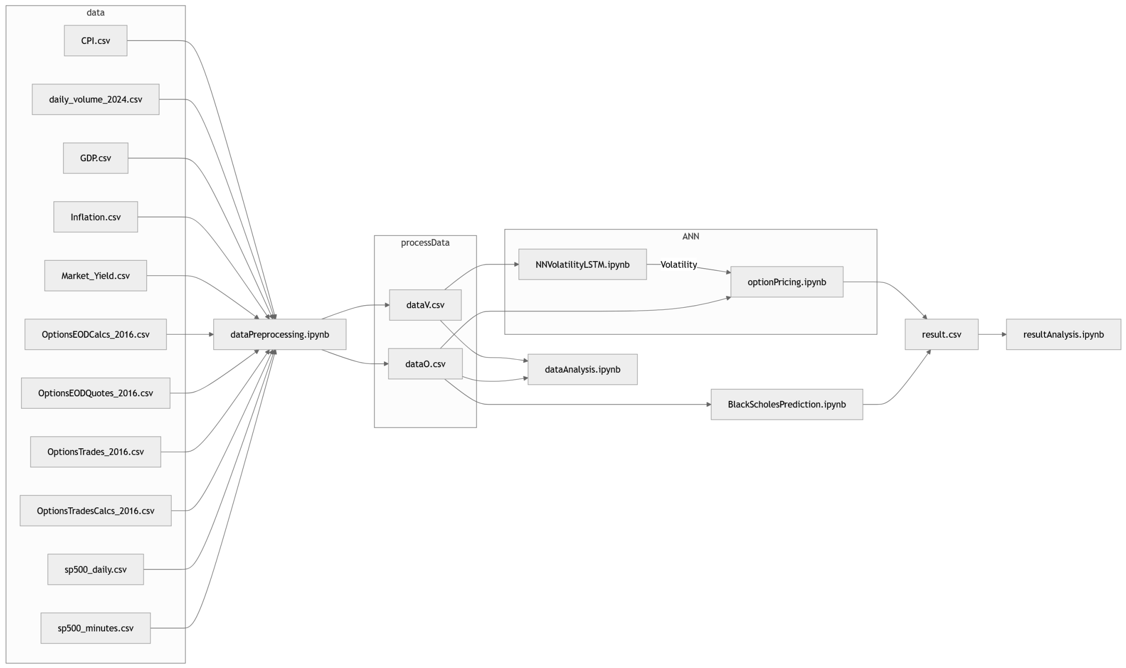
Research Project Report

June 30, 2025

1 Introduction

This document will clearly outline the advancement of the research project. Based on the Scrum and sprint methodology, I will update the document every week, including what is new and what is next.

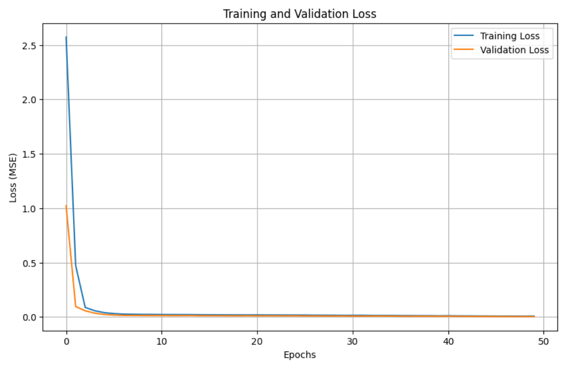


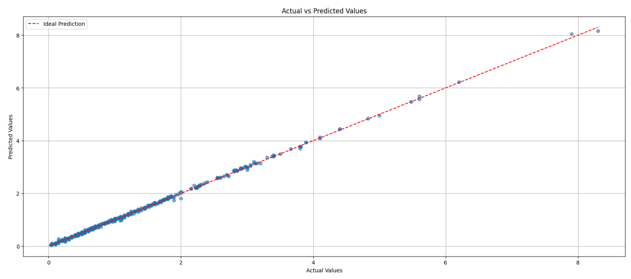
Iteration 1

February 3, 2025

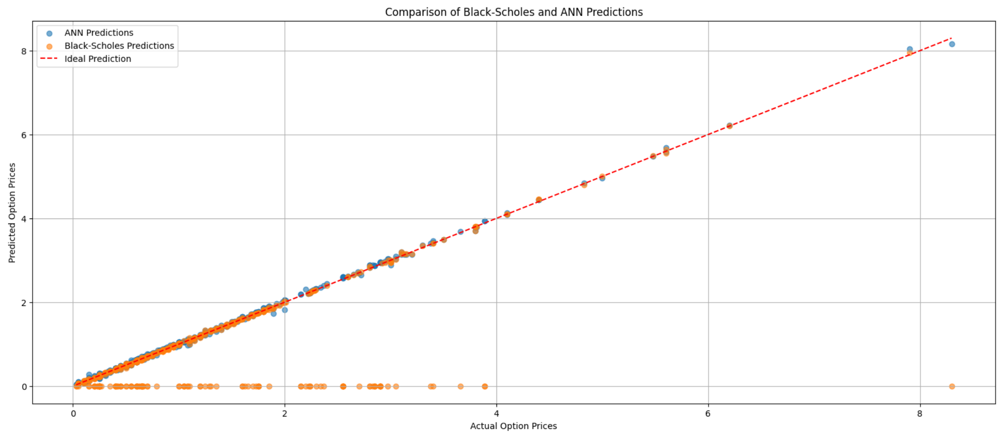
What Is New?

- Created and trained a ANN MLP model for option pricing, using Black-Scholes parameters to target option prices.





- Compared the model's performance against Black-Scholes models:



- Started to build a custom LSTM model with NumPy. For now, I think Python allows better flexibility and development time than C++, while still maintaining decent performance using only NumPy. I want the model to be compatible with TensorFlow formatting for easier use.

What Is Next?

- Finish the custom MLP model.
- Outliers suppression

Iteration 2

February 10, 2025

What Is New?

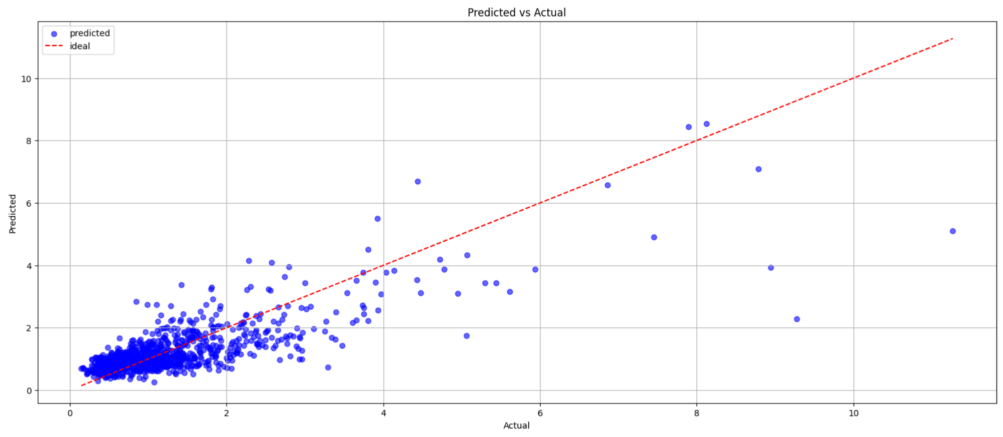
- First principle implementation of artificail neural network **multilayer perceptron**. Can be found here : /code_/models/annModels.py

-
  ```
  mlp = am.MLP(n_input=22, n_hidden1=64, n_hidden2=32, n_output=1)
  epochs = 5000
  learning_rate = 0.001

  #Training
  history = mlp.train(X_train_normalized, y_train, epochs, learning_rate)

  # Predict
  train_preds = mlp.forward(X_train_normalized)
  y_pred = mlp.forward(X_test_normalized)

  #---
  Final Training Loss: 0.41194406219492513
  Final Test Loss: 0.41460153925356924
  ```



What Is Next?

- Paramater optimization for custom model implementation ?
- Would a Transformer work better ? - Wiki Transformer
  - Very likely, However, to get a working transformer model, the data volume is much more advanced than we currently use.

Iteration 3

February 17, 2025

What Is New?

- Finnish data gathering with scipt :/code_/tools/getData.ipynb, all the assets data are gather in /data/stocks (around 200 symbols)
- Transformer implementation in progress
- Benchmark against LSTM model
- Paramater optimization for custom model implementation ?

What Is Next?

- Document on maths behind models (models.pdf)

Iteration 4

February 24, 2025

What Is New?

- LSTM implementaiton in progress
- FFNN MLP and LSTM mathematics models

What Is Next?

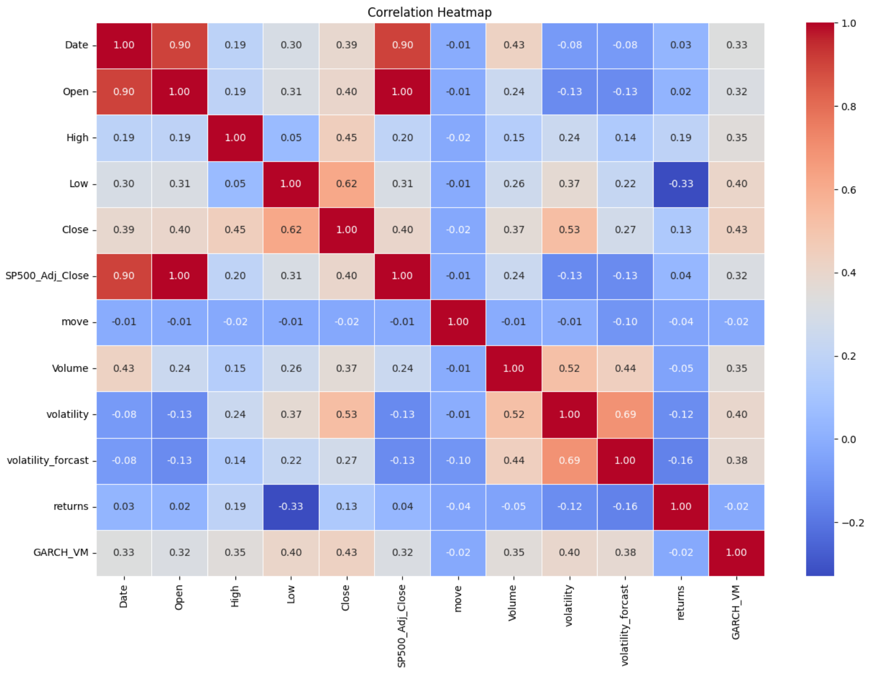
- Identifies specific aspect of volatility time series (mean reversion, volatility clustering, heavy tail)
- identifies drawback in LSTM architecture for specific financial time series
- Optimize model for financial time series
- identify best loss function for volatility time series

Iteration 5

March 3, 2025

What Is New?

- Data Work
  - Compare to litterature
  - Normalize data to improve models performances
  - Select relevant feature to work with the model (clean confusion matrix)
- Litterature about new/modify LSTM model for financial time series prediction
- Document on volatility model updated



What Is Next?

- Improve mathematical relationship of LSTM models with litterature and financial time series properties.

Iteration 6

March 10, 2025

What Is New?

- The volatility time series have some properties as for exemple **volatility clustering**. It implied that huge amplitude volatility periode are follow by small volatility changes and back to huge periode. The default LSTM network isn't aware of that so we can try to implement this in the forget gate to keep this informaiton inside the network. For instance we can propose a solution like this

  $$F_t = \sigma \left(W_f \cdot [H_{t-1}, X_t] + b_f - \mathbf{k\sigma_t} \right)$$

  Where the new term $k\sigma_t$ represente the a contante k that is a leanring parameter to scale the impact on the network and σ_t that is the volatility estimation value.

What Is Next?

- Implementaiton
- Benchmark against default LSTM and Black-Scholes
- Litterature

Iteration 7

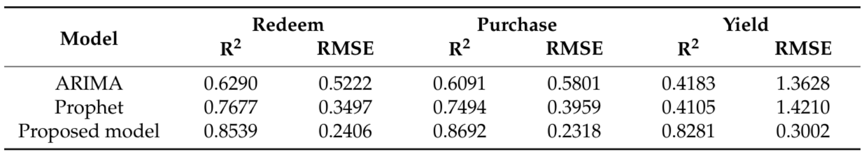
March 24, 2025

Litterature review

1. AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction
   Adding and attention layer to LSTM model. Appliyng weight to input feature thanks to an attention layer. Then, in a second stage, the attention model select all relevant features for LSTM input model.

MAPE on DJIA	
LSTM	*0.00625*
AT-LSTM	*0.00486*

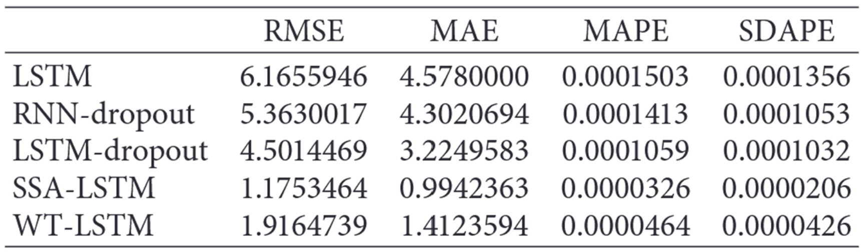
2. Improved Financial Predicting Method Based on Time Series Long Short-Term Memory Algorithm
   Automated capital prediction strategy, first by analysing the fluctuation and tail risk. Then by use ARIMA and Prophet models. Finally time series modeleing of the wavelet LSTM for a two part analysis of the linear separated wavelet and non-linear embedded wavelet to predict volatility.



Model	**Redeem**		**Purchase**		**Yield**	
	R^2	**RMSE**	**R^2**	**RMSE**	**R^2**	**RMSE**
ARIMA	0.6290	0.5222	0.6091	0.5801	0.4183	1.3628
Prophet	0.7677	0.3497	0.7494	0.3959	0.4105	1.4210
Proposed model	0.8539	0.2406	0.8692	0.2318	0.8281	0.3002

3. Prediction of Financial Time Series Based on LSTM Using Wavelet Transform and Singular Spectrum Analysis
   Imporove LSTM prediction capabilities by using data denoising methods including wavelet transformation (WT) and singular spectrum analysis (SSA) on the closing DJIA, divided in short, meduim and long term time periode. The LSTM data denoising performe better than raw data for data prediciton on all tree time periodes.

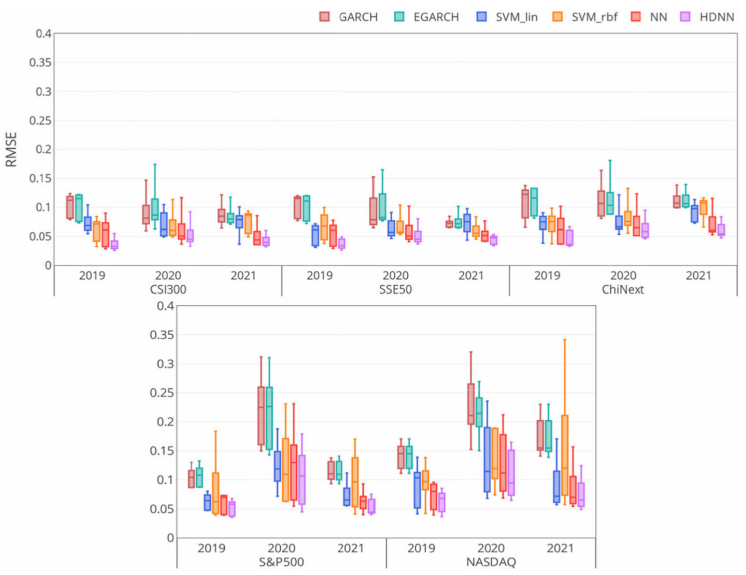
Table 5: 6-hour DJIA closing price forecast results.



	RMSE	MAE	MAPE	SDAPE
LSTM	6.1655946	4.5780000	0.0001503	0.0001356
RNN-dropout	5.3630017	4.3020694	0.0001413	0.0001053
LSTM-dropout	4.5014469	3.2249583	0.0001059	0.0001032
SSA-LSTM	1.1753464	0.9942363	0.0000326	0.0000206
WT-LSTM	1.9164739	1.4123594	0.0000464	0.0000426

4. Black-Scholes-Artificial Neural Network : A novel option princing model
   Comparaison of multiple option pricing model and intruduction of a new model call BSANN, a basic ANN MLP model in [11-15-1] performing better than tranditionnal methodes.

5. Volatility forcasting using deep neural network with time-series feature embedding
Propose a hybrid deep neural network model (HDNN). Encoding one-dimensionnal time-series data into two-dimensionnal GAF images to use a CNN with 2D concolutions layers, then performe feature embeding and dense layers regression to predict the volatility



6. Volatility forcasting using deep recurrent neural networks as GARCH models
Propose new method to predict volatility time series by using a combination of GARCH and and deep neural network. Also introduce a mehanisme to identifiy ideal sliding windows side for volatilty. With evaluation of GRU, LSTM, BiLSTM

Table 6 Performance results for the Volatility prediction of the ASX200 Time Series using Recurrent Neural Networks

	Train dataset			Test dataset		
	ALL-GARCH(1,1) with			ALL- GARCH(1,1) with		
	BILSTM	LSTM	GRU	BILSTM	LSTM	GRU
RMSE	**0.1205**	0.2979	0.3391	**0.2106**	0.2226	0.2594
MAE	**0.0934**	0.1952	0.2163	**0.1382**	0.1737	0.2078
MAPE(%)	**7.7217**	9.49156	10.3561	**8.5580**	10.9473	13.1652
R^2	**0.9725**	0.8321	0.7825	**0.4968**	0.4380	0.2368
Spearman	**0.9570**	0.8788	0.8711	**0.7785**	0.6719	0.6788

7. Machine Learning for Options Pricing: Predicting Volatility and Optimizing Strategies – Explore how ML models can outperform traditional pricing models (like Black-Scholes), enhancing option traders' decision-making.

8. NEURAL NETWORK LEARNING OF BLACK-SCHOLES EQUATION FOR OPTION PRICING

9. Option Pricing with Deep Learning
This paper propose a deep learning approach to option pricing with 3 models, 2 MLP(1&2)

and a LSTM model. MPL1 as a MLP predicting the option price, while MLP2 predicting the bid & ask of the underlying price. Furthermore, LSTM model extimating volatility to feed its outpur to the MLP1 and then having a prediction of the option price.

	Model	train-MSE	MSE	Bias	AAPE	MAPE	PE5	PE10	PE20
Call	BS	322.95	321.37	-0.05	78.79	4.81	50.52	59.33	67.43
	MLP1	23.71	24.00	0.01	24.49	2.12	61.04	68.39	74.33
	MLP2	7.70	15.21	0.09	23.45	1.73	63.03	70.10	75.54
	LSTM	30.61	30.97	0.13	26.58	2.33	58.94	66.35	72.42
Put	BS	543.48	533.25	97.37	68.00	97.46	12.87	18.22	23.58
	MLP1	15.65	15.66	5.03	43.73	18.48	30.46	40.51	51.13
	MLP2	2.03	8.84	3.85	39.59	14.32	33.74	44.25	55.01
	LSTM	22.81	23.15	6.01	48.32	26.05	27.45	36.24	46.17

Table 1: Error metrics comparing MLP1 price and MLP2 equilibrium price with Black-Scholes prices. Note all metrics beside MSE are percentages.

10. Volatility forecast using hybrid Neural Network models

Iteration 8

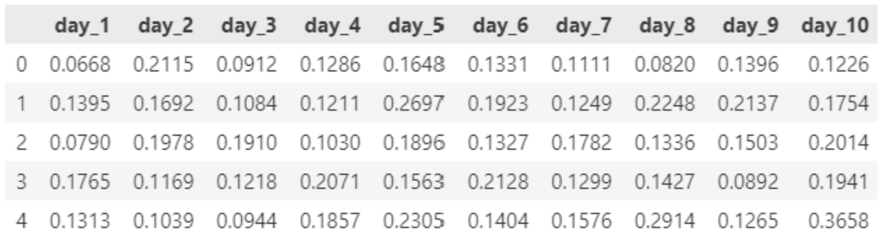
March 31, 2025

LSTM implementation

Model architecture : [8-16-1]

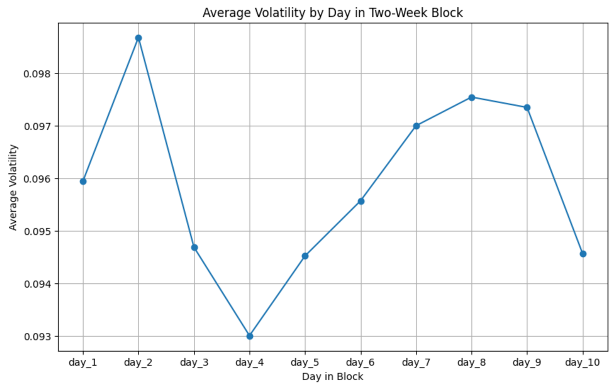
Data work

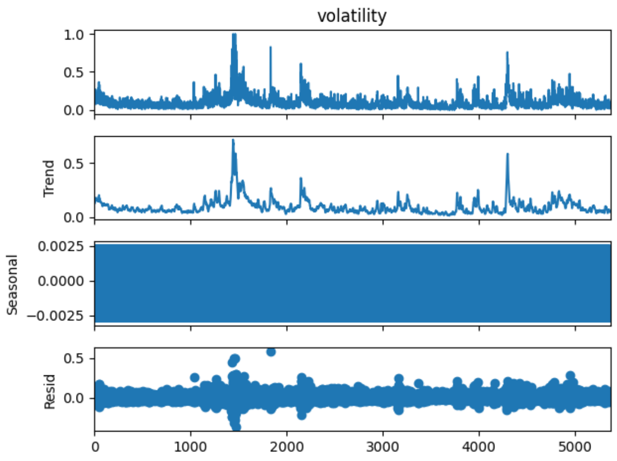
- **Re-sizing of the data** - Data work
  In the file dataResize.csv the size of the data have been resize to 2 weeks per exemple.



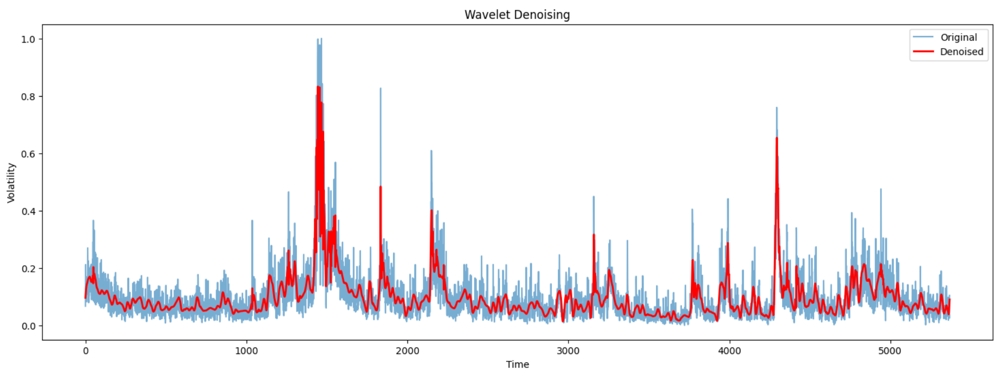
	day_1	day_2	day_3	day_4	day_5	day_6	day_7	day_8	day_9	day_10
0	0.0668	0.2115	0.0912	0.1286	0.1648	0.1331	0.1111	0.0820	0.1396	0.1226
1	0.1395	0.1692	0.1084	0.1211	0.2697	0.1923	0.1249	0.2248	0.2137	0.1754
2	0.0790	0.1978	0.1910	0.1030	0.1896	0.1327	0.1782	0.1336	0.1503	0.2014
3	0.1765	0.1169	0.1218	0.2071	0.1563	0.2128	0.1299	0.1427	0.0892	0.1941
4	0.1313	0.1039	0.0944	0.1857	0.2305	0.1404	0.1576	0.2914	0.1265	0.3658

- **Data seasonality** - Analysis
  Analysis reccurent pattern in dataResize to use the 10-days seasonnality.





- **De-noising (Wavelet)** - Data work
  De-noising the raw data to better capture the trend in the time serie



- **Sliding windows** - Analysis
  Papers : Volatility forcasting using deep recurrent neural networks as GARCH models and Single-scale time-dependent window-sizes in sliding-window dynamic funcitonal connectivity analysis

Statistical models

- ARIMA
- GRU
- BiLSTM

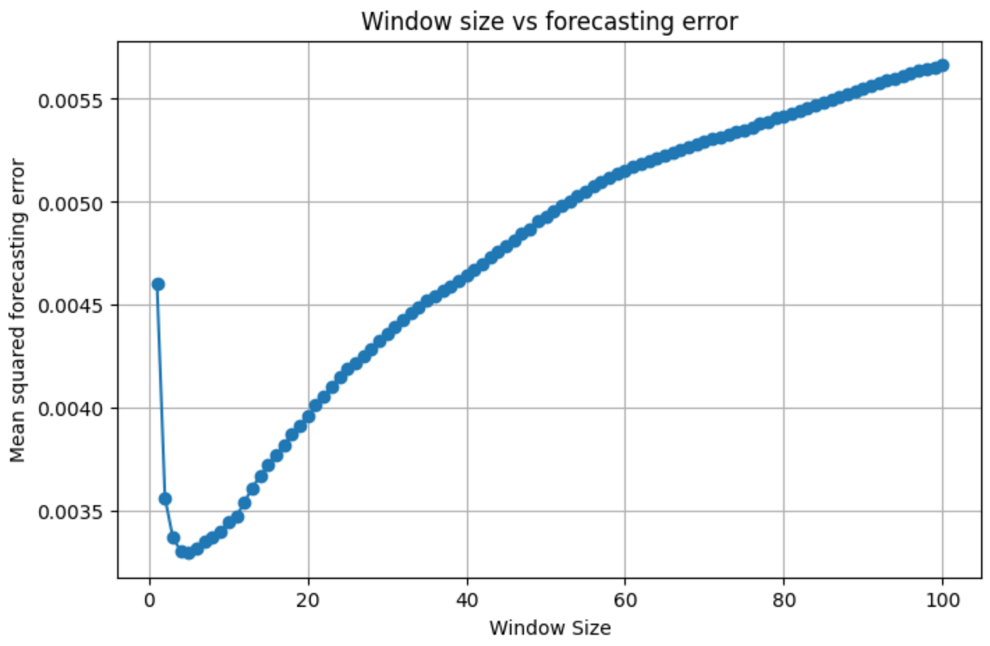
Iteration 9

April 7, 2025

Sliding window

Implementation

Implementation of static sliding windows on the volatility time serie



Best window size given by the EMD is currently 5 days (a trading week) with MSE estimator.

Improvement - Dynamic sliding window

Iteration 10

April 14, 2025

LSTM

Need to improve LSTM performance, not as good as TensorFlow implementation.

Sliding Window

The dynamic sliding vector may cause a problem with the LSTM vectore size.

Iteration 11

April 21, 2025

From the paper : Volatility forcasting using deep recurrent neural networks as GARCH models

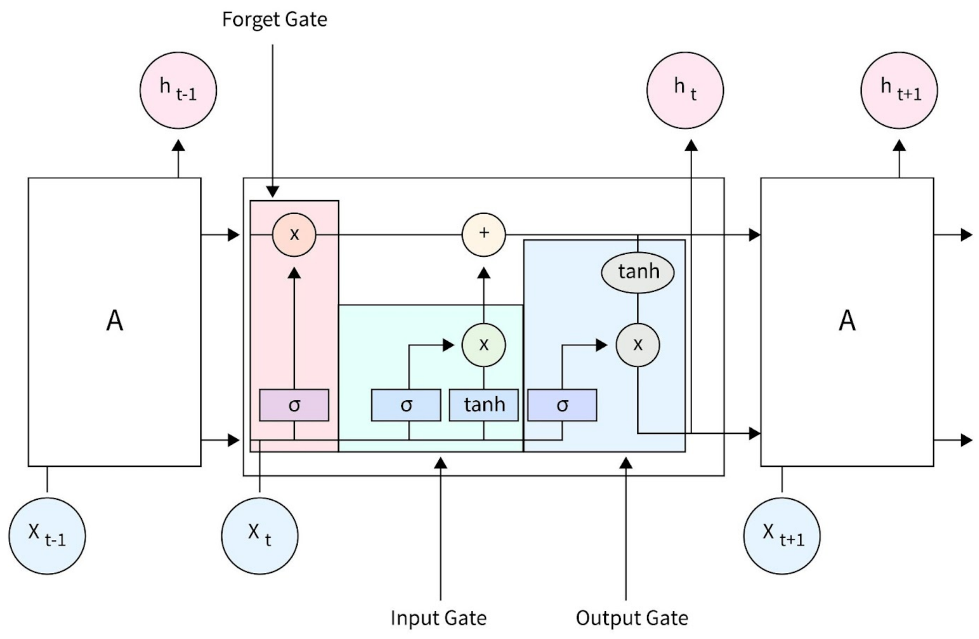
1. Decompose the series (returns R_t and volatilities V_t) into K intrinsic mode functions (IMFs) via EMD:
$$R_t \xrightarrow{\text{EMD}} \{c_i^R(t)\}_{i=1}^K, \qquad V_t \xrightarrow{\text{EMD}} \{c_i^V(t)\}_{i=1}^K$$
2. Hilbert-transform each IMF to get its instantaneous phase $\phi_i(t)$, then frequency
$$f_i(t) = \frac{1}{2\pi} \frac{\mathrm{d}}{\mathrm{d}t} \phi_i(t),$$
and thus instantaneous period
$$p_i(t) = \frac{1}{f_i(t)}$$
3. Energy-weight, by computing each IMF's average energy over the sample:
$$E_i = \frac{1}{T} \sum_{t=1}^{T} \big[c_i(t)\big]^2$$
4. Weighted average period at time t:
$$p^R(t) = \frac{1}{\sum_{i=1}^K E_i^R \sum_{i=1}^K E_i^R \, p_i^R(t)}, \qquad p^V(t) = \frac{1}{\sum_{i=1}^K E_i^V \sum_{i=1}^K E_i^V \, p_i^V(t)}$$
5. Combine returns and vol by taking the max (as the paper does) and admit τ as your ideal window size :
$$\tau(t) \;=\; \max\big\{\, p^R(t),\, p^V(t)\big\}$$

Another approach could be to

1. Compute the energie based on instantaneous amplitudes from each IMF via the Hilbert transform
$$E_i = \frac{1}{T} \sum_{t=1}^{T} \big[c_i(t)\big]^2$$
2. Normalize and forme energy-based weights :
$$w_R(t) \;=\; \frac{E_R(t)}{E_R(t) + E_V(t)}, \quad w_V(t) \;=\; \frac{E_V(t)}{E_R(t) + E_V(t)}$$
3. Weighted average of periods :
$$P(t) \;=\; w_R(t)\, P^R(t) \;+\; w_V(t)\, P^V(t),$$
then round up to get an integer window length:
$$\tau(t) \;=\; \big\lceil P(t) \big\rceil.$$

However, I am concerned that this solution limits performance by providing a smaller average window size than the previous maximization function. As a result, it may miss periodic information that the original maximization approach captured effectively.

With an LSTM model, a problem arises with a changing sliding window size: the dynamic changes of the input vector. Even though we can force the vector to change its size at each time point, the long-term memory information will be retained. One solution could be to modify the forget gate to enable less information retaining when the size of the window is smaller.



Related papers :

- Sliding Window Empirical Mode Decomposition -its performance and quality
- Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation
- An attention-based multi-input LSTM with sliding window-based two-stage decomposition for wind speed forecasting

Iteration 12

April 28, 2025

Comparaison between different methods to choose ideal size of the sliding window ; default, EMD and proposed :

$Metric/Method$	$Default$	EMD	$Proposed$
MSE	0.002921	0.003081	0.003104
MAE	0.038735	0.039244	0.039849
$MAPE$	60.86	55.60	64.80
R^2	0.4224	0.4712	0.2427

No time to look the density metrics foracting idea

Iteration 13

May 12, 2025

Weight EMD model

Write the model for each IMF instead of the global static one size window.

$$E_i = \frac{1}{T}\big[c_i(t)\big]^2$$

$$P^R(t) = \frac{1}{E_i^R E_i^R\, P_i^R(t)}, \qquad P^V(t) = \frac{1}{E_i^V E_i^V\, P_i^V(t)}$$

$$\tau(t) \;=\; \max\big\{\, P^R(t),\, P^V(t)\big\}$$

Forget gate modification to enable dynamic LSTM model

On way to approach this probleme could be to change the long term memory proportionnaly to the size of the window by weighting it. For exemple a deacresing window, retaining less information further back in the past would induced a long term memory value weighted less. The weights can be calculated by the ration of the changing size.

The forget gate could change from :

$$\mathbf{F}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{H}_{t-1}, \mathbf{X}_t] + \mathbf{b}_f)$$

To :

$$\mathbf{F}_t = \sigma(\mathbf{W}_f \cdot [\frac{\mathbf{H}_{t-1}}{\frac{\mathbf{S}_{t-1}}{\mathbf{S}_t}}, \mathbf{X}_t] + \mathbf{b}_f)$$

With **S** the size of the sliding window.

Test result :

Metric/Method	*Base*	*Proposed*
MSE	0.003	0.056
MAE	0.040	0.038
$MAPE$	63.31%	48.34%
R^2	0.457	0.3703

Regarding these result, it appear the proposed method can improve accuracy for most of the point, however, looking at the MSE and $\mathbf{R^2}$ metric some few big misses are occuring. The process show improvement but need to be more stable to correctly predicte all values without big misses that dragues metrics the wrong way.

Probability density model

Using density distribution of the volatility to improve forecasting. By giving a model the caracteristics of the probability density function, (σ, μ, mediane, mode, etc...)

Iteration 14

May 19, 2025

Forget gate

Controling the information retention by replacing with zero some memory information.

Let's take the input vector :

$$(seq\_lenght, batch\_size, input\_size)$$

Where :

- seq_lenght : the size of the time serie data point fed to the model (sliding window size)
- $batch\_size$: the size of the batch for each weight and bias update
- $input\_size$: number of feature fed to the model (in our case 1 as there is only volatility time serie for now)

In order to dynamicly adjust the seq_lenght we need to set a max_seq_lenght as we can decrease the vectore size but not increase it.

For a given vectore :

$$[1; 2; 3; 4; 5]$$

Admit $max\_seq\_lenght = 3$, let's say in a time step the ideal sliding window size is 2 we set $seq\_lenght = 2$ so the vector changes to :

$$\begin{array}{cccc} 1 & 2 & 3 & \text{seq} = 3 \\ 0 & 3 & 4 & \text{seq} = 2 \\ 3 & 4 & 5 & \text{seq} = 3 \end{array}$$

Test result of the implementation :

$Metric/Method$	$Base$	New
MSE	0.003	0.003
MAE	0.040	0.049
$MAPE$	60.76%	97.64%
R^2	0.465	0.320

We also have to keep in mind that the computing time of the new methode is much higher, (around 8 minutes here against 30 sec)

Iteration 15

May 26, 2025

Index masking for LSTM model

Here the goal is to add a discreat value, either $[0;1]$ for now to allow the total forgetting of time serie value, in our case ie, the value associate with 0.
For exemple :

$$\begin{matrix} 0 & 0 & 1 & 1 \\ 3 & 4 & 5 & 6 \end{matrix}$$

In that case, 3 and 4 will be forgotten by the model, while still being in a vector of size 4.

The input tensor now is a 4d matrix in order to add this information to each data point.

$$(seq\_lenght, batch\_size, input\_size, index\_mask)$$

After implementation :

$Metric/Method$	$Base$	$Masking$
MSE	0.003	0.002
MAE	0.040	0.038
$MAPE$	60.76%	55.76%
R^2	0.465	0.462

Adding more features

List of the features added back to the volatility model :

- Present volatility
- open price
- close price
- high
- low
- volume

After implementation :

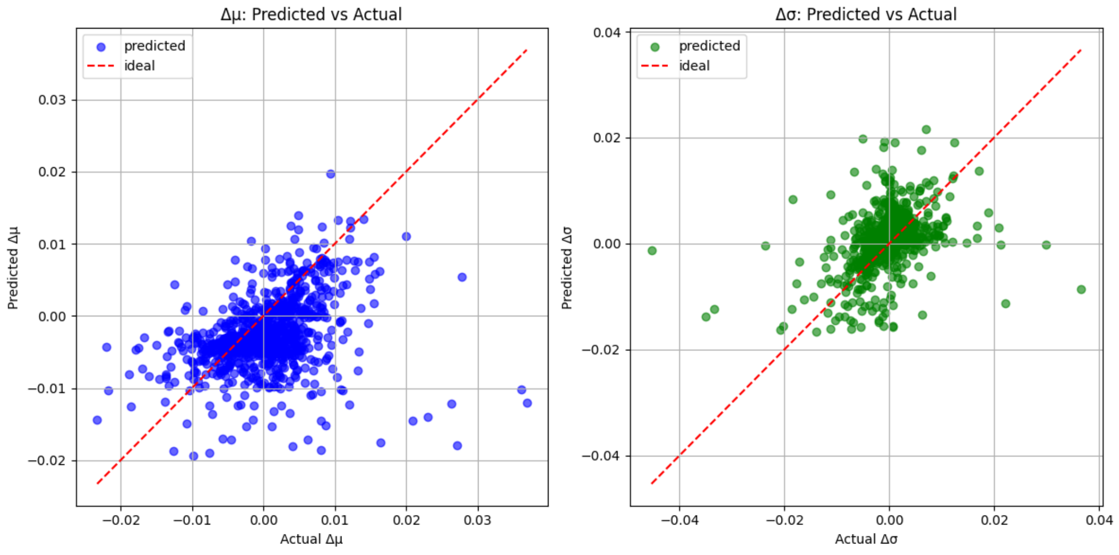
$Metric/Method$	$Base(vol)$	$Withmorefeatures$
MSE	0.003	0.010
MAE	0.040	0.079
R^2	0.322	-0.996

From these results, it appear the volatility prediction actually gets worse when adding features. This can be due to the model being to small and not capturing the complex relationship, or, the features not being the most relevant (see correlation matrix).

Probabilistic model

In addition of doing a regression on the volatility, also predict the parameters of the distribution, (μ, σ) With target set to next day (μ, σ) results gives :

$Metric/Target$	mu	$sigma$
MSE	0.00007	0.0002
MAE	0.006	0.010
R^2	0.727	0.822



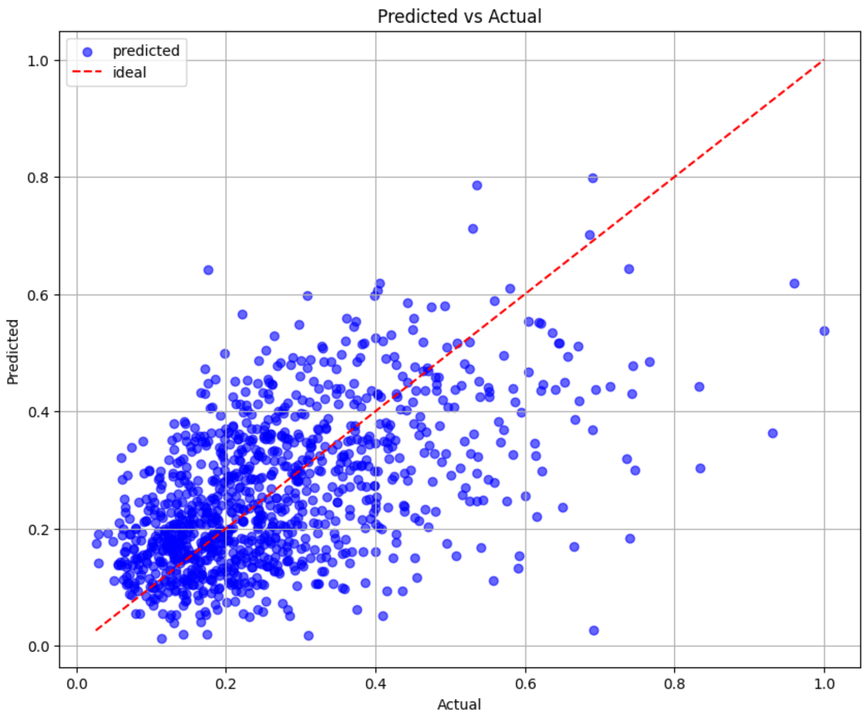
Back to option price model

Model with input :

```
df[['Open', 'volatility', 'High', 'Close', 'Volume', 'returns', 'EWMA_VM', 'yang_zhang']]
```

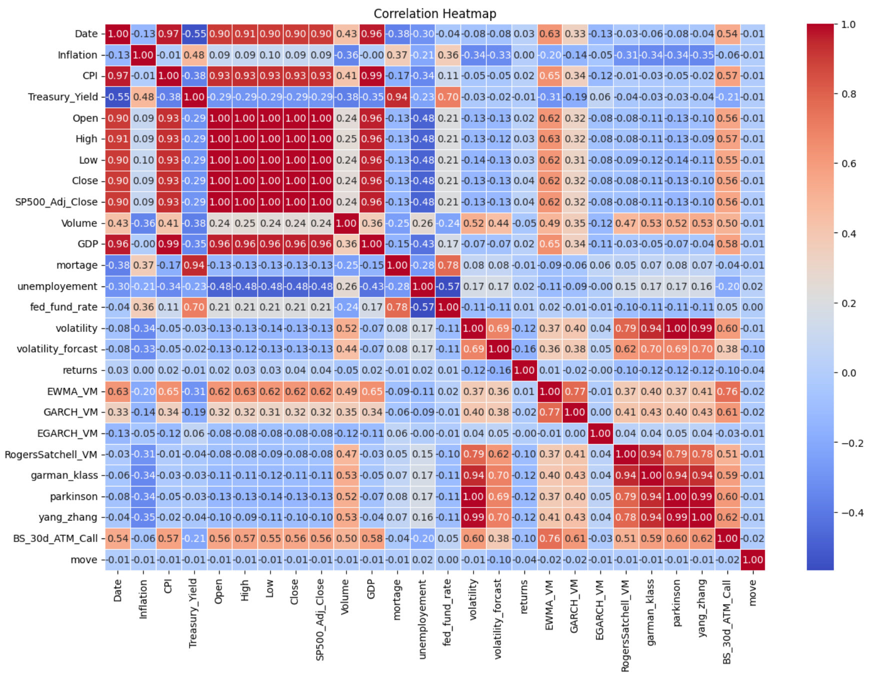
One hidden layer of size 64, no EMD, sliding window size of 20. Result :

$Metric/Method$	$Base\_option\_model$
MSE	0.017
MAE	0.101
R^2	0.182



Correlation matrix

Relationship between input features and target (option price) using the BS model. Sample of option prices can be ofund online, but for large samples, (days to week) they are pay to download



Iteration 16

June 02, 2025

Change R^2 formula

Bounding R^2 between 0 and 1 either by

- Normalizing, the issue is that it change the scale, so a 0 before can become a 0.5.
- Or, by taking $max(0.0; R^2)$ the downside is the lost of the information "how bad the model is", in our case the model should stay above 0 and if it is 0, the information lost isn't to bad since we already know its really bad.

Try a bigger model for relationship complexity

From a model of size :

$$[Input; 64; output]$$

Let's try a model of size :

$$[Input; 256; 128; output]$$

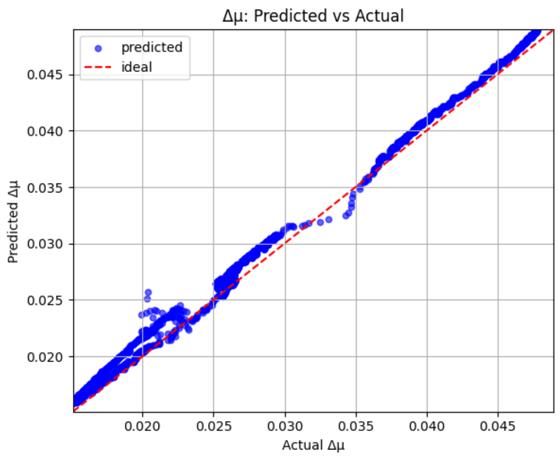
Result :

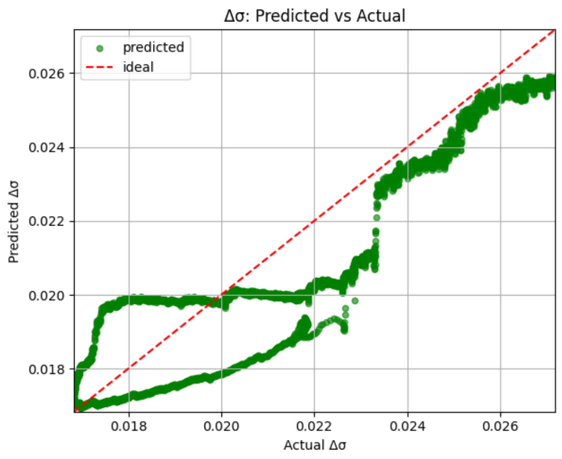
$Metric/Method$	$Base$	$Biggermodel$
MSE	0.003	0.003
MAE	0.040	0.041
R^2	0.322	0.309

Probabilistic model

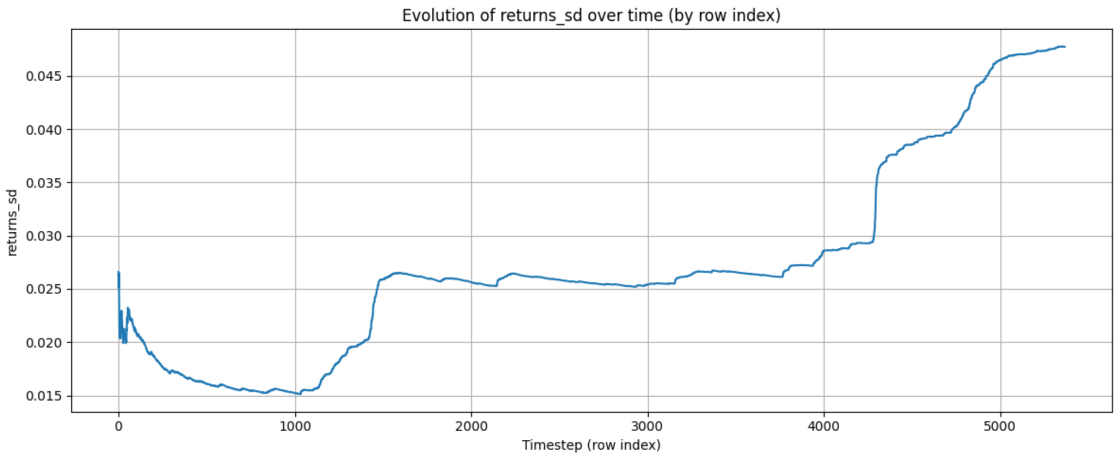
Density funciton of σ

Calculating the sd of returns, then predicting the target means of the returns's sd and sd of the returns's sd.
The result given by a [64;32] model are :

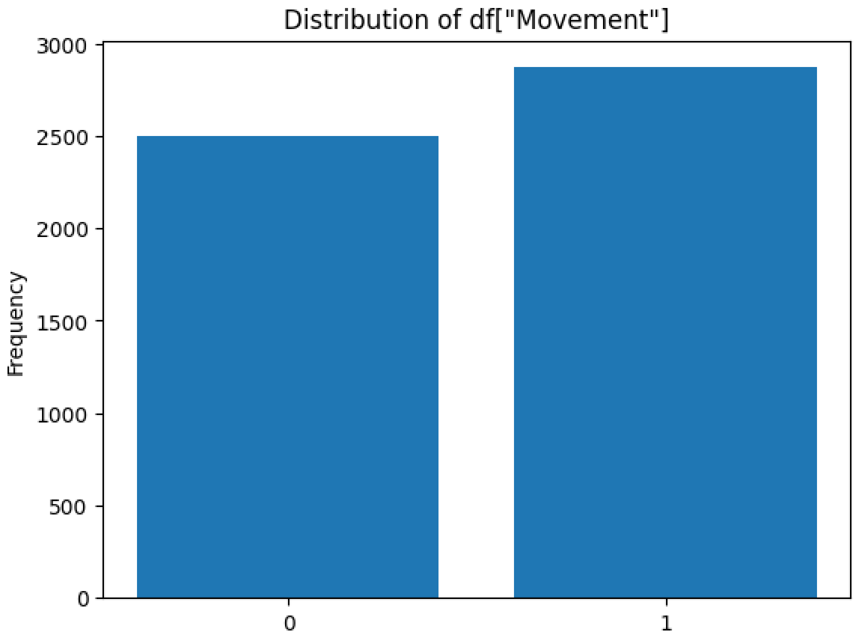




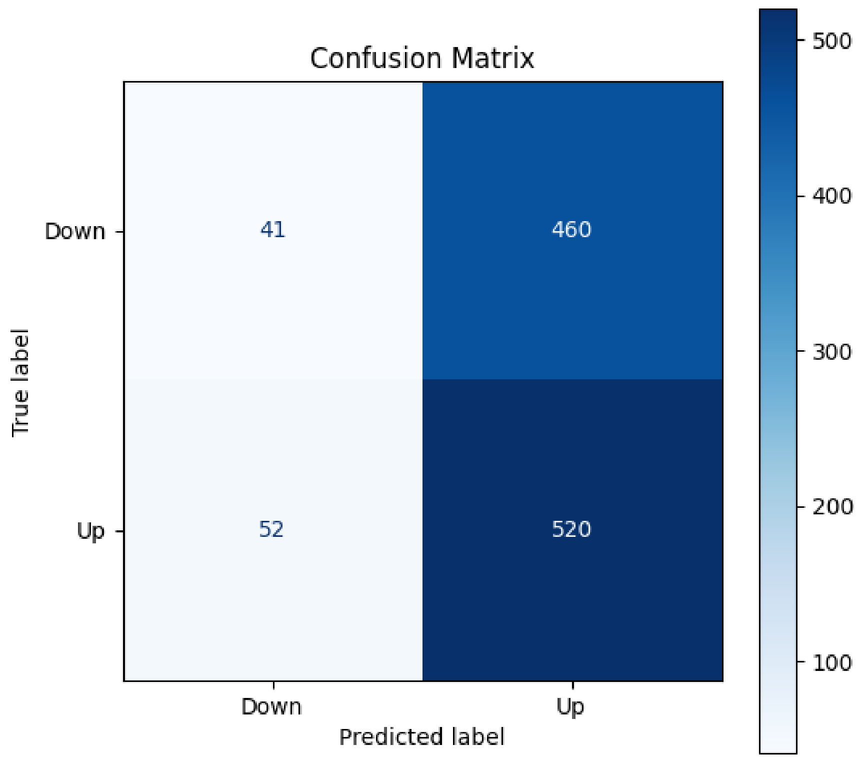
It appear the model have better result predicting the mean, wich isn't surprising, due to the more stable nature of this metric. However, for the sd, the results aren't as good as for the mean, we can observe a similarity between the returns's sd time serie and the retust given by the model, indicating harder prediction when volatility increases.



Price model with bernoulli distribution



This confusion matrix indicates the modle choose 1 or "up" a lot more than it should, This can be caused by the more important presence, of 1 in the distribution. However, this show that the model overestimate that number.



Confidence interval

Assuming a normal distribution of the returns we can calculating CI with :

$$\mathrm{CI}_{1-\alpha} = \bar{x} \ \pm \ z_{1-\frac{\alpha}{2}} \frac{s}{\sqrt{n}}$$

$$\left[\bar{x} - z_{1-\frac{\alpha}{2}} \, \tfrac{s}{\sqrt{n}} \ , \ \bar{x} + z_{1-\frac{\alpha}{2}} \, \tfrac{s}{\sqrt{n}}\right]$$

Result to :

```
n          = 5369
mean       = 0.5866
std        = 0.0477
95% CI     = [0.5854, 0.5879]
```

Adding more features

$Metric/Method$	$Base$	$Withmorefeatures$
MSE	0.003	0.007
MAE	0.040	0.067
R^2	0.322	0.000

Adding more featured worsten the performances compare to a simple time serie. On explenation can be a noisy data that confuses the model compare to the volatility data that can be noisy but with only one feature dim, this can be easier for the model. Also we can assume other data type could be more relevant, for exemple macro-economic data like news analysis, but this it out of scope.

Other type of numerical data could be relevant but adding too much features seems to lead to worse results.

Option price model with enhance volatility model + benchmark

$Metric/Method$	$Base$	vol_LSTM_EMD
MSE	0.022	0.017
MAE	0.121	0.102
R^2	0.000	0.000

Around 20% improvement of the MSE
Same for MAE
R^2 to investigate...

Hyper-parameters tunning

```
param_grid = {
    'hidden_dim1': [128, 64],
    'hidden_dim2': [64, 32],
    'lr': [1e-3, 1e-4],
    'weight_decay': [1e-5, 1e-6],
    'dropout': [0.1, 0.2]
}
```

Not enough time yet to do the computation

Iteration 17

June 09, 2025

Bigger model

- RandomizedSearchCV (HL1 size, Batch size)

```
param_dist = {
    'module__hidden_dim1': randint(32, 256),
    'module__hidden_dim2': randint(32, 128),
    'batch_size': randint(32, 256),
    'optimizer__lr': loguniform(1e-4, 1e-2),
}
```

- Training size
- Loose function

Probabilistic model

.expanding, dist of each days, or past ten days

Option pricing model benchmark

In progress : need to calculate the 50/50 expected value of winning, then compare BS VS EMD-LSMT + RNN option price

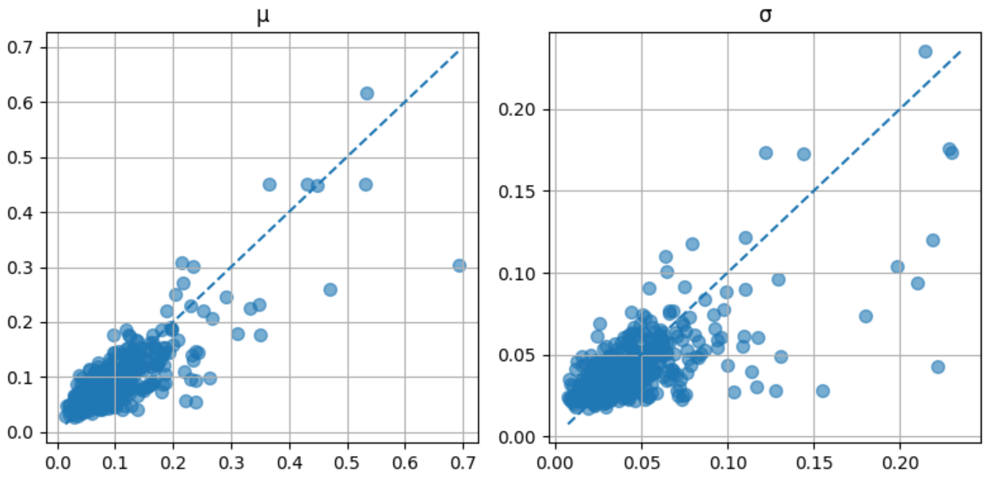
Iteration 18

June 16, 2025

Probabilistic model

Following a logic of a 10 days window to predict the next distribution of a 10 days period. For exemple days 0 to 9 are used as D1 to predict D2 ie the distibution of day 10 to 19.

Result :



```
MSE : 0.0002
```

Option model benchmark

After Louis Bachelier work, we consider a true fair price to be a nil expectation for both the seller and the buyer of the option. In that case, knowing histirical data it is possible to determine the true faire price of an option.
Then training the model on it.

The goal is to compare to that reference the BlackScholes model, as well as volatility LSTM and volatility EMD_LSTM.

Hyperparameters optimization

Best result found :

```
  Best parameters: {'batch_size': 225, 'module__hidden_dim1': 118, 'module__hidden_dim2': 47,
```

```
Test MSE : 0.003
Test MAE : 0.048
Test R²  : 0.353
```

Compression

$Metric/Method$	$Base$	$Compressed\_model$
MSE	0.003	0.003
MAE	0.039	0.043
R^2	0.430	0.432

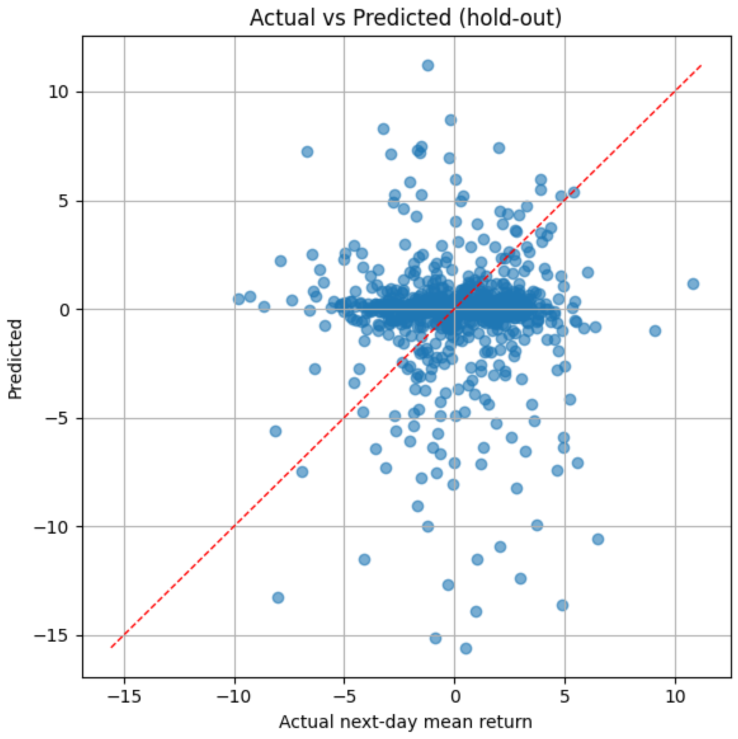
Iteration 19

June 23, 2025

Probabilistic model

11 days pred with 10 days rolling window $p \sim \mathcal{N}(\mu, \sigma)$

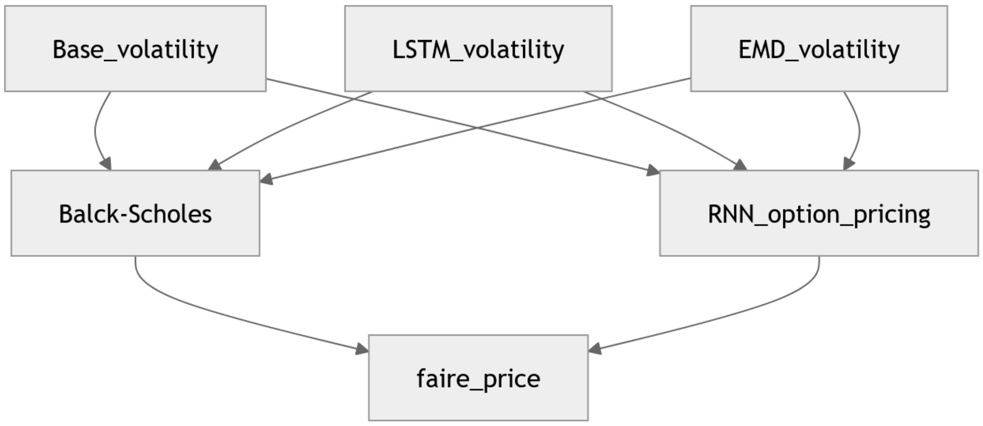
- μ 11th day returns based on the last 10 days returns distribution
- σ LSTM part with volatility prediction



Calculate a confidence interval to gain an option price distibution.

Option pricing model

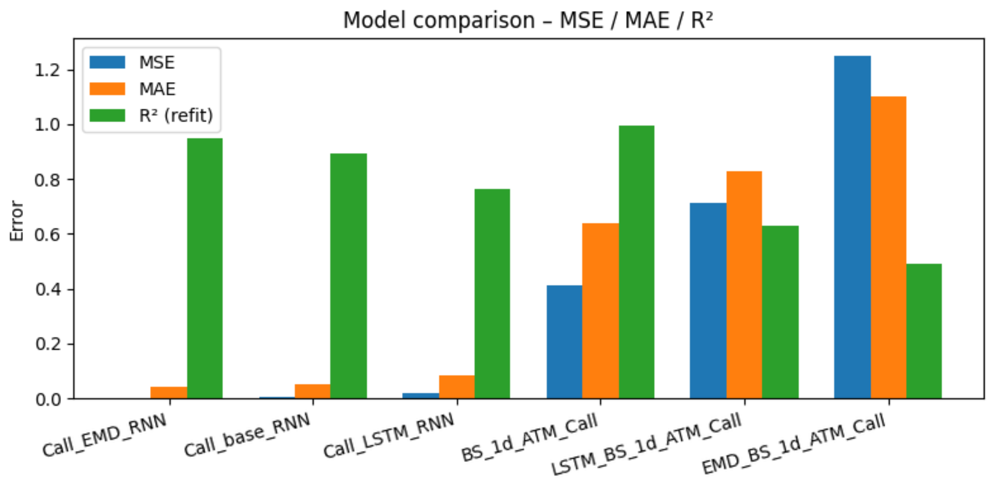
The benchmarking of the option pricing model with different methode and volatility estimators can be represented as follow :



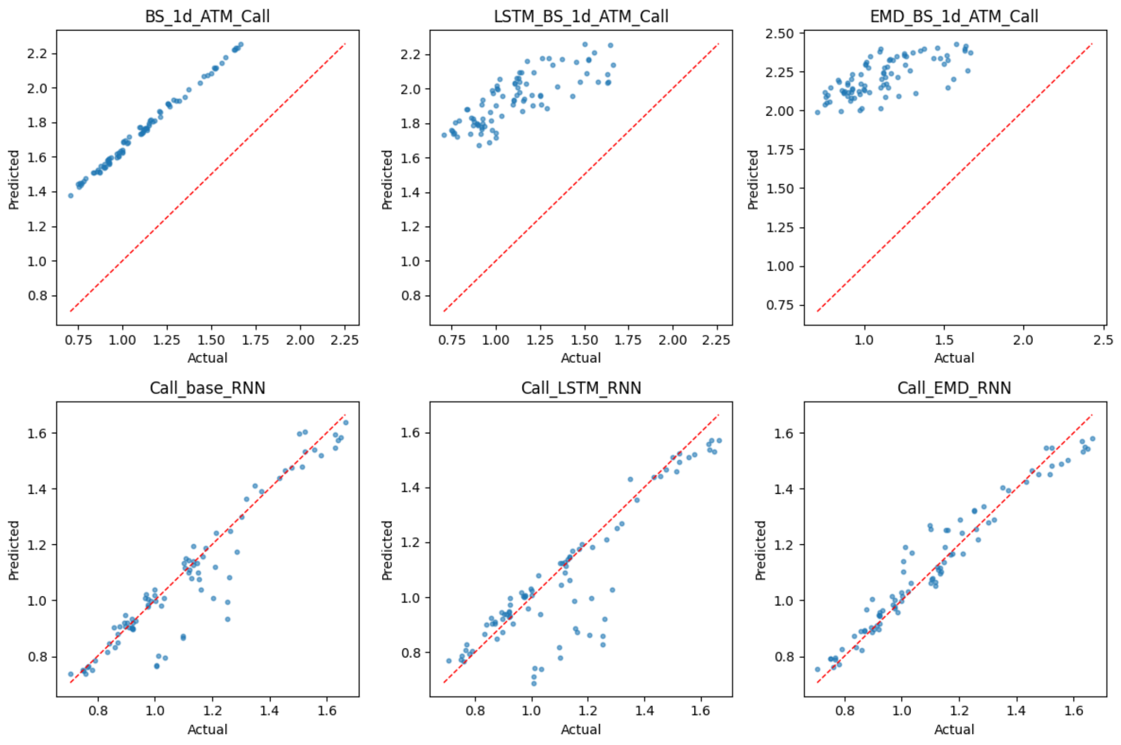
After testing each combinations we get :

Metric/Method	*BS_Base*	*BS_LSTM_vol*	*BS_EMD_vol*	*RNN_base*	*RNN_LSTM_vol*	*RNN_EMD_vo*
MSE	0.41	0.71	1.24	0.007	0.017	0.003
MAE	0.64	0.82	1.10	0.05	0.08	0.04
R^2	0.99	0.63	0.48	0.89	0.76	0.94

Ploting all these metrics on a bar graph give :



Comparing predicted vs actual value of these different methodes :



Among the RRN methodes, we can see that the EMD+LSTM volatility one is the best performing. Also, among the Black-Scholes, we can observe a bias, due the the faire price calculation methode.

Hyper parameters optimization

HL1 : 100
HL2 : 48
Batch size : 152
lr : 0.0005