.404	1271.223

Value of minimum AIC: 1265.547

Values of estimated p and q: 2, 3

Thus, we obtain ARIMA (2,2,3)

ARIMA (2,2,3) MODEL RESULTS:

ARIMA Model Results						
=====						
Dep. Variable:	D2.daily_vaccinations	No. Observations:	152			
Model:	ARIMA(2, 2, 3)	Log Likelihood	-625.774			
Method:	css-mle	S.D. of innovations	14.404			
Date:	Fri, 13 Aug 2021	AIC	1265.547			
Time:	23:44:53	BIC	1286.715			
Sample:	2	HQIC	1274.146			
=====						
	coef	std err	z	P> z	[0.025	0.975]

const	-0.0725	0.030	-2.440	0.015	-0.131	-0.014
ar.L1.D2.daily_vaccinations	-0.2383	0.061	-3.886	0.000	-0.358	-0.118
ar.L2.D2.daily_vaccinations	-0.8098	0.056	-14.489	0.000	-0.919	-0.700
ma.L1.D2.daily_vaccinations	-0.6876	0.028	-24.226	0.000	-0.743	-0.632
ma.L2.D2.daily_vaccinations	0.6874	0.033	20.544	0.000	0.622	0.753
ma.L3.D2.daily_vaccinations	-0.9999	0.037	-26.963	0.000	-1.073	-0.927

Roots				
	Real	Imaginary	Modulus	Frequency
AR.1	-0.1471	-1.1015j	1.1113	-0.2711
AR.2	-0.1471	+1.1015j	1.1113	0.2711
MA.1	1.0000	-0.0000j	1.0000	-0.0000
MA.2	-0.1562	-0.9878j	1.0001	-0.2750
MA.3	-0.1562	+0.9878j	1.0001	0.2750

The obtained **ARIMA (2,2,3)** model is given by:

$$Y_t = (-0.0725) - (0.2383)Y_{t-1} - (0.8098)Y_{t-2} + \varepsilon_t - (0.6876)\varepsilon_{t-1} + (0.6874)\varepsilon_{t-2} - (0.9999)\varepsilon_{t-3}$$

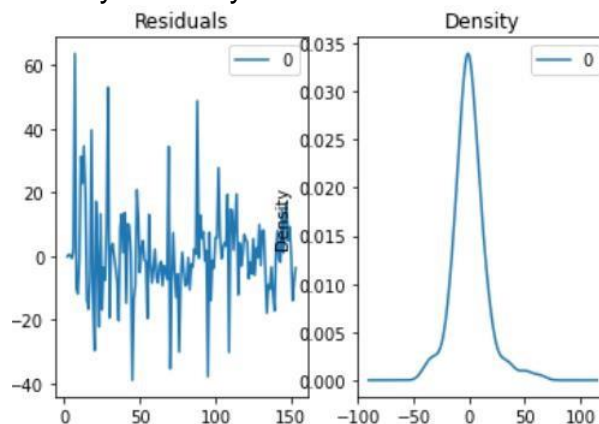
CORRELATION MATRIX OF THE PARAMETERS ESTIMATES:

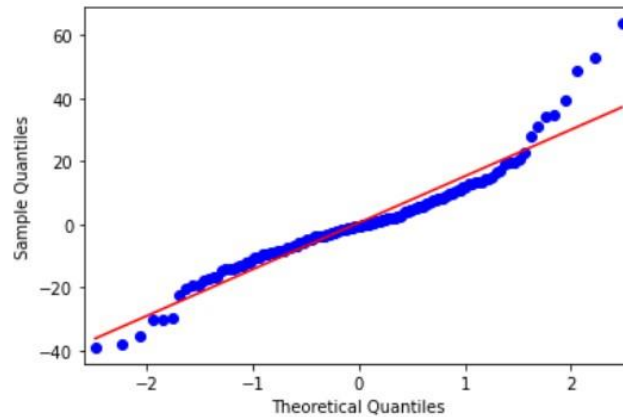
```
[ [ 1.00000000e+00 -4.58600679e-03 2.65377603e-02 6.50590754e-03 -5.60483681e-03 7.61952248e-05]
[ -4.58600679e-03 1.00000000e+00 6.59723438e-02 -3.99086634e-01 3.37797371e-01 6.48927910e-04]
[ 2.65377603e-02 6.59723438e-02 1.00000000e+00 1.24565244e-01 -1.08142475e-01 2.23521059e-03]
[ 6.50590754e-03 -3.99086634e-01 1.24565244e-01 1.00000000e+00 -1.38282521e-01 2.32334624e-01]
[ -5.60483681e-03 3.37797371e-01 -1.08142475e-01 -1.38282521e-01 1.00000000e+00 -5.65097784e-01]
[ 7.61952248e-05 6.48927910e-04 2.23521059e-03 2.32334624e-01 -5.65097784e-01 1.00000000e+00]]
```

We can see from correlation matrix that all the parameters are uncorrelated.

RESIDUALS DIAGNOSTICS:

Below are the line plot and density plot and Q-Q plot of the residuals. All represents that the residuals are nearly normally distributed around mean zero.





ANDERSON-DARLING TEST FOR NORMALITY OF THE RESIDUAL DISTRIBUTION:

The Anderson-Darling test (Stephens, 1974) is used to test if a sample of data came from a population with a specific distribution. It is a modification of the Kolmogorov-Smirnov (K-S) test and gives more weight to the tails than does the K-S test. The K-S test is distribution free in the sense that the critical values do not depend on the specific distribution being tested (note that this is true only for a fully specified distribution, i.e. the parameters are known). The Anderson-Darling test makes use of the specific distribution in calculating critical values. The Anderson-Darling test is an alternative to the chi-square and Kolmogorov-Smirnov goodness-of-fit tests.

The Anderson-Darling test is defined as:

H_0 : The data follow a specified distribution.

Test statistics, A is given by,

$$A^2 = -n - S,$$

where

$$S = \sum_{i=1}^n \frac{2i-1}{n} [\ln(F(Y_i)) + \ln(1 - F(Y_{n+1-i}))].$$

H_a : The data do not follow the specified distribution

Where,

F is the cumulative distribution function of the specified distribution. **Y_i** are the *ordered* statistics.

We reject the null hypothesis if the p-value is less than 0.05 level of significance which allows us to claim that our data is statistically different from a specified distribution. On the other hand, if p-value is higher than 0.05, we can state that our data is not statistically different from a specified distribution.

Critical values in a statistical test are a range of pre-defined significance boundaries at which the H_0 can be failed to be rejected if the calculated statistic is less than the critical value.

Rather than just a single p-value, the test returns a critical value for a range of different commonly used significance levels.

Statistic: 2.810

Significance level: critical values

15.000: 0.562, data does not look normal (reject H_0)

10.000: 0.640, data does not look normal (reject H_0)

5.000: 0.768, data does not look normal (reject H_0)

2.500: 0.895, data does not look normal (reject H_0)

1.000: 1.065, data does not look normal (reject H_0)

We apply the Anderson-Darling test and find that at each significance level, p values are lesser than critical value and we conclude that the residuals do not follow a normal distribution.

SHAPIRO-WILK TEST FOR NORMALITY TESTING OF RESIDUAL DISTRIBUTION:

The Shapiro–Wilk test tests the null hypothesis that a sample x_1, \dots, x_n come from a normally distributed population.

Statistics:

$$W = \frac{(\sum_{i=1}^n a_i x_{(i)})^2}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

Where,

$x_{(i)}$ is the i^{th} order statistic.

The coefficients a_i are given by:^[1]

$$(a_1, \dots, a_n) = \frac{m^T V^{-1}}{C},$$

where C is a vector norm:^[2]

$$C = \|V^{-1}m\| = (m^T V^{-1} V^{-1} m)^{1/2}$$

and the vector m ,

$$m = (m_1, \dots, m_n)^T$$

Which is made of the expected values of the order statistics of independent and identically distributed random variables sampled from the standard normal distribution and V is the covariance matrix of those normal order statistics.

Null Hypothesis (H0): residuals are normally distributed.

Alternate Hypothesis (H1): residuals are not normally distributed.

If the p value is less than 0.05, then the null hypothesis is rejected means that there is evidence that the residuals are not normally distributed. On the other hand, if the p value is greater than the 0.05, then the null hypothesis cannot be rejected means there is evidence that the residuals are normally distributed.

Statistics=0.931, $p=0.000$

We see that p value is less than 0.05 level of significance and hence we reject the null hypothesis and conclude that residuals are not normally distributed.

MANN-WHITNEY U TEST (non-parametric test):

Null hypothesis H_0 : the distributions of two populations are equal.

Alternative hypothesis H_1 : the distributions are not equal.

Let X_1, X_2, \dots, X_n be an i.i.d. random sample from population X , and Y_1, Y_2, \dots, Y_m be an i.i.d. random sample from population Y , and both samples independent of each other. The corresponding Mann-Whitney U statistic is defined as:

$$U = \sum_{i=1}^n \sum_{j=1}^m S(X_i, Y_j),$$

with

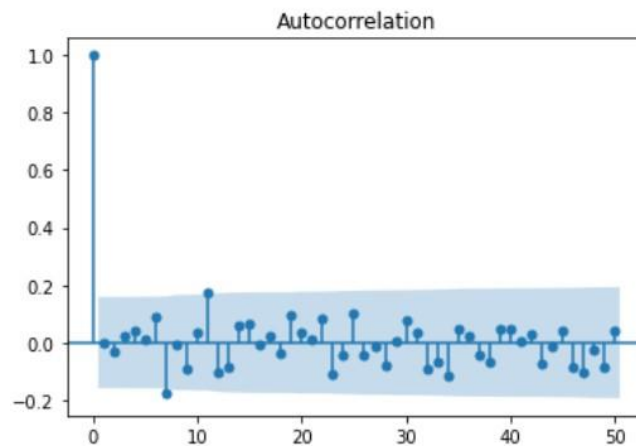
$$S(X, Y) = \begin{cases} 1, & \text{if } Y < X, \\ \frac{1}{2}, & \text{if } Y = X, \\ 0, & \text{if } Y > X. \end{cases}$$

Here, we have taken a random sample from $N(0,1)$ to compare the distribution of residuals.

Statistics=4824.000, $p=0.268$

After applying the test, we can see that p value is 0.268 which is greater than 0.05 level of significance hence we fail to reject the null hypothesis and conclude that both samples have the same distribution that means residuals are normally distributed.

ACF PLOT OF RESIDUALS:



LJUNG BOX TEST TO CHECK WHETHER RESIDUALS ARE CORRELATED AT DIFFERENT LAGS:

Ljung Box test is a test for testing the *absence* of serial autocorrelation, up to a specified lag k .

Null hypothesis H_0 : residuals are uncorrelated.

Alternate hypothesis H_a : residuals are correlated.

Test statistics:

$$Q(m) = n(n+2) \sum_{j=1}^m \frac{r_j^2}{n-j},$$

Where,

r_j = the accumulated sample autocorrelations, m

= the time lag.

If p value is less than level of significance 0.05, then we reject the null hypothesis and conclude that residuals are correlated else we fail to reject the null hypothesis and conclude that residuals are uncorrelated.

	lb_stat	lb_pvalue
1	0.000264	0.987031
2	0.180994	0.913477
3	0.266900	0.966129
4	0.512444	0.972278
5	0.527548	0.991080
10	8.372105	0.592540
12	15.301489	0.225361
14	17.076177	0.252141
15	17.871433	0.269485
17	17.990221	0.389458

From the above, we see that p values are greater than 0.05 level of significance at each lag. This suggests that residuals are uncorrelated.

FORECASTING:

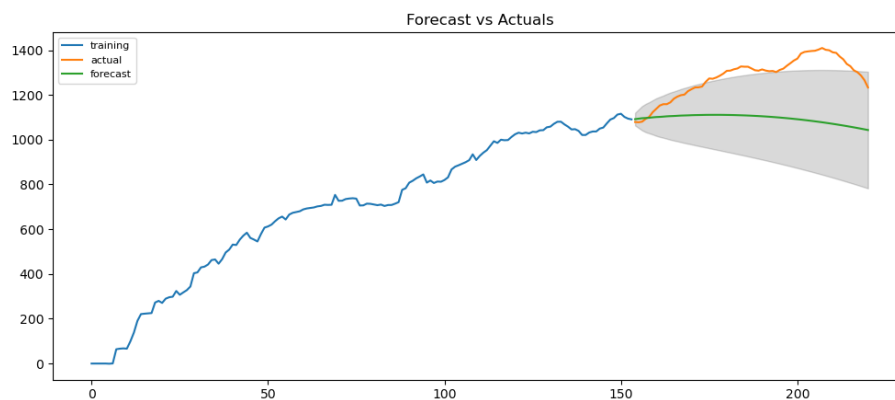
Now we are ready for forecasting stage. We forecasted the observations and show these with it's actual values.

```
>Predicted=1091.764, Actual=1079.103
>Predicted=1093.917, Actual=1077.794
>Predicted=1095.920, Actual=1080.561
>Predicted=1096.837, Actual=1093.100
>Predicted=1097.987, Actual=1101.820
>Predicted=1099.811, Actual=1122.861
>Predicted=1101.138, Actual=1137.756
>Predicted=1101.888, Actual=1152.561
>Predicted=1103.030, Actual=1158.684
>Predicted=1104.397, Actual=1158.790
>Predicted=1105.245, Actual=1166.756
>Predicted=1105.886, Actual=1183.164
>Predicted=1106.848, Actual=1192.663
>Predicted=1107.752, Actual=1198.854
>Predicted=1108.262, Actual=1201.617
>Predicted=1108.764, Actual=1216.995
>Predicted=1109.438, Actual=1225.743
>Predicted=1109.930, Actual=1233.399
>Predicted=1110.176, Actual=1234.054
>Predicted=1110.481, Actual=1237.764
>Predicted=1110.822, Actual=1258.280
>Predicted=1110.958, Actual=1273.680
>Predicted=1110.965, Actual=1272.322
>Predicted=1111.021, Actual=1277.730
>Predicted=1111.020, Actual=1285.712
>Predicted=1110.846, Actual=1295.329
>Predicted=1110.609, Actual=1307.748
>Predicted=1110.380, Actual=1308.602
>Predicted=1110.051, Actual=1314.726
>Predicted=1109.591, Actual=1318.176
>Predicted=1109.095, Actual=1327.349
>Predicted=1108.564, Actual=1326.187
>Predicted=1107.923, Actual=1326.183
>Predicted=1107.187, Actual=1318.347
>Predicted=1106.415, Actual=1310.194
>Predicted=1105.580, Actual=1308.360
>Predicted=1104.640, Actual=1314.151
>Predicted=1103.628, Actual=1308.886
>Predicted=1102.569, Actual=1306.060
>Predicted=1101.432, Actual=1307.177
>Predicted=1100.203, Actual=1302.148
>Predicted=1098.910, Actual=1311.936
>Predicted=1097.559, Actual=1317.256
>Predicted=1096.124, Actual=1330.214
>Predicted=1094.608, Actual=1341.731
>Predicted=1093.031, Actual=1353.685
>Predicted=1091.385, Actual=1362.308
>Predicted=1089.657, Actual=1385.093
>Predicted=1087.855, Actual=1392.144
>Predicted=1085.989, Actual=1394.616
>Predicted=1084.049, Actual=1396.415
>Predicted=1082.031, Actual=1397.545
```

```

>Predicted=1079.942, Actual=1402.792
>Predicted=1077.785, Actual=1409.851
>Predicted=1075.553, Actual=1401.888
>Predicted=1073.246, Actual=1399.736
>Predicted=1070.869, Actual=1390.572
>Predicted=1068.421, Actual=1387.936
>Predicted=1065.897, Actual=1371.595
>Predicted=1063.301, Actual=1359.171
>Predicted=1060.634, Actual=1338.366
>Predicted=1057.895, Actual=1328.543
>Predicted=1055.081, Actual=1309.856
>Predicted=1052.196, Actual=1301.623
>Predicted=1049.239, Actual=1287.105
>Predicted=1046.209, Actual=1266.057
>Predicted=1043.105, Actual=1233.303

```



PYTHON CODES THAT IS USED FOR THIS PROJECT:

```
#importing the libraries
```

```
import numpy as np import
```

```
pandas as pd
```

```
import matplotlib.pyplot as plt import
```

```
random
```

```
#reading dataset into dataframe as data1 data1 =
```

```
pd.read_csv("C:/Users/Lenovo/Desktop/time series project summer/vaccinations.csv")
```

```
data1.head(5)
```

```
#checking dimation of data1 data1.shape
```

```
#CHECKING FOR NULL VALUES in data1
```

```
data1.isnull().sum()
```

```
#Replacing NA by 0 data1
```

```
= data1.fillna(0) data1
```

```
#CHECKING FOR NULL VALUES IN EACH COLUMNS
```

```
data1.isnull().sum()
```

```
#CHECKING COUNT OF UNIQUE VALUES IN COLUMNS
```

```
data1['date'].value_counts()
```

```
#DROPPING THE locations and ISO code COLUMNS FROM data1
```

```
data1.drop(['location','iso_code'], axis=1, inplace=True)
```

```
data1.head() data1.shape
```

```
#finding list of numerical value columns into list num_col num_col
```

```
= [i for i in data1.columns if data1.dtypes[i]!='object'] num_col
```

```
#sorting dataframe by Date in ascending order
```

```
data1 = data1.sort_values(by = 'date') data1
```

```
#grouping dataframe by Date and summing them date wise data1 as data2
```

```
data2 = data1.groupby(['date'], as_index = False).sum() data2
```

```
#copying the date column
```

```
col_date = data2['date'].copy()
```

```
len(num_col) np.random.seed(10)
```

```
#now we make our dataframe data2 continuous by adding random number using unniform(0,1) and name it as added_df
```

```
added_df = data2.iloc[:,1:].add(np.random.rand(221, 9)) added_df
```

```
#adding col_date column in added_df dataframe at right most column
```

```
added_df['Date'] = col_date added_df
```

```
#making list of column names of added_df dataframe as cols cols=list(added_df.columns)
```

```
#taking total_vaccinations and date columns in new dataframe df_date_tot to fit the model in this dataframe to forecast
```

```
total_vaccinations df_date_tot = added_df[[cols[4]] + [cols[-1]]] df_date_tot
```

```
#plotting the df_date_tot dataframe date vs total_vaccinations all over the wworld
```

```
plt.figure(figsize=(3,3),dpi=300)
```

```
plt.plot(df_date_tot['Date'],df_date_tot['daily_vaccinations']) plt.show()
```

```
plt.subplot(212) plt.hist(df_date_tot['daily_vaccinations']) plt.show()
```

```
# splitting dataframe by row index into two parts
```

```
part_1 = df_date_tot.iloc[:111,:] part_2 =
```

```
df_date_tot.iloc[111:,:] df_date_tot.describe()
```

```
part_1.shape part_2.shape
```

```
part_1.var() part_2.var()
```

```
df_daily_vaccinations = df_date_tot.copy()
```

```
df_daily_vaccinations.shape
```

```
#box cox method for variance stablization from scipy.stats import boxcox
```

```
df_daily_vaccinations['daily_vaccinations'], lam = boxcox(df_daily_vaccinations['daily_vaccinations'])
```

```
print('Lambda: %f' % lam)
```

```

# line plot plt.figure(figsize=(3,3),dpi=300)
plt.plot(df_daily_vaccinations['Date'],df_daily_vaccinations['daily_vaccinations'])
plt.show() # histogram plt.subplot(212)
plt.hist(df_daily_vaccinations['daily_vaccinations']) plt.show()
df_daily_vaccinations['daily_vaccinations'] train =
df_daily_vaccinations['daily_vaccinations'][:154] test =
df_daily_vaccinations['daily_vaccinations'][154:]
plt.figure(figsize=(3,3),dpi=300)
plt.plot(df_daily_vaccinations['Date'],df_daily_vaccinations['daily_vaccinations'])
plt.show()

#testing the stationarity by DF test which results it as stationary time series
#by looking at the graph and adf test it is clearly non_stationary.
from statsmodels.tsa.stattools import adfuller result =
adfuller(df_daily_vaccinations['daily_vaccinations']) print('ADF
Statistic: %f' % result[0]) print('p-value: %f' % result[1])
print('Critical Values:') for key, value in result[4].items():

(key, value)) # KPSS test from statsmodels.tsa.stattools
import kpss result =
kpss(df_daily_vaccinations['daily_vaccinations'])
print(result)

#shows series is non stationary #ACF plot upto lag 50 from
statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_daily_vaccinations['daily_vaccinations'], lags = 50)
plt.show()

#first order differencing of raw data
df_daily_vaccinations['daily_vaccinations_with_diff_1'] = df_daily_vaccinations['daily_vaccinations'] -
df_daily_vaccinations['daily_vaccinations'].shift(1) df_daily_vaccinations

#ploting the time series after first order differning plt.figure(figsize=(5,5),dpi=300)
plt.plot(df_daily_vaccinations['Date'],df_daily_vaccinations['daily_vaccinations_wi
th_diff_1']) plt.show()

#acf plot of dataframe of diff 1 from statsmodels.graphics.tsaplots import
plot_acf plot_acf(df_daily_vaccinations['daily_vaccinations_with_diff_1'][1:],
lags = 50) plt.show()

#pacf plot of dataframe of diff 1 from statsmodels.graphics.tsaplots import
plot_pacf plot_pacf(df_daily_vaccinations['daily_vaccinations_with_diff_1'][1:],
lags = 50) plt.show()

#DF test on data with diff 1 from statsmodels.tsa.stattools import adfuller
result = adfuller(df_daily_vaccinations['daily_vaccinations_with_diff_1'][1:])
print('ADF Statistic: %f' % result[0]) print('p-value: %f' % result[1])
print('Critical Values:') for key, value in result[4].items():

(key, value))

```

```

#shows non stationary # KPSS test from statsmodels.tsa.stattools
import kpss result =
kpss(df_daily_vaccinations['daily_vaccinations_with_diff_1'][1:])
print(result)

#shows series is non stationary #second
order differencing of raw data
df_daily_vaccinations['daily_vaccinations_with_diff_2'] = df_daily_vaccinations['daily_vaccinations_with_diff_1'] -
df_daily_vaccinations['daily_vaccinations_with_diff_1'].shift(1) df_daily_vaccinations

#ploting the time series after second order differencing plt.figure(figsize=(5,5),dpi=500)
plt.plot(df_daily_vaccinations['Date'],df_daily_vaccinations['daily_vaccinations_with_diff_2'])
plt.show() # histogram plt.subplot(212)
plt.hist(df_daily_vaccinations['daily_vaccinations_with_diff_2']) plt.show()

#acf plot of dataframe of diff 2 from statsmodels.graphics.tsaplots import
plot_acf plot_acf(df_daily_vaccinations['daily_vaccinations_with_diff_2'][2:],
lags = 50) plt.show()

#pacf plot of dataframe of diff 2 from statsmodels.graphics.tsaplots import
plot_pacf plot_pacf(df_daily_vaccinations['daily_vaccinations_with_diff_2'][2:],
lags = 50) plt.show()

#df test for data with diff 2 from statsmodels.tsa.stattools import adfuller
result = adfuller(df_daily_vaccinations['daily_vaccinations_with_diff_2'][2:])
print('ADF Statistic: %f' % result[0]) print('p-value: %f' % result[1])
print('Critical Values:') for key, value in result[4].items():

                                                                    print('\t%s: %.3f' %
(key, value))

#shows series is stationary
# KPSS test from statsmodels.tsa.stattools import kpss result =
kpss(df_daily_vaccinations['daily_vaccinations_with_diff_2'][2:])
print(result)

#shows series is stationary df_date_tot #fitting arima
models from statsmodels.tsa.arima_model import
ARIMA from statsmodels.tsa.arima_model import
ARMAResults import re import statsmodels.api as sm
from statsmodels.graphics.gofplots import qqplot import
pylab import scipy.stats as stats from scipy.stats import
anderson from scipy.stats import shapiro from
scipy.stats import normaltest

#ARIMA Model
def arima(p,d,q):
    model = ARIMA(train, order=(p,d,q))    model_fit =
model.fit(dis=0)    print(model_fit.summary())
print(ARMAResults.cov_params(model_fit))    diag_elt =
np.sqrt(ARMAResults.cov_params(model_fit).diagonal()) D =

```



```

np.diag(diag_elt) corr_mat =
np.linalg.inv(D).dot(ARMAResults.cov_params(model_fit)).dot(np.linalg
g.inv(D)) print(corr_mat)

# Plot residual errors residuals =
pd.DataFrame(model_fit.resid) fig, ax =
plt.subplots(1,2) residuals.plot(title="Residuals",
ax=ax[0]) residuals.plot(kind='kde', title='Density',
ax=ax[1]) plt.show()

plt.savefig('C:/Users/Lenovo/Desktop/resdi_plots.png',dpi=100)

# q-q plot of residuals plt.subplot(1,1,1)
qqplot(model_fit.resid, line='r', ax=plt.gca()) plt.show()
plt.savefig('C:/Users/Lenovo/Desktop/qq_plots.png',dpi=500)

#anderson darling test of normality of residual distribution
result = anderson(model_fit.resid) print('Statistic: %.3f' %
result.statistic) p = 0 for i in
range(len(result.critical_values)):
    sl, cv = result.significance_level[i], result.critical_values[i]
if result.statistic < result.critical_values[i]:
    print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))
else:
    print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

# Shapiro-Wilk Test of normality of residual distribution
stat, p = shapiro(model_fit.resid) print('Statistics=%.3f,
p=%.3f' % (stat, p))
# interpret
alpha = 0.05 if
p > alpha:
    print('Sample looks Gaussian (fail to reject H0)')
else:
    print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test from numpy.random
import randn from scipy.stats import
mannwhitneyu data1 = randn(67) stat, p =
mannwhitneyu(data1, model_fit.resid)
print('Statistics=%.3f, p=%.3f' % (stat, p))

```

```

# interpret
alpha = 0.05    if
p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')

#stats.probplot(model_fit.resid, dist="norm", plot=plt)

# acf plot of residuals
plot_acf(residuals, lags = 50)    plt.show()

plt.savefig('C:/Users/Lenovo/Desktop/acf_resid.png',dpi=100)

# Forecast    fc, se, conf = model_fit.forecast(67,
alpha=0.05) # 95% conf

for i in range(len(test)):
    print('>Predicted=%.3f, Actual=%.3f' % (fc.tolist()[i], test.reset_index(drop=True).tolist()[i]))

#ljung box test
print(sm.stats.acorr_ljungbox(residuals, lags=[1,2,3,4,5,10,12,14,15,17], return_df=True)) #if statistics p value is less than
critical value than reject H0

#means residuals are correlated

# Make as pandas series    fc_series = pd.Series(fc,
index=test.index)    lower_series = pd.Series(conf[:, 0],
index=test.index)    upper_series = pd.Series(conf[:, 1],
index=test.index)

# Plot
plt.figure(figsize=(12,5), dpi=100) plt.plot(train, label='training')
plt.plot(test, label='actual')    plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series,
                color='k', alpha=.15)    plt.title('Forecast vs
Actuals')    plt.legend(loc='upper left', fontsize=8)
plt.savefig('C:/Users/Lenovo/Desktop/forecast_f.png',dpi=100)
plt.show()

```

```

arima(1,1,1) #bad arima(2,1,1) #bad arima(1,1,2) #bad
arima(2,1,2) #second best arima(2,2,4) arima(3,1,3) #third best
arima(1,2,1) #less AIC than 2,1,2 but not all variables are signifi
arima(2,2,1) #bad arima(1,2,2) #bad arima(2,2,2) #bad
arima(2,2,3) #best arima(0,1,0) import pmdarima as pm model
= pm.auto_arima(train, start_p=1, start_q=1,
                 test='adf',      # use adftest to find optimal 'd'
                 max_p=4, max_q=4, # maximum p and q
m=1,            # frequency of series
d=None,         # let model determine 'd'
seasonal=False, # No Seasonality
start_P=0,      D=0,
trace=True,     error_action='ignore',
suppress_warnings=True,
stepwise=True)

print(model.summary()) model.plot_diagnostics(figsize=(7,5))
plt.show() arima(4,2,4) arima(1,2,1) arima(1,2,0)
arima(0,2,2) arima(1,2,2) arima(2,2,2) arima(2,2,1)
arima(2,2,0) arima(0,2,3) arima(1,2,3) arima(2,2,3)
arima(3,2,3) arima(3,2,2) arima(3,2,1) arima(1,2,4)

```

REFERENCES:

- Machine Learning Mastery
- Wikipedia
- Time Series Analysis: Forecasting and Control Fifth Edition by Box and Jenkins
- Kaggle
- Stackexchange