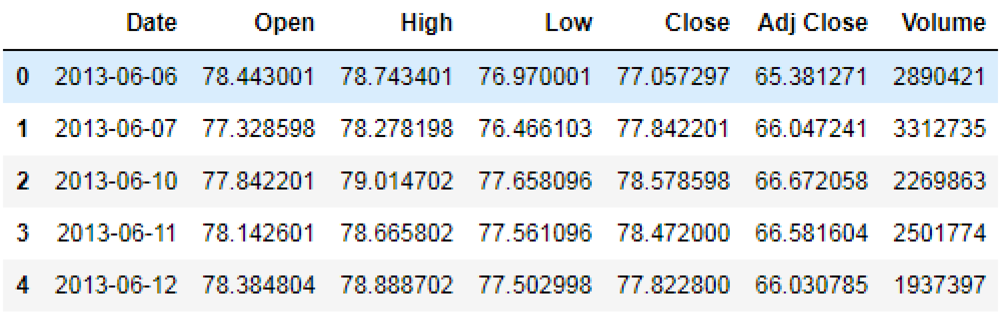
I am going to try to predict the values of Siemens stock price, for the last 30 days, using as a variable, the closing values of the previous 20 days, by training a **Recurrent Neural Network.**

We will use data downloaded from the following link:

https://finance.yahoo.com/quote/SIE.DE/history?period1=1496696400&period2=1528232400&interval=1d&filter=history&frequency=1d

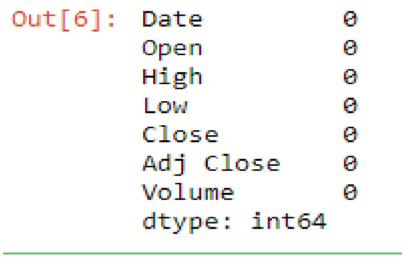
I have downloaded data of the last 5 years.

These are the five first rows, and just by their names it is very clear what they represent



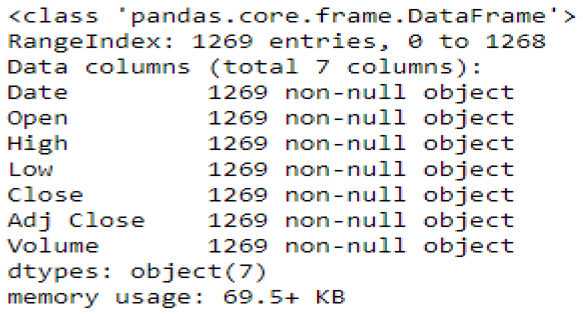
	Date	Open	High	Low	Close	Adj Close	Volume
0	2013-06-06	78.443001	78.743401	76.970001	77.057297	65.381271	2890421
1	2013-06-07	77.328598	78.278198	76.466103	77.842201	66.047241	3312735
2	2013-06-10	77.842201	79.014702	77.658096	78.578598	66.672058	2269863
3	2013-06-11	78.142601	78.665802	77.561096	78.472000	66.581604	2501774
4	2013-06-12	78.384804	78.888702	77.502998	77.822800	66.030785	1937397

Checking each column for missing values



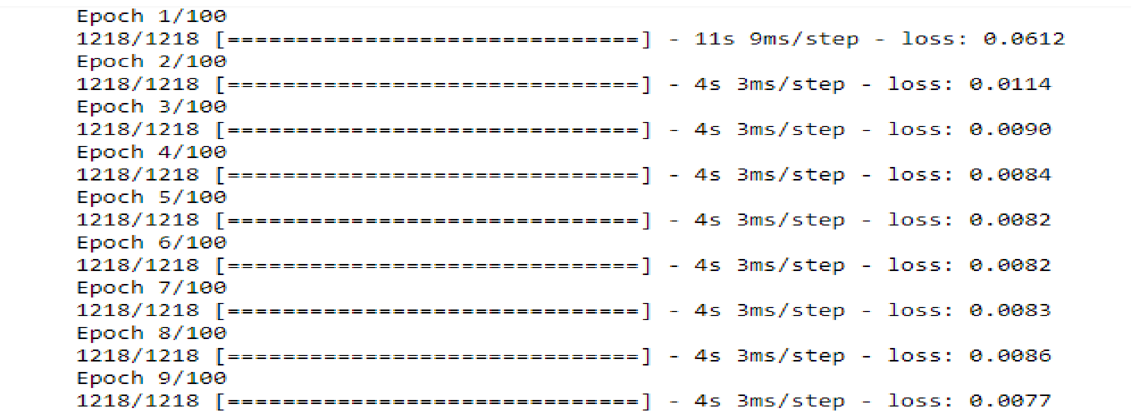
```
Out[6]: Date         0
        Open         0
        High         0
        Low          0
        Close        0
        Adj Close    0
        Volume       0
        dtype: int64
```

Here, we can see the type of our variables, all of them are objects, so the “Close” column, must be transformed.



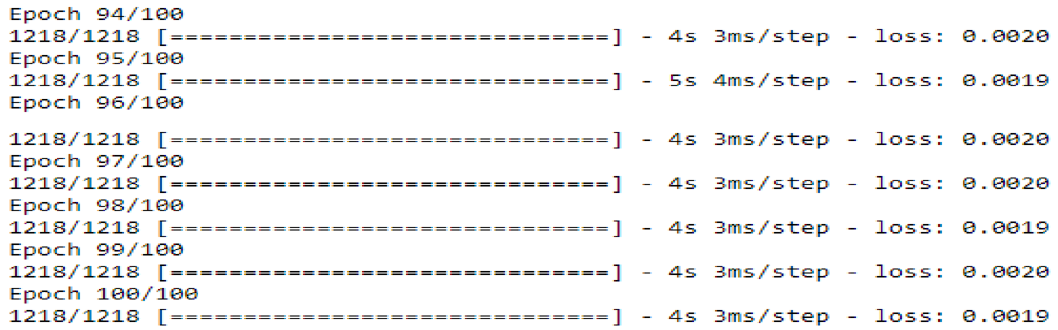
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1269 entries, 0 to 1268
Data columns (total 7 columns):
Date         1269 non-null object
Open         1269 non-null object
High         1269 non-null object
Low          1269 non-null object
Close        1269 non-null object
Adj Close    1269 non-null object
Volume       1269 non-null object
dtypes: object(7)
memory usage: 69.5+ KB
```

After, all the necessary manipulations, described In the ipynb file, we trained the RNN.



```
Epoch 1/100
1218/1218 [==============================] - 11s 9ms/step - loss: 0.0612
Epoch 2/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0114
Epoch 3/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0090
Epoch 4/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0084
Epoch 5/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0082
Epoch 6/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0082
Epoch 7/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0083
Epoch 8/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0086
Epoch 9/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0077
```

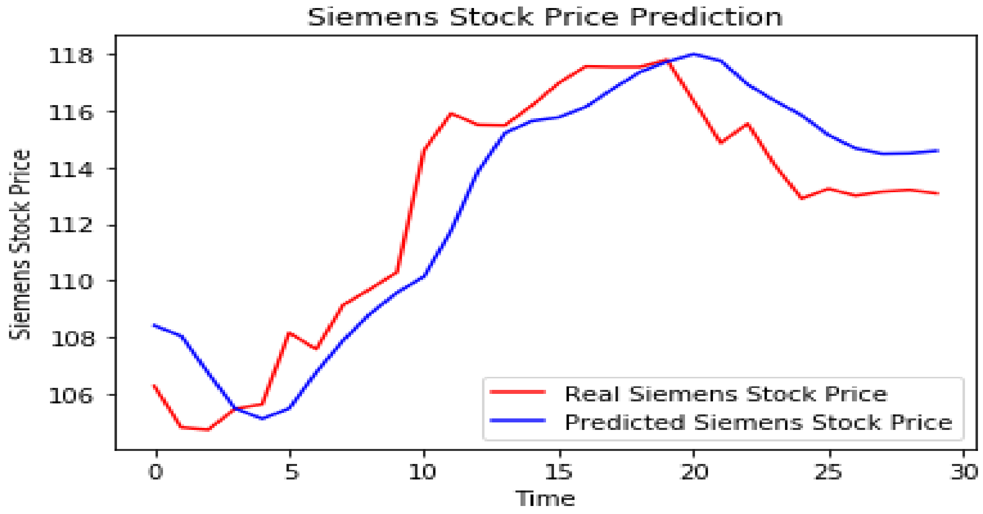
The last part of the training



```
Epoch 94/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0020
Epoch 95/100
1218/1218 [==============================] - 5s 4ms/step - loss: 0.0019
Epoch 96/100

1218/1218 [==============================] - 4s 3ms/step - loss: 0.0020
Epoch 97/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0020
Epoch 98/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0019
Epoch 99/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0020
Epoch 100/100
1218/1218 [==============================] - 4s 3ms/step - loss: 0.0019
```

These are the results. Our line, is smoother than the real, and of course it cannot predict sudden changes like for instance day 6 or day 19. Still, it depicts quite well the "trend" of the stock price.


Siemens Stock Price Prediction
Siemens Stock Price
Time
Real Siemens Stock Price
Predicted Siemens Stock Price