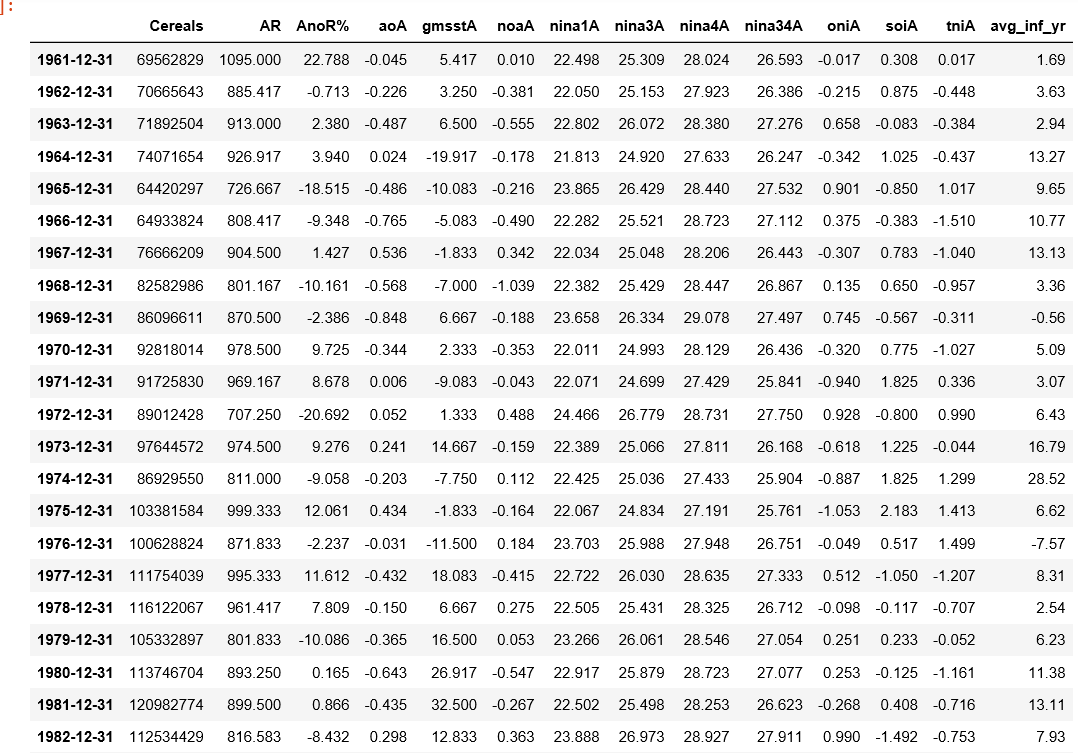
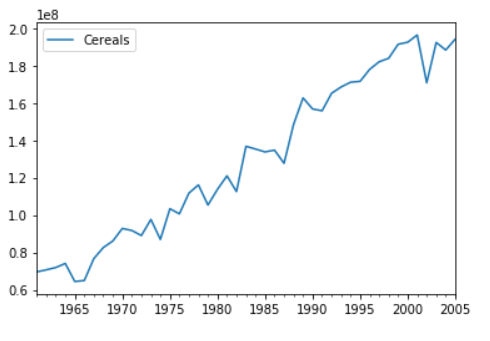
**Non-Linear Time Series Analysis Using Machine Learning, Time Series Analysis and Deep Learning**

Seven models were built for univariate and multivariate time series analysis using Machine Learning, Time Series Analysis techniques and Deep Learning as a tool and variation in the outcome of different models were observed.

Non Linear Time Series dataset provided was such that there was 1 dependent variable (outcome) i.e. the production of cereals annually of order of e+08 and 13 independent variable (features) on which the production of cereals depend, and on which five models were made to check the accuracy of each on the same dataset.



**Sample Dataset**



**Cereals Production over the years shows it is time dependent and is not stationary.**

* **Multivariate Regression Model :**

Three Multivariate Polynomial Regression Model (**A Supervised Learning technique**) were made to be deployed on the dataset.

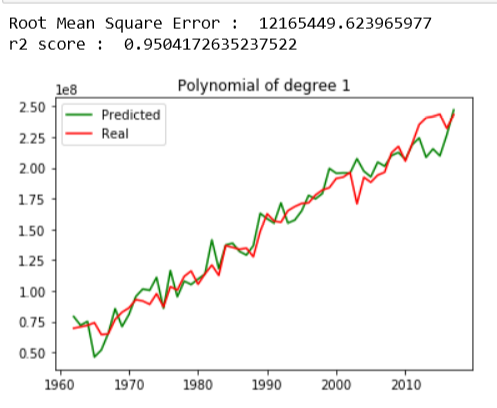
It is one kind of **structural model** and time component is not present in structural models because of problem of auto correlation, which is due to the reason that errors of time series models are correlated with each other and under Linear Regression assumption, errors has to be uncorrelated.

The Libraries used for building the model :

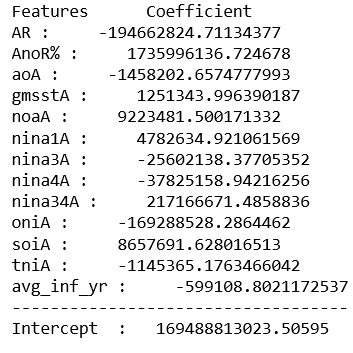
* **sklearn**.**preprocessing** for model selection i.e. Polynomial feature.
* **sklearn**.**linear**\_**models** for fitting the Polynomial Regression features using Linear\_Regression.
* **sklearn**.**metrics** for importing loss and correlation function i.e. to calculate the mean\_squared\_error and r2\_score.
* **pandas** to read csv file.
* **matplotlib**.**pyplot** for visualization purpose.

The **r2\_score** tells us about how much the two data sets are correlated.

First, a polynomial function of degree 1 was deployed on the dataset to see the variation in expected versus the real outcome and a root mean square error of 12165449.623965977 and r2 score of 0.9504172635237522 was observed.

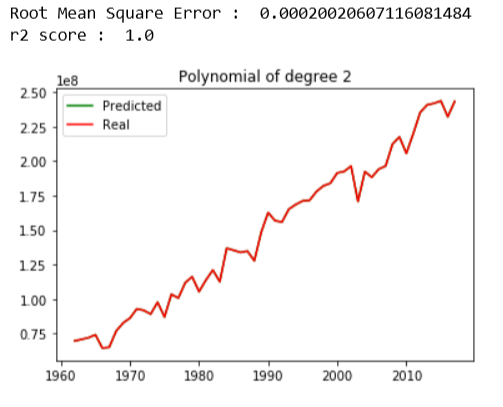


**A r2 score of 0.95 shows, the Linear Regression model is very well fitting on the dataset.**

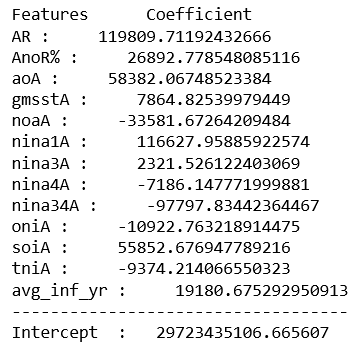


**Coefficient of Independent Variables and Intercept in Linear Regression Model.**

Then, a polynomial function of degree 2 was deployed on the dataset to see the variation in expected versus the real outcome and a root mean square error of 0.00020020607116081484 and r2 score of 1.0 was observed.

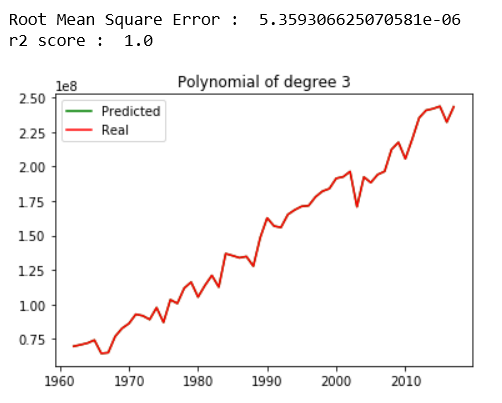


**A r2 score of 1 shows, the Polynomial Regression model of degree 2 is overfitting on the dataset as the dataset is small and has many features.**

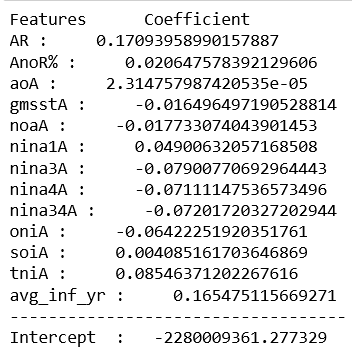


**A r2 score of 1 shows, the Polynomial Regression model of degree 2 is overfitting on the dataset as the dataset is small and has many features.**

Then, a polynomial function of degree 3 was deployed on the dataset to see the variation in expected versus the real outcome and a root mean square error of 5.359306625070581e-06 and r2 score of 1.0 was observed.



**A r2 score of 1 shows, the Polynomial Regression model of degree 2 is overfitting on the dataset as the dataset is small and has many features.**



**A r2 score of 1 shows, the Polynomial Regression model of degree 3 is overfitting on the dataset as the dataset is small and has many features.**

**Conclusion :**

The predicted and true plot of outcome in case of polynomial of degree 2 and 3 overfitted the dataset and which is due to the reason that the dataset has very less training examples and also it has many features to be trained upon, whereas polynomial of degree 1 showed very less variation in predicted and true outcome and so is the best generalized model for the data among the three machine learning models.

* **ARIMA (Auto Regressive Integrated Moving Average) Model :**

An Univariate Auto Regressive Integrated Moving Average Model was deployed on the dataset available, which used the lag observations of the outcome itself as an input, (considering no exogeneous variable) and based on which, using Auto Regressive(AR), Integration(I) and Moving Average(MA) method predicted the next outcome.

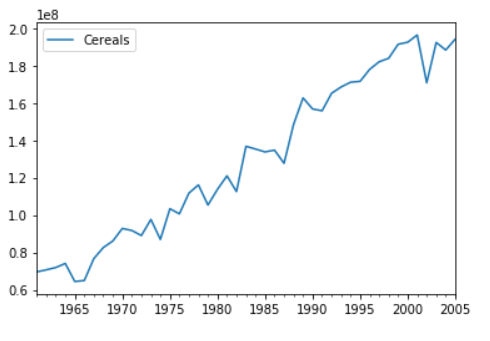
**ARIMA** is simply an ideology that captures auto correlation in the time series by modeling it directly.

The Libraries used for building the model :

* **pandas** to read csv file.
* **matplotlib**.**pyplot** for visualization purpose.
* **statsmodels**.**api** used for econometric purposes and to import plot\_acf and plot\_pacf.
* **stastmodels**.**tsa** for importing stattools for augmented dickey-fuller test and to import ARIMA model.

Lags of stationarised series are called “**Auto Regressive**” that refers to AR terms and lags of the forecast errors are called “**Moving Average**” which refers to the MA terms.

Residual errors are the difference between actual values and what we predicted. For an ideal model, it can be white noise (very random with no pattern or sequence).



**Cereals Production over the years shows it is time dependent and is non stationary, so before proceeding with building the model, it is necessary to stationarize the dataset.**

For an ARIMA model the time series to be considered for training and fitting must be stationary, as it becomes quite easy to train a stationary time series rather than a non-stationary time series, as the non-linear property of time series is not easy to be captured by many of the algorithms. A time series is said to be stationary, if its **statistical properties** such as mean, variance and auto co-variance is not a function of time i.e. does not change with time. A stationarised regression model uses lags of dependent variables or lags of forecast errors as regressors.

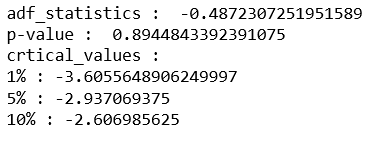
**Need of making a time series stationary:**

* An auto regressive forecasting model, are essentially linear regression (**ordinary least square**) model that utilize the lags of the series itself as predictors.
* We know that linear regression works best, if the predictors (features) are not correlated against each other. So stationarising the time series solves this problem.

Beside looking at the dataset, there are various statistical methods to check if a time series is stationary, the one used here in ARIMA model building was: **Augmented Dickey-Fuller Test.**

Augmented Dickey-Fuller test is a kind of statistical test called **unit root test**. The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend. The null hypothesis of the test is that the time series can be represented by a unit root, that it is non-stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary. We interpret this result using the p-value from the test.

* p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
* p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.



**In this case p-value came to be 0.8944843392391075 so it was necessary to make the time series stationary.**

whereas, other results such as adf\_statistics compares itself with the critical values and tells us, by what percent we can confidently say the dataset is stationary, like here adf\_statistics is more than all the critical values i.e. 1%, 5% and 10%, so in this case we have less than 90% confidence to say that this data is stationary.

**Methods to make a time series stationary:**

* **Differencing**: In this method, we compute the difference of consecutive terms in the series. Differencing is typically performed to get rid of the varying mean. Mathematically, first order differencing can be written as:

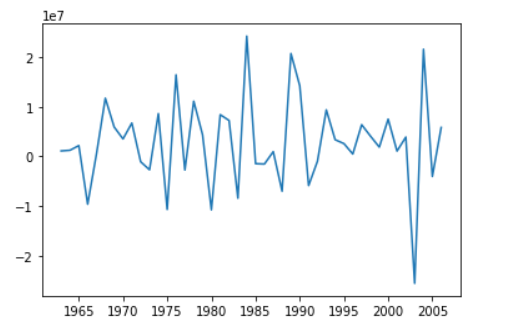
yt‘ = yt – y(t-1)

where yt is the value at a time t.

* **Transformations**: Transformations are used to stabilize the non-constant variance of a series. Common transformation methods include power transform, square root, and log transform.

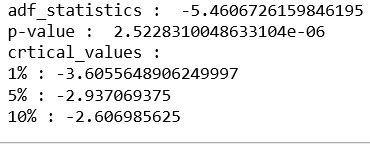
After differencing, it is necessary to reverse it (undifferencing) so as to obtain a forecast for the original series.

The dataset is first order differenced to obtain a stationary series.



**Plot observed after first order differencing of Cereals dataset, it clearly shows that now the dataset is stationarised as mean seems to be constant at any point in time.**

Now, we will perform **augmented dickey fuller test** on the new obtained dataset after differencing.



**In this case p-value came to be 2.5228310048633104e-06 so the time series is now stationary.**

Other results as adf\_statistics is also less than 1% critical value, showing that with 99% confidence we can say that the time series is now stationary.

There are essentially three parameters involved in ARIMA model i.e. p, d and q. The parameter ‘**p**’ incorporates the effect of past values into our model (**AR terms**). For ex. If last 3 days has been warm then tomorrow will also be warm.

The parameter ‘**q**‘ incorporates the number of lagged forecast errors in prediction equation (**MA terms**). It allows us to set error of our model as a linear combination of the error values observed at previous time points in the past.

The parameter ‘**d**’ incorporates the number of times the raw observations are differenced to make the dataset stationary, also called the **degree of differencing**.

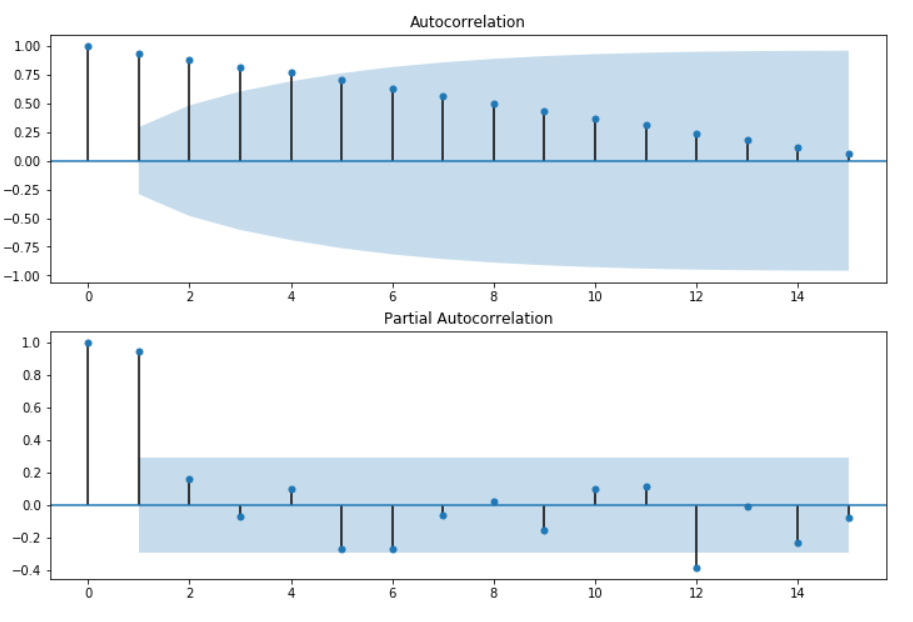
As now, we have ‘d’ parameter as 1, we have to select other parameters.

**There are different methods to find the values of these parameters** :

* From the **Autocorrelation plot** of the data (will help if Moving Average (MA) model is appropriate) we can tell whether or not we need to add MA terms (‘q’).
* From the **Partial Autocorrelation plot** of the data (will help if Auto regressive (AR) model is appropriate) we can tell whether or not we need to add AR terms (‘p’).

**Autocorrelation function** (ACF) plots the correlation between a series and its lags.

**Partial Autocorrelation function** (PACF) also conveys similar information but is conveys the pure correlation of a series and its lags, excluding the correlations from intermediate lags.

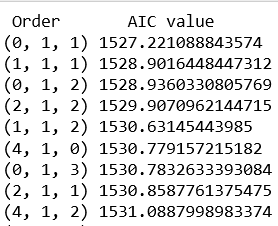


**ACF and PACF plot for Cereals Dataset shows the AR (AutoRegressive) and MA (Moving Average) term present**

It can be observed from the **ACF plot** that it is geometrically decreasing so it can be inferred that AR term is present and to get the order of AR term PACF plot is used.

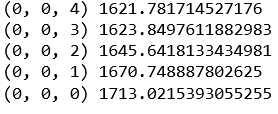
From the **PACF plot**, it can be observed that there is significant correlation at first lag, followed by correlations that are not significant, this also infers that AR term is present in the data.

But to make the best fitting ARIMA model, **Akaike Information Criterion** (AIC) is used here, which is an estimator of the relative quality of statistical models for a given set of data. Given a collection of data, AIC estimates the quality of each model, relative to each of the other models. The AIC value allows us to compare how a model fits the data and takes into account the complexity of model, so model that has a better fit while using fewer features will require a better (lower) AIC score than similar models that utilize more features.



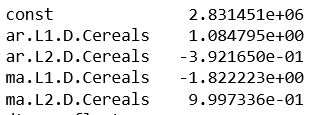
**Order (p,d,q) and their corresponding AIC values (Best models).**

In estimating the amount of information lost by a model, AIC deals with the trade-off between the goodness of fit of the model and the simplicity of the model. In other words, AIC deals with both the risk of over-fitting and the risk of under-fitting.

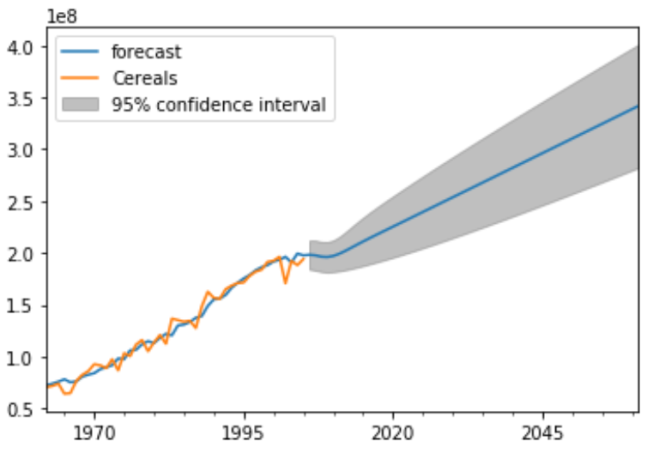


**Order (p,d,q) and their corresponding AIC values (Worst models).**

The value of parameters chosen for this dataset were ‘p’ : 2, ‘d’ : 1, ‘q’ : 2.



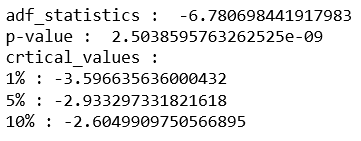
**Parameters of ARIMA model of order (2,1,2) – a constant, coefficient of two time series lags and coefficient of two forecast error lags, corresponding to p and q.**



**The forecast produced for next 100 years using the ARIMA model of order (2,1,2). The model is well fitted over the data, with a 95% confidence that the prediction will be in the shaded region.**

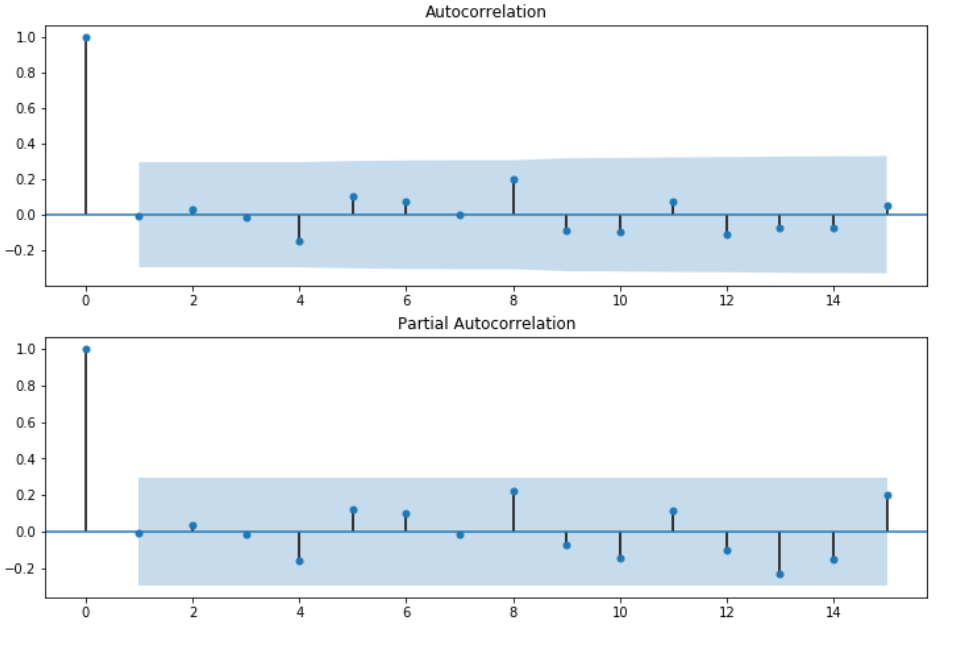
Just like input observations, residual errors from time series can themselves have temporal structure like trends, bias and seasonality. Any temporal structure in the time series of residual forecast errors is useful as a diagnostic, as it suggests information that could be incorporated into the predictive model. This is what **Moving Average** term does, it models the lagged forecast error values.

An **ideal model** will not have any structure in residual error, just random fluctuations which cannot be modeled.

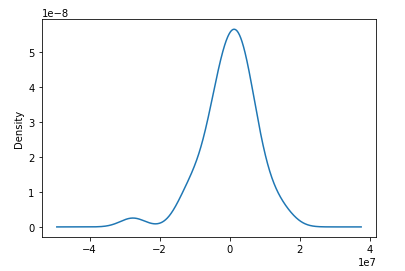


**Augmented Dickey-Fuller Test is applied on the residual error to check its stationarity, and it is observed to be stationary.**

**Autocorrelation plot** and **Partial Autocorrelation plot** can be used for residual error to check if any of the lags we missed that contained vital information to be modeled.



**The plots clearly shows that the model is perfect over the dataset and the residual error is just random fluctuations and do not contain any temporal structure.**

****

**Kernel Density Estimate plot of residual shows it is spanning around 0 and model is optimised.**

* **Conclusion :**

The results obtained were quite good and fitted well to the dataset.

* **Vector Auto Regressive (VAR) Model :**

The VAR model is one of the most commonly used methods for multivariate time series analysis. In VAR model, each endogeneous variable (dependent i.e. to be predicted) is a linear function of the past values of itself as well as the past values of all other variables.

In this model, we have considered two endogeneous variable i.e. time series (which are strongly correlated) and all other 12 features as exogeneous variables i.e. time independent variables to train the model.

A Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This is what is used in building this model, order of lag value of cereals production, which shows the best dependency of itself is selected by **AIC** information.

**VAR** is able to understand and use the relationship between several variables. This is useful for describing the dynamic behavior of the data and also provides better forecasting results.

The libraries used for building the model :

* **pandas** to read csv file.
* **matplotlib**.**pyplot** for visualization purposes.
* **numpy** for making an array, so to store the predictions.
* **statsmodels**.**api** used for econometric purposes and to import plot\_acf and plot\_pacf.
* **stastmodels**.**tsa**.**vector**\_**ar**.**vecm** for importing coint\_johansen package to test for stationarity of the multivariate time series.
* **statsmodels**.**tsa**.**vector**\_**ar**.**var**\_**model** for importing VAR model.

A general VAR model is similar to ARIMA model except that each quantity is vector valued and matrices are used as coefficient as the multivariate time series not only depends on its own lags but also on other feature lags and other exogeneous variables (independent).

As discussed for the ARIMA model, for any time series analysis, it is necessary to first make the time series stationary.

For a Multivariate Time Series, there exists a test to check the stationarity of the time series and that is used for this dataset i.e. **Johansen Test for Cointegration.**

**Cointegration :** If there exists a stationary linear combination of non-stationary random variables, the variables combined are said to be cointegrated**.**

In the Johansen test, we check whether lambda has a zero eigenvalue. When all the eigenvalues are zero, that would mean that the series are not cointegrated, whereas

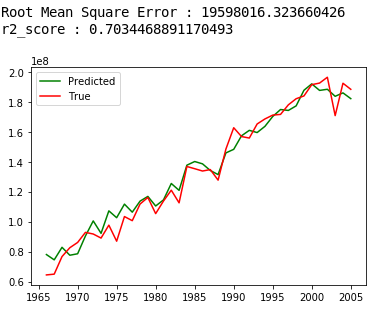
when some of the eigenvalues contain negative values, it would imply that a linear combination of the time series can be created, which would result in stationarity.

The linear combination of these prices represents the net market value of the portfolio. If the change in the value of the portfolio is related to its current value by a negative regression coefficient or in this case a negative eigenvalue, then we would have a **mean reverting** or stationary portfolio. This is the essence of the **Johansen Test.**

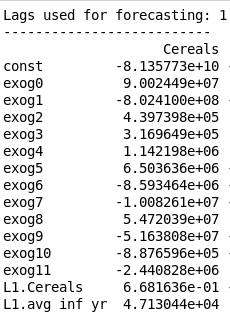
**The Johansen Test for Cointegration** takes three arguments as parameter: time series, order of null hypothesis (a value of 0 implies that it is a constant term i.e. there is no time trend in the polynomial) and number of lagged difference terms. In this model I have used order of null hypothesis as 0 and number of lagged difference terms as 1.

The output of this test provides us with trace statistics and eigen statistics. The trace statistics tell us whether the sum of the eigenvalues is 0. .

The eigen statistics stores the eigenvalues in decreasing order of magnitude, they tell us how strongly cointegrated the series are or how strong is the tendency to mean revert.



**VAR model prediction plot on true dataset, which shows that the model is fitted well, by using 2nd order lag of itself and one other variable as obtained by coint johansen test (eigenvalues i.e. the one having best correlation with the outcome) and 12 other exogeneous variables.**



**Parameters of VAR model taking 13 exogeneous (12 independent variables and 2nd lag of Cereals (dependent variable itself) ) and 2 endogeneous variable – a constant, coefficient of 13 exogeneous variables and coefficient of lags of endogeneous variables.**

* **Conclusion :**

Then the model fitted was used for forecasting on the same dataset and quite better results were obtained as compared to ARIMA model. Then using the matplotlib library, plot of required (to be predicted i.e. cereal production annually) time series was plotted to observe the difference between the true and predicted values.

* **Long Short-Term Memory (LSTM) Model : (A Deep Learning Model) :**

Neural Networks like Long Short-Term Memory (LSTM) recurrent neural networks are able to solve almost any multiple variable input problem and so with this incite I made a model for time series forecasting with multiple independent variable to predict an outcome i.e. dependent variable.

RNN are capable of learning nonlinearities, and specialized nodes like LSTM nodes are even better at this and hence differencing in time series is not required in building a LSTM model.

LSTM is a special kind of recurrent neural network. Recurrent Neural Networks are networks with loop in them. For sequential problem, where output depends on previous output, RNN is very useful. For ex. stock prediction, weather prediction but for small data. For large dataset, RNN has limitations and to overcome which LSTM is used.

LSTM has gates and cell states which are additional interactions as compared to RNN. LSTM have a chain like structure, but the repeating module has four layers interacting in a very special way. These layers are :-

* Forget gate layer (sigmoid neural net layer).
* Input gate layer (sigmoid neural net layer).
* Input gate layer for creating new information (tanh neural net layer).
* Output gate layer (sigmoid neural net which combines with tanh neural net through point wise multiplication).

LSTM has three of these gates to protect and control the cell state.

The core idea behind LSTM is :-

* The key to LSTM is the cell state.
* This cell state is like a conveyor belt, it runs straight down the entire chain, to which LSTM has the ability to remove or add information, carefully regulated by structures called gates.
* Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a point wise multiplication operation.
* The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through. A value of 0 means “let nothing through”, while a value of 1 means “let everything through”.

For **non-linear univariate time series** analysis the dependent variable (outcome) depends on its past values i.e. lags.

For **non-linear multivariate time series** analysis the dependent variable (outcome) depends not only on its past values, but also on past values of other independent variables.

Both the non-linear univariate and multivariate time series analysis models are trained by **supervised learning technique**.

The libraries used for building the LSTM model :

* **pandas** to read csv file.
* **numpy** for making an array of input dataset to feed into the neural network.
* **matplotlib**.**pyplot** for visualization purpose.
* **sklearn**.**preprocessing** to import MinMaxScaler for normalizing the variables.
* **keras** for constructing the neural network.
* **keras**.**models** to import Sequential model, which is a class consisting of linear stack of layers.
* **keras**.**layers** to import Dense (a fully connected neural network layer), Dropout (to set activations of some random nodes zero to avoid overfitting) and LSTM.

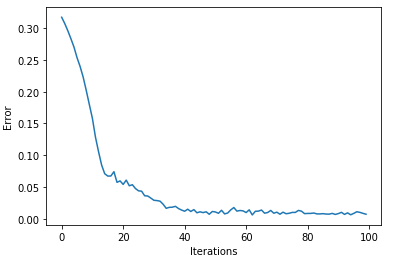
Both an non-linear univariate time series LSTM model as well as non-linear multivariate time series LSTM model were built. Based on these two models, slightly different results were produced.

Before feeding the input to the neural nets, **normalization** is a very important step in Deep Learning techniques. So on the dataset available, normalization is done, the goal of **normalization** is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

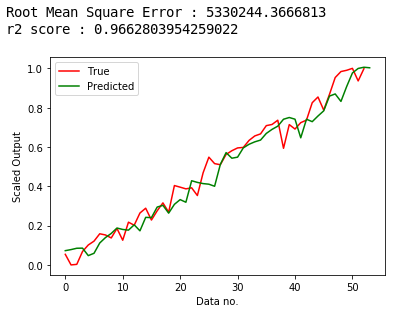
From various observations, I decided to use lag values of last 2 time points to feed as an input to the neural net for non-linear univariate time series analysis to train for the next point in time. For Non-Linear Multivariate Time Series Analysis, 13 different features and

1st lag of outcome itself are used and and is trained on next time point outcome i.e. cereals production. Both the models are trained using 50 input LSTM layers and 3 hidden LSTM layers each of 50 input nodes, the neural net then is connected to a Dense layer to predict the outcome. To compile the model, the loss function chosen was **mean square error** and **ADAM** as the optimization algorithm. The model is then trained with 100 epochs and a batch size of 10.

* **Non Linear Univariate Time Series**

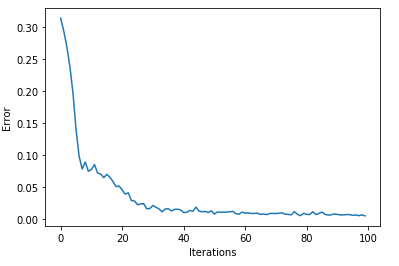
****

**The loss obtained for univariate time-series model was 0.0065, after 100 epochs**.

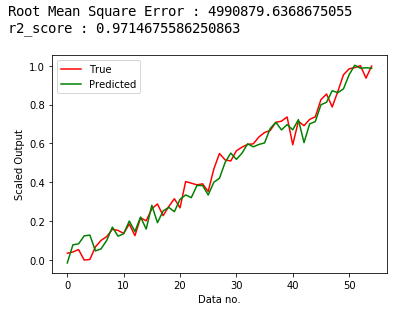


**For Univariate Time Series, the prediction plot obtained on the true dataset, it shows that the model has fitted somewhat well to the dataset.**

* **Non Linear Multivariate Time Series**



**The loss obtained for multivariate time-series model was 0.0055, after 100 epochs.**



**For Multivariate Time Series, the prediction plot obtained on the true dataset, it shows that the model has fitted well to the dataset.**

**Conclusion :**

The predicted outcome can be observed through the plots, the result obtained was much better than ARIMA and VAR model and almost fitted to the dataset. The predicted outcome for the **multivariate** time series model fitted much better to the dataset than the **univariate** time series model.

* **Comparative Analysis between the Machine Learning, Time Series Analysis and Deep Learning models built on the dataset :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Technique** | **Model** | **Parameters** | **Outcome plot** |
| **Machine Learning** | **Multivariate Linear Regression model** | MR1_coeff.PNG | MR1.PNG |
| **Multivariate Polynomial Regression model of degree 2** | MR2_coeff.PNG | MR2.PNG |
| **Multivariate Polynomial Regression model of degree 3** | MR3_coeff.PNG | MR3.PNG |
| **Time Series Analysis** | **ARIMA model of order (2,1,2)** | ARIMA_params.PNG | true_forecast_ARIMA.PNG |
| **VAR model lag order selected optimally by AIC value** |  |  |
| **Deep Learning** | **Non-Linear Univariate Time Series** | Hyper-parameters :  Epochs = 100  Batch-Size = 10  Input = Last 2 univariate values (outcome itself), determined by autocorrelation function plot as supervised learning to train for the next outcome in an iterative way for all the data points. |  |
| **Non-Linear Multivariate Time Series** | Hyper-parameters :  Epochs = 100  Batch-Size = 10  Input = Independent variables and dependent outcome as supervised learning to train for the next outcome |  |