About Stock Market Analysis:

- Stock market prediction is the process to determine the future value of company stock or other financial instruments traded on an exchange. The successful forecast of a stock’s future price could yield significant profit. My motivation in this project is that a good prediction helps us make better financial decisions (buy or sell) about the future. The main objective is to identify a high price for the next day to understand the movement of stocks in the market.I worked on a stock dataset from ***Investing.com***

Steps involved are:

- Importing libraries
- scraping datas
- Model Building
- Train the model
- Predict the model

  ### Library used

Scrap and Reading given dataset

investpyis a Python package to retrieve data from Investing.com, which provides data retrieval from up to 39952 stocks, 82221 funds, 11403 ETFs, 2029 currency crosses, 7797 indices, 688 bonds, 66 commodities, 250 certificates, and 4697 cryptocurrencies. I used **Natural Gas** Data's for this prediction

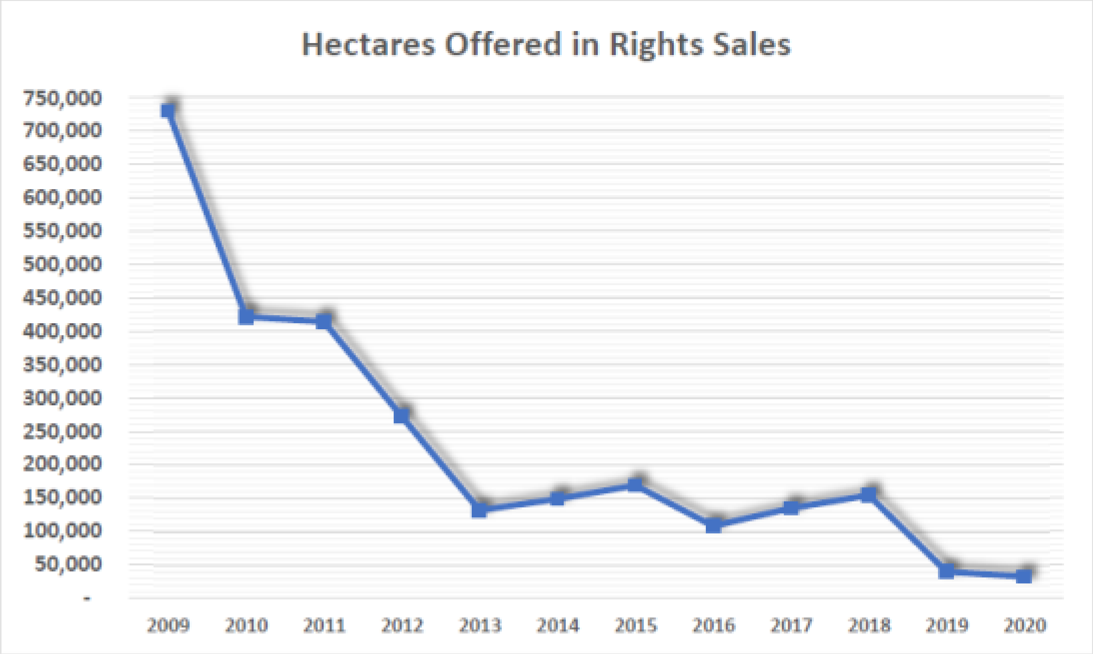
```
             Open   High    Low  Close  Volume Currency
Date
2010-01-04  5.685  5.784  5.595  5.703  455600      USD
2010-01-05  5.766  5.839  5.613  5.757  280100      USD
2010-01-06  5.766  6.046  5.730  5.938  295200      USD
2010-01-07  5.938  6.010  5.757  5.920  236300      USD
2010-01-08  5.956  6.046  5.830  6.001  222000      USD
```

- Government defers these revenue and recognizes it over time. Until 2011, the period was eight years, then it was nine years. In FY 2019, the revenue recognition period was changed to ten years.

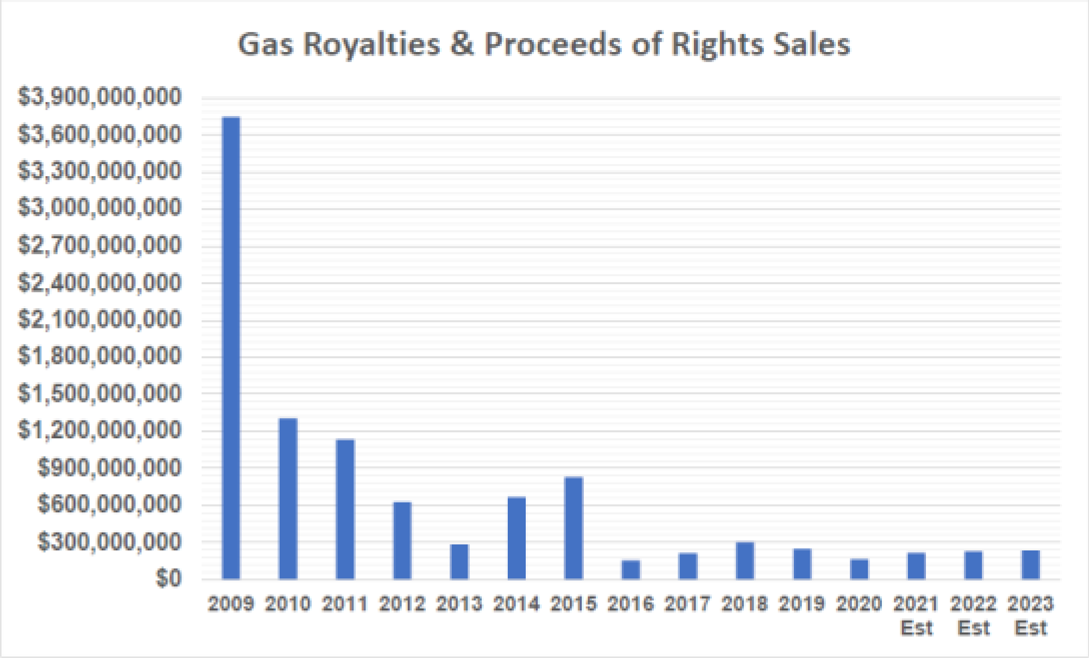
- In the years large sums were coming in, the revenue deferrals served Liberal governments. Hiding revenue allowed them to freeze social benefits and justify program cuts.

- Accountants might find it odd that revenue is spread over ten years on contracts that typically last three or five years. But, there is another factor at play. The Energy Ministry renews land deals administratively rather than re-tendering properties when contract period end.

- This works in favour of companies already involved in the gas fields and works against new entrants to the industry. This process of administrative renewals also works against the public. Fewer and fewer land parcels are available in the monthly auctions. This chart illustrates:



With the growth of royalty credit programs, that segment of natural gas revenue is disappearing as well. When looking at this chart, look back at the production figures.



Like Liberals before them, the Horgan NDP has no intention of maximizing the public share of natural resources.

They continue the royalty credits program without change and the total available to reduce producers' future payments is now between $2.7 and$3.0 billion.

Basic Checks

Out[3]:

	Open	High	Low	Close	Volume	Currency

Date	Open	High	Low	Close	Volume	Currency
Date						
2010-01-04	5.685	5.784	5.595	5.703	455600	USD
2010-01-05	5.766	5.839	5.613	5.757	280100	USD
2010-01-06	5.766	6.046	5.730	5.938	295200	USD
2010-01-07	5.938	6.010	5.757	5.920	236300	USD
2010-01-08	5.956	6.046	5.830	6.001	222000	USD

```
            Open  High    Low  Close  Volume Currency
Date
2022-05-25  5.78  5.78  5.560   5.71  166267      USD
2022-05-26  5.67  5.84  5.670   5.72  144295      USD
2022-05-27  5.79  5.84  5.680   5.76  379050      USD
2022-05-31  5.82  5.88  5.515   5.59  219186      USD
2022-06-01  5.63  5.71  5.570   5.66  150886      USD
```

Out[5]:

```
(3124, 6)
```

Out[6]:

```
18744
```

Out[7]:

```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Currency'], dtype='object')
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Open	3124.0	5.916586	2.616466	1.90	3.9200	5.1800	7.6500	15.25
High	3124.0	6.042227	2.665943	1.97	3.9875	5.3125	7.8025	15.25
Low	3124.0	5.785679	2.554219	1.90	3.8500	5.0500	7.4625	14.30
Close	3124.0	5.912754	2.611505	1.90	3.9200	5.1660	7.6505	14.61
Volume	3124.0	278740.730794	538382.273952	0.00	129202.5000	201417.5000	311676.2500	19669424.00

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3124 entries, 2010-01-04 to 2022-06-01
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Open      3124 non-null   float64
 1   High      3124 non-null   float64
 2   Low       3124 non-null   float64
 3   Close     3124 non-null   float64
 4   Volume    3124 non-null   int64
 5   Currency  3124 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 170.8+ KB
```

Checking for missing values

Out[10]:

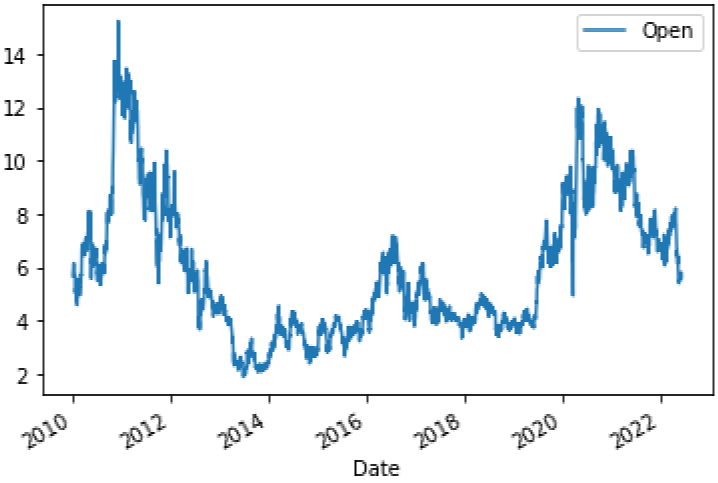
```
Open        0
High        0
Low         0
Close       0
Volume      0
```

```
Currency    0
dtype: int64
```

Remarks:

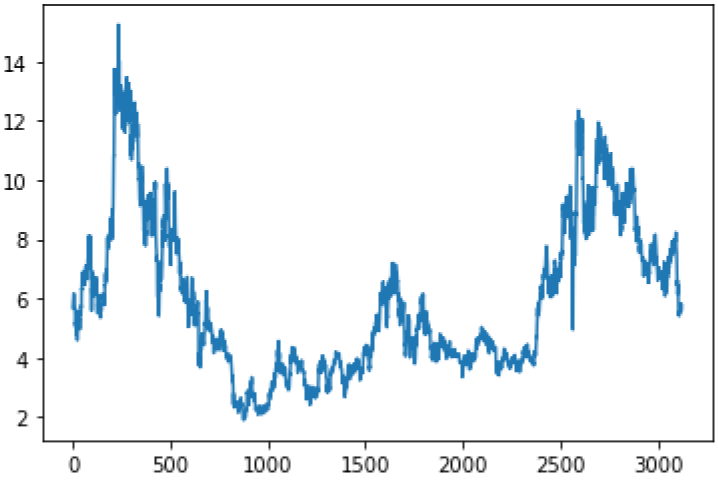
There are no missing values

```
Out[12]: <AxesSubplot:xlabel='Date'>
```



```
Out[14]: array([[5.685],
                [5.766],
                [5.766],
                ...,
                [5.79 ],
                [5.82 ],
                [5.63 ]])
```

```
Out[15]: [<matplotlib.lines.Line2D at 0x1259d39aa60>]
```



I normalized stock prices by using min-max normalization for each stock. The goal of normalization is to change the values of price columns in a dataset to a common scale without distorting differences in the range of the values. This can be applied when features have different ranges (scales of inputs are wildly different).

$$p_{\ scaled} = \frac{p - p_{min}}{p_{max\,-}\, p_{min}}$$

```
Out[18]: (3124, 3124)
```

```
[[0.2835206 ]
 [0.28958801]
 [0.28958801]
 ...
 [0.29138577]
 [0.29363296]
 [0.27940075]]
```

```
Out[20]: (2186, 938)
```

```
Out[22]: (2186, 938)
```

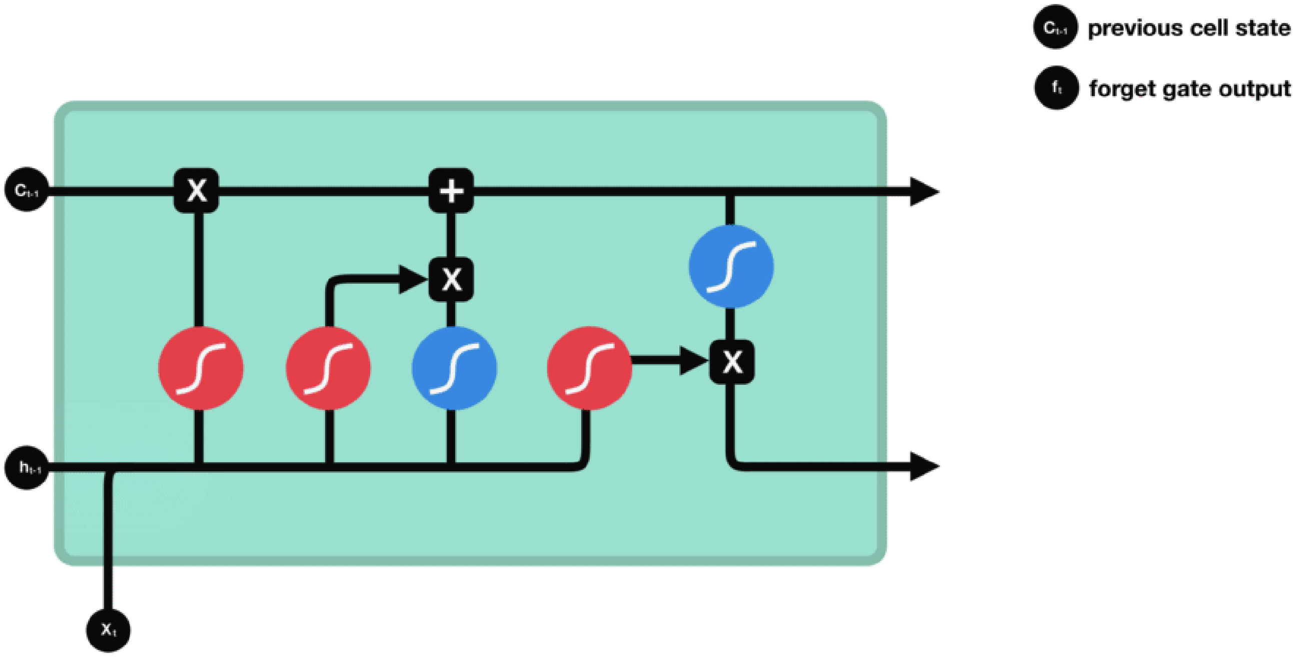
Introduction to Long Short Term Memory

- **LSTM**s were introduced by Hochreiter & Schmidhuber (1997), and they are explicitly designed to avoid the long-range issue that a vanilla RNN faces. They are slightly different than RNNs by using a different function to compute the hidden state. LSTM network consists of several memory blocks called cells. Two states are being passed to the next cell; the cell state and the hidden state. The LSTMs can add or remove information to the cell state via gates.

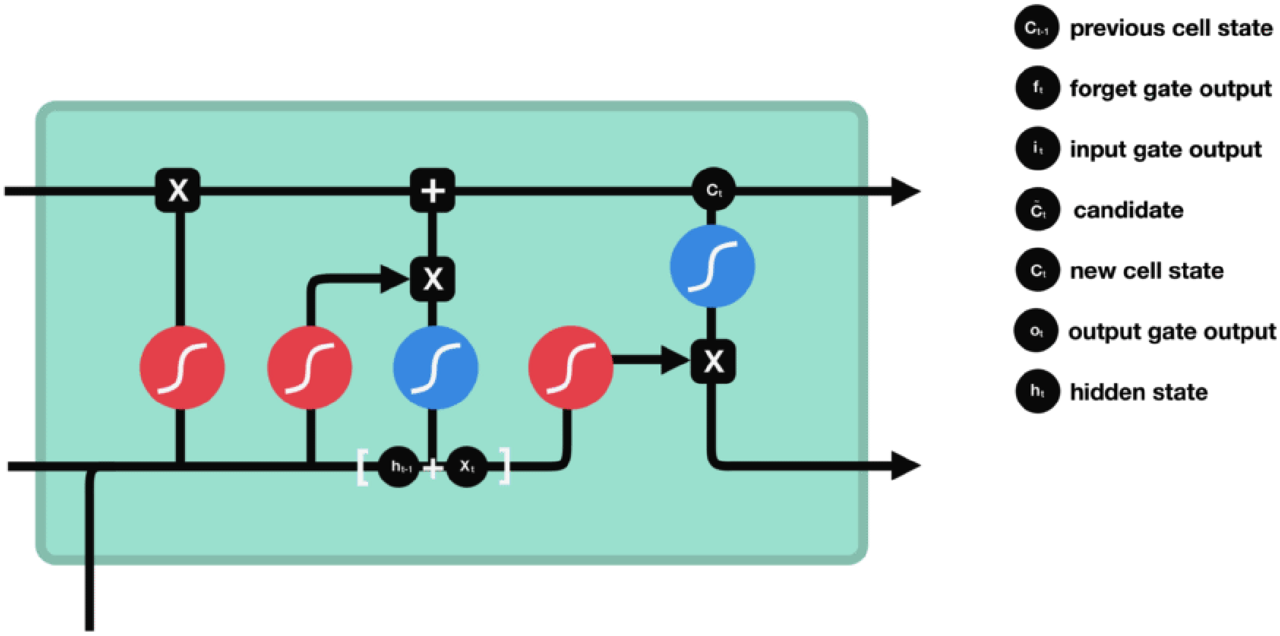
- An **LSTM** cell has 5 vital components that allow it to utilize both long-term and short-term data: the cell state, hidden state, input gate, forget gate and output gate.

- **Forget gate layer:** The decision of what information is going to pass from the cell state is done by the “forget gate layer.” It gives a number between 0 and 1 for each number in the cell state by using the sigmoid function. While 1 shows “let input through”, 0 means “do not let input through”.

- **Input gate layer:** It manages the process of the addition of information to the cell state (decide which values to update). Firstly, it regulates what values need to be added to the cell state by using a sigmoid function. Then, it creates a vector including all possible values that can be added to the cell state by using the tanh function, which outputs values from -1 to +1. It multiplies the value of the filter (the sigmoid gate) to the created vector (the tanh function) and so it transfers this useful information to the cell state via addition operation.



- **Output gate layer:** In that step, the network selects useful information from the current cell state and shows as output is done via the output gate.



```
Out[25]:  ((2085, 100), (2085,))
```

```
Out[26]:  ((837, 100), (837,))
```

Build the Model

- We define the reconstruction LSTM Autoencoder architecture that expects input sequence with 3

```
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Training model with adam optimizer and mean squared error loss function

```
Epoch 1/100
33/33 [==============================] - 7s 122ms/step - loss: 0.0175 - val_loss: 0.0049
Epoch 2/100
33/33 [==============================] - 3s 104ms/step - loss: 0.0021 - val_loss: 0.0029
Epoch 3/100
33/33 [==============================] - 4s 129ms/step - loss: 0.0018 - val_loss: 0.0027
Epoch 4/100
33/33 [==============================] - 3s 102ms/step - loss: 0.0018 - val_loss: 0.0025
Epoch 5/100
33/33 [==============================] - 3s 101ms/step - loss: 0.0016 - val_loss: 0.0022
Epoch 6/100
33/33 [==============================] - 3s 100ms/step - loss: 0.0016 - val_loss: 0.0021
Epoch 7/100
33/33 [==============================] - 3s 100ms/step - loss: 0.0014 - val_loss: 0.0023
Epoch 8/100
33/33 [==============================] - 4s 110ms/step - loss: 0.0014 - val_loss: 0.0028
Epoch 9/100
33/33 [==============================] - 3s 101ms/step - loss: 0.0014 - val_loss: 0.0016
Epoch 10/100
33/33 [==============================] - 4s 133ms/step - loss: 0.0012 - val_loss: 0.0017
Epoch 11/100
33/33 [==============================] - 4s 125ms/step - loss: 0.0011 - val_loss: 0.0016
Epoch 12/100
33/33 [==============================] - 4s 119ms/step - loss: 0.0010 - val_loss: 0.0014
Epoch 13/100
33/33 [==============================] - 4s 117ms/step - loss: 9.9947e-04 - val_loss: 0.00
13
Epoch 14/100
33/33 [==============================] - 4s 123ms/step - loss: 0.0010 - val_loss: 0.0012
Epoch 15/100
33/33 [==============================] - 4s 110ms/step - loss: 0.0010 - val_loss: 0.0013
Epoch 16/100
33/33 [==============================] - 3s 105ms/step - loss: 9.1073e-04 - val_loss: 0.00
15
Epoch 17/100
33/33 [==============================] - 4s 111ms/step - loss: 8.3409e-04 - val_loss: 0.00
11
Epoch 18/100
33/33 [==============================] - 4s 129ms/step - loss: 7.8086e-04 - val_loss: 0.00
11
Epoch 19/100
33/33 [==============================] - 4s 120ms/step - loss: 7.4905e-04 - val_loss: 9.73
72e-04
Epoch 20/100
33/33 [==============================] - 4s 116ms/step - loss: 7.1724e-04 - val_loss: 0.00
```

```
13
Epoch 21/100
33/33 [==============================] - 4s 113ms/step - loss: 8.2165e-04 - val_loss: 0.00
13
Epoch 22/100
33/33 [==============================] - 5s 140ms/step - loss: 8.2116e-04 - val_loss: 9.67
94e-04
Epoch 23/100
33/33 [==============================] - 4s 116ms/step - loss: 7.1031e-04 - val_loss: 0.00
11
Epoch 24/100
33/33 [==============================] - 4s 114ms/step - loss: 6.0076e-04 - val_loss: 8.38
04e-04
Epoch 25/100
33/33 [==============================] - 3s 104ms/step - loss: 6.2057e-04 - val_loss: 8.53
10e-04
Epoch 26/100
33/33 [==============================] - 4s 107ms/step - loss: 5.5274e-04 - val_loss: 8.81
82e-04
Epoch 27/100
33/33 [==============================] - 4s 116ms/step - loss: 5.8108e-04 - val_loss: 7.97
76e-04
Epoch 28/100
33/33 [==============================] - 4s 106ms/step - loss: 6.3044e-04 - val_loss: 7.82
43e-04
Epoch 29/100
33/33 [==============================] - 4s 129ms/step - loss: 5.0241e-04 - val_loss: 7.57
88e-04
Epoch 30/100
33/33 [==============================] - 4s 121ms/step - loss: 5.0791e-04 - val_loss: 8.48
38e-04
Epoch 31/100
33/33 [==============================] - 3s 106ms/step - loss: 4.7929e-04 - val_loss: 7.20
53e-04
Epoch 32/100
33/33 [==============================] - 3s 98ms/step - loss: 4.9232e-04 - val_loss: 9.924
2e-04
Epoch 33/100
33/33 [==============================] - 3s 97ms/step - loss: 5.1013e-04 - val_loss: 9.521
2e-04
Epoch 34/100
33/33 [==============================] - 4s 110ms/step - loss: 4.9599e-04 - val_loss: 6.32
33e-04
Epoch 35/100
33/33 [==============================] - 4s 115ms/step - loss: 4.1381e-04 - val_loss: 6.58
47e-04
Epoch 36/100
33/33 [==============================] - 4s 114ms/step - loss: 4.5563e-04 - val_loss: 0.00
10
Epoch 37/100
33/33 [==============================] - 4s 108ms/step - loss: 5.1789e-04 - val_loss: 6.63
22e-04
Epoch 38/100
33/33 [==============================] - 4s 115ms/step - loss: 4.4510e-04 - val_loss: 7.64
86e-04
Epoch 39/100
33/33 [==============================] - 4s 120ms/step - loss: 4.1058e-04 - val_loss: 9.22
13e-04
Epoch 40/100
33/33 [==============================] - 4s 107ms/step - loss: 4.1141e-04 - val_loss: 5.64
07e-04
Epoch 41/100
33/33 [==============================] - 4s 122ms/step - loss: 3.7087e-04 - val_loss: 5.55
44e-04
Epoch 42/100
33/33 [==============================] - 3s 105ms/step - loss: 3.8128e-04 - val_loss: 5.39
```

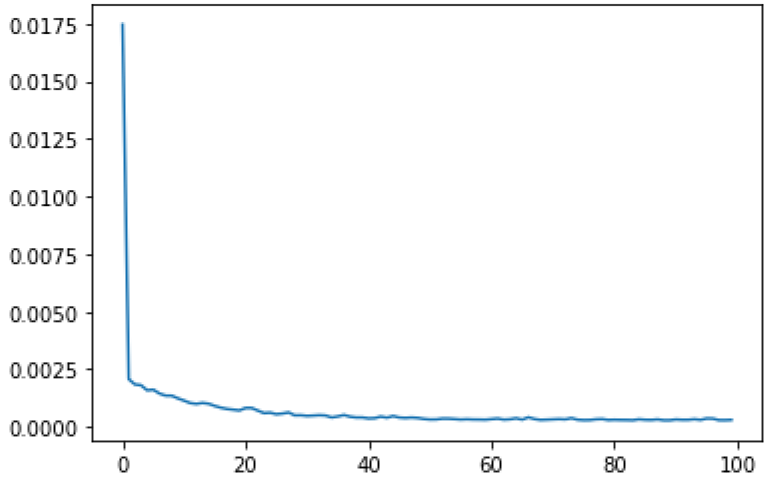
```
70e-04
Epoch 43/100
33/33 [==============================] - 4s 112ms/step - loss: 4.4983e-04 - val_loss: 6.14
78e-04
Epoch 44/100
33/33 [==============================] - 3s 99ms/step - loss: 4.0198e-04 - val_loss: 5.348
7e-04
Epoch 45/100
33/33 [==============================] - 3s 100ms/step - loss: 4.7285e-04 - val_loss: 9.01
59e-04
Epoch 46/100
33/33 [==============================] - 3s 106ms/step - loss: 4.0830e-04 - val_loss: 6.68
56e-04
Epoch 47/100
33/33 [==============================] - 3s 103ms/step - loss: 3.8950e-04 - val_loss: 5.37
65e-04
Epoch 48/100
33/33 [==============================] - 4s 107ms/step - loss: 4.0864e-04 - val_loss: 0.00
11
Epoch 49/100
33/33 [==============================] - 4s 107ms/step - loss: 3.8962e-04 - val_loss: 5.27
37e-04
Epoch 50/100
33/33 [==============================] - 4s 108ms/step - loss: 3.4870e-04 - val_loss: 5.24
19e-04
Epoch 51/100
33/33 [==============================] - 3s 106ms/step - loss: 3.2413e-04 - val_loss: 5.50
72e-04
Epoch 52/100
33/33 [==============================] - 4s 118ms/step - loss: 3.2686e-04 - val_loss: 5.98
41e-04
Epoch 53/100
33/33 [==============================] - 4s 108ms/step - loss: 3.6134e-04 - val_loss: 5.57
16e-04
Epoch 54/100
33/33 [==============================] - 4s 114ms/step - loss: 3.6021e-04 - val_loss: 5.25
47e-04
Epoch 55/100
33/33 [==============================] - 4s 111ms/step - loss: 3.4369e-04 - val_loss: 5.17
56e-04
Epoch 56/100
33/33 [==============================] - 3s 100ms/step - loss: 3.2206e-04 - val_loss: 8.29
54e-04
Epoch 57/100
33/33 [==============================] - 3s 99ms/step - loss: 3.3563e-04 - val_loss: 5.009
0e-04
Epoch 58/100
33/33 [==============================] - 3s 97ms/step - loss: 3.2337e-04 - val_loss: 4.974
4e-04
Epoch 59/100
33/33 [==============================] - 3s 97ms/step - loss: 3.2100e-04 - val_loss: 5.062
2e-04
Epoch 60/100
33/33 [==============================] - 3s 98ms/step - loss: 3.1436e-04 - val_loss: 5.003
3e-04
Epoch 61/100
33/33 [==============================] - 3s 99ms/step - loss: 3.4000e-04 - val_loss: 6.910
2e-04
Epoch 62/100
33/33 [==============================] - 4s 108ms/step - loss: 3.6517e-04 - val_loss: 5.21
06e-04
Epoch 63/100
33/33 [==============================] - 3s 103ms/step - loss: 3.1739e-04 - val_loss: 5.23
79e-04
Epoch 64/100
33/33 [==============================] - 4s 118ms/step - loss: 3.3364e-04 - val_loss: 5.41
```

```
75e-04
Epoch 65/100
33/33 [==============================] - 4s 110ms/step - loss: 3.7732e-04 - val_loss: 5.62
81e-04
Epoch 66/100
33/33 [==============================] - 3s 102ms/step - loss: 3.1925e-04 - val_loss: 5.74
92e-04
Epoch 67/100
33/33 [==============================] - 3s 101ms/step - loss: 4.1837e-04 - val_loss: 5.02
39e-04
Epoch 68/100
33/33 [==============================] - 3s 100ms/step - loss: 3.4337e-04 - val_loss: 5.59
66e-04
Epoch 69/100
33/33 [==============================] - 3s 102ms/step - loss: 3.0509e-04 - val_loss: 5.47
37e-04
Epoch 70/100
33/33 [==============================] - 3s 105ms/step - loss: 3.1619e-04 - val_loss: 5.77
31e-04
Epoch 71/100
33/33 [==============================] - 3s 102ms/step - loss: 3.2936e-04 - val_loss: 7.87
36e-04
Epoch 72/100
33/33 [==============================] - 4s 109ms/step - loss: 3.4250e-04 - val_loss: 5.12
39e-04
Epoch 73/100
33/33 [==============================] - 3s 103ms/step - loss: 3.2956e-04 - val_loss: 4.94
67e-04
Epoch 74/100
33/33 [==============================] - 3s 97ms/step - loss: 3.8327e-04 - val_loss: 5.921
6e-04
Epoch 75/100
33/33 [==============================] - 3s 101ms/step - loss: 3.1429e-04 - val_loss: 5.25
94e-04
Epoch 76/100
33/33 [==============================] - 3s 99ms/step - loss: 2.9577e-04 - val_loss: 5.209
7e-04
Epoch 77/100
33/33 [==============================] - 3s 98ms/step - loss: 3.0017e-04 - val_loss: 5.004
0e-04
Epoch 78/100
33/33 [==============================] - 3s 100ms/step - loss: 3.3314e-04 - val_loss: 8.18
23e-04
Epoch 79/100
33/33 [==============================] - 3s 104ms/step - loss: 3.4362e-04 - val_loss: 4.93
17e-04
Epoch 80/100
33/33 [==============================] - 3s 103ms/step - loss: 3.0069e-04 - val_loss: 5.90
03e-04
Epoch 81/100
33/33 [==============================] - 3s 104ms/step - loss: 3.1166e-04 - val_loss: 4.96
64e-04
Epoch 82/100
33/33 [==============================] - 3s 102ms/step - loss: 3.0333e-04 - val_loss: 4.98
76e-04
Epoch 83/100
33/33 [==============================] - 3s 101ms/step - loss: 3.0056e-04 - val_loss: 5.04
32e-04
Epoch 84/100
33/33 [==============================] - 4s 110ms/step - loss: 2.9303e-04 - val_loss: 5.65
01e-04
Epoch 85/100
33/33 [==============================] - 3s 102ms/step - loss: 3.3273e-04 - val_loss: 7.91
84e-04
Epoch 86/100
33/33 [==============================] - 4s 110ms/step - loss: 3.0948e-04 - val_loss: 4.95
```

```
82e-04
Epoch 87/100
33/33 [==============================] - 4s 112ms/step - loss: 3.0294e-04 - val_loss: 5.54
49e-04
Epoch 88/100
33/33 [==============================] - 3s 102ms/step - loss: 3.2767e-04 - val_loss: 5.03
83e-04
Epoch 89/100
33/33 [==============================] - 3s 99ms/step - loss: 2.9028e-04 - val_loss: 5.319
6e-04
Epoch 90/100
33/33 [==============================] - 3s 96ms/step - loss: 2.9034e-04 - val_loss: 8.550
2e-04
Epoch 91/100
33/33 [==============================] - 3s 104ms/step - loss: 3.2456e-04 - val_loss: 7.53
85e-04
Epoch 92/100
33/33 [==============================] - 3s 103ms/step - loss: 3.0863e-04 - val_loss: 5.94
87e-04
Epoch 93/100
33/33 [==============================] - 3s 96ms/step - loss: 3.0945e-04 - val_loss: 5.602
7e-04
Epoch 94/100
33/33 [==============================] - 3s 97ms/step - loss: 3.3652e-04 - val_loss: 5.054
4e-04
Epoch 95/100
33/33 [==============================] - 3s 97ms/step - loss: 3.0621e-04 - val_loss: 5.383
2e-04
Epoch 96/100
33/33 [==============================] - 3s 97ms/step - loss: 3.6758e-04 - val_loss: 8.816
2e-04
Epoch 97/100
33/33 [==============================] - 3s 98ms/step - loss: 3.6169e-04 - val_loss: 5.442
6e-04
Epoch 98/100
33/33 [==============================] - 3s 106ms/step - loss: 2.9573e-04 - val_loss: 5.19
85e-04
Epoch 99/100
33/33 [==============================] - 3s 100ms/step - loss: 2.9331e-04 - val_loss: 5.67
64e-04
Epoch 100/100
33/33 [==============================] - 3s 99ms/step - loss: 3.0567e-04 - val_loss: 6.804
2e-04
```

```
Out[30]: <keras.callbacks.History at 0x156ec868ac0>
```

```
Out[31]: [<matplotlib.lines.Line2D at 0x156812162e0>]
```

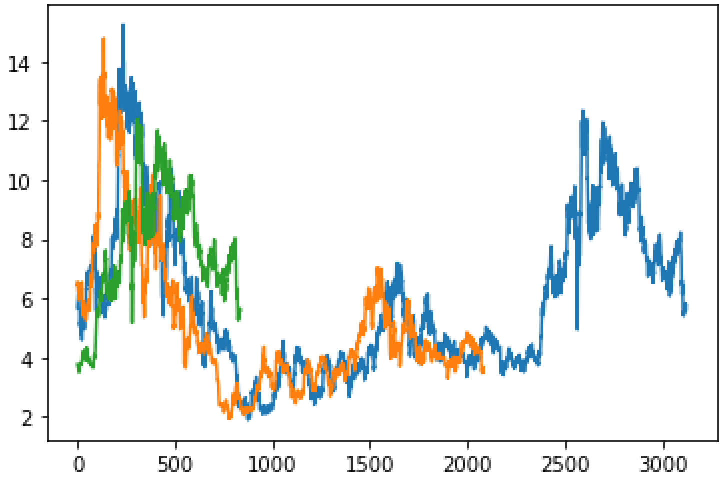


- That is, loss is a number indicating how bad the model's prediction was on a single example.

- If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.
- The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.

Out[124…

```
[<matplotlib.lines.Line2D at 0x15685061fd0>]
```



Out[125…

```
numpy.ndarray
```

Out[126…

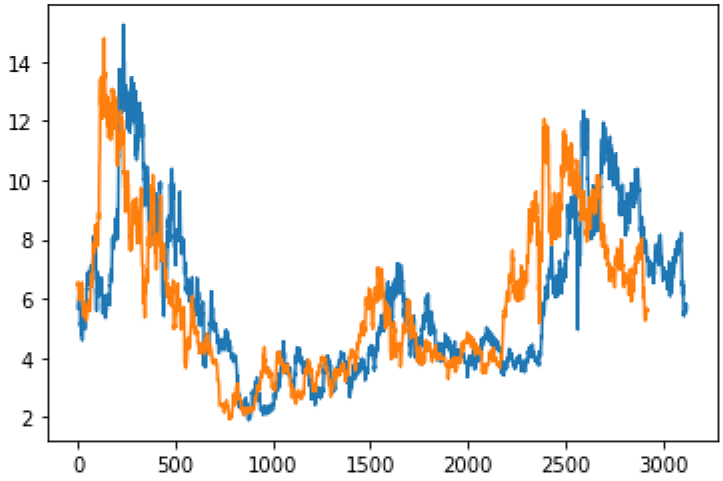
```
5.3660773295323025
```

Out[127…

```
7.43215022209674
```

Out[129…

```
[<matplotlib.lines.Line2D at 0x1568506c190>]
```



Out[130…

```
938
```

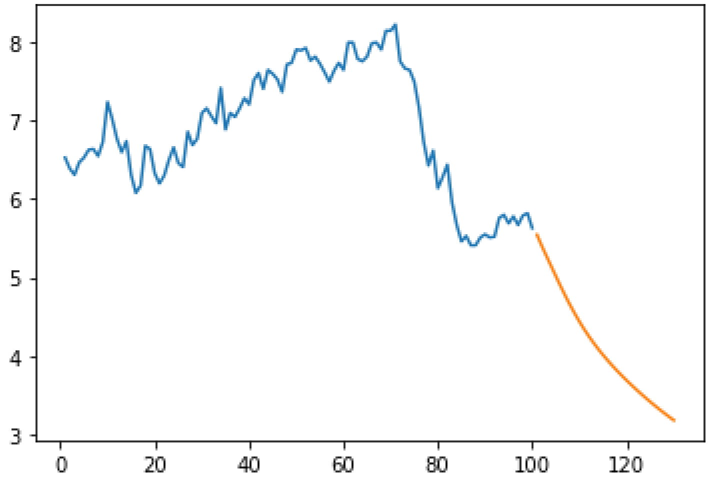
Out[134…

```
(1, 100)
```

```
[[0.2730453610420227], [0.2625875473022461], [0.25278881192207336], [0.24312253296375275],
[0.23345814645290375], [0.22394904494285583], [0.21475909650325775], [0.2059978693723678
6], [0.19772593677043915], [0.18996651470661163], [0.18271346390247345], [0.17594011127948
76], [0.16960634291172028], [0.16366510093212128], [0.15806734561920166], [0.1527660042047
5006], [0.14771823585033417], [0.1428869366645813], [0.13824161887168884], [0.133757859468
46008], [0.12941730022243042], [0.12520654499530792], [0.12111657857894897], [0.11714143306
016922], [0.1132776215672493], [0.10952312499284744], [0.10587689280509949], [0.1023382321
0000992], [0.09890638291835785], [0.09558042883872986]]
```

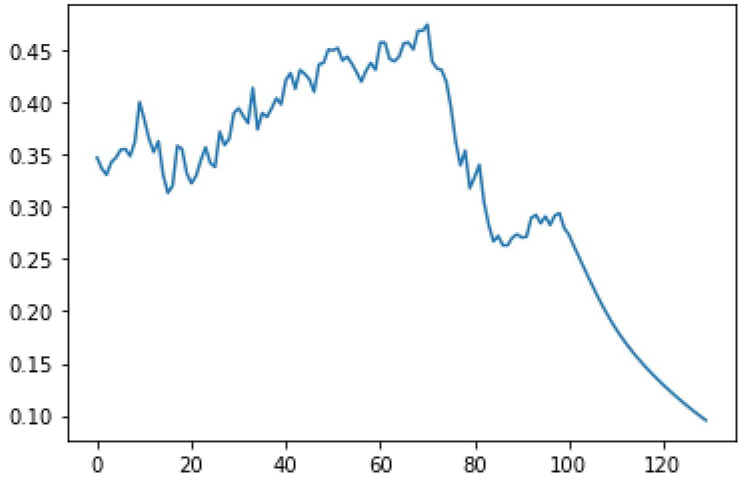
```
Out[137…  3124
```

```
This green line is the predicted line
```

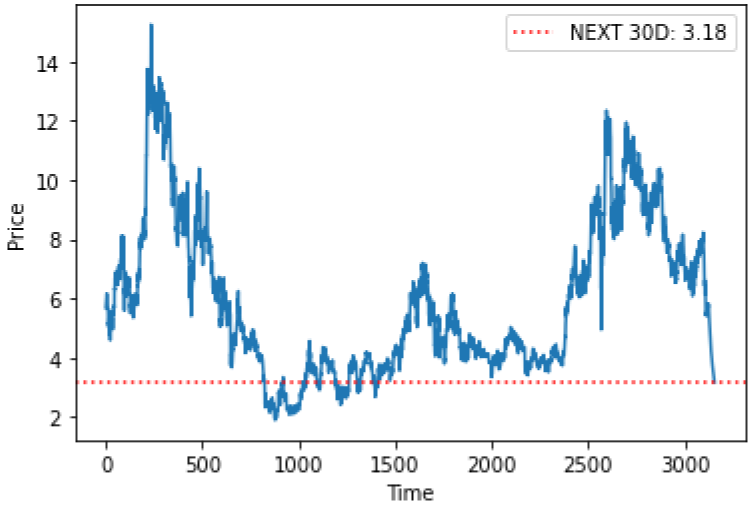


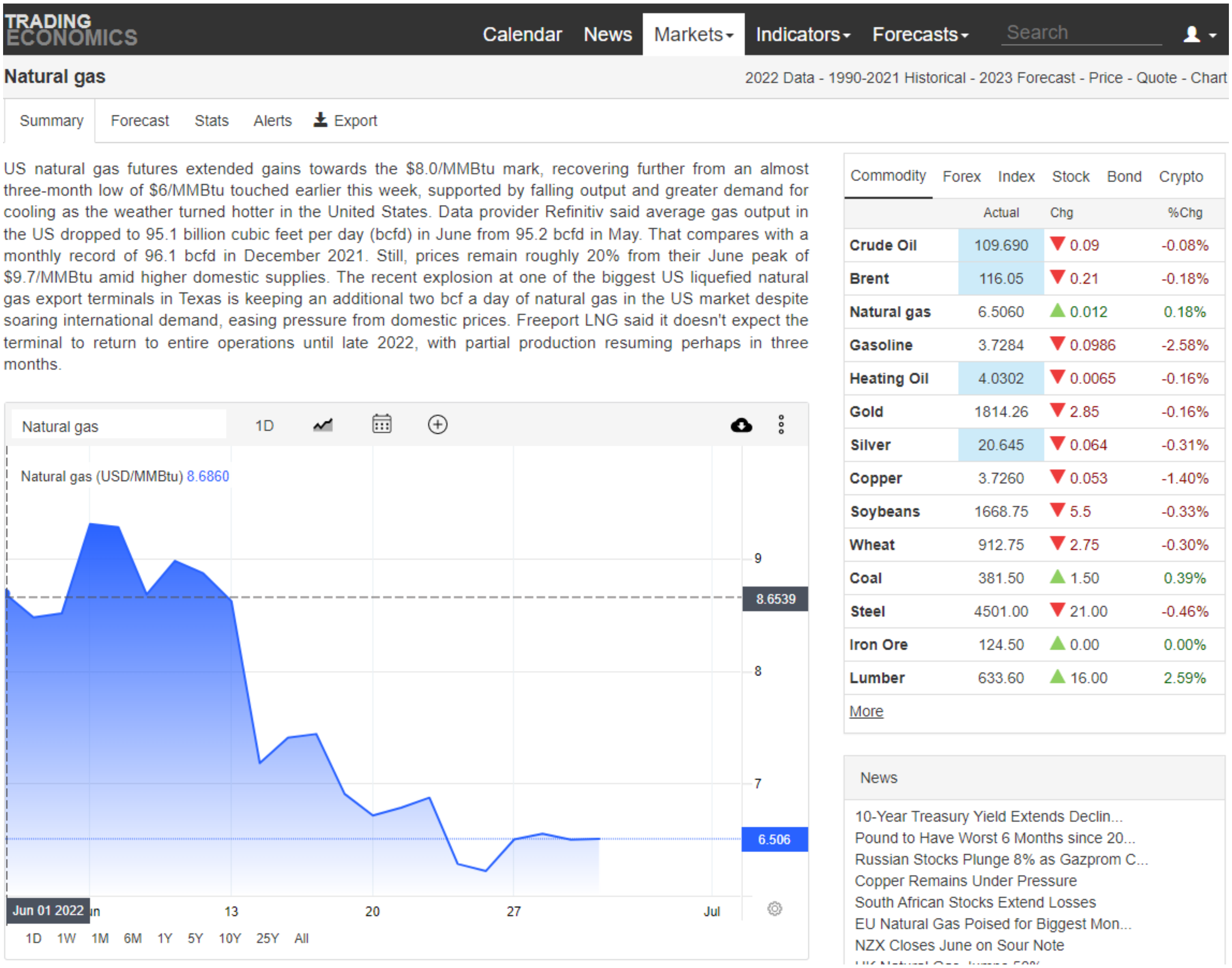
```
Out[115…  3124
```

```
Out[116…  [<matplotlib.lines.Line2D at 0x156828eb430>]
```



```
Out[63]:  <matplotlib.legend.Legend at 0x156818a57f0>
```





Natural gas

2022 Data - 1990-2021 Historical - 2023 Forecast - Price - Quote - Chart

US natural gas futures extended gains towards the $8.0/MMBtu mark, recovering further from an almost three-month low of $6/MMBtu touched earlier this week, supported by falling output and greater demand for cooling as the weather turned hotter in the United States. Data provider Refinitiv said average gas output in the US dropped to 95.1 billion cubic feet per day (bcfd) in June from 95.2 bcfd in May. That compares with a monthly record of 96.1 bcfd in December 2021. Still, prices remain roughly 20% from their June peak of $9.7/MMBtu amid higher domestic supplies. The recent explosion at one of the biggest US liquefied natural gas export terminals in Texas is keeping an additional two bcf a day of natural gas in the US market despite soaring international demand, easing pressure from domestic prices. Freeport LNG said it doesn't expect the terminal to return to entire operations until late 2022, with partial production resuming perhaps in three months.

	Actual	Chg	%Chg
Crude Oil	109.690	▼ 0.09	-0.08%
Brent	116.05	▼ 0.21	-0.18%
Natural gas	6.5060	▲ 0.012	0.18%
Gasoline	3.7284	▼ 0.0986	-2.58%
Heating Oil	4.0302	▼ 0.0065	-0.16%
Gold	1814.26	▼ 2.85	-0.16%
Silver	20.645	▼ 0.064	-0.31%
Copper	3.7260	▼ 0.053	-1.40%
Soybeans	1668.75	▼ 5.5	-0.33%
Wheat	912.75	▼ 2.75	-0.30%
Coal	381.50	▲ 1.50	0.39%
Steel	4501.00	▼ 21.00	-0.46%
Iron Ore	124.50	▲ 0.00	0.00%
Lumber	633.60	▲ 16.00	2.59%

More

News

10-Year Treasury Yield Extends Declin...
Pound to Have Worst 6 Months since 20...
Russian Stocks Plunge 8% as Gazprom C...
Copper Remains Under Pressure
South African Stocks Extend Losses
EU Natural Gas Poised for Biggest Mon...
NZX Closes June on Sour Note
UK Natural Gas Jumps 50%

In this model i tried to show next 30Days predicton data from the date 01.06.2022 and comparing my predicted data with actual stock graph which graph i took from **Trading Economy**

Model create by : **Soumyadarshan Dah**