

**Course Name:Business Intelligence and Analytics**  
**Professor Name:Prof. Saji.K.Mathew**  
**Department Name:Department of Management Studies**  
**Institute Name:Indian Institute of Technology Madras**  
**Week:11**  
**Lecture:43**

**IMPLEMENTATION IN PYTHON: ANN | BI&A | Prof. Saji K Mathew**

Okay, so different steps, what is data preparation? You are aware of it. So, one of the, so one of the steps in the data preparation is partitioning the data into training set, testing set and validation set. And in this kind of an exercise which involves a stock price series, we would split the data and the general recommendation is use the oldest data 70 percent and include major patterns in the data, when you split data. So, therefore, a time series like this should be observed first. First plot the time series and see if you make 70 percent as training data, does it really include all the major patterns.

## Data preparation

- ▶ Input, output selection
- ▶ Data transformation
  - ▶ Range (say  $[-1, 1]$ )
  - ▶ First differences
  - ▶ Logarithmic



- ▶ Data sets for training, testing & validation
  - ▶ Training: oldest, ~70%, inclusive of major patterns in the data
  - ▶ Testing: 20-30% of training data
  - ▶ Validation: remaining, most recent data

For example, here you can see perhaps there is a stock split or something that has happened here. So, you have to ensure that this particular pattern is included for training. Otherwise the model of, the model training will not be efficient, it does not have the all the patterns to train the model. And then 20 to 30 percent for testing and most recent data for validating the final model and reporting what is the prediction error.

So, that is the validation set. Validation and testing are similar, but in testing you know that you will generate many models. You may actually generate, say 5 or 10 constituent models or, sorry candidate models, not constituent. And then use the, so the test data will give you the, you know accuracies of each model and you will finally select one model as the final model whichever is most accurate. And then once again test it using the validation data and report what is the error, prediction error with respect to a very recent set of data. That is the approach.

## Network topology (architecture/structure)

- ▶ **Number of hidden layers**
  - ▶ Determines non-linearity
  - ▶ Problem of over-fitting
  - ▶ Thumb rule: 1-2 hidden layers
- ▶ **Number of neurons in each layer**
  - ▶ i/p layer: no neurons=no of i/ps
  - ▶ hidden layer
    - ▶ No magic formula- follow experimentation
    - ▶ Thumb rule:  $\sqrt{\text{no of i/p neurons} * \text{no of o/p neurons}}$
  - ▶ o/p layer: based on application

So, next step in this project is choosing the network topology and I think we discussed this already in connection with neural networks in the previous session. So, topology is also known as architecture or structure. In literature, you will find this different words they all mean the same. What is a network? It is a feed forward network, but how many hidden layers and how many neurons in each layer? These are all design that you have to do.

So, especially the choice of number of hidden layers and number of neurons in each hidden layer is a choice or they actually form part of the hyper parameters. These are actually parameters that you decide for the algorithm. So, these are already discussed and then, also the transfer function for each layer. So, we discussed the general principle that for hidden layer the transfer function can be non-linear, but for the output layer it has to be linear. And there are of course, reasons for that and we are not going into that.

## Training

- ▶ Transfer functions
  - ▶ Subject to experimentation
  - ▶ Thumb rule for time series
    - ▶ output layer: linear
    - ▶ rest all: sigmoid
- ▶ Algorithm
  - ▶ Back Propagation (BP) most widely used
    - ▶ Gradient Descent (GD)
    - ▶ Gradient Descent with Momentum (GDM)

So, then the training process itself. So, we have discussed this already. So, now a days the ReLU is very commonly used as a transfer function. So, we will look at it when we actually start modeling. So, the algorithm, also you have choices when you use typical library or software for training neural networks, the gradient descent and gradient descent with momentum. So, these are different types or different variants of the back propagation algorithms.

And you know that in modeling, especially for the purpose of forecasting and prediction, you also need to estimate what is the prediction error. So, prediction error since it is a time series continuous value data, you can use indicators or measures like MSE, RMSE, mean absolute error, mean absolute percentage error, MAPE. And you can see that

MAPE is a ratio and rest all measures are not ratios. But for the same data set of course, you can compare using any one of them and MSE is widely used.

## Model performance & prediction errors

- ▶ Mean Square Error (MSE)
- ▶ Root Mean Square Error (RMSE)
- ▶ Mean Absolute Error (MAE)
- ▶ Mean Absolute Percentage Error (MAPE).

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2, \quad \text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2},$$
$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t| \quad \text{and} \quad \text{MAPE} = \frac{1}{n} \sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{|y_t|} (100\%)$$

So, that concludes the introduction to stock price modeling and ANN is something as a technique we learnt already. Now, we are going to actually get some data and apply this principles which we learnt, to model the data and then subsequently see, visualize and see how the model performs. So, that is our next exercise.

So, the data set that we are going to use in this session is the daily stock price series of Infosys Limited listed in NASDAQ. Infosys perhaps is the first Indian IT company to get listed in NASDAQ and that was a historic moment.

So, you can see the Infosys price series starting from 1999 to the point in time when I downloaded this data, that was 2015. So, you have about 16 years of daily price series, daily price series and daily means that is everyday stock price, but only for the trading days. How many trading days are there in a week? 5 trading days. So, it is every 5 days data, it does not have Saturday, Sunday included.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Open	High	Low	Close	Volume	Adj Close						
2	19/05/15	31.8	31.93	31.61	31.7	1913300	31.7						
3	18/05/15	31.25	31.61	31.07	31.41	2658800	31.41						
4	15/05/15	30.88	31.13	30.67	31.07	3025600	31.07						
5	14/05/15	30.77	30.84	30.63	30.7	1645500	30.7						
6	13/05/15	30.86	31.11	30.69	30.75	1812200	30.75						
7	12/05/15	30.63	30.89	30.6	30.68	3742800	30.68						
8	11/05/15	31.59	31.67	31.23	31.3	2953800	31.3						
9	08/05/15	31.13	31.25	30.99	31.22	2502400	31.22						
10	07/05/15	31.03	31.12	30.64	30.82	3765400	30.82						
11	06/05/15	30.87	30.92	30.51	30.62	2939300	30.62						
12	05/05/15	31.16	31.35	31.1	31.22	2368000	31.22						
13	04/05/15	31.75	32	31.54	31.54	2867600	31.54						
14	01/05/15	31.03	31.26	30.99	31.19	1707700	31.19						
15	30/04/15	30.97	31.07	30.87	30.98	3527300	30.98						
16	29/04/15	31.08	31.34	31.05	31.15	3759400	31.15						
17	28/04/15	31.28	31.57	30.99	31.28	5483500	31.28						
18	27/04/15	31.56	31.88	30.98	31.31	6278900	31.31						
19	24/04/15	32.47	33.04	31.45	31.81	15491300	31.81						
20	23/04/15	34.56	35.18	34.01	34.96	5499900	34.96						
21	22/04/15	34.34	34.72	34.19	34.56	5192700	34.56						
22	21/04/15	34.37	34.76	34.37	34.55	4681500	34.55						
23	20/04/15	34.48	34.65	34.28	34.52	4884900	34.52						
24	17/04/15	35.08	35.19	34.66	35.1	2389300	35.1						
25	16/04/15	35.38	35.54	35.15	35.31	2629900	35.31						
26	15/04/15	35.75	36.19	35.52	36.15	2283500	36.15						
27	14/04/15	36.23	36.4	36.1	36.12	1587900	36.12						
28	13/04/15	36.1	36.44	36	36.22	1836400	36.22						
29	10/04/15	36.16	36.32	35.99	36.31	1637800	36.31						

**BUSINESS INTELLIGENCE & ANALYTICS**



So when you look at the data, read it that way. So, daily series not weekly, not monthly. So keep that in mind and daily series is easy to predict, but experts will say when you actually change the frequency, the prediction accuracy would be different. And when you look at the price series, when you download this as a csv file what you get is a few attributes of the time series like the date, opening price, highest price during the day, closing, lowest price during the day, closing price, trade volume and adjusted closing price. So, which of the attributes of the series we should be using? Should it be opening price, closing price, highest, lowest or adjusted closing price? Experts would recommend, when I talk to experts in finance about this, they strongly recommended closing price, not adjusted closing price.

And there is a rationale that they gave, the rationale is adjusted closing price does not exactly capture the behavior, because if the company adjust the closing price and what, how they adjust the closing price is not an information that is known. So there is some intervention to change the series there, but instead of that if you look only at the closing price, it captures the exact behavior of the stock price. So, but I have seen in papers or in research papers, some use the adjusted closing price also, but for our exercise we will use the closing price which is the, which is one of the columns in the data series. So, we should extract this and that is the idea. And also if you look at the series, you can see the series starts with the current data which is the latest data, 2015 data and then it. So, the starting is with the current data and the ending is with the old data. So, we may have to reverse this. So that you know we use the initial data for training and the final data for testing and validation. So, there is a flipud, flipud function both in matlab as well as in

python for doing this, you had, you can inverse the data series. I started this modeling with matlab in the beginning, may be 15 years ago, but which was a proprietary software and quite expensive, but today with R and python you can do this without having to pay for a software.

The screenshot displays a Jupyter Notebook interface with the following content:

```
In [1]: #IMPORTING LIBRARIES
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import sklearn.neural_network as nn
import matplotlib.pyplot as plt

In [2]: #READING AND EXPLORING DATA
data=pd.read_csv("/Users/sajimathew/Documents/DoMS/Teaching/EMBA/DMBI/DMBI-2020/Data/infy.csv")
price=data['Close']
Nandata=np.count_nonzero(pd.isnull(price))
Nandata
plt.plot(price)
#reversing the data series
price=np.flipud(price)
plt.plot(price)
#price.describe()
```

Out [2]: [<matplotlib.lines.Line2D at 0x7f82082abd90>]

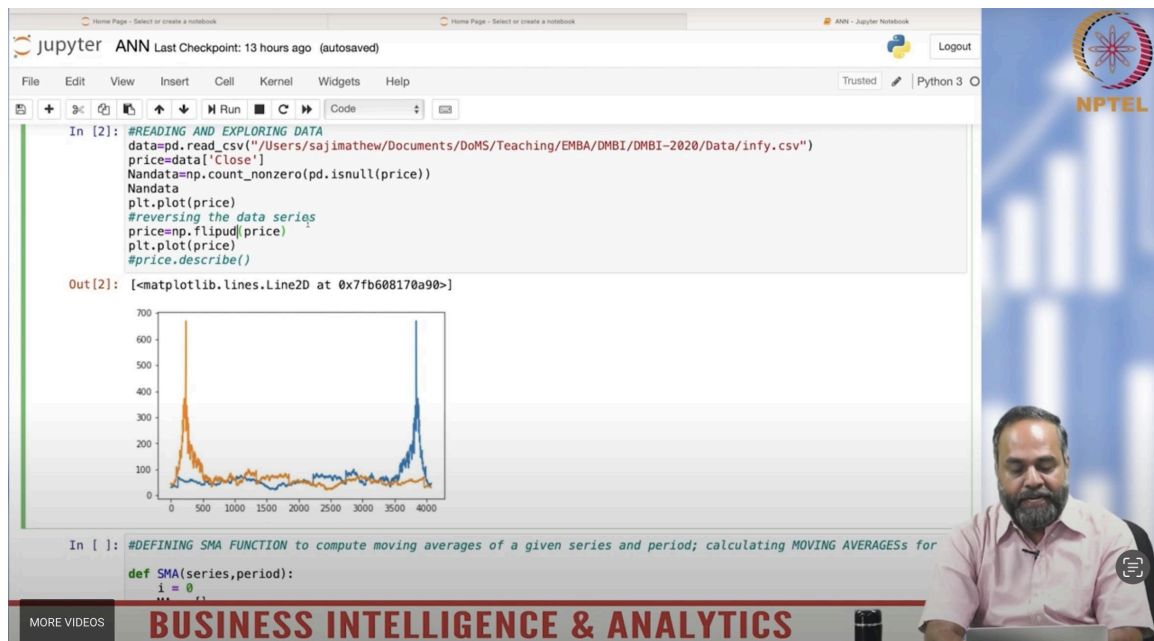
The plot shows a time series of stock prices with a vertical line at the end. The y-axis ranges from 300 to 700. The plot is titled 'BUSINESS INTELLIGENCE & ANALYTICS'.

So now we move from here, having looked at the data series how the data looks like, let us now go on to model this using python. So, there is a python script which is already available, which I migrated from matlab to R and then R to python. So and it is, the algorithm is the same or the methods are the same, but you use different programming languages. Here is my jupyter notebook and I am going to use ANN script which is already uploaded here in the directory and let me go there and here is the data set infi.csv and we are going to import python libraries to do modeling and comments have been given at different points for you to understand what each instruction does. So we are going to use of course, pandas and numpy in this project and of course, scikit learn, we are going to scale the data or standardize the data and so we have standard scalar and scikit learn library has a neural network function for modeling the data using neural network and then the plot function, plot library, matplotlib is used.

So these are called first and then I am going to read the data using the read.csv which we used earlier also and that is read as data. So, this becomes a panda table which is already dead. So, price is equal to data, the close, close is the column which we saw that has the closing price. So that is our, what, that is our time series actually we are going to use only that, closing price series, close.



So, our time series basic data is that which is now named as price here and we are checking if there is any missing data and actually you will find there is no missing data and as I said we need to reverse this data series because we need to start with the oldest data and then end with the most recent data. So, the flipud function is used to, is used to reverse the data series and then I am plotting it. So, I am going to plot the data series and before reversing and after reversing and you can see that the data series is reversed and you can also see that there is a major change that is at the end of the data and we need to ensure that this particular pattern is captured in the training data, in the training data. So I am just plotting the reversed series alone.



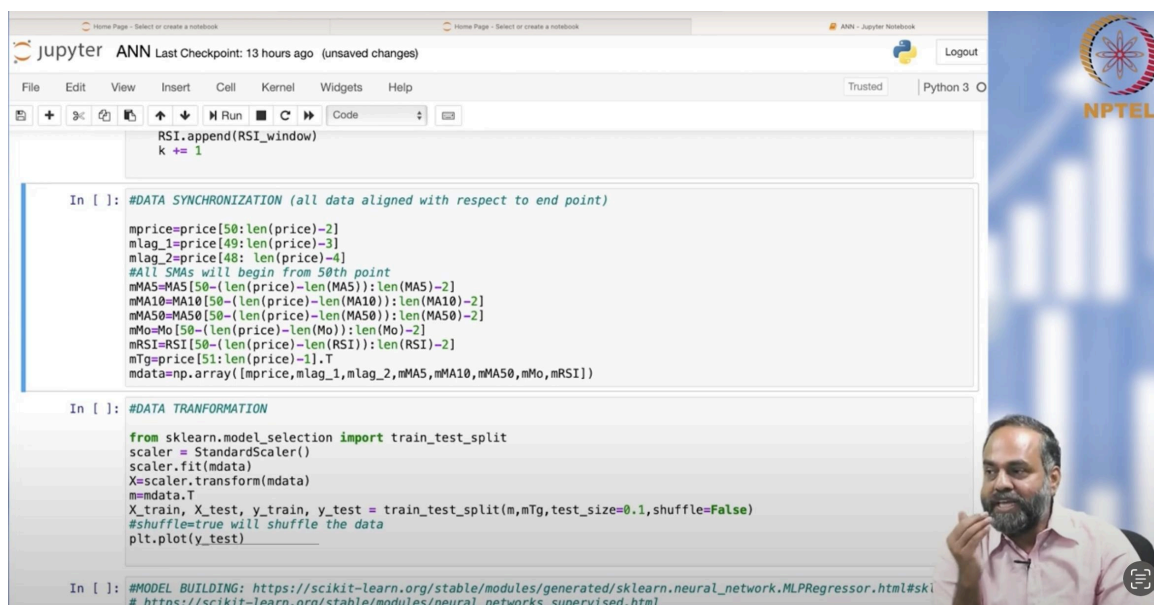
So, that it does not confuse. So, you can see that the, that major changes happened only in the beginning. So therefore, if you take 70 percent of the data the major change is already contained in the training data. So, just to be sure about it and now in the next cell, I have defined a function for moving average and literature suggest that in order to train a neural network for stock prices effectively, one must use three moving averages. Moving average for 5 days, moving average for 10 days, moving average for 50 days meaning moving average for 1 trading week, moving average for 2 trading weeks that is 10 days and moving average for 5 trading weeks, that is the meaning.

So, that is what is done. So, since we need to repeat this function. So, I wrote a function, I defined a function SMA and then I am actually giving values or arguments to that function SMA, to create moving average 5, moving average 10 and moving average 50. This is based on literature. And in the next cell, I am calculating momentum. So we discussed what is momentum? Momentum is old closing price minus current closing

price.

So, there is simple wide loop that is created to compute this values for the whole time series, the whole series. And since momentum is only 1, I do not have to define a function for it, it is calculated. And the next cell computes relative strength indicator or RSI. So I explained or I showed you the formula for RSI, which is a function of number of positive changes and number of negative changes. This particular cell just implements that formula using a wide loop again because you have to do it for the entire series.

So, RSI is calculated here and the variable RSI would finally have, the RSI will have the complete data for the time series once it is calculated. That is done, so far so good. And we can see that we are still putting all our time to prepare data. We are doing, we are still preparing data, we have not started modeling at. So the next step is the step that requires all a researchers attention because the idea here is to use the time series and derived indicators from the time series to train the model.



The image shows a Jupyter Notebook interface with the following code cells:

```
RSI.append(RSI_window)
k += 1
```

```
In [ ]: #DATA SYNCHRONIZATION (all data aligned with respect to end point)
mprice=price[50:len(price)-2]
mlag_1=price[49:len(price)-3]
mlag_2=price[48:len(price)-4]
#All SMAs will begin from 50th point
mMA5=MA5[50-(len(price)-len(MA5)):len(MA5)-2]
mMA10=MA10[50-(len(price)-len(MA10)):len(MA10)-2]
mMA50=MA50[50-(len(price)-len(MA50)):len(MA50)-2]
mMo=Mo[50-(len(price)-len(Mo)):len(Mo)-2]
mRSI=RSI[50-(len(price)-len(RSI)):len(RSI)-2]
mTg=price[51:len(price)-1].T
mdata=np.array([mprice,mlag_1,mlag_2,mMA5,mMA10,mMA50,mMo,mRSI])
```

```
In [ ]: #DATA TRANSFORMATION
from sklearn.model_selection import train_test_split
scaler = StandardScaler()
scaler.fit(mdata)
X=scaler.transform(mdata)
m=mdata.T
X_train, X_test, y_train, y_test = train_test_split(m,mTg,test_size=0.1,shuffle=False)
#shuffle=true will shuffle the data
plt.plot(y_test)
```

```
In [ ]: #MODEL BUILDING: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#skl
# https://scikit-learn.org/stable/modules/neural_networks_supervised.html
```

Now, there is a challenge here. There is a time series which starts at some point in time and ends at some point in time. Now, when I calculate indicators like the moving averages, that series will not be starting at the same point in time. So, when you calculate moving averages, some data is lost and therefore, it is important to synchronize the data from the same starting point. So, the effort in this particular module is to synchronize the data with respect to starting point 50. Although the data that we have starts at 1, since we are calculating a moving average with respect to 50, it cannot start before 50.

It has to start at 50th point. So, if you closely observe this data, it is all starting from the



50th point, from the 50th point. The formula has been implemented to start all data series from the 50th point, but you can see also that I am using lag, lags of the series. We found that lags also inform modeling. So, you can see my starting point is with the price, m price is the starting series, the original series which starts at 50th point.

I use lag 1. So, lag 1 will start at the 49th point and end at a point before the end point. So, that is what it is and lag 2 will start at the 48th point. Moving average 5, moving average 10, moving average 50 will all start at 50th point. I am synchronizing it with respect to the 50th point. Momentum also starts at 50th, RSI starts at 50th and you see there is a series called mT or target data.

Target data is starting at 51st point. What is the rationale for it? We already discussed how prediction work. When I give 50th point, the target is 51st point. So, target means, you advance the data series by one point, that is your target data set. So, I created it by starting it at 51st point. When I actually use the target data, if I take the data point for other series up till the last point, then the target data will not have the last point.

So, it will be short of one data point. So, that is the reason why I am not going to the last point. I am going to the point last but two, not even last but one, to reserve data for the target. All these you have to take care in synchronizing the data, but good question. Now, having prepared the data and prepared your data matrix as consisting of how many indicators? m price that is the original price, lag 1, lag 2, moving average 5, moving average 10, moving average 50, momentum RSI, 3+2, 5+ 2 7+ 1 8, 8 attributes or 8 indicators. Actually, these are derived from the original series, but I use derived indicators and create a matrix to train the model which I am going to build.

This particular methodology I have derived from a research work which is given as reference in your course outline. So, that is a particular empirical method that is followed. It is a lot based on prior observations and experience.

The screenshot shows a Jupyter Notebook interface with the following code in cell [8]:

```

In [8]: #DATA TRANSFORMATION
from sklearn.model_selection import train_test_split
scaler = StandardScaler()
scaler.fit(mdata)
X=scaler.transform(mdata)
m=mdata.T
X_train, X_test, y_train, y_test = train_test_split(m,mTg, test_size=0.1, shuffle=False)
#shuffle=true will shuffle the data
plt.plot(y_test)

```

The output of cell [8] is a line plot showing the test data. The x-axis ranges from 0 to 400, and the y-axis ranges from 30 to 70. The plot shows a fluctuating time series that ends with a sharp spike reaching approximately 70.

Below the plot, the code for cell [1] is partially visible:

```

In [1]: #MODEL BUILDING: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sk
# https://scikit-learn.org/stable/modules/neural_networks_supervised.html

```

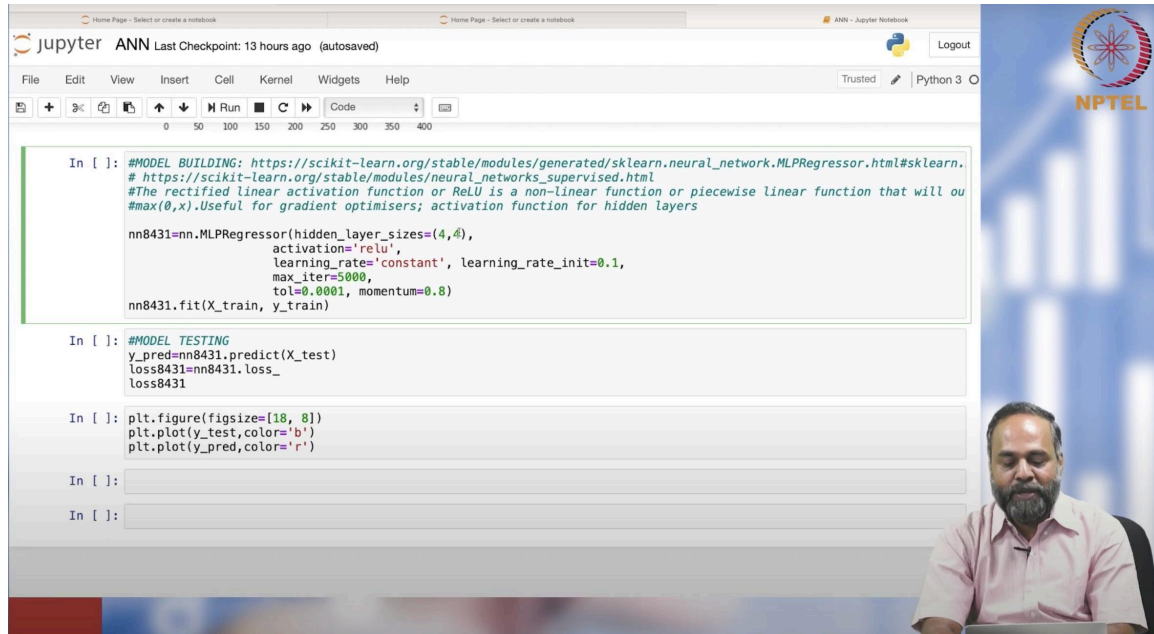
The notebook is titled "ANN" and has a last checkpoint 13 hours ago. The NPTEL logo is visible in the top right corner.

Now again we are in data preparation. Now we are going to transform the data using the standard scalar available in Python library and I am also going to split the data using the split function that is available in scikit-learn, into training and test data. I am not creating a validation set separately here, for simplicity. So, what I have plotted is the test data, the transformed test data. It is not really transformed data that I have used `m` here, but let us go ahead with that. So, I just wanted to be sure I am saying the right thing to you when I look at.

I have used `m` here. We will come back and use the `m` data later. Let us go with `m` to start with. `m` is the original data that I am splitting here and the plotted data is actually in the same value units.

Let us go to the next module. In the next module, so far we worked on data preparation, up till the point of splitting the original data into training data and test data. Now, we are going to model this data using a neural network and therefore, neural network has its own hyper parameters and we need to set this up or we need to specify the model. So, what is being done here is, I am giving a name to the model `nn` neural network 8431, but yes 8431 means what, we learn to define or code networks. A network design can be coded.

What does 8431 mean? 8 neurons. How many layers are there first of all? 4 layers. 4 layers, 8431, 4 layers. How many neurons in the input layer? 8 neurons because we have 8 variables. That is correct. How many neurons in the output layer? 1 neuron because we only want to predict the closing price.



So, 1 is the output layer. So, how many hidden layers? 2 hidden layers. So, 2 hidden layers and the first hidden layer has 4 neurons and second hidden layer has 3 neurons. That is the interpretation of 8431 and now I am calling MLP regressor which is the neural network model in the python library, scikit library and so I am specifying hidden layer sizes, but I have given 4 4. If it has to be 8431, it has to be 3 here, not 4, 4 3 1.

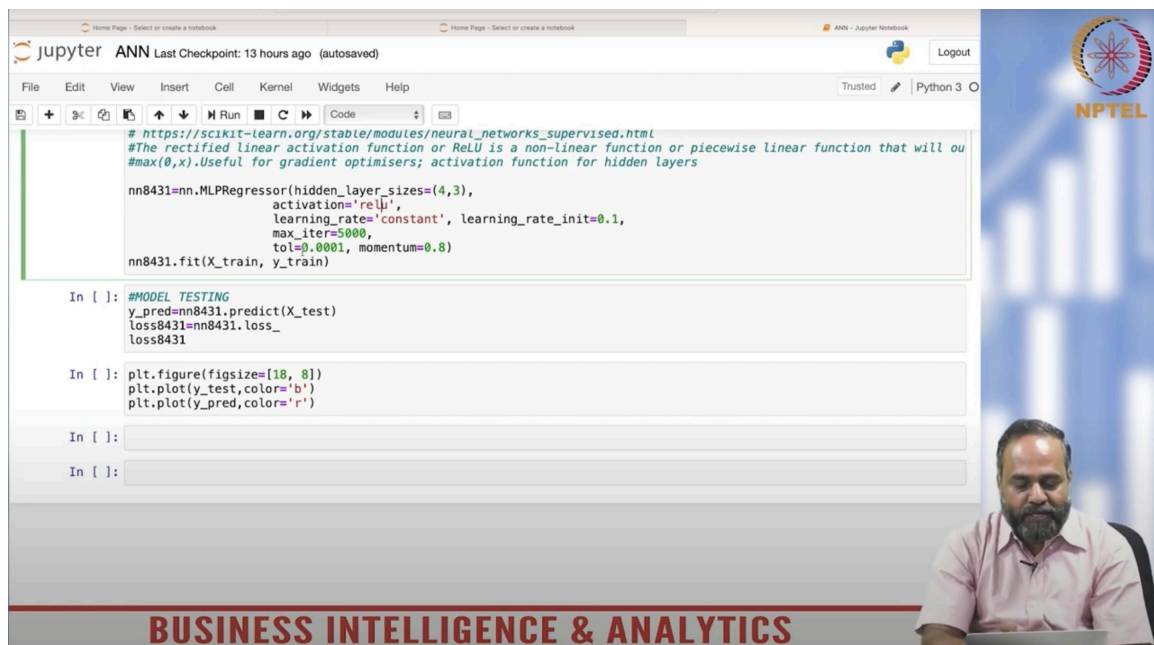
So, I am giving it as 4 and 3. It has to be given only for hidden layers here. Activation function is a ReLU function. ReLU is widely used like sigmoid. What does ReLU means? Those who have learned electronics and rectifiers, ReLU is a rectifier function, meaning what? If you have a sinusoidal function, you input the sinusoidal, it will give only the positive side, the negative side is dropped. So, that is a rectifier. So, if there is an input value, the output will be same as the input if it is a positive value. If it is a negative value, 0 is the output. So, it is a rectifier function, ReLU function.

Learning rate is constant and learning rate is initiated as 0.1. 0.1 is a learning rate value. Learning rate  $\eta$  is something that is a hyper parameter which you can adjust. So the value is between 0 and 1. When you work with neural networks, you can give any value between 0 and 1, but the implication is if you give a value close to 1, the training may not converge. There will be lot of oscillations, you can see. So, therefore, start with a low value, for learning rates like 0.1, 0.01 and train it again if it is converging, then give 0.2, 0.3 and see what value gives you best performance and, convergence and best performance. You can actually change these values, learning rate. And similarly, momentum is another parameter you can use to tune the algorithm.

Now, GDM, this is gradient descent with momentum. So, momentum is another parameter, hyper parameter you can adjust. Momentum follow the opposite principle. Start with a high value. If you give a low value for momentum, the training may not converge. Start with a 0.9 or 1 even if it is not converging, 0.9, 0.8 like that. So, these are tricks. So for learning rate, start from a low value, for momentum start from a high value. So, the learning rate is the rate at which the algorithm learns the patterns. So it is, at the end of the day it is an optimization that is happening and there is something called convergence.

So, all these values help in reaching convergence. So, if you put a high value as I said it may not converge, but higher the learning rate, faster the convergence. It is a rate. It will learn faster, but it may not converge also. So, learning rate and momentum in fact, tunes the training algorithm and it will have opposite effects. Let me not take you to other domains, but that is the method or the hyper parameters.

Then maximum number of iterations, number of times the data set will be used for deciding the weights, for computing the weights. So, you can decide the number of iterations as 5000 or 10000, but it has computational cost. More the number of iterations, more time it will take to train, but thanks to your computers today, it does not take much time. As was this exercise, say 20 years ago or 30 years ago you can start training a network and then go for your coffee and combine.



```
# https://scikit-learn.org/stable/modules/neural_networks_supervised.html
#The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will ou
#max(0,x).Useful for gradient optimisers; activation function for hidden layers

nn8431=nn.MLPRegressor(hidden_layer_sizes=(4,3),
                       activation='relu',
                       learning_rate='constant', learning_rate_init=0.1,
                       max_iter=5000,
                       tol=0.0001, momentum=0.8)

nn8431.fit(X_train, y_train)

In [ ]: #MODEL TESTING
y_pred=nn8431.predict(X_test)
loss8431=nn8431.loss_
loss8431

In [ ]: plt.figure(figsize=[18, 8])
plt.plot(y_test,color='b')
plt.plot(y_pred,color='r')

In [ ]:

In [ ]:
```

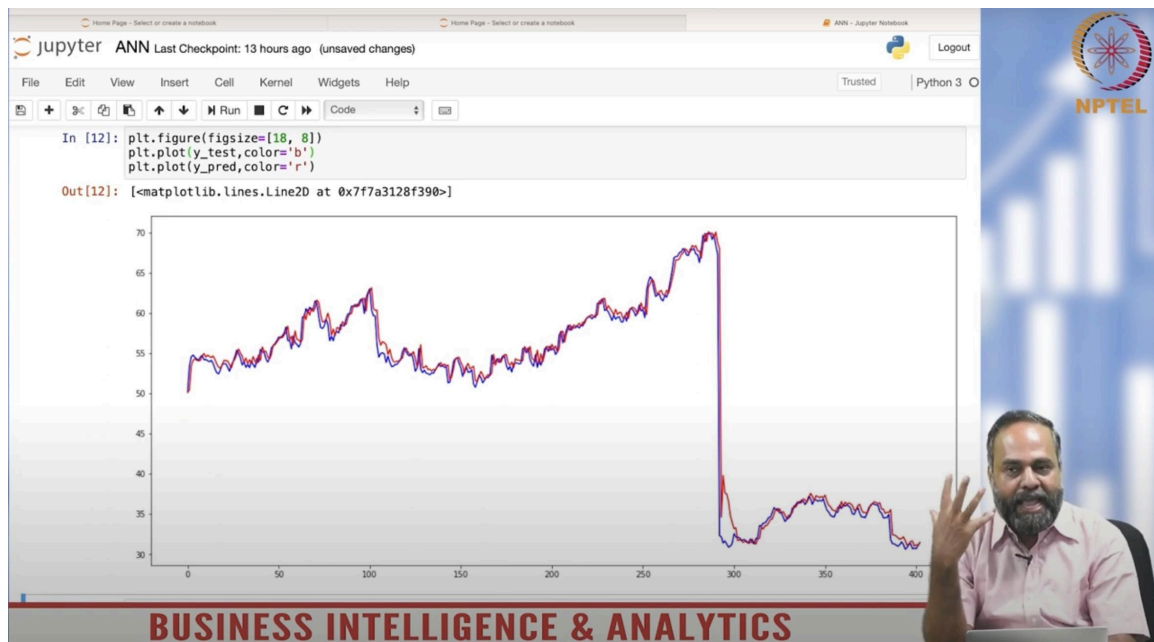
**BUSINESS INTELLIGENCE & ANALYTICS**

So, because this takes a lot of computation. So, these are the parameters that we input,

tolerance, the change between errors, if it actually becomes very low, then the training converges. So, these are actually hyper parameters that you use for training the algorithm. And after deciding or after specifying this, then you fit the data. So, 8 4 3 1 actually the data is, the training data is input, training data x train and y train. So that is the training it has done.

You have built the model and how far it took, you have 16 years of data and of course, training data is 70 percent of that, but it did it fast, quick. Now our effort is to look at the model, take the model and make it predict. So, we are giving x test as an input, x test as the input. The test data is given as input and then you get a predicted y pred, the predicted output is obtained. We already know what is the actual y test, which is the actual data, y pred is the predicted data and it also outputs the loss, loss is nothing but the error, the prediction error and it is MSE value, it is MSE which is labeled as loss.

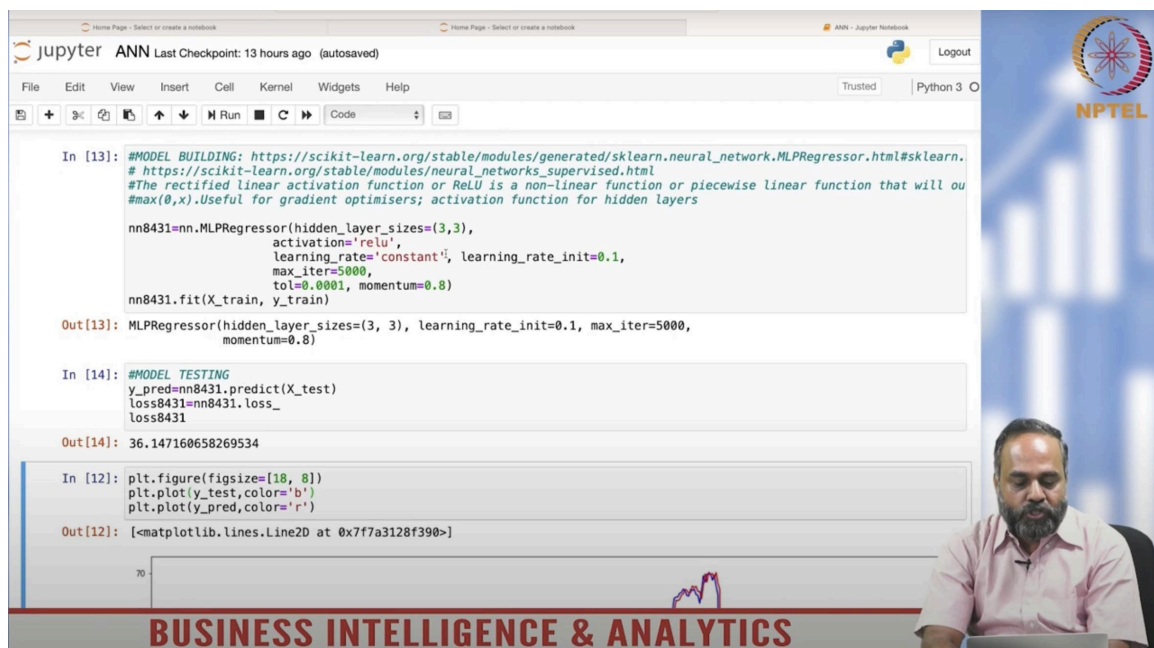
So, loss is the MSE or the prediction error. So, what is the prediction error? 34.57 and in the next cell, I am visualizing how well the model is performing. I am trying to visualize the model. Do you like it? What is it plotting? The predicted output and the actual value.



So, the prediction is closely following the actual. Do you agree? The predicted color is red and the test data, the actual data is blue. The blue and the red are close to each other. Now this visualization helps you appreciate ANN because ANN has the ability to model noisy data and non-linear data and predict with good accuracy. The prediction performance as visualized here, looks good. So, therefore, ANN is a very valid approach to modeling time series data, which is noisy and non-linear, widely used.

So, as researchers you may actually become curious at this point. I want to use an ARIMA for the same series and see how ANN versus ARIMA is better or worse. You can add scripts here, use that modeling technique and then compare, using the python scripts and compare the performance of models. And here as researchers you can actually, as I said change the hyper parameters like instead of 4 3, you can use 4 4 and see if the model performs better.

I am looking at my prediction error which is 34.57. Let me see instead of using 4 3, let me use a 3 3 and see if the model performance changes better. It is 36.147. So it is, slightly you know, it is not improving, it is actually getting worse, but you have to take many trials to establish this and between trials there could be changes because all the time the model starts with some random weights. The starting point, the random weight vector will be used. So, there will be changes between the different trainings. So, the trained models could be slightly different in terms of its performance because of the randomness involved.



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [13]: #MODEL BUILDING: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.
# https://scikit-learn.org/stable/modules/neural_networks_supervised.html
#The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will ou
#max(0,x).Useful for gradient optimisers; activation function for hidden layers

nn8431=nn.MLPRegressor(hidden_layer_sizes=(3,3),
                        activation='relu',
                        learning_rate='constant', learning_rate_init=0.1,
                        max_iter=5000,
                        tol=0.0001, momentum=0.8)
nn8431.fit(X_train, y_train)

Out[13]: MLPRegressor(hidden_layer_sizes=(3, 3), learning_rate_init=0.1, max_iter=5000,
momentum=0.8)

In [14]: #MODEL TESTING
y_pred=nn8431.predict(X_test)
loss8431=nn8431.loss_
loss8431

Out[14]: 36.147160658269534

In [12]: plt.figure(figsize=[18, 8])
plt.plot(y_test,color='b')
plt.plot(y_pred,color='r')

Out[12]: [<matplotlib.lines.Line2D at 0x7f7a3128f390>]
```

The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom that reads "BUSINESS INTELLIGENCE & ANALYTICS". On the right side, there is an NPTEL logo and a video feed of a man with a beard wearing a pink shirt.

So, my job was to get you started with ANN modeling. How you can design ANN network? How you can prepare data? You can see a lot of our time, I would say 70 percent of our effort was in data preparation, to prepare the data because if you do not prepare the data well, the model will not do anything, it will not behave. So therefore, all our effort went for that and then of course, calling the regressor and specifying the hyper parameters and then testing the model and then finally modulating, it is the next step.



Now, as a project we cannot end here, this is one model. Now, train an 8 4 3 1 model, train an 8 4 1 model, train an 8 3 1 model and report the performance of each model and then from a set of trained models you select the best model it could be an 8 4 3 1 or an 8 4 1 or 8 3 1, you have to decide that. Then use some validation data to finally, report its performance. That is how this exercise can end.

So, I have seen this is a master project in some students too, but then you learn a lot about different stock markets and see what model functions best in certain markets or certain assets etc. Those are all details of the application of this approach. I will close here.