Stock Market Prediction Using Neural Networks

Machine learning for finance FIN-418

Author

Mahmoud Sellami

Professor:

Damien Edouard Ackerer

Abstract

A blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts, said Burton Malkies, author of ”A random walk down wall street”.

However the recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic.

This project focuses on use of financial indicators and of Recurrent neural networks and their different variants to predict future direction of stocks

Table of contents:

1) Introduction:

A correct prediction of stocks can lead to huge profits for the seller and the broker.. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the accuracy. Introduction of machine learning to the area of stock prediction has appealed to many researches because of its efficient and accurate measurements . The vital part of machine learning is the dataset used. The dataset should be as concrete as possible because a little change in the data can perpetuate massive changes in the outcome . The goal of this project is to predict the direction of the stocks in the next few days either going up or going down.

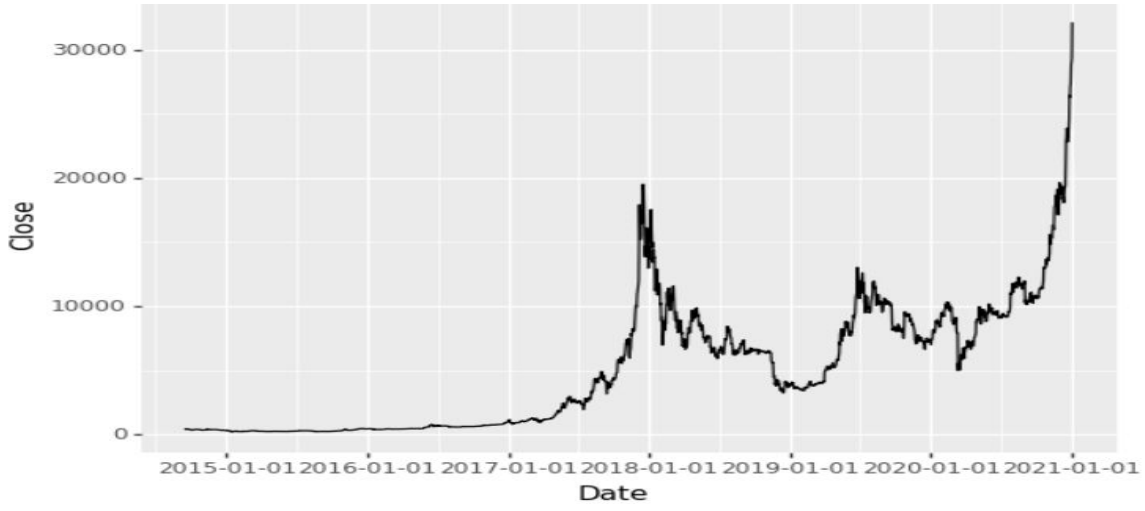
2) Data preprocessing

2.1) Data Collecting

In this project, supervised machine learning is employed on a dataset obtained from Yahoo Finance. I chose BitCoin's stock since these days it has been growing exponentially reaching historical record values . This dataset comprises the following five variables: open, close, low, high and volume from 01-09-2014 until the end of 2020

Open, close, low and high are different bid prices for the stock at separate times with nearly direct names.

The volume is the number of shares that passed from one owner to another

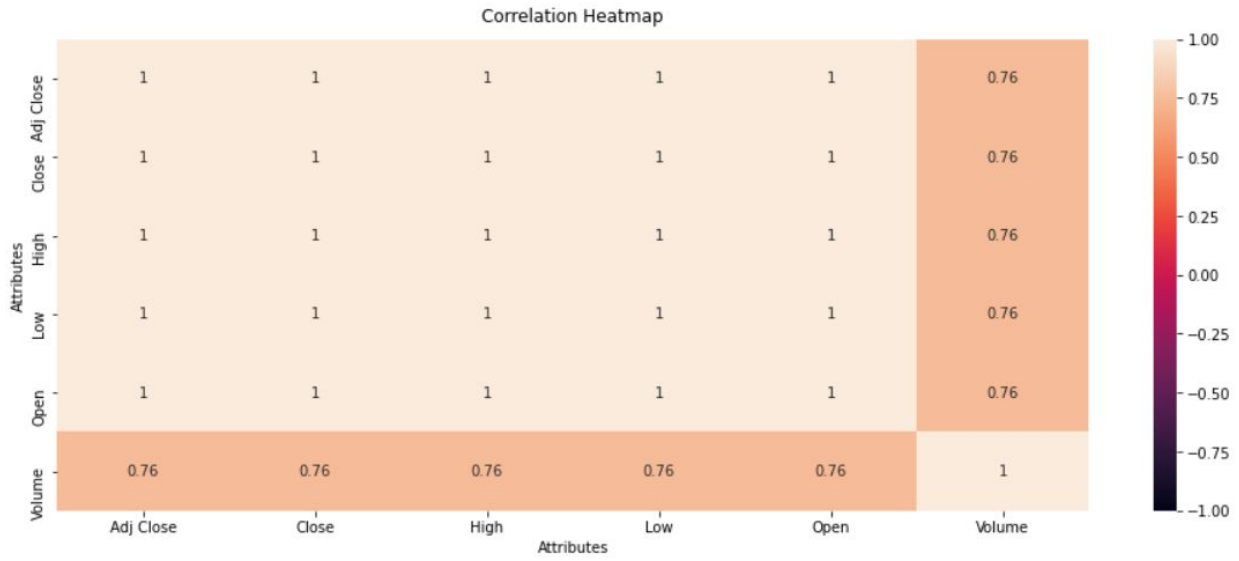


2.2) Data cleaning

In order to clean our data I had to select the attributes that can make the predicted variable more accurate or eliminate those attributes that are irrelevant and can decrease the model accuracy and quality.

In that sense, data correlation is used as a way to understand the relationship between multiple variables and attributes in the dataset

Here is the correlation matrix heatmap of the dataset:



Hence I dropped the highly correlated attributes and only kept close and volume values.

2.3) Stationary data

Financial data is usually non-stationnary (this is the case here too).

Many research papers highlight the fact that stationnary datasets improve the accuracy of predictive models(1)

We perform the Augmented Dicky-Fuller stationnarity test on the differentiated Close and Volume columns.

If p-value > 0.05: we fail to reject the null hypothesis (H0) so the data has a unit root and is non-stationary.

If p-value <= 0.05: we reject the null hypothesis (H0) so the data does not have a unit root and is stationary.

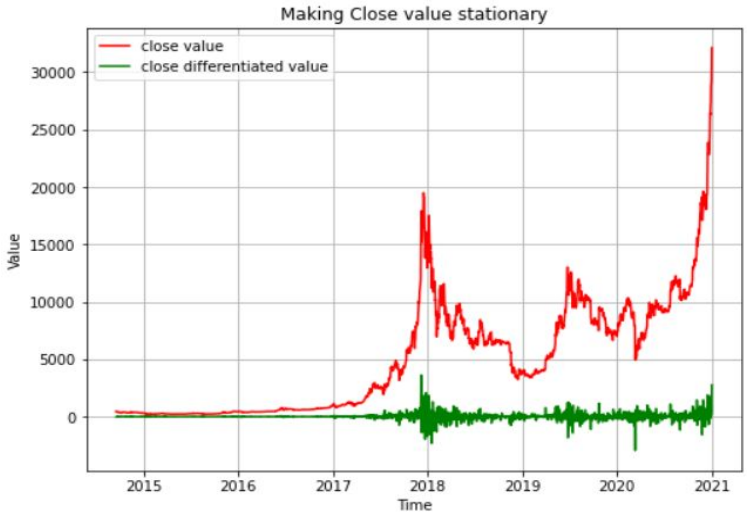
	Close	Volume
ADF statistic	2.008150	0.035457
p-value	0.998683	0.961368
critical values	1%: -3.433 5%: -2.863 10%: -2.567	1%: -3.433 5%: -2.863 10%: -2.567

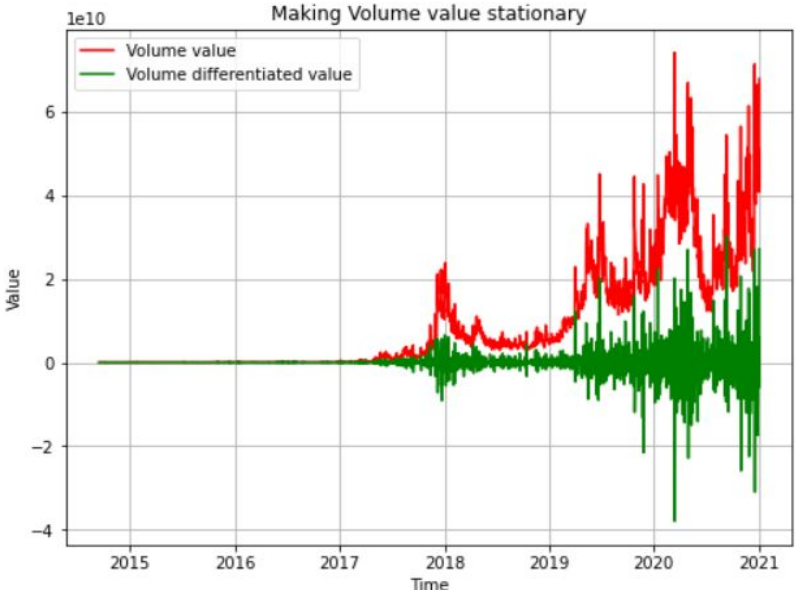
From the table It is clear that the Close and Volume are non-stationnary.

Therefore, we will compute their first order derivatives in order to have two stationnary columns and we get the following statistics:

	Close_diff	Volume_diff
ADF statistic	2.008150	0.035457
p-value	0.998683	0.961368
critical values	1%: -3.433 5%: -2.863 10%: -2.567	1%: -3.433 5%: -2.863 10%: -2.567

Here are plots to compare the stationarity of the Volume and Close times series respectively:





2.4) Denoising Data

Financial data often contains a lot of noise, which makes it harder for predictive models to extract any useful and viable information from this kind of data.

I will be using the wavelet theory in order to denoise the Close and Volume values

What is a wavelet ?

A wavelet is a function that has mean 0

The wavelets are generally grouped by families created with a mother wavelet and its images obtained using affine transformations (translation , dilation , contraction).

Wavelet analysis is a tool for simultaneously characterizing the components time and frequency of a signal, unlike the Fourier representation.

In fact, Wavelets complement Fourier theory to allow multi-resolution analysis (time and frequency) of signals.

What is a signal's wavelet decomposition ?

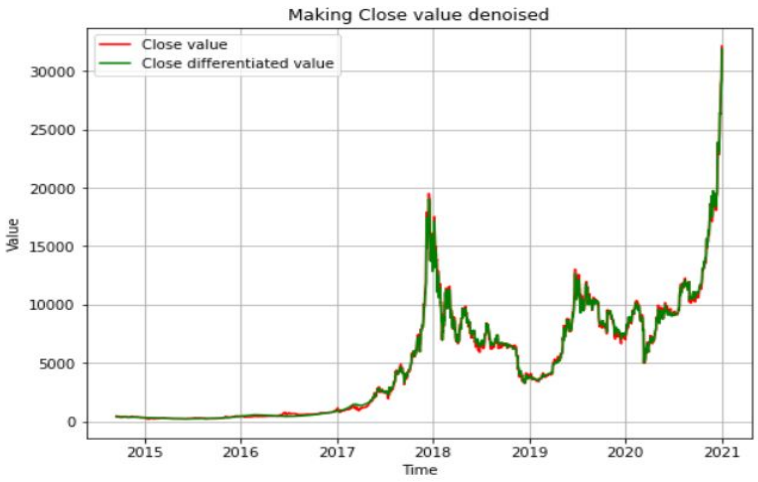
Decomposing a signal S into wavelets consists of calculating the set of its dot products with a family of wavelets. The numbers obtained are called wavelet coefficients and the operation associating its wavelet coefficients with a functions is caller a wavelet transformation

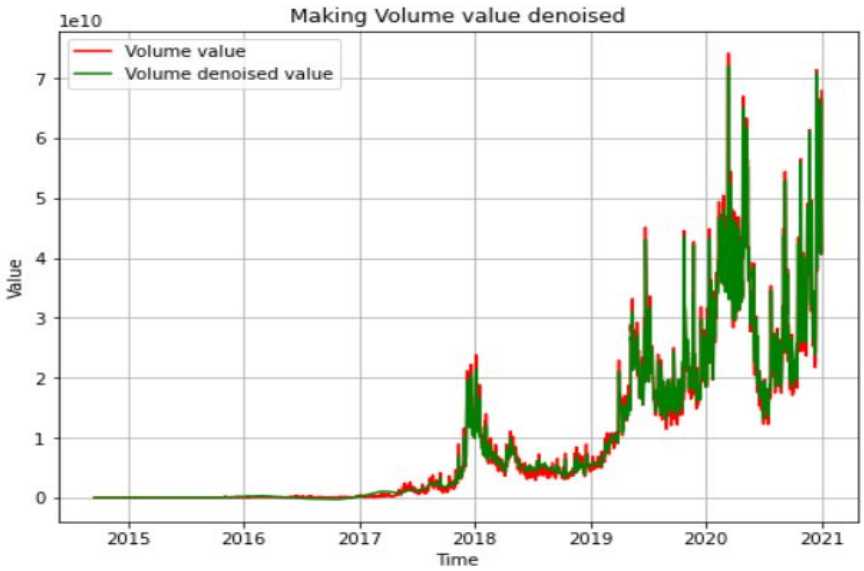
How to denoise a signal using wavelets ?

When decomposing a signal S into wavelets, we can eliminate the associated wavelets that have very low wavelet coefficients or lower than a certain threshold s to then reconstruct a denoised signal.

The threshold is set empirically depending on the wavelet family used or visually by observing the spectrogram of the wavelet decomposition of the signal

We use the pywt library to denoise our data. We reach a maximum level of decomposition of 8 for both 'Volume' and 'Close'





2.5) Financial and statistical features:

Since I want to predict future stock values, it would be wise to take advantage of the tools used by trades to analyze the movements of a price for decision making. Thus I have added several technical indicators (TA library), using (open, close , high , low, volume) to compute them. Here are some examples of them:

volatility_atr : The indicator provide an indication of the degree of price volatility. Strong moves, in either direction, are often accompanied by large ranges, or large True Ranges.

volatility_kch : Showing a simple moving average line (high) of typical price.

volume_adi : Accumulation/Distribution Index (ADI) , Acting as leading indicator of price movements.

The rest of the features can be found in the description of TA (2).

Statistical features:

In the case of times series, several statistical properties can be calculated and exploited in order to analyze the signals in which I am interested. To do this I use Python's tsfresh

library to extract a multidtude of variables characterizing the prices and volumes of the stock. Tsfresh library provides three types of variables:

descriptives statistics: minimum, maximum, quantiles, median, mean, variance etc..

graphical properties: peaks, acquity coefficient, asymmetry coefficient, passing times..

spectral analysis: autocorrelation, partial autocorrelation, fourrier coefficients, Augmented Dicky-Fuller test ..

2.6) Feature selection

Due to the high number of features (5 from original data , 22 financial features , 110 statistical features), I need to do a dimension reduction . So I use SelectKbest from sklearn to select the best K features to use for the machine learning part. K is a hyperparameter that I tuned and found to be 10.

I also noticed that no matter what K I choose none of the statistical features makes it to the top-10 features. This is also good since it takes a lot of time(3 hours) to compute them. Moreover, I noticed that using statistical features makes my models' predictions worse ! So I decided not to use them . Here are the features left:

volume_adi	volatility_atr	volatility_bbhi	volatility_bbli	volatility_kcl	trend_visual_ichimoku_b	trend_psar_up_indicator	trend_psar_down_indicator	others_cr	Close_denoised
-2.454020e+07	0.0	0.0	0.0	415.635005	4791.162412	0.0	0.0	-7.192558	391.365262
-4.448391e+07	0.0	0.0	0.0	396.784899	4791.162412	0.0	1.0	-13.674475	391.169544
-3.937687e+07	0.0	0.0	0.0	391.075673	4791.162412	0.0	0.0	-10.589639	390.870224
-5.037763e+07	0.0	0.0	0.0	389.306740	4791.162412	0.0	0.0	-12.794369	390.516734
-4.974146e+07	0.0	0.0	0.0	389.802287	4791.162412	0.0	0.0	-12.066018	390.133779

3) Prediction using Machine Learning:

3.1) Preparing data and splitting it:

For this part I used the prepare_data function provided in week 7 with the following parameters: prepare_data(num_days=100, step=5, horizon=5, test_train_ratio=0.8).

Only num_days was tuned and hence I ended up with 100.

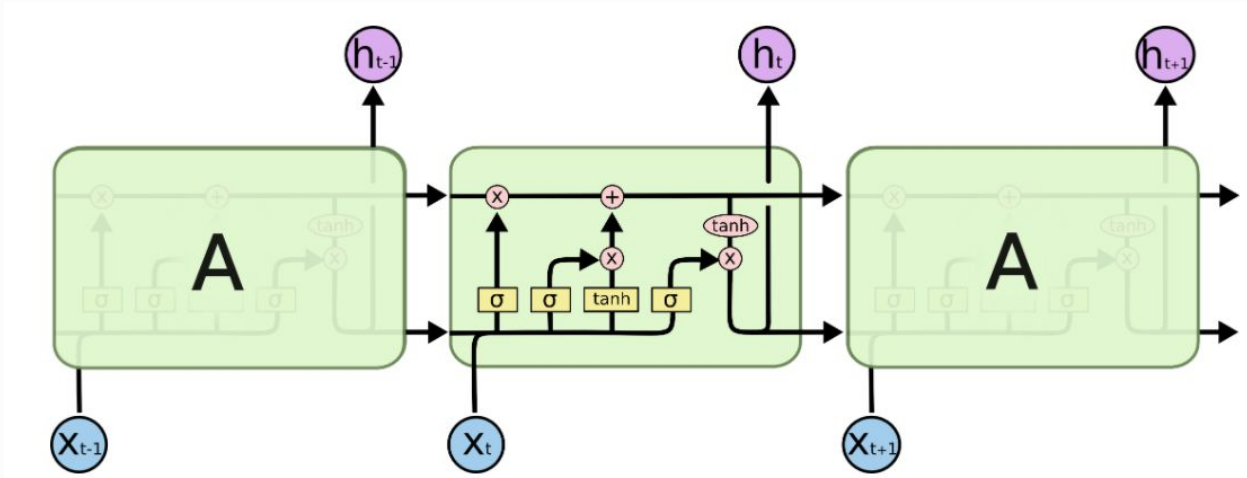
Finally with this function I have x_train with the shape (n_samples, 100, 10) y_train with the shape (n_samples, 2) . The target is binary; it takes 1 if the data after the horizon is greater than the actual data. It takes 0 otherwise

3.2) LSTM Model:

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. (3)



The repeating module in an LSTM contains four interacting layers.

3.3) Models comparison :

I compared two mdoels LSTM and CNN

Summary of LSTM model:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           12200
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 2)                 102
=================================================================
Total params: 52,702
Trainable params: 52,702
Non-trainable params: 0
```

Summary of CNN model:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 100, 10)           0
_________________________________________________________________
conv1d (Conv1D)              (None, 100, 8)            1280
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 100, 8)            1032
_________________________________________________________________
lambda (Lambda)              (None, 8)                 0
_________________________________________________________________
flatten (Flatten)            (None, 8)                 0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 18
=================================================================
Total params: 2,330
Trainable params: 2,330
Non-trainable params: 0
_________________________________________________________________
None
```

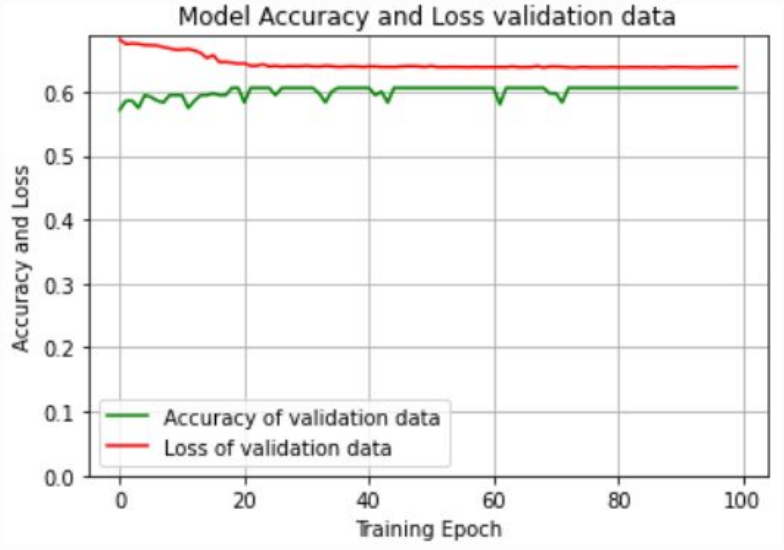
Since I am dealing with a binary classification (Up/ Down) I used “categorical-crossentropy” as a loss function for the training.

Adam was also the fastest optimizer to converge, in fact for SGD for reasonable learning rate the model does not converge.

Training results

CNN model:

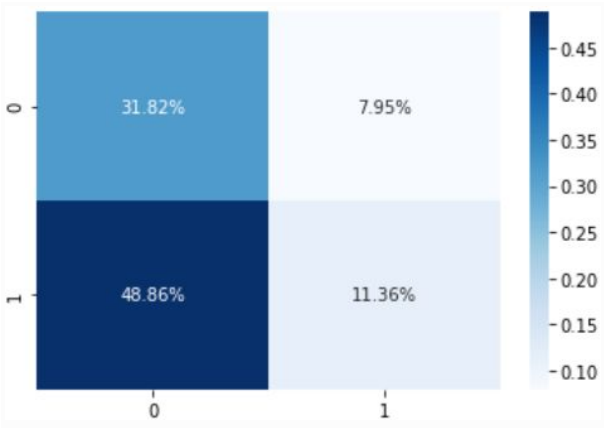
LSTM model:



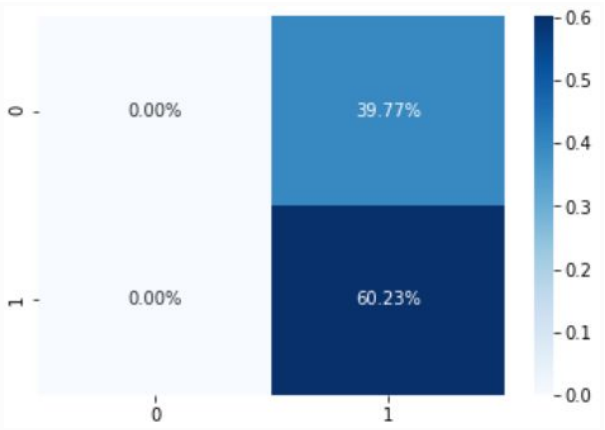
Since this is a categorical classification, the use of confusion matrix and f score is better for the assessment of the models:

Here I presented my best LSTM model after trying different architectures and playing with the features

CNN confusion matrix



LSTM confusion matrix



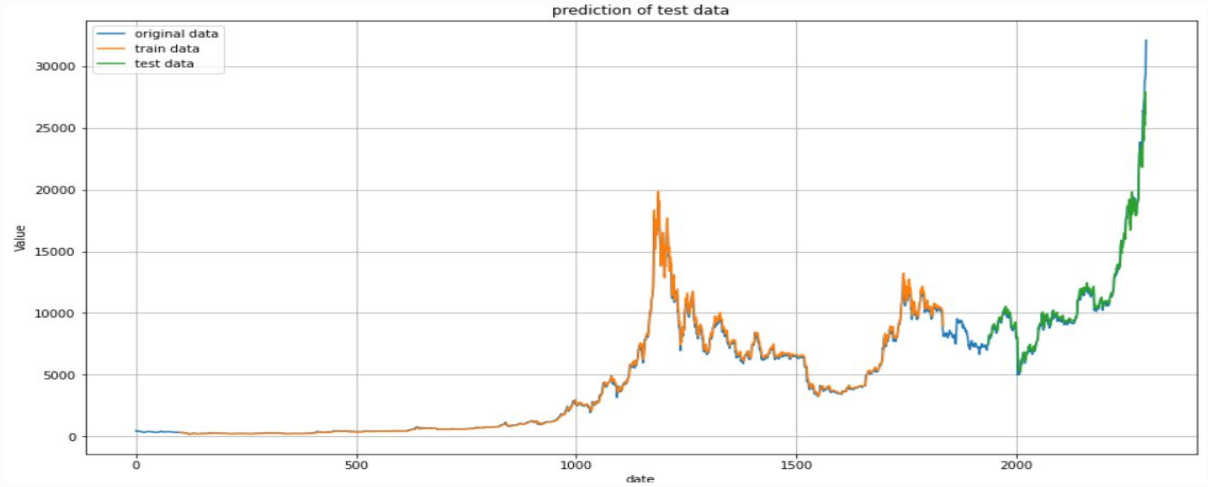
We compute the f1 score which is the harmonic mean of the precision and recall

	CNN	LSTM

F1-score	0.29	0.75

As we can see the lstm model performed a lot better than the cnn model in terms of F1-score. In fact CNN was used as a benchmark.

However, When I try to predict tomorrow's value instead of the direction(up/down) I managed to have quite an impressive model LSTM with similar architecture



4) Conclusion

Two models have been utilized in this project: LSTM and CNN, on the Yahoo finance dataset for BitCoin stock. LSTM was more effective for the accuracy of the predictions thereby yielding positive results. The use of financial features helped our model perform better as well. It has led to the conclusion that it is possible to predict stock markets with more accuracy and efficiency using machine learning techniques. However some improvements could be added by using attention models with giving more importance to certain past times instead of using all the 100 days for example.

5) References:

1:
https://www.sciencedirect.com/science/article/pii/S0925231219304461?casa_token=ims7tz1Rc8cAAAAA:pBZbRvSr6bDdtPay40_X3R6fNH8IqlTUsfABVZms9jT4ZglqDgTm3hjb65rJUDOAagOSsJQwMOE

2:
https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#volatility-indicators

3-https://colah.github.io/posts/2015-08-Understanding-LSTMs/